TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

AUTOMATION OF THE WAREHOUSE

LAO AUTOMATISEERIMINE

MASTER THESIS

Student: Adeel Nawab

Student code: 196392MAHM

Supervisor: Dr. Dmitry Shvarts, Researcher

Tallinn 2021

(*On the reverse side of the title page*)

AUTHOR’S DECLARATION

Hereby I declare that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

“.......” .... 20…..

Author: ....
/signature /

Thesis is in accordance with terms and requirements.

“.......” .... 20….

Supervisor: …........................
/signature/

Accepted for defence

".......".....................20….

Chairman of theses defence commission: ....
/name and signature/

Non-exclusive Licence for Publication and Reproduction of GraduationTthesis[1]

I, Adeel Nawab (date of birth:24/08/1990), hereby

1. grant Tallinn University of Technology (TalTech) a non-exclusive license for my thesis Automation of the Warehouse.

Supervised by

Researcher, Dr.Dmitry Shvarts,

1.1 reproduced for preservation and electronic publication, incl. to be entered in the digital collection of TalTech library until the expiry of the term of copyright;

1.2 Published via the web of TalTech, incl. to be entered in the digital collection of TalTech library until the expiry of the term of copyright.

1.3 I am aware that the author also retains the rights specified in clause 1 of this license.

2. I confirm that granting the non-exclusive license does not infringe third persons' intellectual property rights, the rights arising from the Personal Data Protection Act, or rights arising from other legislation.

[1] *Non-exclusive Licence for Publication and Reproduction of Graduation Thesis is not valid during the validity period of restriction on access, except the university`s right to reproduce the thesis only for preservation purposes.*

______________ (*signature*)

______________ (*date*)

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: Adeel Nawab, 196392MAHM

Study programme, MAHM31/18- Mechatronics

main speciality: Mechatronics

Supervisor(s): Researcher, Dr. Dmitry Shvarts

Consultants: …………………………………………………………….(name, position)

……………………………………………………………………………… (company, phone, e-mail)

Thesis topic:

(in English) *Automation of the warehouse*

(in Estonian) *Lao Automatiseerimine*

Thesis main objectives:

1.Install TC3 and how to use the RS232 serial communication and use its slave. It is associated with it with the help of Transponder and Reader.

2. After learning serial communication, we will learn how to read and write RFID tags or cards.

3. Database management system using SQLite in the embedded PC CX2042 and learn how to use it and deploy according to our needs with the help of a prototype.

Thesis tasks and schedule:

No	Task description	Deadline
1.	Literature Review and Introduction Chapter	January 2020
3.	Design development and final working	March 2021
4.	Making a prototype of the system	April 2021
5.	Experimental results and optimization	April 2021
6.	Thesis defence	May 2021

Language: ………… **Deadline for submission of thesis:** "……."………20….a

Student: …………………….. ………………… "……."……………………………..20….a

/signature/

Supervisor: ………………… …………………….. "……."…………………20….a

/signature/

Consultant: ………………… ……………………. "……."…………………20….a

/signature/

Head of study programme: …………… ………………… "……."…………………20…..a

/signature/

CONTENTS

LIST OF FIGURES

PREFACE

This thesis was proposed by Researcher Dr. Dmitry Shvarts who encouraged me and guided me on the right way to complete this master thesis, and the author has enthusiastically accepted the challenge of pursuing it as my Masters' Thesis. The author has a passion for Automation Industry, telecommunications, and RND to make the solutions more user-friendly, efficient, and affordable. All the experimental and practical work on the thesis has been done in Tallinn, Estonia, using the hardware and software available at the mechatronics laboratories in the Department of Mechatronics of Tallinn university of technology.

The thesis is about teaching how the Automation can be made very efficient and safe and designed a whole automated environment using Beckhoff embedded PC, and RFID module made by pepper+fuchs both are connected using RS232 serial communication. These are connected using an SQLite database, used as a record for entry and dispatching of pallets from the warehouse. The main reason behind this is to make warehouses much safe and advance to help the workers.

I want to express my profound thanks to Professor Mart Tamre, and the Mechatronics department staff for the huge support provided to me whenever I needed to conduct my studies and the thesis. Also, Taltech University accepts me as a master's student and provides all the facilities for a remarkable study experience.

Warming and encouraging thanks to my family and friends for all the support given to me to fulfill the dreams in the best way possible and achieve the target in the desired time frame.

Keywords: Automated Warehousing, Warehouse safety, RFID Automation, Master's Thesis, Beckhoff enhancements, SQLite Database.

List of abbreviations and symbols

RFID- Radiofrequency Identification

PLC- Programmable Logic Controller

SKU- Storage keeping Unit

SQL- Structured Query Language

TC3- TwinCat 3

DWMS- Digital warehouse management system

UHF- Ultra High Frequency

ECA- Event condition action

DP- Decentralized periphery

IoT- Internet of things

SSD- Solid-state drive

KB- Kilo Bytes

UPS- Uninterruptible Power Supply

NVRAM- Non-Volatile Random-Access Memory

GB- GigaByte

CPU- Central Processing Unit

LTSB- Long-term Servicing Branch

PCI- Peripheral Component Interconnect

PDA- Personal Digital Assistant

CRDF- Continuous Review system with Demand Forecasting

WSN- Wireless sensor network

DBMS- Database management system

WCS- Warehouse control system

1. INTRODUCTION

The work aims to design and develop an automated system for the warehouse process.

Currently, the workers must face problems with manual inventory location. They are unable to use the warehouse effectively as per capacity, and due to this, many loopholes are created because Bar code on pallets needs to be in front of the sensor, and whenever there is a bulk of pallets moving in or out, the occurrences of error are reported, a traditional sensor cannot work it is way out. The focus is to improve the worker problem by using radio frequency identification (RFID) technology, and therefore, pallets do not need to be aligned with the sensor. It can read the pallets using radiofrequency in a specified zone.

The warehouse will be pre-defined inventory by using SQLite. Even the warehouse has its method of dealing with SQLite, and it is always customized according to the customer need, and it is evolved in any facility.

In this project, there will be an implementation of RFID tags on pallets, and these pallets will be moved on the floor by the forklift, and whenever these pallets cross a gate there will be an operator who will add the information in the database with the date and all the information required to locate the pallets and it will be removed when the pallet is dispatching from the warehouse.

The motivation behind this is, finding a cost-effective solution for this problem where we can find an easy solution, and which is easy for the end customer and any automation engineer can upgrade it in the future and this includes a high efficiency because of automatic entry and removal of pallets from the system which will decrease the number of mistakes and provides a better solution for the warehouse.

The expectations and further improvement of this topic include the design and prototype of the expected system and further development.

Initially, install associated TC3 Software from the internet and learn how to use the serial communication Rs232 module and use its slave—associated with control head interface with Transponder and Reader's . Develop visualization in twincat XAE and with these buttons it will be easy for the operator to read and write data in the database.

Database management system using SQLite in the PLC and learn how to use it and deploy according to our needs. After learning serial communication, we will learn how to read and write RFID tags.

Embedded PC series CX2042 with the RS232 communication module and TwinCat software – TC3 integrated with visual studio 2013 with Packages:

- TF6600 TC3 RFID Reader Communication
- Pepperl+Fuchs RFID
- TF6420 TC3 Database Server

The CX2042 has an Intel Xeon CPU with a clock rate of 2.2 GHz (4 cores). A fan operating with ball bearings and speed monitoring is integrated into the basic CPU module. CPU is the basic module that contains the main memory with a size of 8 GB RAM, optionally available with up to a capacity of 32 GB. It boots from a CFast flash memory card where both the operating system and user programs and data are stored. The CPU has a Built-in 128 kB NVRAM, which keeps data memory if no UPS is available.

Microsoft Windows 10 IoT Enterprise LTSB 64 bit is used as the operating system. Twin CAT 3 enables automation projects to be distributed across the Intel Xeon Central Processing Unit's different cores.

All system modules of the CX2000 series for left- or right-sided functional extensions can be connected or add more modules. Internally the modules are interlinked with PCI Express's help and can be plugged into the CPU in the real environment.

The main power supply for the CPU module comes from a CX2100-0014 power supply module. Up to two mass storage modules (either CX2550-0010 CFast modules or CX2550-0020 2½-inch SSD modules) can be plugged in between the power supply unit and the CPU, the use of up to three mass storage devices in total.

1.1 Thesis structure

This part will give a brief description of the chapters on this thesis a short outline for the reader to understand better what to read.

Chapter 1 Introduces the topic and the study's related to the thesis accomplishments, and the visions are detailed in the study.

Chapter 2 contains an analysis of previous studies conducted concerning the thesis's objectives referring to the published data. The section further discusses the explanation of the importance of the thesis.

Chapter 3 consists of the methodology of the thesis where it describes the proposed solution, selection of the hardware and software that will be used in the project. Furthermore, it provides details on two approaches to the solution and selection of the most suitable approach.

Chapter 4 comprises a detailed explanation of how the selected approach is developed. This includes the process flow chart, the programming methods used in PC, implementation of data communication methods, and explanation of the user interface and testing.

Chapter 5 overview and discussion provides the details on explanations of the project, improvement in the future, and summary of the thesis.

2. LITERATURE REVIEW AND BACKGROUND

A data warehouse may be a data store that gives online expository handling instruments for the intelligently information examination and investigation. Decision-makers might investigate the information warehouse with diverse measurements, and suitable concept progressions to report comes about they require. In any case, for genuine library practices, the current common ways to deliver reports are to inquiry related tables of library robotization frameworks by fitting outlined SQL commands based on their claim proficient works. It is time-consuming and resolute. In this manner, this considers endeavors to create information distribution centers pertinent to library applications. We analyzed the reports necessities of Taiwan public city and college libraries and after that plan the truth tables and measurement tables of comparing information stockroom. We executed such information stockroom from the information of a few open library robotization frameworks. It appears that applying information distribution centers to create reports of libraries is feasible and efficient.[1]

Increased clarity and efficiency of material and product flow in warehouses will determine logistical units' properties (e.g., small bins, palletized goods, number of pallets, containers). It presents pallets counting or monitoring its item, determining the pallet's number and characteristics, such as its location, weight, and time. If we see it, this is done by combining the Marlo Automotive positioning that uses passive planar markers, an RFID system, a dimensioning system that employs depth sensors, and a loaMarlonge detection system mounted on vehicles. The proposed approach was developed and evaluated in a real-world testbed. This enabled us to transfer the subsystems' accuracy to our new pallet monitoring system, i.e., we achieved a pallet positioning accuracy real-world testbed dimensioning accuracy of up to 5 cm in each dimension and highly accurate pallet identification. By fusing the data from these subsystems, we could generate the pallet as mentioned above information for subsequent monitoring and control of warehouse operations in real-time.[2]

The warehouse's typical logistics software is Warehouse Management System(WMS) and Warehouse Control System(WCS). WCS is different from WMS, and WCS aims to manage a broad range of material handling equipment in the warehouse. We analyzed the vital functions and limitations of the existing WSC and suggested a new architecture for WCS. New architecture and functions of Smart WCS/ECS are suggested to address such requirements and limitations. SMART WCS/ECS will enhance the efficiency of warehouse operation.[3]

Aiming at product information management in modern enterprises, the warehouse management system uses the J2EE architecture based on B/S mode. It uses SSH, the more popular integrated open-source framework in a web application, to improve the system's security and reliability. It mainly introduces the architecture and database design of the system. The design of user permissions and data decisions of this system is amiable, thus providing users a better experience. The mobile terminal based on IOS service is also accessible to the system, increasing the system's operability. Practical results show that the system has the advantages of simple operation, accurate data analysis, and processing, significantly improving warehouse management efficiency.[4]

The logistics industry is paying more attention to warehouse management with the continual growth of IT, using the latest computer networking communication technologies, and changing warehouse management. The method of introducing new inventions for warehouse management has become a very vital need of giant warehouses. The Hongxing logistics company's warehouse management information system and planning the business process, setting up feature modules for warehouse management. The need and demand for smart warehousing services improve with creating new ideas and developing warehouses where everything is precisely managed, and that is possible only with the help of a good database system that will organize the statistics of the smart warehouse without any need for human resources.[5]

The project research discusses patterns in constructing current logistics principles in warehouse management. The meaning, the practical advantages, and prospects of such methods are given. It is considered in the article to use modern kinds of storage control. Detailed definitions of creative logistics systems are given in the article. The study of logistics principles enables their attributes, details, and application to be defined.[6]

A data warehouse is a system for collecting, organizing, holding, and sharing historical data. Business users use it for decision support. Business users run queries in one manner or another on the data warehouse environment to support their process. Hence, Data warehouse testing plays an essential role in the system. Data warehouse testing comprises exhaustive testing of a Data warehouse during its design and on an ongoing basis for the incremental activities. Data warehouse testing is standard today because of the Increase in companies and the load of the goods, so all that must be managed in a system that will be perfect. The manual y data validation and verification process are time-consuming and not accurate. This work introduces an automated data validation strategy to reduce the data warehouse testing time, thus reducing the testing cost and attaining good data quality in less time.[7]

With the modern development of Internet technology, the unit's warehouse still uses paper records to register and go to storage. The following combination of the actual situation, the use of the current popular front-end framework to do a suitable unit application lightweight warehouse management system, the front end mainly uses angular + bootstrap for page management, the back end using Java + spring-boot framework for business logic and data processing, the server uses Nginx to implement reverse proxy for pages and tomcat.[8]

With the intelligent warehousing and logistics industry's vigorous development, how to carry out efficient, accurate, and large-scale asset identification has become an important research direction, and development goal, but traditional warehousing management generally adopts non-Automatic and paper documents as the entry and exit—library credentials, completely manual and less efficient. A radio frequency identification (RFID) technology is combined with a warehouse automation transmission system to design an RFID radio frequency identification device. Simultaneously, a radio frequency server program and an upper application service program are constructed, and a set of RFID-based warehouse automation solutions is proposed and implemented. The experiment results show that the system can fully meet the needs of extensive warehousing and logistics enterprises.[9]

This study gives an overview of the design and implementation of an intelligent warehouse management system. Our solution's main emphasis consists of three software components: An Xamarin IoT application, which acts as the exchange interface with the end-users. A web browser contains a database and ensures the persistence and communication between the system's different components and a robot that moves products in the warehouse according to storage and shipping requests. Our solution is designed to allow the various actors to have real-time information on the different workflows within the warehouse and all stock movements. Hence the need for a system that controls all zones and locations ensures communication between the various actors and Software components while optimizing data exchange and load consumption for IoT equipment.[10]

The need for this study is to give an overlook and comparison of best-known data warehouse architectures. Single-layer, two-layer, and three-layer architectures are structure-oriented depending on the number of layers used by the architecture. In independent data marts architecture, bus, hub-and-spoke, centralized, and distributed architectures, the primary layers are differently combined. Listed data warehouse architectures are compared based on organizational structures, with their similarities and differences. The second comparison looks at information quality (consistency, completeness, accuracy) and system quality (integration, flexibility, scalability). Bus, hub-and-spoke, and centralized data warehouse architectures got the highest information and system quality assessment scores.[11]

Tracking the location of items on the warehouse storage rack automatically is required by manufacturing companies to improve operational efficiency, productivity, and resource utilization. Due to the long reading range and faster data transfer rate, a passive ultra-high frequency (UHF) RFID system is explored to localize the rack's items in this study. A tag is incorporated into an item on the rack for localization using radio signals. The received signal strength indication (RSSI) and radio frequency (RF) phase are utilized for localization. A supervised clustering approach is developed based on location tags' measurements at the different zone of the rack. The localization accuracy is improved by using the kernel as a similarity measure. If the desired granularity of item location is not small, the lower price point per Tag makes passive RFID systems more economical than active RFID systems.[12]

According to this study, an automated storage and retrieval system (AS/RS) controls logistics operations using radio frequency identification (RFID) system recognition. The system has been designed and constructed focused on warehouse needs, especially for the transportation of pallets at short or long distances and accurate positioning of the pallets on the shelves. For the prototyping of the three-axis AS/RS system, two stepper motors for x- and y-axes and a DC motor with a speed reducer for the z-axis were used. Control is achieved by a Microchip RISC microcontroller (namely the ATmega32A). The RFID system succeeded in controlling the fully automated placement of pallets based on the PN532 transceiver. Furthermore, the AS/RS communicates via RFID with another material handling equipment, i.e., the conveyor belt system, and therefore it has been designed to support Internet-of-Things (IoT). Moreover, it has been designed to support ERP systems.[13]

The wholesale trading companies face new challenges in terms of the level of service, which is an essential factor that determines the revenue ratio and company profitability. Thus, businesses currently want to submit orders on schedule, in decent shape, and full to their corresponding clients. This file focuses on the value of warehouse management for manufacturers of mass consumption, as warehouses assess their operating costs and the cost of productivity. This improves the working environment through groundbreaking approaches such as lean warehouse and allocation methods to reduce downtime and distances. A pilot program of implementing the instruments listed in a wholesale distributor's warehouse has been implemented. The analysis resulted in a 22 percent improvement in the distance covered in the factory, the decrease in picking hours, an optimum arrival period, and a rise in 215,720, 22 sole. These findings are decisive for the delivery chain efficiency (inventory, expense, delay, and risk), as this industry's key to success is to fulfill customers' needs in the shortest possible time.[14]

It represents a concept for the semi-automatic Warehouse Inventory management Device (WIND) installed on reach truck (RT) vehicles. Inventory counting is a compulsory operational process in supply chain management. Current counting methods often threaten employee's health and safety and are not cost-efficient. The WIND is designed to take an automatic image recording to perform a manual counting process. The WIND system consists of a high-resolution industrial camera for visual inventory counting. A wheeled encoder and a laser distance measurement sensor are ready on the system to maintain semi-automatic functions. A sensor fusion algorithm runs in the background and provides localization among the warehouse racks. While an RT is on the move, its current location can be matched data sensors' data using the DHL Supply Chain warehouse management system. The camera unit has an automatic triggering mechanism that switches only when the RT is in the correct location. The captured photos are saved to an industrial computer on the WIND instantly and uploaded to the central database upon completing whole tasks. In the final stage, users can count inventories manually on a computer screen and then save them into the database. Organizations can correct records by the proposed system, identify flaws in the inventory process, and avoid discrepancies.[15]

The goal is both decision analysis and data warehousing technologies to increase administrative management quality and performance. First, the control framework is arranged into five modules: 1) Management module; 2) Service module for decisions; 3) Personal management module; 4) Management module for data provision; and 5) Maintenance module for systems. Second, four subsystems, i.e., 1) the database management system; 2) the foundation management model system; 3) the knowledge base managing system. And 4) the basic management methodology. Afterward, to build a system for administrative management, a decision analysis algorithm combined with a data warehouse is designed. Finally, to test the performance of our system, we implement an administrative management system. Experimental results show that utilizing the data warehouse technology, the quality of administrative management is significantly improved in four aspects: 1) Administration efficiency evaluation, 2) Quality administration evaluation, 3) Administration management satisfaction evaluation, and 4) Administrative management coverage evaluation.[16]

Database management systems (DBMS)have been in use for a long time, and they are used to store and save private or confidential data. DBMS must ensure the data stored is safe from malicious hackers' attacks. There are methods for detecting and preventing malicious SQL Injection in a standalone DBMS or Cloud space DBMS in this study.[17]

SQL injection is a security vulnerability resulting from unauthorized access to the victim database and services by attackers. Harmful codes can be injected into the victim database, all information in the database can be accessed, or the database can be deleted entirely and rendered unusable by using the SQL injection method. SQL injection attacks are a simple and effective method used frequently in hacking attacks against Web services. Despite all measures taken, SQL injection attacks continue to increase. The academic studies in this field are usually theoretical, having limited practical aspects. In this study, an SQL injection attack case against a web service was examined in detail. The study aims to evaluate SQL injection attacks in practice.[18]

Digital Business is an environment where nearly all physical activities turn into virtual assets in the value chain. The project described the historical benefits described in many businesses and use RFID cases in several industries around product logistics in the last years and the vision D-Business. An existing RFID solution architecture based on the reference EPCglobal/GS1 framework was modified to be extended to the IoT domain to fulfill these requirements. The project environment, the RFID automation strategies, and the innovative IoT solution architecture are presented.[19]

Automated warehouse management based on RFID and even object detection and remote accessing of data with cloud storage technology are proposed. The RFID program helps to identify better each RF tag, each of which contains a unique ID. In a warehouse, it is possible to effectively observe and view any object or container that has RF Tags. In addition, the cloud storage facility improves this performance. It implemented scanning multiple RF tags for this purpose and stored the Tag and its locations in the cloud. NODEMCU is a small device for handling the cloud by storing data to monitor the scanning process via Wi-Fi and Arduino. As the warehouse or household products and their location are stored in the cloud, the web or smartphone application can access them remotely.[20]

2.1 Conclusion

- The RFID transponder and receiver are required in this type of project because of no accuracy in the distancing of pallets containing the Tag, which a transponder must read for the entry and exit of the pallet.
- There is cloud usage in some studies, but most companies prefer their servers for the database because of security and easy access.
- Prototyping has been done using Arduino, which is not reliable for industrial use and the challenging environment.
- The technique of storing images in the database means it consumes high capacity and needs more storage, so most of the system's chances get stuck or overloaded.
- There is still the use of manpower in counting the pallets from the camera photos, which is not an accurate way of managing the warehouse and can make severe mistakes.
- Electric forklift busy counting instead of dispatching and unloading from trucks and wasting much time.
- A mechanical device, which is called WIND(warehouse inventory management device), is used. It is not an efficient and safe way of working in a warehouse environment, and lots of mechanical parts are needed.

2.2 The objective of the thesis:

- This project aims to provide an autonomous environment to the warehouse, allowing easy management of pallets consisting of RF tags.
- I am testing and doing different experiments with the transponder and receiver using communication module RS232 with CX2042 embedded PC by Beckhoff.
- Use of SQLite for the database management system to add and remove pallet tags in the warehouse.

- The database is upgradable and therefore enables the addition of new Tags and other modifications.
- This solution will provide high accuracy and efficient work in the warehouse environment.
- The project will be affordable and easy to operate.
- To provide good quality and innovative logistics services.

3. METHODOLOGY

The chapter will focus on a detailed explanation of the hardware implementation of the proposed solution and will explain the overview of the proposed solution. The hardware used, the selected software, and the communication parameters used in the project.

3.1 Proposed solution

As discussed in chapter 2, the Automation of the warehouse system to be completed should program the selected Embedded PLC using Twincat 3 integrated with visual studio 2013. In the solution, the PLC should read and write the pallets numbers using the RFID module and save all the information in the SQLite database. There are no specific tools required as used in previously made warehouses, for example, WIND. Once the PLC is programmed, it will keep the pallets and relevant data in the database storage and perform required tasks at any given time or requirement.

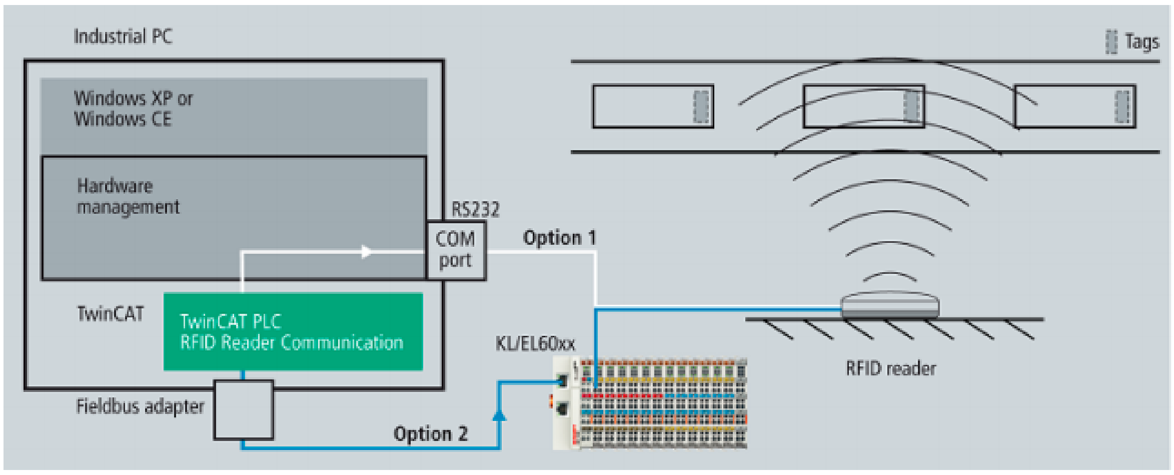


Figure 3-1 Overview of the system[21]

The proposed solution is described in the above Figure 3-1 describes the outline of the solution. Since the system needs to be read and write the passive tags, the movements of the pallets in the warehouse will be captured by using the RFID module, which consists of a Transponder, receiver, and tags for capturing that data. An easy way is provided to the worker to efficiently operate the warehouse database by checking the database inventory and the desired pallets' real-time location. The exact location of the pallets can be identified by the zones or some lights, which will indicate the pallet, and the dispatcher will know the place of pick and drop of the pallet. The user interface will have different controls and explain that in the stuff inside the warehouse.

At least one worker needs to monitor and stay in front of the PC and perform the database management to provide the information related to the pallet, which is going for the storage and its position and other measurements data. Once the data collection is done, the other process can be carried out, and the operator should move the pallet in the warehouse to its point to be store. Upon the connection, the operator can determine the position of the pallet and perform the dispatch of the stuff and goods according to the timing schedule. The entry and exit of the pallet from the database will make sure and make it more reliable than the pallet is removed, and there will be no mistake will happen because of accurate information available in the database.

All the information and related information of the pallet will be saved in a database for future reference and prove that data can be used to see that the pallet is being removed from the warehouse.

3.2 Hardware selection

The hardware selection is an essential section of the solution. It consists of many different automation Embedded PC and other hardware. The detailed description will be as below.

1. Beckhoff Embedded PC CX2042-0155
2. Serial Interface RS232 EL6002
3. RFID Control Interface
4. RFID Read/Write Head
5. RFID Tags

3.2.1 Beckhoff Embedded PC

Beckhoff incorporates pc technology and modular I/O to produce a space-saving industrial controller on the DIN rail with the CX range of Embedded PCs. By default, terminals can be attached to the right side of each embedded PC from the Beckhoff I/O range. The terminals are connected simply to the embedded PC; the connection is formed by clicking on it without further measures.[22]

Since the I/O units are directly connected, extra coupler costs are reduced, less wiring work needs to be performed, and the control unit can also be used in a lightweight and space-saving manner in the controlling cabinet. This allows for smaller and much more economical control cabinets or terminal boxes.[22]

The CX product set is ideal for all power ranges and blends the industrial PC and hardware PLC worlds. In the Embedded-PC product range, every user will find a suitable device: adapted to budget, performance class, and control complexity. The CX family contains several simple CPU modules for optimal adjustment to the respective mission. For various processors. The product line includes small control units with ARM processors, high-performance multi-core systems, and Intel® Xeon® processors, thereby providing the customer with an optimum value for money.[22]

The most suitable CX controller is selected based on the expected complexity and scope of the automation program, and there are many Embedded PCs available, and CX2042-0155 is the best for the project.[22]

Beckhoff CX2042-0155 Embedded PC, the CX20x2 Embedded PC series, is a modular control system intended for installation on a DIN rail. The system is scalable so that the basic PC modules, power supply units, system modules, and extension modules can be assembled and installed in the control cabinet or terminal box as required. The CX20x2 Embedded PC series is used in conjunction with Bus Terminals for recording digital and the transmittal and transmission of analog signals from sensors to actuators or controls.CX20x2 Embedded PC series combines the industrial and hardware PLC and worlds. The Embedded PC is configured to satisfy the defense class IP20 specifications. This includes defense against fingers and rigid extraneous artifacts up to 12.5 mm, but not water protection. Unless otherwise stated, system operation in wet and dusty conditions is not allowed. The electrical and technological data limits listed must be observed.[23]

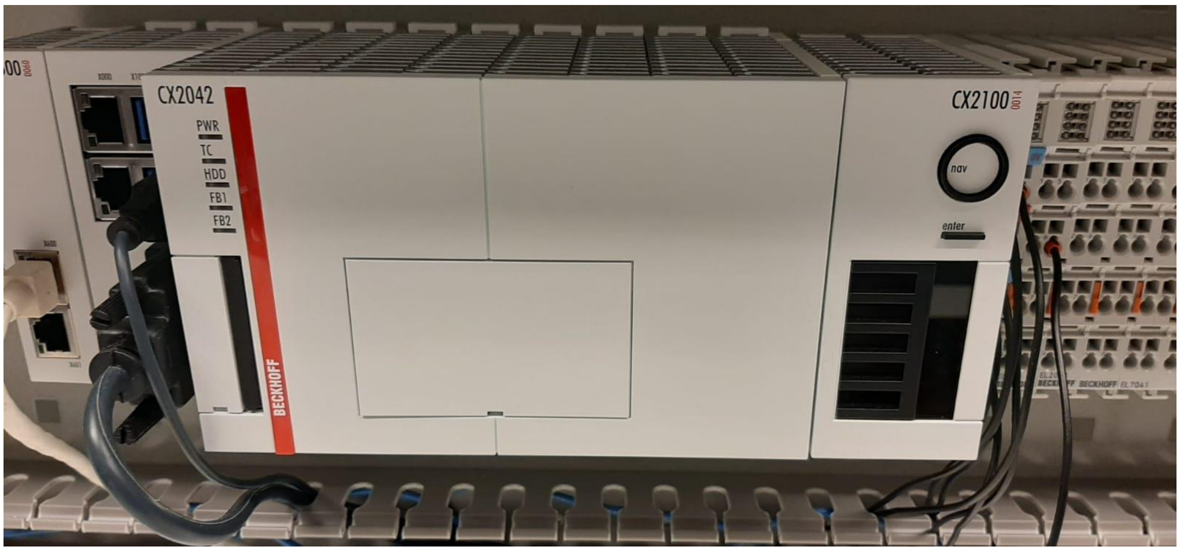


Figure 3-2 Beckhoff CX2042-0155.[23]

A module of a Generic CPU A completely working PC is the primary CPU module that can be used along with the smallest possible setup with a CX2100 power supply unit. The basic setup of the CX20x2 Embedded PC includes:

- a CFast card slot,
- two independent Gbit Ethernet interfaces,
- four USB 3.0 interfaces,
- a DVI-I interface.

Additional interfaces or storage media may be used to expand the standard CPU module, as necessary. Up to four device or Fieldbus modules can be connected on the left-hand side of the basic CPU board. The right side of the CPU module and the power supply assembly can be attached to up to four extension modules: two extra storage media extension modules (CFast card, SSD) and two USB extension modules. Microsoft Windows 10 IoT Enterprise LTSB can be used as the operating system. The standard CPU module features 128 kB of NOVRAM, which can be used as an alternative to a capacitive UPS or an external battery pack. In case of a power outage, critical data are processed and re-started in the NOVRAM.[24]

USB 3.0 interfaces (X100, X101, X102, X103) Figure3-3 USB 3.0 interfaces X100, X101, X102, X103. The Embedded PC has four independent USB 3.0 interfaces for connecting keyboards, mice, touchscreens, and other input or data storage devices. USB 3.0, pin numbering, interface. Notice that each system uses electricity. Any GUI is 900 mA restricted. The port is USB 3.0 and is of form An according to USB 3.0.[25]

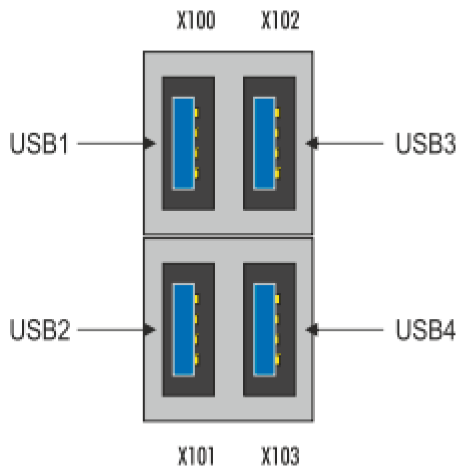


Figure 3-3 USB interface.[23]

Ethernet RJ45 (X000, X001) There is no switch incorporated. The two Ethernet interfaces are separate. Differently designed the individual Ethernet interfaces. The Ethernet interfaces (X000, X001) are designed to communicate with EtherCAT in the distribution state. Note that for a line topology, an additional switch is required.[23]

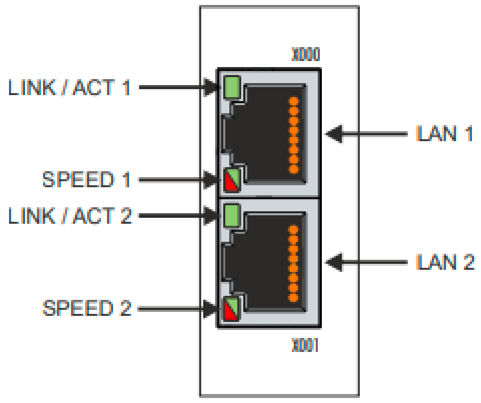


Figure 3-4 Ethernet Interface.[23]

X000 X001 LAN 1 LAN 2 LINK / ACT 2 SPEED 2 LINK / ACT 1 SPEED 1 . X000, X001 Ethernet interface. Both interfaces of Ethernet achieve 10/100/1000 Mbit rates. The connection status is shown by the LEDs on the left of the interfaces. The upper LED (LINK/ACT) shows the network connection of the interface. The LED is green will glow if this is the case. When the data is being transferred, the LED blinks.[25]

Speed of interaction is indicated by the lower LED (SPEED). The LED will not illuminate if the speed is 10 Mbit. The LED is orange if the speed is 100 Mbit. If the speed is 1000 Mbit, the LED lights red (gigabit).[25]

DVI-I (X200): Digital data is transferred by the DVI-I (X200) interface, connected to a digital or analog display. Distance to the input system determines the resolution on the display or the Beckhoff Control Panel. The total length is 5 meters. Beckhoff sells some panels with a "DVI expansion" integrated. This allows for a cable length of up to 50 m.[26]

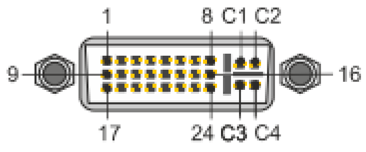


Figure 3-5 DVI (X200).[23]

3.2.2 Serial Interface Terminal (RS232)

Dual channels the serial EL6002 interfaces allow devices with both RS232 interfaces and a single D-Sub connector to be connected (9 pins). The interfaces are separated electrically from and from the EtherCAT. Devices attached to the EtherCAT terminals of EL6002 are connected via the coupler to the automation system.

The active transmission channel runs in an absolute duplex with 300 baud up to 115.2 kbaud independently of the EtherCAT higher-level structure. The interfaces RS232 ensure a high level of immunity to interference with electrically isolated signals. The EL6002 can be used as a standard Windows COM interface and the TwinCAT Virtual Serial COM Driver.[27]

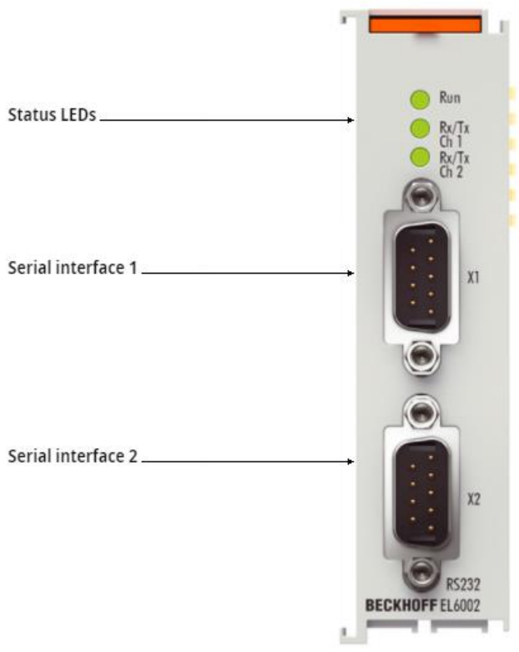


Figure 3-6 RS232 module. [28]

3.2.3 RFID control interface

The RFID IC-KP2-2HRX-2V1 IDENTControl Compact from Pepperl+Fuchs, with its revolutionary architecture, provides a wide variety of advantages relative to other systems. The IDENTControl and IDENTControl Compact control interfaces are at the machine core.

The IDENTControl Compact control interface can be conveniently and flexibly customized to your specifications with optimized interfaces to all commercially available Fieldbus networks such as Profibus, Profine, Ethernet, CC-Link, standard communication devices (RS 232 or RS 485), and various reading/write heads solutions for frequency ranges LF, HF, and UHF.[29]

A Led indicates bus contact, linked read-write heads, and working read-write commands on the front of the building. The trigger sensors also improve the reliability of the program.

The device is also suitable for use in control cabinets and field applications under IP67. In the box is access to the higher-level bus, and all links are connected to the box. In case of system failure, this enables easy installation and quick, trouble-free replacement. The EMC specification is compatible with a high level of the reliable metal holder, grounding, and protective wires.[29]

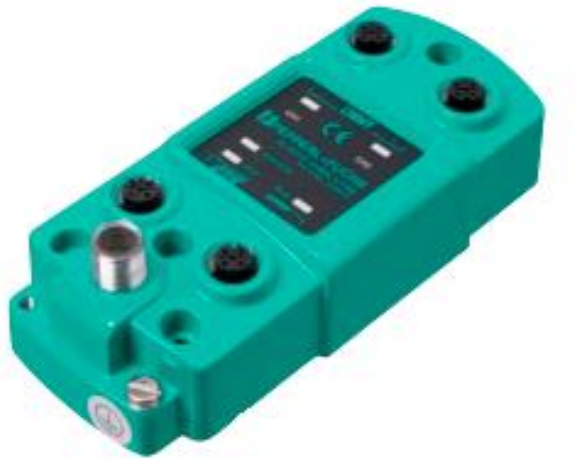


Figure 3-7 Control Interface IC-KP2-2HRX-2V1.[30]

"IDENTControl" family handling data RFID control units. They manage read/write operations and filter the raw data flow until the higher-level control scheme is reached. Both standard Fieldbus device interfaces (PROFIBUS, PROFINET, EtherNet/IP, TCP/IP, MODBUS TCP/IP, and EtherCAT) and serial interfaces are well-equipped for this purpose in our RFID controls. The Ethernet link makes multiple control units on a single network PC simple to parameterize remotely. If process control variables detect a hardware error, you are instantly alerted by an automatically created SMS response.

As these RFID control units allow identity data exchange on four different networks, up to four different RFID read/write heads can be attached to a single IDENTControl (LF, HF, or UHF). Because the machines can be easily mounted and shared, all links can be connected, and continuous processes are ensured. In small-scale RFID applications and restricted installation areas, the "Identitcompact" offers a spot-on solution for connecting one or two reads and write RFID heads.[31]

Both IDENTControl and IDENTControl Compact feature strong metal housing and guarantee optimal electromagnetic compatibility of the full RFID device solution by continuous shielding of all pepper+Fuchs RFID components. There are also advanced features such as multiplexing channels, triggering, or digital display.[29]

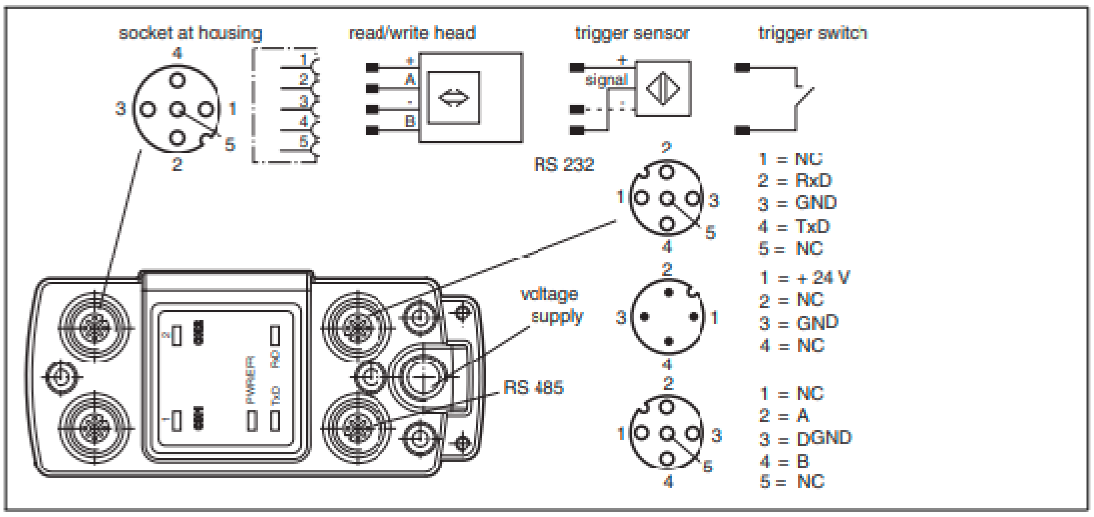


Figure 3-8 Control interface pin configuration[30]

Power supply Connect the power supply to an integrated voltage and reverse polarity indicator via M12 connector (green: correct polarity, red: reverse polarity). There is a connector with the following pin assignment on the housing and connection cables for the power supply. The compatible M12 sockets are available in various lengths and have an open cable end to link the IDENTControl Compact to a power supply.[32]

Core1:brown

Core2:white

Core3:blue

Core 4: black

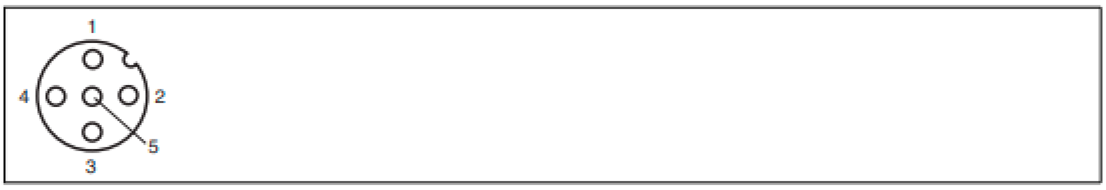
Figure 3-9 Connecting cable for power supply.[30]

Read/Write Head and Trigger Sensors A maximum of two reads/write heads (125 kHz, or 13.56 MHz) or read/write heads can be connected to the IDENTControl Compact electromagnetic connections (UHF with 868 MHz). A trigger sensor can be connected to sockets 1 and 2 rather than a read-write head. A reading/writing head may be assigned to the trigger sensor. PNP must be the activator sensor.[32]

Different transfer rates can be used 1200, 2400, 4800, 9600, 19200, 38400 bits/s. The interface works with (permanent) the following parameters:

- 8 data bits
- One start bit
- One-stop bit
- No parity

Connect the RS 232 interface with the M12 socket. You must place the cable shield on the thread in the connector plug.



Pin assignment of the M12 socket for RS 232

1	NC
2	RxD
3	GND
4	TxD
5	NC

You can use adapter V1S G-0.15M-PUR-SUBD for the connection.

Connection layout of the adapter for the RS 232 interface

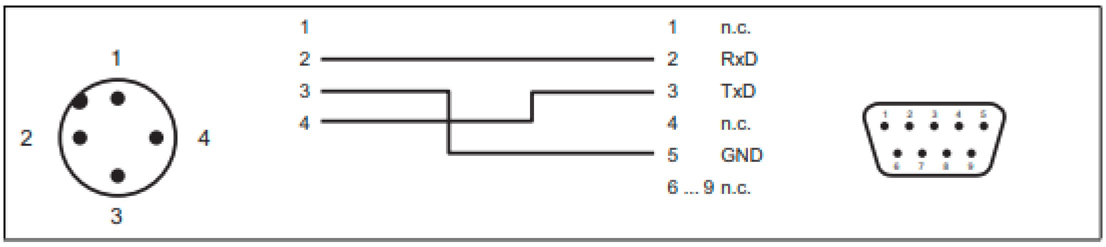


Figure 3-10 Connection layout for RS232.[30]

The serial interface link cable is attached to the host through a suitable cable through an M12 socket.

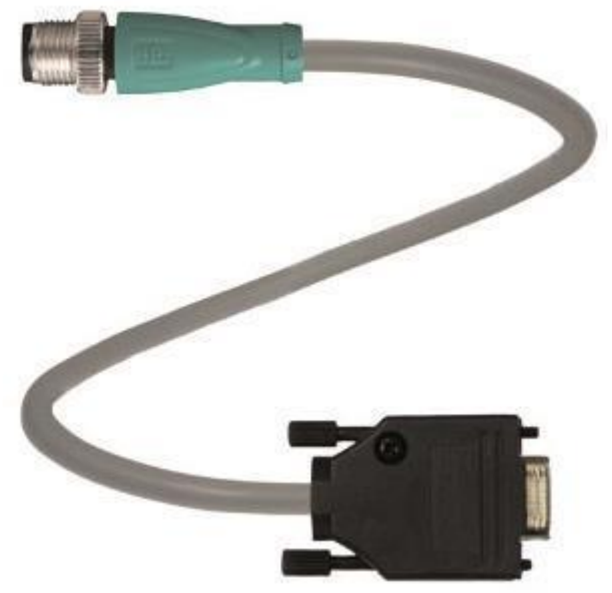


Figure 3-11 Connecting cable for Rs232.[30]

3.2.4 RFID Read/Write head

The header records the minimum distance from a metallic setting and neighboring read/write heads for the mounting. This attribute can be found in the read/write header datasheets. If they are positioned away from the metal surface or inclination in the metal surface plane, the cube-shaped heads, constructed on concrete, have a minimum of 75 percent of the nominal working distance. FP has a little work gap of at least 75 percent, even though it is entirely metallic.[33]

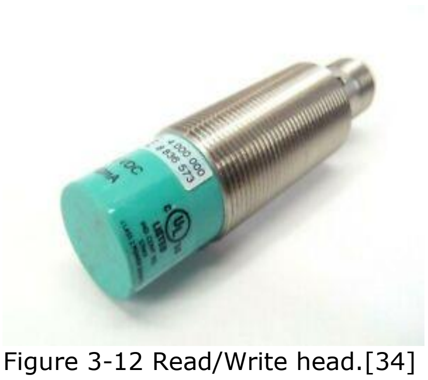


Figure 3-12 Read/Write head.[34]

Time and speed of reaction Values for reading/writing depend on the IDENT device technology used. However, the same calculation formula calculates the possible passing speed regardless of the technology used. Passing speed is calculated by:

$$V_{max} = \frac{\text{width of the reading field in meter}}{\text{Reading time in seconds}}$$

When the motion is passed at about half of the reading area, the read field width applies approximately to the side length of the read head for inductive devices. The reading field width applies. The read head is F15, for instance, 0.14 m, and the reading time for fixed code is 40 ms.[33]

$V_{max} = 0.14\,m \,/\, 0.04\,sec$ = 3.5 meter per second

However, one half of the speed of passage can be used because of potential noise effects in the environment:

$V_{practical}$ = Vmax / 2 = 1.75 meter per second

One-third of passing speed is recommended for practical use in devices with an operating frequency of 13.56 MHz, according to ISO 15693:

$V_{practical}$ = V max / 2 = 1.75 meter per second.

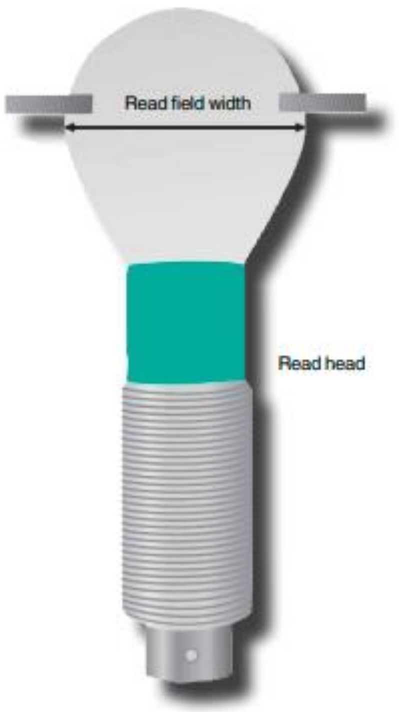


Figure 3-13 Range of Read/Write head.[34]

Time and speed of reaction Writing/reading values depend on the type of technologies used in IDENT. However, the same formula is used for estimating the potential passing speed irrespective of the technology employed. Vmax = read field width [m] calculates the passing speed and [s] Time for reading. [33]

Table 3-1 Read/write ranges in/on plastic (125 kHz read/write Tag at 25 °C, in mm)

Read/write head	IPH-18GM-V1	
Read only / Read/write tag	Reading	writing
IPC03-C1	0……………………….40 mm	0……………………….35 mm

Table 3-2 Read/write ranges, directly on steel with 10 mm spacing (125 kHz read/write Tag at 25 °C, in mm)

Read/write head	IPH-18GM-V1	
Read only / Read/write tag	Reading	writing
IPC03-C1	0……………………….30 mm	0……………………….30 mm

3.2.5 RFID Transponder

Conditions of assembly Conversion There are different aspects to the reading gap between the read/write head and transponders. There are two main factors: metal and liquid. The selection is affected by a humid climate. As opposed to a dry pallet, a wet pallet as a transporter cuts the range by 80%. When the transponder is mounted near or directly on metal, the range is often smaller. The reading area approaches its maximal extension by the increasing distance of the transponder to a metallic surface.[33]

The appropriate distance from the transponder is 20 mm from the metallic surface. Running transponders, for example, then exceed approx—90% of the reading distance in a non-metallic environment. Special spacers will be used on all our transponders and can be used accordingly. Furthermore, there are special transponders for inductive systems available for installation in metal and the surface mounting on metal.

Figure 3-14 RFID Transponder.[34]

Reading only code (IPC03-C1):

Reading time: time= 40 ms = 0.04 s

s = 140 mm = 0.14 m

$V_{practical}$ = 0.14 m x (2 x 0.04 s)

= 1.75 m/s

Reading 8byte:

Reading time (t) =(2 x 30 ms) + (100 ms) = 160 ms = 0.13 s

s = 140.00 mm = 0.141 m

$V_{practical}$ = (0.14 m) x (2 x 0.16 s) = 0.43 m/s

3.3 Software selection

Since I am using Beckhoff embedded PC, the software selection according to the hardware, and I have used other company RFID modules, it needs some libraries to select the compatible software, and it will be discussed deeply below.

3.3.1 Twincat 3.1 (TC3)

The simplification of TwinCAT 3 is one of the key approach's software engineering. Instead of developing own standalone tools, it is worthwhile to integrate them into familiar and developed environments for software development. For TwinCAT 3, Microsoft Visual Studio is the creative environment. Integrate TwinCAT 3 in the Visual Studio as an expansion.[24]

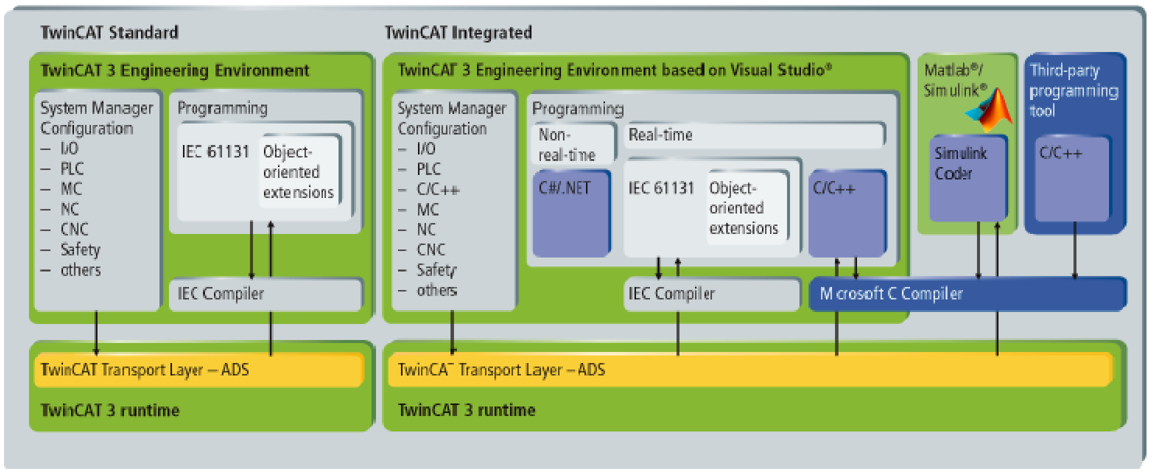


Figure 3-15 Twincat Standard and integrated.[35]

There are three more packages for the requirement for interfacing the hardware with the embedded PC are as follows:

i. TF6340 TC3 Serial Communication.
ii. TF6420 TC3 Database Server.
iii. TF6600 TC3 RFID Reader Communication.

3.3.2 TF6340 TC3 Serial Communication

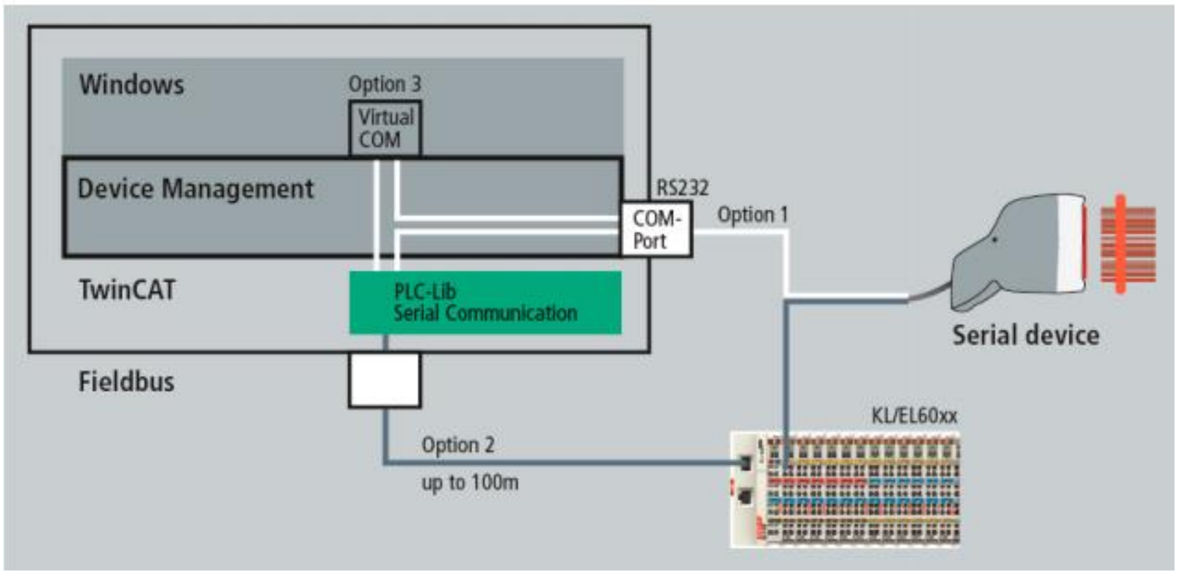


Figure 3-16 overview of serial communication.

Beckhoff terminals the network-based Fieldbus system from Beckhoff to access serial terminals at up to 100 m. The following Beckhoff terminals are supported:

- KL6xxx Bus Terminals
- EL60xx EtherCAT Terminal

3.3.3 TF6420 TC3 Database Server

Data interchange between TwinCAT systems and different database systems is permitted on the TwinCAT database server. Without any interference in the current computer code for small applications, the program can be used with a configurator. A massive library of PLC functional blocks for complicated tasks is available for the Database Server. For instance, SQL commands like Insert or Select can be used directly from the PLC. Procedures (Stored Processes) may be stored and then called from databases to remove loads from the PLC if necessary. In this case, databases are used in combination with the Stored Procedure, and the outcomes can be returned to the controller using parameters transmitted via a corresponding PLC function block.[35]

A broad array of various database systems, MS SQL, MS SQL Compact, MS Access, MySQL, PostgreSQL, DB2, Oracle, Interbase, Firebird, ASCII, XML files (e.g., TTX or.csv) and now NoSQL databases with MongoDB support, are supported by TwinCAT Database Server. [35]

3.3.3.1 Components of TF6420

• TwinCAT Database Server: The service along with TwinCAT begins and is terminated. The connection between TwinCAT and the database is created.

• Configurator: The TwinCAT Database Server enables the database parameter required for simple communication with the corresponding database to be visualized.

• PLC library: There are some feature blocks in the PLC library. They allow a database link to be established, a new table to be created, data to be written into any table using Insert commands and select commands to be read. Database entries can also be modified or removed and saved processes triggered. NoSQL databases have their feature blocks, for example, designed for the use of lightweight PLC JSON records. The operating theory is the same.[35]

3.3.3.2 Database Principle

Within the TwinCAT system, the Database Server communicates via ADS. Externally it links to the respective configured database.

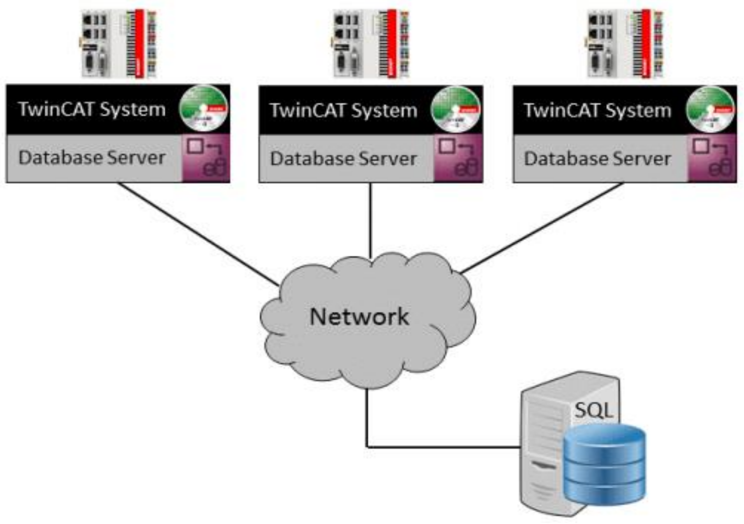


Figure 3-17 Database principle

SQLite is an ideal embedded device database. No installation is needed in this file-based SQL database, which is already implemented in the TwinCAT Database Server. The relationship database has the most SQL database capabilities and follows the SQL92 basic commands. The database allows fast and secure data storage. The database does not, therefore, encourage users to be distinguished. This is particularly suitable for the secure storage of local device variables.[36]

3.3.4 TF6600 TC3 RFID Reader Communication

The Communication Library of TC3 RFID Reader enables communication from the PLC application with the RFID reader. Read-only and read/write interfaces can be used with RFID readers.

The TwinCAT RFID library facilitates a variety of programs with various RFID reader functions. Implementation costs are minimal, so there is no requirement for detailed examination and implementation of a single manufacturer's interface protocol. The library automatically handles frame setup, telegram composition, classification of the command, telegram identification, and other protocol features.

The Compact RFID Reader is optionally available for Beckhoff Multi-Touch Control Panels. Compared to other versions of RFID Readers, TwinCAT does not approach the Compact RFID Reader with a TF6600 RFID Reader, but rather with TF6340 Serial Communication.[37]

3.3.5 Visual Studio 2013 integration

Visual Studio 2013 is a development environment by Microsoft™, and it was used in the project to the program required for Structured Text language to interface the Embedded PC with RFID pepper+Fuchs module.

3.4 Connectivity of the modules

As described in the overview of the system in Figure 3-1, the RFID module is connected to Embedded PC CX2042 using RS232 (EL6002) serial communication using Ethernet.

The connectivity between Embedded PC and the PC is done via an Ethernet connection. The PLC will not work correctly without the routing of the connection using an IP address. As per the specification, without the ethernet connectivity, the EtherCAT modules will not work at all, and it impossible to work without this because if there is no connection, you cannot access the connection of the modules and link their I/O's with your project. [21]

The connectivity between the PLC and the Pepperl+Fuchs RFID is done via a serial port connection. The RFID module is connected using M12 cables, and it is using 24v for the power. The Local Area Network (LAN) connection is made between the PLC and the server for the TCP/IP connection is done in the PC, and the client is the PLC. [21]

3.5 Different approaches to the solution

At the start of the project, the implementation of the approach began discussing. Two options were proposed. Use Beckhoff PLC, particularly by siemens.

i. Using siemens to make the solution more generic, the most used components are as follows:

- Communication module CM 1243-5 for connection of SIMATIC S7-1200 to PROFIBUS as DP Master module; PG/OP communication; S7 communication.
- RFID communication module RF160C for PROFIBUS DP-V0; 2 readers can be connected; without Connection block for PROFIBUS.
- SIMATIC RF300; Reader RF310R (GEN2); RS422 interface (3964R); IP67. -25 to +70 °C, 55x 75x 30 mm, with integrated antenna.
- SIMATIC RF300 Transponder RF360T EPOXY card, 8 KB FRAM, IP67. -25 to +70 °C, 85x 54x 2.5 mm, Minimum order quantity 10 unit.
- SIMATIC S7-1200, CPU 1214C, compact CPU, AC/DC/relay, onboard I/O: 14 DI 24 V DC; 10 DO relay 2 A; 2 AI 0-10 V DC, Power supply: AC 85-264 V AC at 47-63 Hz, Program/data memory 100 KB.
- PC Intel i5 or i7.

The system consists of the following major implementations:

- Install associated Siemen's software TIA portal from the internet and learn how to use the Profibus master communication card and use its slave—associated with it with the help of Transponder and Reader.
- Develop screen and display essential records on the computer screen in live mode.
- Database management system in the PLC and learn how to use it and deploy according to our needs.
- After learning Profibus communication, we will learn how to read and write the RFID tags.

In this solution, it was planned to use the same RS232 communication and programming language like ladder logic, but then because of some availability of hardware, this solution has not proceeded.

ii. Initially, install associated TC3 Software from the internet and learn how to use the serial communication Rs232 module and use its slave—associated with it with Transponder and Reader's help. Develop screen and display essential records on the computer screen in simulation mode.

Database management system using SQLite in the PLC and learn how to use it and deploy according to our needs. After learning Profibus communication, we will learn how to read and write RFID tags.

Embedded PC series CX2042 with the RS232 serial communication module.

TwinCat software – TC3 integrated with visual studio 2013.

Packages:

- TF6600 TC3 RFID Reader Communication
- TF6340 TC3 Serial Communication
- TF6420 TC3 Database Server

The first step is to start learning the connectivity of the packages using their manuals and the hardware according to the datasheets. The next step is to perform its operations are planned to conduct in two stages. The next is to interface the libraries with the required calling structured text. For this stage, some parameters must initialize in the program for the relation with hardware. [23]

4. MAKING THE PROTOTYPE

This chapter will discuss in detail how the selected approach was developed. It is important to provide a detailed outline of the system working that was developed and the database and how it was implemented.

The RFID read and writes setup. It will explain the user interface and its functionality and provide a detailed description of the programming of the Embedded PC and how to read and write the tags that are saved and recalled from the database and interact with the user interface using SQLite Database.

4.1 Warehouse Design and structure

Inbound Products which are coming for the storage and when the truck will arrive at the warehouse before going for the storage all the pallets will be categorized according to the companies and their designated zones in the warehouse.

The RFID read/write head installed near the door which can be read and then the specific tag will be read in the database system and operator will place all the information for the product on the pallet for example weight, company, date of arrival ,time and dispatching time. All these information will be saved in the database linked with the UID non changeable Low Frequency RF tag.

After reading of the tag for the pallet and saving the information by operator then it will move towards the zone which is linked with RF card and this will be taken by the forklift driver who will place it in the correct place so the tag will be remain attached so when it will need to be dispatched then the pallet will be easily recognized with RF card and this make it more easy for the forklift driver to load and unload the pallets from the warehouse.

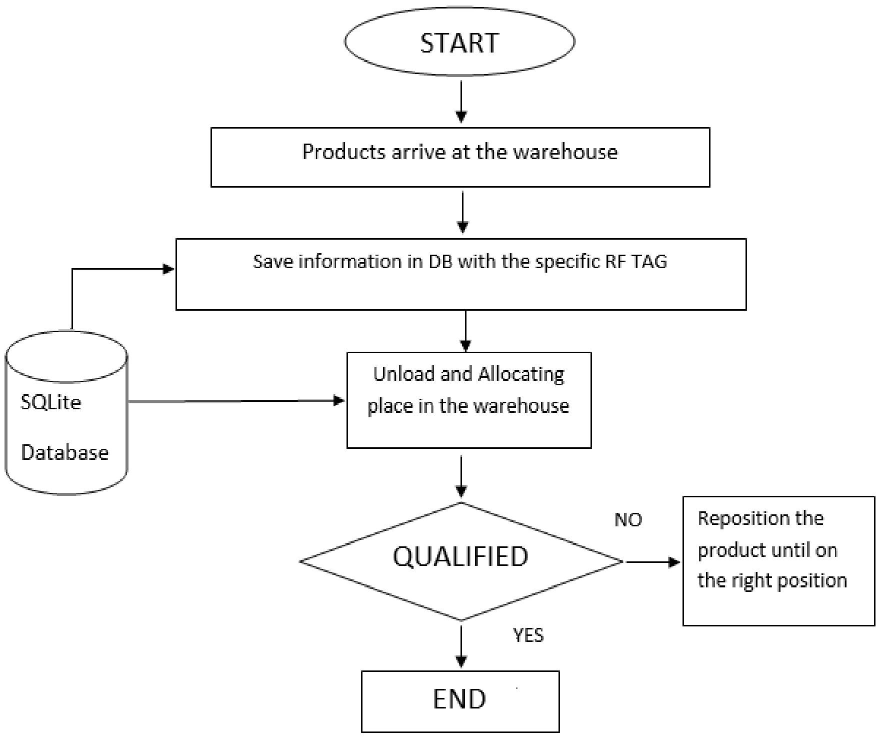

START
Products arrive at the warehouse
Save information in DB with the specific RF TAG
SQLite Database
Unload and Allocating place in the warehouse
QUALIFIED
NO
Reposition the product until on the right position
YES
END

Figure 4-1 Inbound Process

Outbound products when the product inside and need to be dispatch the operator will check in the database according to the date and time of dispatching and will be pick up using the forklift and when it is near to door it will be read again by the RF sensor and operator will update in the database that this pallet is dispatched from the warehouse, and it will keep the inventory save in the database for the further records.

According to the database all the pallets inside the warehouse linked with UID which means that every tag have its own UID, and it is used to store the information linked using the Beckhoff plc and database server are connected so everything the operator will write in database it will be save in the inventory for the records and will be call next time when it is needed.

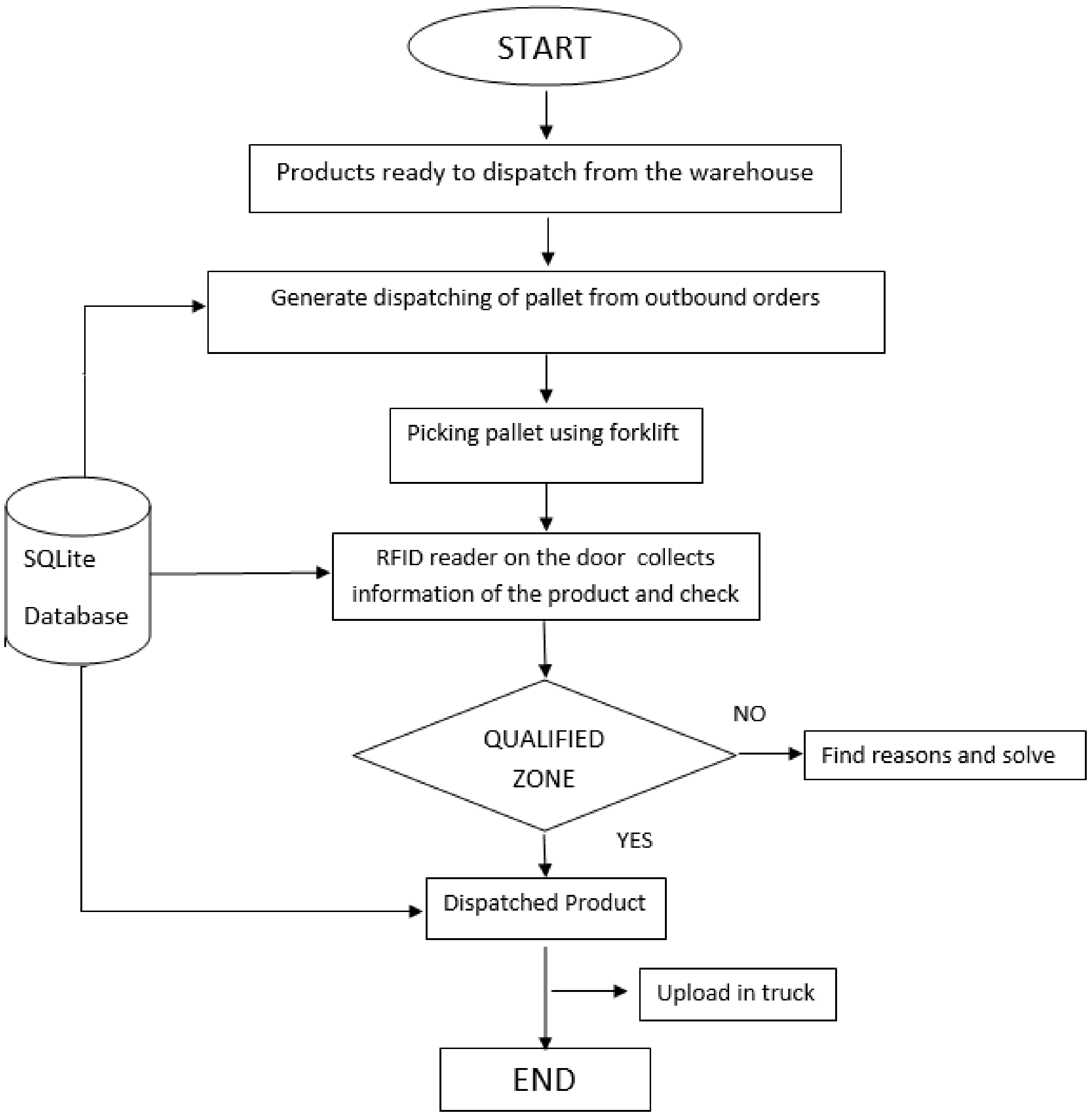

START
Products ready to dispatch from the warehouse
Generate dispatching of pallet from outbound orders
Picking pallet using forklift
SQLite Database
RFID reader on the door collects information of the product and check
QUALIFIED ZONE
NO
Find reasons and solve
YES
Dispatched Product
Upload in truck
END

Figure 4-2: Outbound process

4.2 Connecting with PLC

Connecting the hardware with the PLC is the main section of the program. This section is responsible for making a secure connection with PC and PLC with the serial communication module by adding the Embedded PC and its I/O devices. The control interface is connected using M12 cable for power and it is further connected with serial communication module using M12 to Sub-D adapter. After that step to be taken in the user interface to calibrate, checking the channel one lights and power of RFID control interface, powering light of RFID read/write head, operation of the RFID on how to start and run the program. The configuration is shown in Figure 4-3 below.

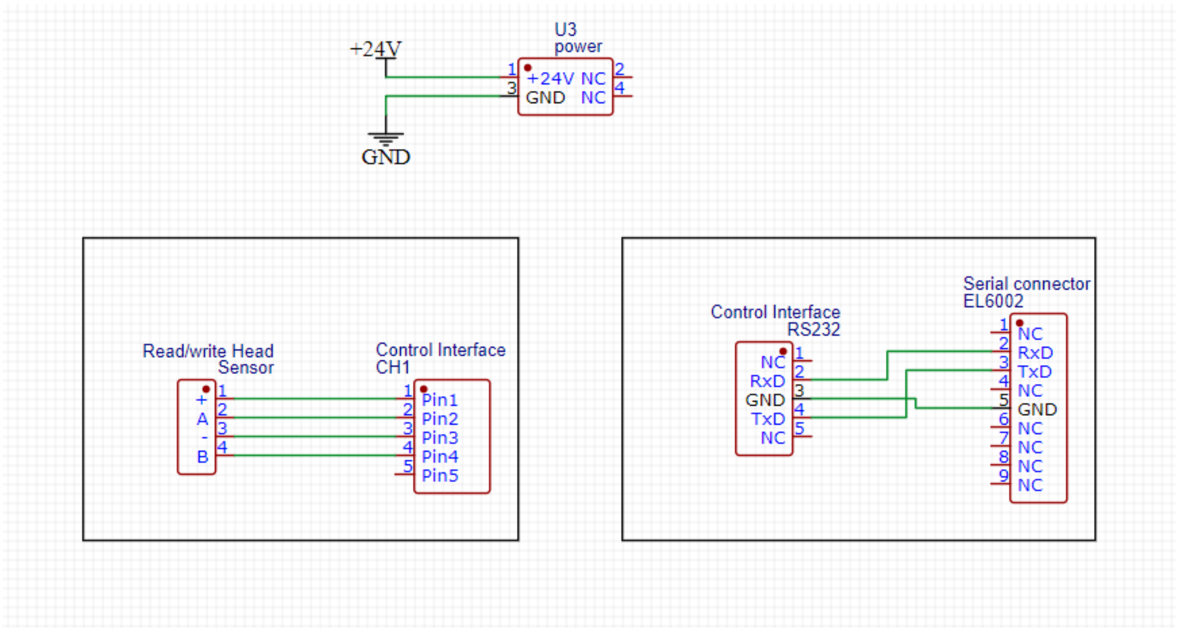


Figure 4-3: Pin's configuration.

First, both Embedded PC and Twincat XAE (Vs2013) software needs to be started. When all the modules and wirings are attached, then turn on the PLC, and it is connected using the ethernet connection for the establishing of IP connection between the PC and the hardware. It will automatically search the desired targeted PLC. After the connection with the CX2042 is done, then I/O devices must be scanned. After adding all the devices then it is ready to write the program.

After the connections and integration of the software, the Control interface will read and write the tags from the head interface, which will be further saved in the SQLite database.TF is the library series that is used to connect all the functions. Once the programming is completed, it will save all the warehouse data in the database and be easily read by the user.

Accordingly, the different tags will be assigned to the different pallets, and it is possible to add more tags to the program according to the warehouse requirements. When the RFID reads the data, it will be saved and can be recalled anytime when needed.

4.3 Database server

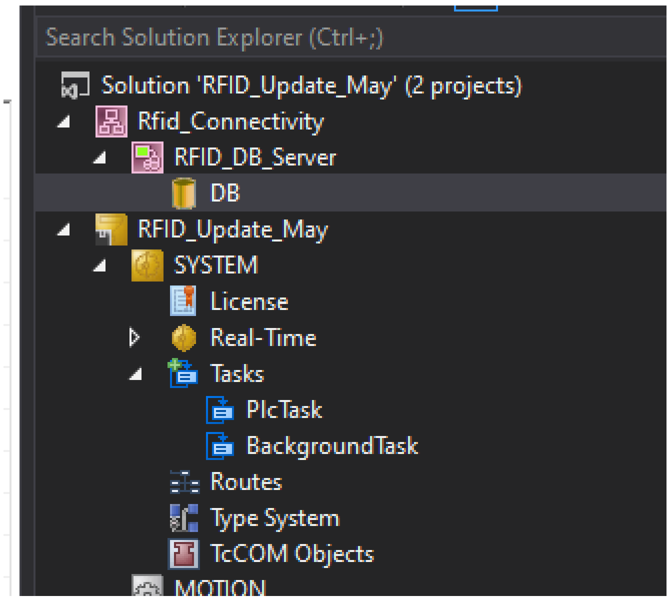
TwinCAT Database Server linked via an extension of connectivity and database server. The extension for the database server is needed to give access to the SQLite database. After the extension, SQLite can be selected as the database which will be used further.

This SQLite file database does not need to be installed as it is already built into the TwinCAT Database Server. The relational database provides much of the SQLite database capabilities and follows the SQL92 basic commands. The database allows fast, secure storing of data. The database does not permit users to be distinguished. It is thus especially suitable for secure storage of local device variables.

4.3.1 Database setup

This part will show how to connect the database using SQLite using the extensions of visual studio. Sqlite (PRG) (POU File Name): use to create Table, insert data and Read the Data from the Database. It is a program, so I cannot pass Input parameters like the tag I read from the RFID head. For this issue, I used Function block.

Database Read and Write: First, for database reading & writing, we first need to connect to the SQLite File using the Function block of FB_SQLDatabaseEvt and call its Connect function and pass the DB ID on Connect function like that FB_SQLDatabaseEvt.Connect(2); and then we check if there is no error, then it means our Connection is successfully created. After the successful connection, we can execute the queries to create the table, insert and read data using the function block of FB_SQLCommandEvt. We use FB_SQLCommandEvt.Execute method to execute the queries. Before executing, we need to write a SQL query in string format and pass it on to the Execute Method, and now our query has been finished its work.


Search Solution Explorer (Ctrl+;)

Solution 'RFID_Update_May' (2 projects)
Rfid_Connectivity
RFID_DB_Server
DB
RFID_Update_May
SYSTEM
License
Real-Time
Tasks
PlcTask
BackgroundTask
Routes
Type System
TcCOM Objects
MOTION

Figure 4-4: Database connectivity.

Once the database connectivity is installed, there is a DB_server that needs to be created to start the programming, as shown in Figure 4-3. It enables the programmer to easily select the features that need to be enabled in the program. In this case, the author has enabled the database access to the visual studio, and it will continuously run in the background. Following Structured text is used to connect the database to the PLC.

```
IF fbSqlDatabase.Connect(Database_ID) THEN
        IF fbSqlDatabase.bError THEN
                dbState := 255;
        ELSE
                dbState := dbState+1;
        END_IF
END_IF
```

Once the database is connected, the user must start creating a database and making the database table. If the database is created, it will make it easy to read and write the data from the RFID and create the database. The following code did it.

```
IF fbSqlDatabase.CreateCmd(ADR(fbSqlCommand)) THEN
        IF fbSqlDatabase.bError THEN
                dbState := 255;
        ELSE
                dbState := DBMode;

                IF HasTableCreated AND dbState = 2 THEN
                        dbState:= 3; // then insert data if db
                END_IF

        END_IF
END_IF
```

If dbState mode is on table creation and the table is created, then change dbmode to insertion, and then it will start inserting the data.

4.3.2 Write and creating the table in Database

To execute the SQLite query to store data or create the table in a database. The SQLite command must be created to execute the SQLite query. So then it will allow to create a table and insert data in that table.

The database will not be created in the visual studio, but it will be saved in a separate DB file, and it is readable by using a notepad or online SQLite viewer. The program code that is performing this operation is shown below.

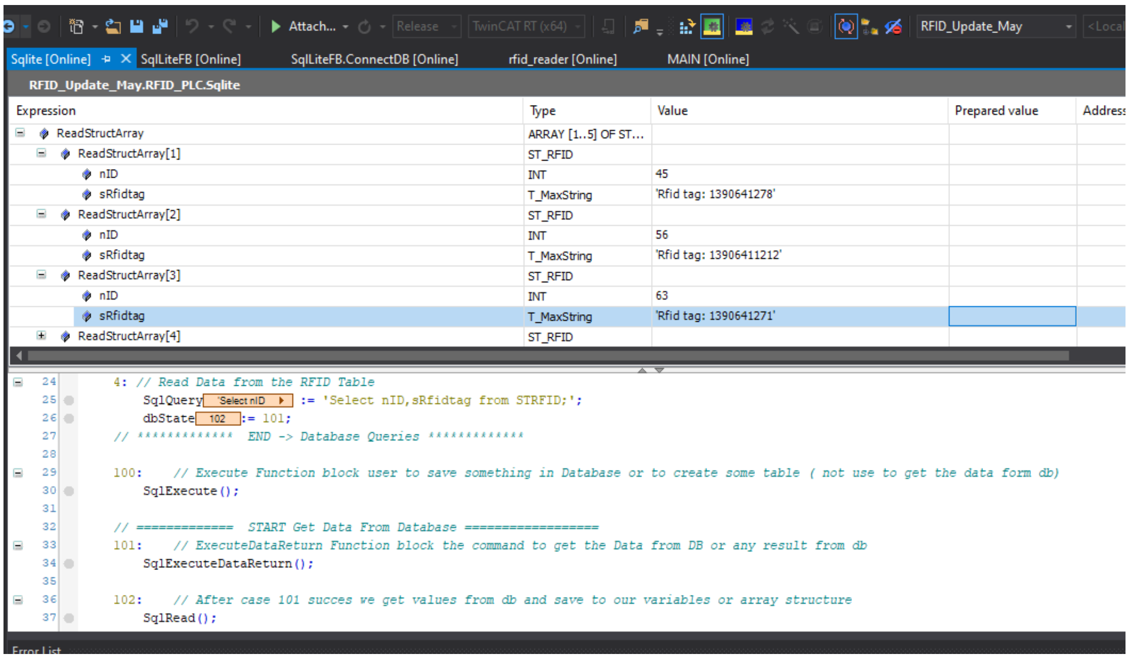
```
IF fbSQLCommand.Execute(ADR(SqlQuery), SIZEOF(SqlQuery)) THEN
        IF fbSQLCommand.bError THEN
                LogMessage := fbSQLCommand.ipTcResult;
                dbState := 255;
        ELSE
                LogMessage := fbSQLCommand.ipTcResult;
                IF DBMode = 2 THEN
                        HasTableCreated:=TRUE;
                END_IF
                IF DBMode = 3 THEN
                        dbState:=3;
                END_IF
END_IF
```

4.3.3 Read function

To read the data from the database, we need to create a SQL command which executes data return (ADR). It will just call the data from the database, and for further, it must write using array variable.

```
IF fbSQLCommand.ExecuteDataReturn(ADR(SqlQuery), SIZEOF(SqlQuery),
ADR(fbSqlResult)) THEN
        IF fbSQLCommand.bError THEN
                LogMessage := fbSQLCommand.ipTcResult;
                dbState := 255;
        ELSE
                LogMessage := fbSQLCommand.ipTcResult;
                dbState := 102;
        END_IF
END_IF
```

The above is the next program that is used to write the data on a structure array using a function ReadStructArray.



Expression	Type	Value	Prepared value	Address
ReadStructArray	ARRAY [1..5] OF ST...			
ReadStructArray[1]	ST_RFID			
nID	INT	45		
sRfidtag	T_MaxString	'Rfid tag: 1390641278'		
ReadStructArray[2]	ST_RFID			
nID	INT	56		
sRfidtag	T_MaxString	'Rfid tag: 13906411212'		
ReadStructArray[3]	ST_RFID			
nID	INT	63		
sRfidtag	T_MaxString	'Rfid tag: 1390641271'		
ReadStructArray[4]	ST_RFID			

```
    4: // Read Data from the RFID Table
        SqlQuery 'Select nID := 'Select nID,sRfidtag from STRFID;';
        dbState 102 := 101;
    // *************  END -> Database Queries *************

    100:    // Execute Function block user to save something in Database or to create some table ( not use to get the data form db)
        SqlExecute();

    // =============  START Get Data From Database ==================
    101:    // ExecuteDataReturn Function block the command to get the Data from DB or any result from db
        SqlExecuteDataReturn();

    102:    // After case 101 succes we get values from db and save to our variables or array structure
        SqlRead();
```

Figure 4-5: Reading Data from DB.

```
IF fbSqlResult.Read(0, 5, ADR(ReadStructArray), SIZEOF(ReadStructArray),
FALSE,FALSE) THEN

        IF fbSqlResult.bError THEN
                LogMessage := fbSQLCommand.ipTcResult;
                dbState := 255;
        END_IF
END_IF
```

iptcResult function is used to interface from the TwinCAT 3 EventLogger, which provides details on the return value.

4.3.4 SQLite (FB)

This function block is used only to Insert and Read the Rfid tag from the Database (use Function block to call the Insert and Read method from anywhere easily). Did not define the repeated methods like ConnectDB, DisconnectDb which having the same code in both SQLite. This function can be a call from anywhere to read and insert data.

Insert (method): First, define the variable SqlQuery: T_MaxString and, after that, use the fbSQL command to execute the SQL query to store data or create a table in the database.

Read (Method): The variable needs to define SqlQuery: T_MaxString and fbSQL command. ExecuteDataReturn query will be used to recall the data from the database. the data will be written in a structure array which is a read structure array

4.3.5 Global Variables

In GVL, all the variables for the serial communication must be defined for the connections.

RxBufferEL6002 is used to Receive data buffer and used with all receive function blocks.TxBufferEL6002 is used to Transmit data buffer and used with all receive function blocks. The input EL6inData22B and output EL6outData22B will be linked with the twincat system manager.

```
VAR_GLOBAL
        //Com Port Global Variables
//      RxBuffer                :ComBuffer;
//      TxBuffer                :ComBuffer;

//      ComInData AT %I* :PcComInData;
//      ComOutData AT %Q* :PcComOutData;
// EL6002 Global Variables which is available to all programs
        RxBufferEL6002 : ComBuffer;
   TxBufferEL6002 : ComBuffer;

                (* I/O variables for a EL6002 terminal*)
   In_EL6002 AT %I*     : EL6inData22B;
        In_EL6002_2 AT %I*     : EL6inData22B;
   Out_EL6002 AT %Q*     : EL6outData22B;
        Out_EL6002_2 AT %Q*     : EL6outData22B;
END_VAR
```

The variable declaration Rx buffer will receive the data, and the Tx buffer will transmit the data used with all function blocks.

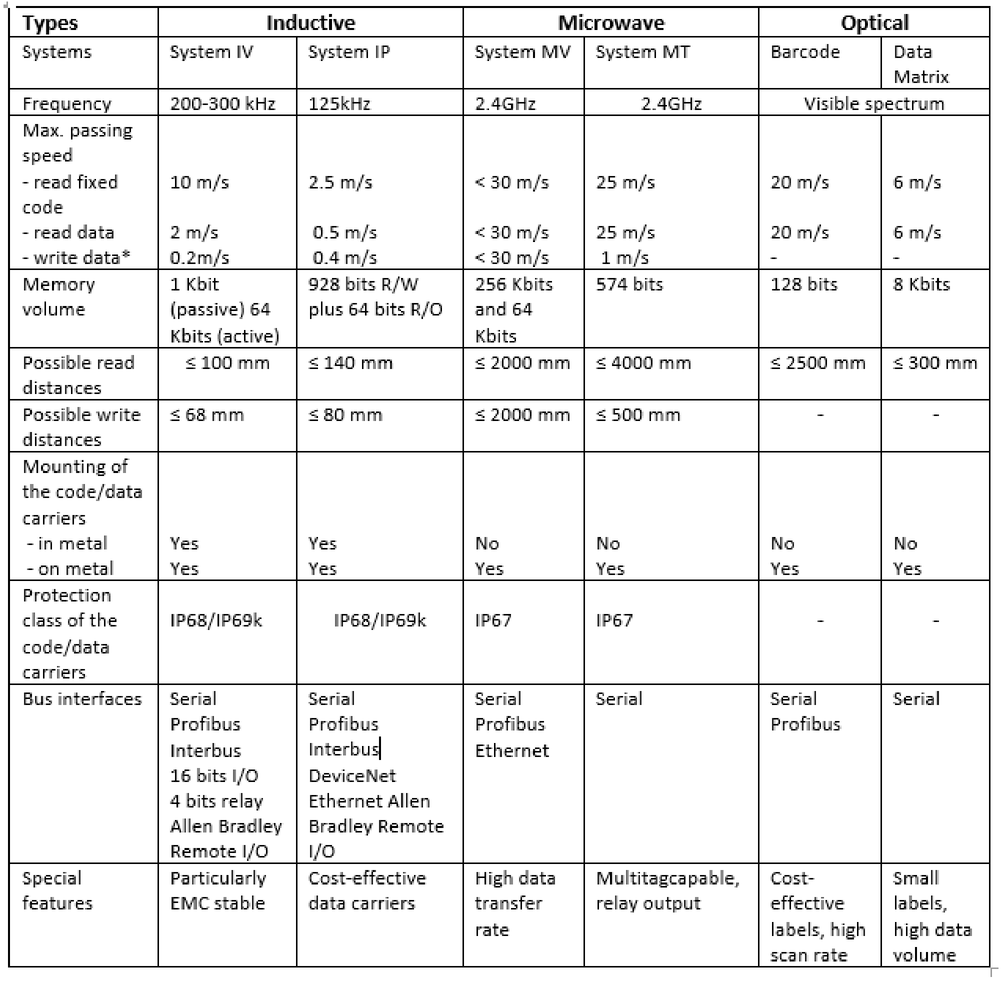
4.4 RFID communication

4.4.1 Inductive identification

The prototype for this project is utilized to employ inductive identification systems and the transformer concept applies to inductive identification systems in close-field, i.e. the read-write cone is locally constrained, and no wave from read/ write heads propagates. The average ranges vary from a few centimeters to almost a mile, depending on the height of the coil. The transmitter signals are often passive and reflect or modify. The energy is extracted from the energy transmitter for this function. These systems are especially useful for automation applications, production technology material control, data acquisition or identification of objects such as storage containers, pads, parts carrier.[38]

In the following table, all systems with their features are compared. The application-specific notes below offer further decision support.

Table 4-1 Comparison of identification systems.[38]



Types	Inductive		Microwave		Optical	
Systems	System IV	System IP	System MV	System MT	Barcode	Data Matrix
Frequency	200-300 kHz	125kHz	2.4GHz	2.4GHz	Visible spectrum	
Max. passing speed - read fixed code - read data - write data*	 10 m/s 2 m/s 0.2m/s	 2.5 m/s 0.5 m/s 0.4 m/s	 < 30 m/s < 30 m/s < 30 m/s	 25 m/s 25 m/s 1 m/s	 20 m/s 20 m/s -	 6 m/s 6 m/s -
Memory volume	1 Kbit (passive) 64 Kbits (active)	928 bits R/W plus 64 bits R/O	256 Kbits and 64 Kbits	574 bits	128 bits	8 Kbits
Possible read distances	≤ 100 mm	≤ 140 mm	≤ 2000 mm	≤ 4000 mm	≤ 2500 mm	≤ 300 mm
Possible write distances	≤ 68 mm	≤ 80 mm	≤ 2000 mm	≤ 500 mm	-	-
Mounting of the code/data carriers - in metal - on metal	 Yes Yes	 Yes Yes	 No Yes	 No Yes	 No Yes	 No Yes
Protection class of the code/data carriers	IP68/IP69k	IP68/IP69k	IP67	IP67	-	-
Bus interfaces	Serial Profibus Interbus 16 bits I/O 4 bits relay Allen Bradley Remote I/O	Serial Profibus Interbus DeviceNet Ethernet Allen Bradley Remote I/O	Serial Profibus Ethernet	Serial	Serial Profibus	Serial
Special features	Particularly EMC stable	Cost-effective data carriers	High data transfer rate	Multitagcapable, relay output	Cost-effective labels, high scan rate	Small labels, high data volume

Most important is to develop the connection and the communication between Embedded PC and software. The Embedded PC has an ethernet interface, and TCP/IP protocol was selected for communication with software PC and serial communication module interfacing with PLC and RFID module using the serial line control command.

Setup for RFID reader Some configurations must be made before the device initialization for seamless contact between the controller and RFID readers. For instance, this includes the serial contact baud rate. To pass these settings to RFID, a proprietary tool from the RFID reader manufacturer may be required.

For all supported RFID reader models, the following standard data transfer settings have been tried and tested:

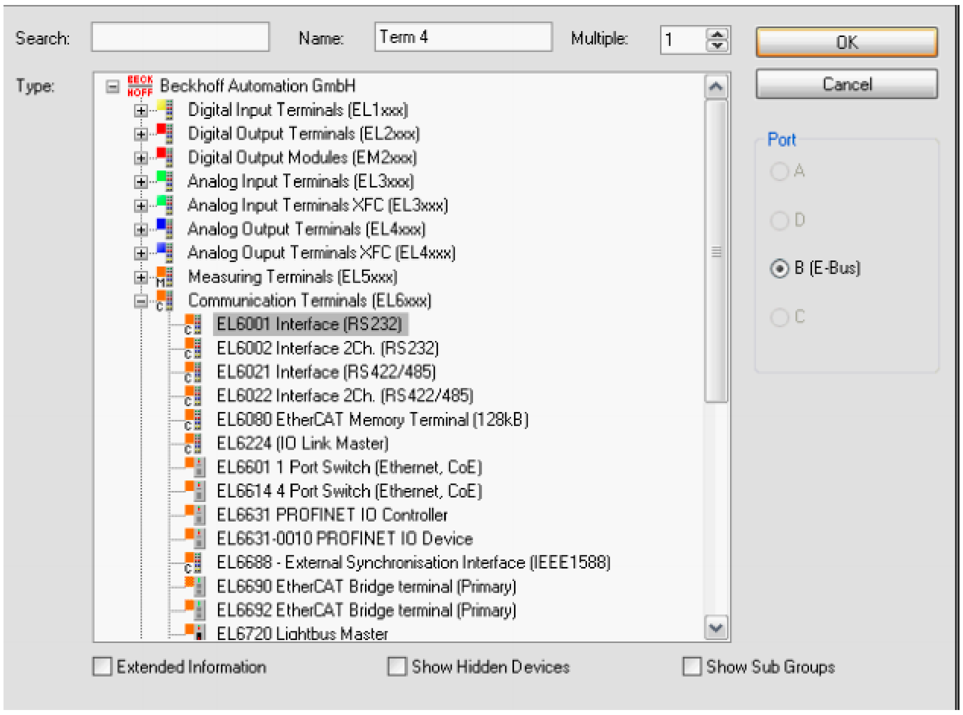
Table 4-2 Data transfer settings.

Setting	Value
Baud rate RS232	38400
Parity Bit	None
Data bits	8
Stop Bit	1

These commands are used to call the serial communication in the program's background using serial line control and EL6002 2 channel. In this project channel 1 is used as the control interface connection.

```
El6002Ctrl(
   Mode:= SERIALLINEMODE_EL6_22B,
   pComIn:= ADR(In_EL6002),
   pComOut:= ADR(Out_EL6002),
   SizeComIn:= SIZEOF(In_EL6002),
   Error=> bEL6002CtrlError,
   ErrorID=> eEL6002CtrlErrorID,
   TxBuffer:= TxBufferEL6002,
   RxBuffer:= RxBufferEL6002 );
```

The figure below shows how the interface to be add in the I/O devices and then it will be mapped with the pin of the channel inputs and outputs. PLC instances with background task must be link one by one for the I/O ports to be ready for the data transfer.


Search:

Name: Term 4

Multiple: 1

OK

Cancel

Type:
- Beckhoff Automation GmbH
  - Digital Input Terminals (EL1xxx)
  - Digital Output Terminals (EL2xxx)
  - Digital Output Modules (EM2xxx)
  - Analog Input Terminals (EL3xxx)
  - Analog Input Terminals XFC (EL3xxx)
  - Analog Output Terminals (EL4xxx)
  - Analog Ouput Terminals XFC (EL4xxx)
  - Measuring Terminals (EL5xxx)
  - Communication Terminals (EL6xxx)
    - EL6001 Interface (RS232)
    - EL6002 Interface 2Ch. (RS232)
    - EL6021 Interface (RS422/485)
    - EL6022 Interface 2Ch. (RS422/485)
    - EL6080 EtherCAT Memory Terminal (128kB)
    - EL6224 (IO Link Master)
    - EL6601 1 Port Switch (Ethernet, CoE)
    - EL6614 4 Port Switch (Ethernet, CoE)
    - EL6631 PROFINET IO Controller
    - EL6631-0010 PROFINET IO Device
    - EL6688 - External Synchronisation Interface (IEEE1588)
    - EL6690 EtherCAT Bridge terminal (Primary)
    - EL6692 EtherCAT Bridge terminal (Primary)
    - EL6720 Lightbus Master

Port: A, D, [✓ B (E-Bus)], C

☐ Extended Information ☐ Show Hidden Devices ☐ Show Sub Groups

Figure 4-6: Communication terminal.[28]

4.4.2 RFID reading

The machine is restarted after the new configuration is triggered. In most cases, the serial settings are already configured in the device setup in the CoE of the EtherCAT terminal. It is recommended that these values be added to the startup list to enable a hardware swap. At runtime, the serial settings can also be customized via CoE.

```
LineControl(    Mode      := SERIALLINEMODE_EL6_22B,
 pComIn    := ADR(EL6ComInData),
 pComOut   := ADR(EL6ComOutData),
  SizeComIn := SIZEOF(EL6ComInData),
    TxBuffer  := gEL6ComTxBuffer,
   RxBuffer  := gEL6ComRxBuffer );
```

For reading the transponder data, we need to create a connection through the head interface with control interface.T_maxString is using to read the tag and store it in the database file.

When we do everything related to the connected database with the RFID, we need to save the tag information to the Database by using SQLite.DB file, which will save the information for later.

```
IF NOT rfid_read.bBusy THEN
                bExe := FALSE;

                stCtrl.bBufferedCmd := FALSE;

                IF NOT rfid_read.bError THEN

                     nSubState := 0;

                     nState := nState + 1;

                     IF   rfid_read.eResponse  =  eRFR_Data_Inventory  THEN
     (* tag already in field: start action directly *)

                          stTag := rfid_read.stTranspInfo;

                tag := CONCAT(',$'', CONCAT(stTag.sSerialNumber, '$');'));

                          WriteQuery  :=  'INSERT  INTO  STRFID  (  nID,
sRfidtag) VALUES (';
                          WriteQuery := CONCAT ( CONCAT( 'INSERT INTO
STRFID ( nID, sRfidtag) VALUES (', INT_TO_STRING( CountN)), tag);
                          database.Insert(WriteQuery);
                          tonActionBlocker[0](PT:=T#3s);
                          IF tonActionBlocker[0].Q THEN
                               tonActionBlocker[0](IN:=FALSE);
                               nState := 4;  (* action at head1 *)
                          END_IF
                     END_IF
                END_IF
           END_IF
```

The above program is used to save the tag read by the control head and save it in the SQLite database, and the Concat function is used to combine two strings.

4.5 Working and testing of hardware

The objective was to run the programme in the plc by connecting it with the Twincat XAE for the connection we used TCP/IP communication and search the plc and after detecting plc we need to add the I/O devices for the mapping of the programming with linked background tasks.

The working and testing of the hardware is done in the laboratory for the results to be taken. In the figures below it is showing the reading of the card by the sensor and the range is 1 mm to 50 mm or 5 cm. Operating frequency is 125kHz and transfer rate 2 kBit/s. The reader is in operating mode when the light is green but when it start reading and writing then the LED turns to orange for indication of the card is being read by the head interface.

All three cards showing the figure have different UID's which will help to access the database to link the card with pallets information and it can be directed to the specific product information when it will be read by the sensor and then the operator can modify the data in the database for the card and pallet. All information will be stored in the database for the product linked with the UID code of the card. Tx and Rx are the data carriers which access through the mapping in the serial port configuration and helps to travel the data with I/O linked in the programming.

RF cards have a unique code and the cards used in this project they are non-changeable, but the data can be written the blocks but instead of writing the data in the card because it has limited memory. It is connected to the database and then the operator will manually insert the data for the warehouse inventory system.

Table 4-3 Cards UID code non-changeable.

Zones	ASCII	HEX	DEC
1	f q<STX>	36 A7 71 02	054 167 113 002
2	ºq<STX>	20 BA 71 02	032 186 113 002
3	6§q<STX>	66 A0 71 02	102 160 113 002

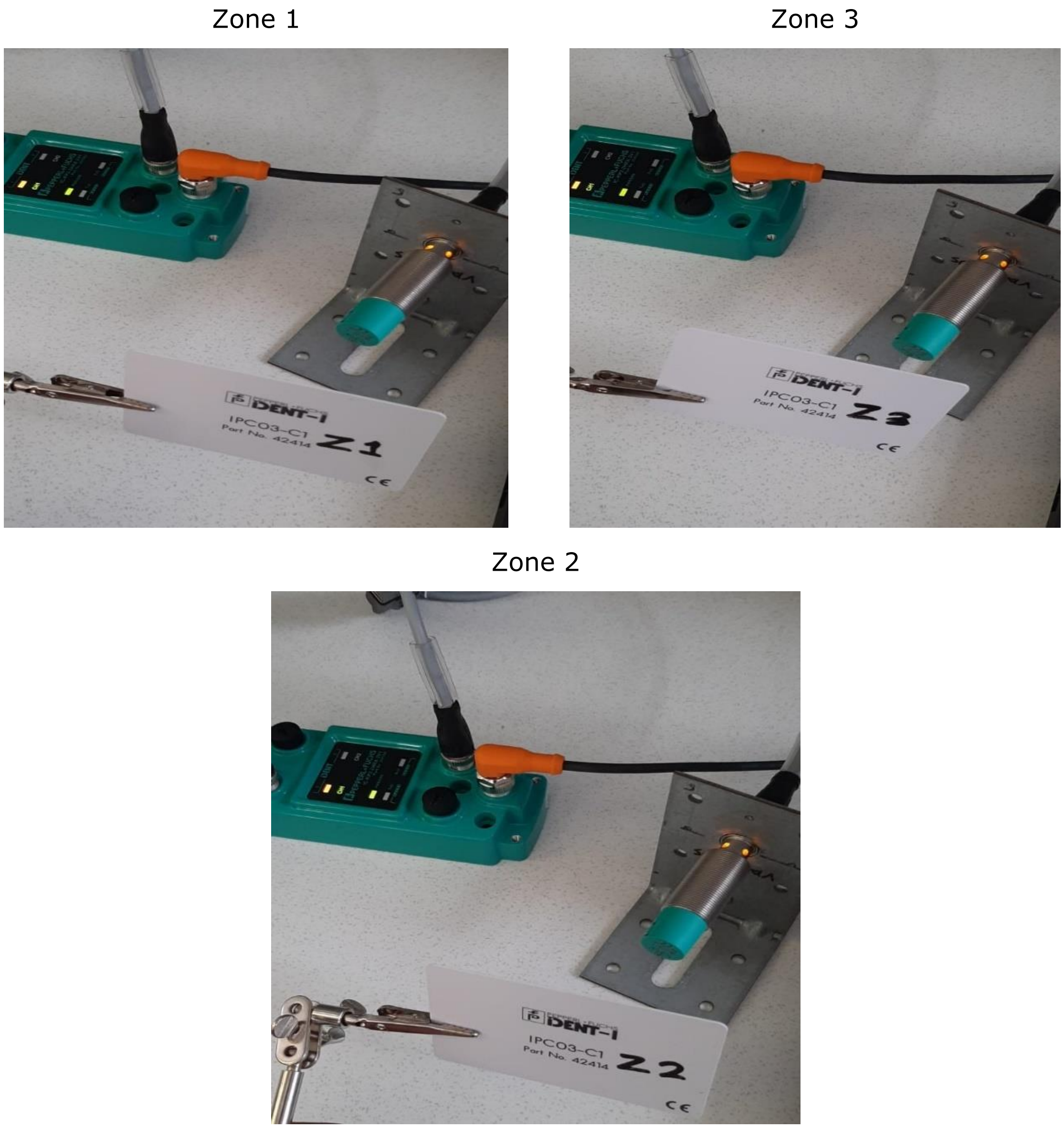
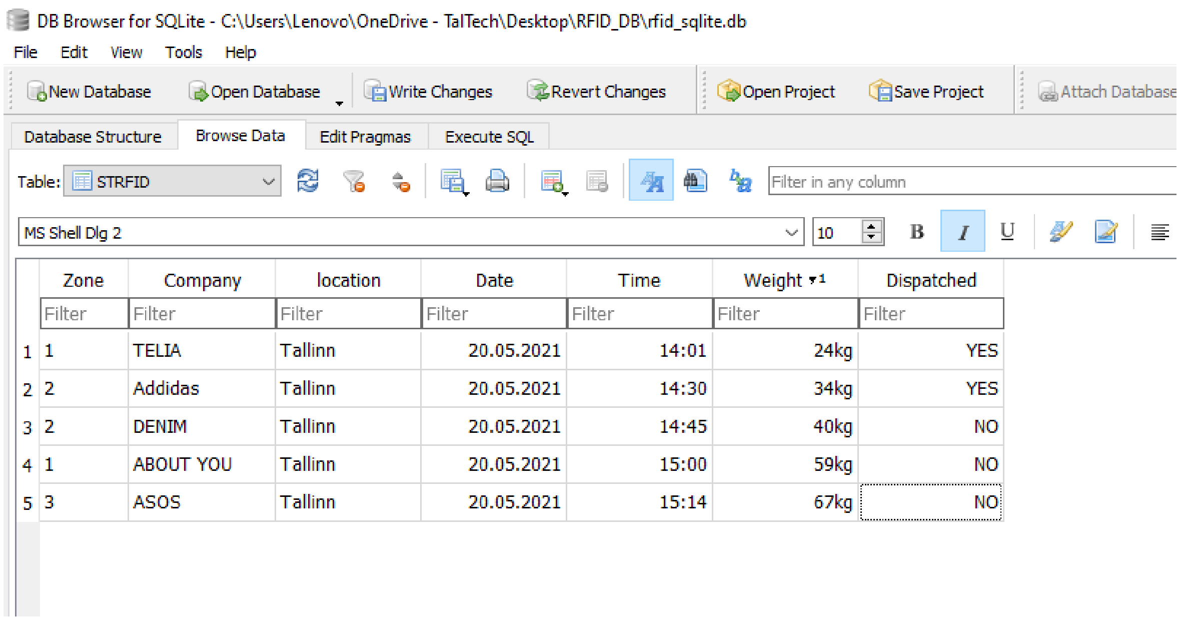


Figure 4-7: Hardware implementation.

4.5.1 Inventory Testing

The database is connected with plc, and it creates a .DB file in the desired folder by using SQLite database library and this database can be operate manually to enter the information of the products coming inside the warehouse. Read string unique code in the card for each product is defined when it was read on entrance of the pallet and further information is saved inside this database by the operator linked with RF card and added in the database table.

Same thing will happen when the product time to be dispatched from the warehouse then it will be updated link with that UID code and status will be updated that it will be dispatched from the warehouse.



	Zone	Company	location	Date	Time	Weight	Dispatched
1	1	TELIA	Tallinn	20.05.2021	14:01	24kg	YES
2	2	Addidas	Tallinn	20.05.2021	14:30	34kg	YES
3	2	DENIM	Tallinn	20.05.2021	14:45	40kg	NO
4	1	ABOUT YOU	Tallinn	20.05.2021	15:00	59kg	NO
5	3	ASOS	Tallinn	20.05.2021	15:14	67kg	NO

Figure 4-8: Database SQLite browser.

5. OVERVIEW AND DISCUSSION

This part of the thesis report addresses project shortcomings, the rationale for the project completed, and some potential proposals for potential future changes to make the final product more user-friendly and effective.

5.1 Explanation of the completed project

The Automation of the warehouse was a master’s thesis project done using the resources with the affordability and availability in the market to access the parts easily all over the world. The Beckhoff PLC are available worldwide quickly, and they have a wide range of Embedded PC, communication modules, and different packages for various type of projects required hardware modules. The embedded PC itself has its operating system, Windows 10, which is very useful in industrial situations. Twincat software is also user-friendly, and they have everything available on their manual to provide help to their customers. The operator can efficiently operate this warehouse database system, and as the system saved the pallets' location according to the zones for example Zone 1, Zone 2 and more around the warehouse, it makes it more efficient in consuming less time as you go and manually find the pallet's location, and then it will be dispatched from the storage. As the indication of the light will be used so the worker can easily see that where is the pallet located in the storage and it will take few minutes to reach there and dispatch the goods, and it is a user-friendly interface where the operator can see the weight and destinations of the storage pallets.

The accuracy of the hardware is discussed is within the acceptable range. However, further accuracy can be obtained by having proper lighting and placements of the sensor.

However, the measurement accuracy of the RFID read/write head depends on the mounting place or material like metal or wood and temperature conditions of the sensor, distance and motion speed of the tags (Transponder). Therefore, it is strictly advised to focus on these factors before the operation of the program.

5.2 Improvements in Future

The project's key objective is to improve the time and quality of the warehouse, so time is wasted by recalling the number of pallets and manually counting them. It will explain further modifications and enhancements in the following section that the author can propose to improve the structure.

- In the future, there will be an implementation of upgraded equipment such as warning lights and indication lights for notification for the workers to quickly reach the place for any problem or occupied space location in the warehouse. Tools such as RGB lights indicate red when it is occupied and green when it has free space for the new pallet. Maybe we can use blue light for the places which are reserved for some clients so no one can occupy those places, and that will make it more efficient because there will be reserved space for the prioritized pallets and cameras can be installed for getting the right image of the pallet for monitoring that it is placed correctly or not and maybe for any dangerous situation it will help.

- From the experiments and upgrades, we can modify the project by using RS485 instead of using RS232. The nominal wire operating distance for the RS232 transmitter from the recipient is 15 meters. However, a trade-off with a higher data rate will stretch the duration considerably.

- The reach from the RS485 is somewhat longer than 1200 meters. This is 24 times RS232's distance. Only if the equipment can run a distance from the transmitter would it be fair to choose RS485.

- The prototype solution can be implemented to real work in a warehouse environment, but it will need some modification according to the warehouse procedures and places to mount the devices. This can be done by upgrading the RFID modules for a better range and precisely working according to the company's workload.

- With future changes, we will upgrade the software and link all the things with the Twincat and Visual studio function versions. Also, if possible, it will be new packages for the new hardware as per the company's requirement or industry. It will be beneficial to consider the programming of the PLC to newer and faster communications platforms such as faster ethernet protocol. This will provide faster communication and allow less or maybe delay in the communication.

SUMMARY

Automation of warehouses A thesis master's subject or any human consumer to function in a cleverly technologically advanced way to minimize the likelihood of injuries and errors caused by human mistakes in a heavy and tiring environment.

To implement this new idea, the first step was to investigate and research the previously worked done technologies and find something that they lacked to improve to remove the errors and increase the warehouse's efficiency. It was concluded from the analysis that there are existing systems that do not have anything like to have such a warehouse management system using RFID and SQL to record the aimed at making it possible for operators' number of pallets in an automated way but found some manual machines to record the number of stored pallets in the warehouse which has zero efficiencies and much risk because of carrying it using a forklift to the height.

Beckhoff Embedded PC CX2042 and RFID pepperl+fuchs were used as hardware for the project using serial communication with the RS232 (EL6002) module. RFID further has three parts Control interface, head interface, and transponder (Tags). It was a combination of Twincat 3 and visual studio 2013 to do the programming using Structured Text (ST), and due to the using pepperl+fuchs RFID module, Twincat needs more extension libraries for the RFID and serial communication.

The program's simulation was tested, and it was found that the range of the transponder and receiver is very important. If the transponder is placed on some metal, it will decrease its range, and when it is placed on some wooden, it will give a reasonable range.

Serial interfaces allowed to make it very convenient for the communication of any hardware together, and SQLite is the database that is easier to use for the operators, and it identifies the place for the storage when to keep it inside the warehouse and when the dispatcher will need to take the pallet it will also locate the pallet which needs to be taken. The other libraries for the database and serial communication are also installed using the twincat 3 to use the features of the SQLite and other devices. Communication between the Embedded PC and RFID was done using Rs232 protocol via Ethercat.

Eventually, the prototype system helps the human operators quickly and effortlessly work in the warehouse and provide better work.

KOKKUVÕTE

Automaatlaod. Magistritöö teemaks on igasuguse inimtarbija käitumine nutikal tehnoloogilisel kõrgtasemel abivahendite kasutamisel, minimeerimaks inimlikest eksitustest tulenevate vigastuste ja vigade tõenäosust raskekaalulises ja väsitavas keskkonnas.

Selle uudse idee rakendamiseks oli esimene samm uurida teavet juba olemasolevate tehnoloogiate kohta ning leida, mis on neis puudu vigade tõhusamal eemaldamisel ja laojõudluse tõstmisel. Analüüsist järeldus, et on olemas süsteemid, mis ei hõlma RFID ja SQL abil automaatselt lao käitaja aluste arvu salvestavaid laohaldussüsteeme; on küll olemas teatud käsitsi juhitavad masinad, mis salvestavad lattu paigutatud aluste arvu – sellised masinad on absoluutselt mittetõhusad ning toovad kaasa ohte seoses kõrgustesse tõstmiseks kahveltõstuki kasutamisega.

Projekti riistvarana olid kasutusel Beckhoff Embedded PC CX2042 ja RFID Pepperl&Fuchs ning toimus jadaedastus RS232 (EL6002) moodulitega. RFID-il on kolm osa: kasutajaliides, pea ühenduskoht ja transponder (Tags). Struktureeritud teksil (ST) põhinevaks programmeerimiseks kasutati Twincat 3 ja Visual Studio 2013 kombinatsiooni; kuna kasutati Pepperl&Fuchs RFID moodulit, siis on Twincat'il RFID ja jadaedastuse tarbeks vaja enam laienduskogusid (extension libraries).

Programmi testiti simulatsioonis ning leiti, et transponderi ja vastuvõtja tegevusulatus on väga oluline. Kui transponder asetatakse metallile, siis selle ulatus väheneb ning puitpinnale asetades on ulatus vastuvõetav.

Jadaliidesed muutsid riisvara omavahelise edastuse väga mugavaks ning SQLite on käitajale lihtsalt kasutatav andmebaas, mis tuvastab ladustamiskoha lao sees ning leiab vajamineva aluse kui dispetšeril on seda vaja. Muud kogud andmebaasi ja jadaedastuse tarbeks paigaldatakse samuti Twincat 3 abil, et kasutada SQLite ja muude seadmete pakutavat. Manusarvuti ja RFID vaheline edastus toimus, kasutades Rs232 protokolli Ethercati abil.

Kokkuvõtteks, prototüüpsüsteem aitab inimkäitajail kiiresti ja tõhusalt laos töötada ning paremaid tulemusi saavutada.

LIST OF REFERENCES

[1] Y. T. Yang and J. C. Shieh, "Data warehouse applications in libraries - The development of library management reports," *Proc. - 2016 5th IIAI Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2016*, pp. 88–91, 2016, doi: 10.1109/IIAI-AAI.2016.129.

[2] H. Borstell, J. Kluth, M. Jaeschke, C. Plate, B. Gebert, and K. Richter, "Pallet monitoring system based on a heterogeneous sensor network for transparent warehouse processes," *2014 Work. Sens. Data Fusion Trends, Solut. Appl. SDF 2014*, 2014, doi: 10.1109/SDF.2014.6954718.

[3] D. W. Son, Y. S. Chang, N. U. Kim, and W. R. Kim, "Design of warehouse contol system for automated warehouse environment," *Proc. - 2016 5th IIAI Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2016*, pp. 980–984, 2016, doi: 10.1109/IIAI-AAI.2016.188.

[4] P. Liu, G. Q. Yao, H. Y. Cai, and Z. Yang, "Design and implementation of warehouse management system based on B/S mode," *2015 Int. Conf. Comput. Sci. Appl. CSA 2015*, pp. 146–150, 2017, doi: 10.1109/CSA.2015.33.

[5] N. Ling, X. Wei, M. M. Ren, and S. H. Fan, "The design and development of warehouse management information system on Hongxing logistics," *2015 Int. Conf. Comput. Sci. Appl. CSA 2015*, pp. 278–282, 2017, doi: 10.1109/CSA.2015.76.

[6] I. G. Fomina and V. V. Samoylov, "Applying of innovative methods in warehouse management," *Proc. 2017 IEEE Russ. Sect. Young Res. Electr. Electron. Eng. Conf. ElConRus 2017*, pp. 1337–1340, 2017, doi: 10.1109/EIConRus.2017.7910814.

[7] S. Savanur and K. S. Shreedhara, "Automated data validation for data warehouse testing," *2016 Int. Conf. Electr. Electron. Commun. Comput. Optim. Tech. ICEECCOT 2016*, pp. 223–226, 2017, doi: 10.1109/ICEECCOT.2016.7955219.

[8] Q. Yunrui, "Front-End and Back-End Separation for Warehouse Management System," *Proc. - 11th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2018*, pp. 204–208, 2018, doi: 10.1109/ICICTA.2018.00053.

[9] L. Yuan, "Research and Practice of RFID-Based Warehouse Logistics Management System," *Proc. - 2019 Int. Conf. Smart Grid Electr. Autom. ICSGEA 2019*, pp.

514–519, 2019, doi: 10.1109/ICSGEA.2019.00123.

[10] M. Dhouioui and T. Frikha, "Intelligent warehouse management system," *DTS 2020 - IEEE Int. Conf. Des. Test Integr. Micro Nano-Systems*, pp. 10–14, 2020, doi: 10.1109/DTS48731.2020.9196063.

[11] G. Blazic, P. Poscic, and D. Jaksic, "Data warehouse architecture classification," *2017 40th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2017 - Proc.*, pp. 1491–1495, 2017, doi: 10.23919/MIPRO.2017.7973657.

[12] S. Huang, O. P. Gan, S. Jose, and M. Li, "Localization for industrial warehouse storage rack using passive UHF RFID system," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, pp. 1–8, 2017, doi: 10.1109/ETFA.2017.8247643.

[13] G. Sinodakis, V. Kostopoulos, M. Drakaki, Y. L. Karnavas, and P. Tzionas, "An RFID-Enabled Automated Storage and Retrieval System via Microcontroller Stepper Motor Control," *2019 8th Int. Conf. Mod. Circuits Syst. Technol. MOCAST 2019*, pp. 2019–2022, 2019, doi: 10.1109/MOCAST.2019.8742069.

[14] Y. Nuñez-Castaneda, M. Moreno-Samanamud, M. Shinno-Huamani, F. Maradiegue-Tuesta, and J. Alvarez-Merino, "Improvement of warehouses of distribution companies through lean warehouse and an allocation algorithm," *Proc. - 2019 7th Int. Eng. Sci. Technol. Conf. IESTEC 2019*, pp. 473–478, 2019, doi: 10.1109/IESTEC46403.2019.00091.

[15] G. M. Cidal, Y. Ayhan Cimbek, G. Karahan, O. Emre Boler, O. Ozkardesler, and H. Uvet, "A study on the development of semi automated warehouse stock counting system," *Proc. - 2019 6th Int. Conf. Electr. Electron. Eng. ICEEE 2019*, pp. 323–326, 2019, doi: 10.1109/ICEEE2019.2019.00069.

[16] Q. Gao, "Study on Administrative Management and Decision Analysis Based on Data Warehouse," *Proc. - 8th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2015*, pp. 856–859, 2016, doi: 10.1109/ICICTA.2015.218.

[17] C. V. Gonzalez and G. Jung, "Database SQL injection security problem handling with examples," *Proc. - 6th Annu. Conf. Comput. Sci. Comput. Intell. CSCI 2019*, pp. 145–149, 2019, doi: 10.1109/CSCI49370.2019.00031.

[18] I. Kara and M. Aydos, "Detection and Analysis of Attacks Against Web Services by the SQL Injection Method [SQL Enjeksiyon Yöntemiyle Web Hizmetine Yönelik Saldiri Tespit ve Analizi]," *3rd Int. Symp. Multidiscip. Stud. Innov. Technol.*

ISMSIT 2019 - Proc., pp. 6–9, 2019.

[19] F. J. Valente and A. C. Neto, "Intelligent steel inventory tracking with IoT / RFID," *2017 IEEE Int. Conf. RFID Technol. Appl. RFID-TA 2017*, pp. 158–163, 2017, doi: 10.1109/RFID-TA.2017.8098639.

[20] M. Rashid, S. M. A. Ahad, S. Siddique, and T. Motahar, "Smart warehouse management system with RFID and cloud database," *2019 Jt. 8th Int. Conf. Informatics, Electron. Vision, ICIEV 2019 3rd Int. Conf. Imaging, Vis. Pattern Recognition, icIVPR 2019 with Int. Conf. Act. Behav. Comput. ABC 2019*, pp. 218–222, 2019, doi: 10.1109/ICIEV.2019.8858546.

[21] J. Blum, " USB Serial Communication ," *Explor. Arduino®*, pp. 141–169, 2019, doi: 10.1002/9781119405320.ch7.

[22] "Embedded PCs - Modular DIN rail industrial PCs | Beckhoff Worldwide." https://www.beckhoff.com/en-en/products/ipc/embedded-pcs/ (accessed May 02, 2021).

[23] E. Pc, "CX20x2," 2020.

[24] Https://infosys.beckhoff.com/english.php?content=../content/1033/cx2000_hw/2671512331.html&id=, "No Title." https://infosys.beckhoff.com/english.php?content=../content/1033/cx2000_hw/2671512331.html&id=.

[25] "No Title." https://infosys.beckhoff.com/english.php?content=../content/1033/cx9020_hw/2744422283.html&id=.

[26] "BECKHOFF CX5120 MANUAL Pdf Download | ManualsLib." https://www.manualslib.com/manual/1389950/Beckhoff-Cx5120.html (accessed May 12, 2021).

[27] "No Title." https://infosys.beckhoff.com/english.php?content=../content/1033/el600x_el602x/1718507531.html&id=.

[28] S. I. Terminals, "EL600x, EL602x," 2020.

[29] "RFID Control Interfaces | RFID Control Units | Browse Literature."

https://www.pepperl-fuchs.com/global/en/classid_2474.htm?view=productgroupliterature (accessed May 06, 2021).

[30] "IC-KP2-1HRX-2V1 IDENTControl Compact unit with serial interface FACTORY AUTOMATION MANUAL."

[31] "RFID Control Interfaces | RFID Control Units | See Overview." https://www.pepperl-fuchs.com/global/en/classid_2474.htm?view=productgroupoverview (accessed May 06, 2021).

[32] "IC-KP2-2HRX-2V1 IDENTControl Compact unit with serial interface FACTORY AUTOMATION MANUAL."

[33] "(No Title)." https://files.pepperl-fuchs.com/webcat/navi/productInfo/doct/tdoct0824f_eng.pdf?v=20200320004754 (accessed May 12, 2021).

[34] T. Data, "Read / write head IPH-18GM-V1," pp. 1–3.

[35] O. No, "TC3 Database Server," 2018.

[36] "databse." https://infosys.beckhoff.com/english.php?content=../content/1033/tf6420_tc3_database_server/1443762187.html&id=.

[37] O. No, "TC3 RFID Reader Communication," pp. 1–77, 2018.

[38] "Identification Systems," 2006.

APPENDICES

Coding of RFID Reader

Connectivity of the Database, and one is to use the Database to save RFID tag number using tc3_database Server in SQLite and read the tag from RFID using tc2_rfid library

Database Connectivity Project

Database Server: RFID DB_Server for the configuration of the database server
DB: It is the Database for the SqlLite, having Database ID (DBID) as 1. The connection string for SQLite is "Data Source=C:\Users\Lenovo\OneDrive - TalTech\Desktop\RFID_DB\rfid_sqlite.DB;" to make a DB in that folder, and it does not require any authentication. Use the helper function to create a database and Check the connectivity from the Database.

RFID PLC

Libraries use in Code (References)
Tc3_DatabaseServer (TF6420): Use to create a table, data insertion, and read from the Database.
Tc2_RFID (TF6600): use to read RFID tag from RFID head.
Tc2_Serial_Com (TF6340) : Use to communication with RFID.
Tc3_EventLogger: Use this library to report the errors when something went wrong.
Tc3_Standard, Tc3_System, Tc3_Module, Tc2_Utilites: Use this library for the basic functionality of Structured Text language and use String functions Concat, etc.
There are two PLC tasks in the PLC project named Background and PLCTask. The background is used for serial communication and connectivity from the RFID head and PLC Task to read the RFID tag and save it in the Database.

POU (Program Organization Unit)

Database Setup

Sqlite (PRG) (POU File Name): use to create Table, Insert Dummy data and Read the Data from the Database. It is a program, so I cannot pass Input parameters like the tag I read from the RFID head. For this issue, I used Function block.

Variables Define:

```
PROGRAM Sqlite
VAR
    fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
    fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);

    LogMessage    : I_TcMessage;
    SqlQuery : T_MAXSTRING;

    // SQL ResultEvent Function block -> Get the Data from the RFID Table
    fbSqlResult : FB_SQLResultEvt(sNetID:='', tTimeout    := T#5S);
    // it is a data structure to store the number of rows or data of RFID table which we get from database
    ReadStructArray : ARRAY[1..5] OF ST_RFID;

    dbState : INT := 0; // our option to do something like read from db , write on db etc
    HasTableCreated: BOOL :=FALSE;

    // User changeable values ( you can also change it on runtime)
    Database_ID : UINT := 1; // DB id for connectivity which is mention on DB Server
    DBMode: Enum_DBMode := Enum_DBMode.TableCreation; // it is our DB Modes cases

    Start: BOOL:= TRUE;

    //==================      DB-MODES Cases=====================
    // 0 -> Connect to the Sqlite db file
    // 1 -> Create a Sql Command reference which query run on sqlite

    // 2 -> Table Creatoion query
    // 3 -> Data insertion query
    // 4 -> Data Read query

    // 100 -> use to create table and insert data into the database
    // 101 -> use to get the data from the database
    // 102 -> its store the data which we get from DB to an array structure

    Count : INT :=1;
END_VAR
```

Code:

```
CASE DB state OF

    // ============  START -> Database Connectivity and Command reference Create ==================
    0:
            IF Start = TRUE THEN
                    ConnectDB();
            END_IF
    1:
            CreateSqlCommand();
    // *************  END -> Database Connectivity and Command reference Create *************

    // =============  START -> Database Queries =================
    2: // Create RFID Table Structure
            SqlQuery := 'CREATE TABLE STRFID( nID INTEGER, sRfidtag TEXT );';
            dbState:= 100;

    3: // Insert data into the RFID table
            SqlQuery:=CONCAT (CONCAT( 'INSERT INTO STRFID ( nID, sRfidtag) VALUES (', INT_TO_STRING( Count)), ',$'RFID tag no : 8912739812638640120$' )');
            Count := Count + 1;
            dbState:= 100;

    4: // Read Data from the RFID Table
            SqlQuery := 'Select nID,sRfidtag from STRFID;';
            dbState:= 101;
    // *************  END -> Database Queries *************

    100:  // Execute Function block user to save something in Database or to create some table ( not use to get the data from DB)
            SqlExecute();

    // =============  START Get Data from Database =================
    101:  // ExecuteDataReturn Function block the command to get the Data from DB or any result from db
            SqlExecuteDataReturn();

    102:  // After case 101 succes we get values from db and save to our variables or array structure
            SqlRead();
    // *************  END Get Data from Database *************

    500:
            DisconnectDB();

END_CASE
```

ConnectDB:

```
// first we connect to the SQLITE DB File and then we do something on that file
```

```
IF fbSqlDatabase.Connect(Database_ID) THEN
   IF fbSqlDatabase.bError THEN
           dbState := 255;
   ELSE
           dbState := dbState+1; // is everthing okay then jump to case 1 and create a command reference
   END_IF
END_IF
```

CreateSqlCommand:

// create a command reference

```
IF fbSqlDatabase.CreateCmd(ADR(fbSqlCommand)) THEN
   IF fbSqlDatabase.bError THEN
           dbState := 255;
   ELSE
           dbState := DBMode;
           // if dbState mode is on table creation and table is created then change db mode to insertion
           IF HasTableCreated AND dbState = 2 THEN
                   dbState:= 3; // then insert data if db
           END_IF

   END_IF
END_IF
```

DisconnectDB:

```
// disconnect from sqlite database when we done anything
IF fbSqlDatabase.Disconnect() THEN
   IF fbSqlDatabase.bError THEN
           dbState := 255;
   END_IF
END_IF
dbState:=0;
```

SqlExecute:

```
// it will execute the sql query to store data or create table in database
IF fbSQLCommand.Execute(ADR(SqlQuery), SIZEOF(SqlQuery)) THEN
   IF fbSQLCommand.bError THEN
           LogMessage := fbSQLCommand.ipTcResult;
           dbState := 255;
   ELSE
           LogMessage := fbSQLCommand.ipTcResult;
           IF DBMode = 2 THEN
```

```
o                       HasTableCreated:=TRUE;
o               END_IF
o               IF DBMode = 3 THEN
o                       dbState:=3;
o               END_IF
o
o       END_IF
o   END_IF
```

SqlExecuteDataReturn:

```
o   // iit will get the data from database
o   IF fbSQLCommand.ExecuteDataReturn(ADR(SqlQuery), SIZEOF(SqlQuery),
    ADR(fbSqlResult)) THEN
o       IF fbSQLCommand.bError THEN
o               LogMessage := fbSQLCommand.ipTcResult;
o               dbState := 255;
o       ELSE
o               LogMessage := fbSQLCommand.ipTcResult;
o               dbState := 102;
o       END_IF
o   END_IF
```

SqlRead:

```
o   // it will write the data on structure array which is ReadStructArray
o   IF fbSqlResult.Read(0, 5, ADR(ReadStructArray),
    SIZEOF(ReadStructArray), FALSE, FALSE) THEN
o       IF fbSqlResult.bError THEN
o               LogMessage := fbSQLCommand.ipTcResult;
o               dbState := 255;
o       END_IF
o   END_IF
```

SqliteFB (FB) (POU File Name): use only to Insert and Read Rfid tag from the Database (use Function block to call the Insert and Read method from anywhere easily).

Note: Did not define the repeated methods like ConnectDB, DisconnectDb which having the same code in both Sqlite.

Variable Define:

```
o   FUNCTION_BLOCK SqlLiteFB
o   VAR_INPUT
o   END_VAR
o   VAR_OUTPUT
```

```
o  END_VAR
o  VAR
o     fbSqlDatabase : FB_SQLDatabaseEvt(sNetID := '', tTimeout := T#5S);
o     fbSqlCommand : FB_SQLCommandEvt(sNetID := '', tTimeout := T#5S);
o
o     LogMessage    : I_TcMessage;
o
o     // SQL ResultEvent Function block -> Get the Data from the RFID Table
o     fbSqlResult : FB_SQLResultEvt(sNetID:='', tTimeout    := T#5S);
o     // it is a data structure to store the number of rows or data of RFID table
   which we get from database
o     ReadStructArray : ARRAY[1..5] OF ST_RFID;
o
o     // User changeable values ( you can also change it on runtime)
o     Database_ID : UINT := 1; // DB id for connectivity which is mention on
   DB Server
o
o     Okay : UINT := 0;
o  END_VAR
```

Insert Variable Define (method):

```
o  METHOD Insert : BOOL
o  VAR_INPUT
o     SqlQuery : T_MaxString;
o  END_VAR
```

Insert Code (Method):

```
o  // it will execute the sql query to store data or create table in database
o  IF fbSQLCommand.Execute(ADR(SqlQuery), SIZEOF(SqlQuery)) THEN
o     IF fbSQLCommand.bError THEN
o              LogMessage := fbSQLCommand.ipTcResult;
o              Okay := 0;
o     ELSE
o              Okay := 200;
o     END_IF
o  END_IF
```

Read Variable Define (Method):

```
o  METHOD Read : BOOL
o  VAR_INPUT
o     SqlQuery : T_MaxString;
o  END_VAR
```

Read Code (Method):

```
o  // iit will get the data from database
o  IF fbSQLCommand.ExecuteDataReturn(ADR(SqlQuery), SIZEOF(SqlQuery),
   ADR(fbSqlResult)) THEN
o     IF fbSQLCommand.bError THEN
o             LogMessage := fbSQLCommand.ipTcResult;
o             Okay := 0 ; // error
o     ELSE
o             LogMessage := fbSQLCommand.ipTcResult;
o
o             // it will write the data on structure array which is ReadStructArray
o             IF      fbSqlResult.Read(1,      5,      ADR(ReadStructArray),
   SIZEOF(ReadStructArray), FALSE, TRUE) THEN
o                     IF fbSqlResult.bError THEN
o                             LogMessage := fbSQLCommand.ipTcResult;
o                             Okay:=0; //Error
o                     END_IF
o             END_IF
o     END_IF
o  END_IF
```

RFID Setup

Background (PRG) (POU File Name): Use to communicate with RFID head

Variable Define (File Name):

```
o  PROGRAM Background
o  VAR
o     //Com port serail communatication
o  // SerialLine      :SerialLineControl;
o
o     (* background communication with the EL6002 terminal *)
o     El6002Ctrl       : SerialLineControl;
o     bEL6002CtrlError   : BOOL;
o     eEL6002CtrlErrorID : ComError_t;
o
o  END_VAR
```

Code:

```
o  (* serial background communication *)
o
o  //SerialLine(
o  // Mode            := SERIALLINEMODE_EL6_22B,
o   // pComIn                  := ADR(ComInData),
o   // pComOut                 := ADR(ComOutData),
o   // SizeComIn     := SIZEOF(ComInData),
o   // TxBuffer      := TxBuffer,
o   // RxBuffer      := RxBuffer
o   //);
o
o   (* background communication with the EL600x terminal *)
o   El6002Ctrl(
o       Mode:= SERIALLINEMODE_EL6_22B,
o       pComIn:= ADR(In_EL6002),
o       pComOut:= ADR(Out_EL6002),
o       SizeComIn:= SIZEOF(In_EL6002),
o       Error=> bEL6002CtrlError,
o       ErrorID=> eEL6002CtrlErrorID,
o       TxBuffer:= TxBufferEL6002,
o       RxBuffer:= RxBufferEL6002 );
```

ErrorList (FB) (POU File Name): Use to report errors at runtime

Variable Define:

```
o   FUNCTION_BLOCK ErrorList
o   VAR_INPUT
o       bErr            :BOOL;
o       nErrId          :UDINT;
o       bEdgeDetect  :BOOL;        (* FALSE: every input is counted; TRUE:
    only rising edge of input is counted *)
o       bReset          :BOOL;
o   END_VAR
o   VAR_OUTPUT
o       arrErr          :ARRAY[1..iHIST_LEN] OF ST_ErrHist;
o       iErrIdx         :UINT;
o       nErrSum                 :UINT;
o   END_VAR
o   VAR CONSTANT
o       iHIST_LEN       :UINT:=100;
o   END_VAR
o   VAR
o       bNewErr                 :BOOL;
o       rtErr           :R_TRIG;
o   END_VAR
```

Code:

```
o   IF bReset THEN
o       bReset := FALSE;
o       iErrIdx := 0;
o       nErrSum := 0;
o       MEMSET(ADR(arrErr),0,SIZEOF(arrErr));
o   END_IF
```

```
o
o   rtErr(CLK:=bErr);
o   IF bEdgeDetect THEN
o       bNewErr := rtErr.Q;
o   ELSE
o       bNewErr := bErr;
o   END_IF
o
o   IF bNewErr THEN
o       nErrSum := nErrSum + 1;
o       IF iErrIdx <= iHIST_LEN THEN
o               IF iErrIdx = 0 THEN
o                       MEMSET(ADR(arrErr),0,SIZEOF(arrErr));
o                       iErrIdx := iErrIdx + 1;
o               ELSIF arrErr[iErrIdx].nErrId <> nErrId THEN
o                       iErrIdx := iErrIdx + 1;
o               END_IF
o
o               arrErr[iErrIdx].nErrId          := nErrId;
o               arrErr[iErrIdx].nErrCount       := arrErr[iErrIdx].nErrCount + 1;
o       END_IF
o   END_IF
```

Rfid_reader (PRG) (POU File Name): use to read RFID tag using RFID library.

Variable Defines:

```
o   FUNCTION_BLOCK rfid_reader
o   VAR_INPUT
o       bReInit                 :BOOL;
o   END_VAR
o   VAR_OUTPUT
o       bReady                  :BOOL;
o       bError          :BOOL;
o   END_VAR
o   VAR CONSTANT
o       iUSEDDCTYPE :USINT          :=99;  (* ADAPT: data carrier type of used
    transponder *)
o   END_VAR
o   VAR
o       // Database readin/writng variable
o       database : SqlLiteFB;
o       ReadQuery : T_MaxString := 'Select nID, sRfidtag from STRFID;';
o       tag : T_MaxString;
```

```
   CountN: Uint:=1 ;

   rfid_read              :FB_RFIDReader :=(eManufacturer:=eRFRM_PepperlFuchs);
   bExe           :BOOL;
   eCmd           :E_RFID_Command;
   stAccData      :ST_RFID_AccessData;
   stCtrl         :ST_RFID_Control;
   stCfg          :ST_RFID_CfgStruct_PepperlFuchsIDENT;
   eErrorID       :E_RFID_ErrID;
   eErrCodeRcv    :E_RFID_ErrCodeRcv_PepperlFuchs;

   nState         :UINT;
   nSubState      :UINT;
   stTag          :ST_RFID_TranspInfo;        (* detected tag -> read/write data from/to tag memory *)
   nRdData                :UDINT;
   nWrData                :UDINT;
   tonActionBlocker:ARRAY[0..1] OF TON;(* to allow only one action for 3 seconds *)

   fbErrHist      :ErrorList;
   arrRcvData     :ARRAY[0..255] OF BYTE;
END_VAR
```

Code:

```
database.ConnectDB();

IF bReInit THEN
   bReInit := FALSE;
   fbErrHist.bReset := TRUE;
   nState := 0;
   nSubState := 0;
END_IF

CASE nState OF
0: (* initialization *)
   bReady := FALSE;
   CASE nSubState OF
   0:
           IF NOT rfid_read.bBusy THEN
                   eCmd := eRFC_GetReaderVersion;
                   bExe := TRUE;
                   nSubState := nSubState + 1;
```

```
o           END_IF
o     1:
o           IF NOT rfid_read.bBusy THEN
o                 bExe := FALSE;
o                 IF NOT rfid_read.bError THEN
o                       IF     rfid_read.bResponseRcv     =     TRUE     AND
   rfid_read.eResponse = eRFR_Data_ReaderVersion THEN
o                             IF       rfid_read.stReaderInfo.eType       =
   eRFRT_PepperlFuchsIC_KP2_2HRX_2V1 OR rfid_read.stReaderInfo.eType =
   eRFRT_PepperlFuchsIC_KP_R2_V1 THEN
o                                   nSubState := nSubState + 1;
o                             END_IF
o                       END_IF
o                 END_IF
o           END_IF
o     2:
o           IF NOT rfid_read.bBusy THEN
o                 eCmd := eRFC_GetConfig;
o                 bExe := TRUE;
o                 nSubState := nSubState + 1;
o           END_IF
o     3:
o           IF NOT rfid_read.bBusy THEN
o                 bExe := FALSE;
o                 IF NOT rfid_read.bError THEN
o                       IF     rfid_read.bResponseRcv     =     TRUE     AND
   rfid_read.eResponse = eRFR_Data_Config THEN
o                             IF             SIZEOF(stCfg)             =
   rfid_read.stReaderCfg.iCfgStructSize THEN
o                                   MEMCPY(ADR(stCfg),
   rfid_read.stReaderCfg.pCfgStruct, rfid_read.stReaderCfg.iCfgStructSize);
o                                   nSubState := nSubState + 1;
o                             END_IF
o                       END_IF
o                 END_IF
o           END_IF
o     4:
o           IF     stCfg.arrHeadCfg[0].iDCType     =     iUSEDDCTYPE     AND
   stCfg.arrHeadCfg[1].iDCType = iUSEDDCTYPE THEN
o                 IF             stCfg.arrHeadCfg[0].eStatus             =
   eRFERRPepperlFuchs_NoError     AND     stCfg.arrHeadCfg[1].eStatus     =
   eRFERRPepperlFuchs_NoError THEN
o                       nSubState := 0;       (* init finshed *)
o                       nState := nState + 1;
o                 ELSE
o                       ;(* error at r/w head *)
o                 END_IF
o           ELSE
o                 nSubState := nSubState + 1;
o           END_IF
o     5:
o           IF NOT rfid_read.bBusy THEN
o                 IF stCfg.arrHeadCfg[0].iDCType <> iUSEDDCTYPE THEN
```

```
o                               eCmd := eRFC_ChangeDCType;
o                               stCtrl.iDCType := iUSEDDCTYPE;
o                               stCtrl.iHeadNumber := 1;
o                               bExe := TRUE;
o                               nSubState := nSubState + 1;
o                       ELSIF  stCfg.arrHeadCfg[1].iDCType  <>  iUSEDDCTYPE
   THEN
o                               eCmd := eRFC_ChangeDCType;
o                               stCtrl.iDCType := iUSEDDCTYPE;
o                               stCtrl.iHeadNumber := 2;
o                               bExe := TRUE;
o                               nSubState := nSubState + 1;
o                       END_IF
o               END_IF
o      6:
o               IF NOT rfid_read.bBusy THEN
o                       bExe := FALSE;
o                       IF NOT rfid_read.bError THEN
o                               IF   rfid_read.bResponseRcv   =   TRUE   AND
   rfid_read.eResponse = eRFR_CmdConfirmation THEN
o                                       nSubState := 2;      (* GetConfig again *)
o                               END_IF
o                       END_IF
o               END_IF
o      END_CASE
o   1: (* set buffered command to detect tag and read its serial number at
   head1 *)
o      CASE nSubState OF
o      0:
o               IF NOT rfid_read.bBusy THEN
o                       eCmd := eRFC_GetInventory;
o                       stCtrl.bBufferedCmd := TRUE;
o                       stCtrl.iHeadNumber := 1;
o                       bExe := TRUE;
o                       nSubState := nSubState + 1;
o               END_IF
o      1:
o               IF NOT rfid_read.bBusy THEN
o                       bExe := FALSE;
o                       stCtrl.bBufferedCmd := FALSE;
o                       IF NOT rfid_read.bError THEN
o                               nSubState := 0;
o                               nState := nState + 1;
o                               IF  rfid_read.eResponse  =  eRFR_Data_Inventory
   THEN     (* tag already in field: start action directly *)
o                                       stTag := rfid_read.stTranspInfo;
o
o                                       // =========== Get tag and save it into
   the Database =================
o                                       tag             :=             CONCAT('$'',
   CONCAT(stTag.sSerialNumber, '$');'));
o
```

```
o                       database.Insert(CONCAT (CONCAT( 'INSERT
  INTO STRFID ( nID, sRfidtag) VALUES (', INT_TO_STRING( CountN)), tag));
o                       CountN := CountN+1;
o                       //
  ************************************************************
  ************
o
o
o                       tonActionBlocker[0](PT:=T#3s);
o                       IF tonActionBlocker[0].Q THEN
o                           tonActionBlocker[0](IN:=FALSE);
o                           nState := 4;  (* action at head1 *)
o                       END_IF
o                   END_IF
o               END_IF
o           END_IF
o   END_CASE
o 2: (* set buffered command to detect tag and read its serial number at
  head2 *)
o   CASE nSubState OF
o   0:
o           IF NOT rfid_read.bBusy THEN
o               eCmd := eRFC_GetInventory;
o               stCtrl.bBufferedCmd := TRUE;
o               stCtrl.iHeadNumber := 2;
o               bExe := TRUE;
o               nSubState := nSubState + 1;
o           END_IF
o   1:
o           IF NOT rfid_read.bBusy THEN
o               bExe := FALSE;
o               stCtrl.bBufferedCmd := FALSE;
o               IF NOT rfid_read.bError THEN
o                   nSubState := 0;
o                   nState := nState + 1;
o                   IF  rfid_read.eResponse  =  eRFR_Data_Inventory
  THEN      (* tag already in field: start action directly *)
o                       stTag := rfid_read.stTranspInfo;
o                       tonActionBlocker[1](PT:=T#3s);
o                       IF tonActionBlocker[1].Q THEN
o                           tonActionBlocker[1](IN:=FALSE);
o                           nState := 5;  (* action at head2 *)
o                       END_IF
o                   END_IF
o               END_IF
o           END_IF
o   END_CASE
o 3: (* react to tag detection *)
o   bReady := TRUE;
o   tonActionBlocker[0](PT:=T#3s, IN:=TRUE);
o   tonActionBlocker[1](PT:=T#3s, IN:=TRUE);
o   IF rfid_read.bResponseRcv THEN
o           IF NOT rfid_read.bError THEN
```

```
o                   IF rfid_read.eResponse = eRFR_Data_Inventory THEN
o                           stTag := rfid_read.stTranspInfo;
o                           IF stTag.iHeadNumber = 1 THEN
o                                   IF tonActionBlocker[0].Q THEN
o                                           tonActionBlocker[0](IN:=FALSE);
o                                           nState := 4;  (* action at head1 *)
o                                   END_IF
o                           ELSIF stTag.iHeadNumber = 2 THEN
o                                   IF tonActionBlocker[1].Q THEN
o                                           tonActionBlocker[1](IN:=FALSE);
o                                           nState := 5;  (* action at head2 *)
o                                   END_IF
o                           END_IF
o                   END_IF
o           END_IF
o   END_IF
o 4: (* action at head1 *)
o   CASE nSubState OF
o   0:
o           IF NOT rfid_read.bBusy THEN
o                   eCmd := eRFC_ReadBlock;
o                   stCtrl.iHeadNumber := 1;
o                   stAccData.iBlockCount       := 1;
o                   stAccData.iBlockSize := 4;
o                   stAccData.iStartBlock       := 0;
o                   stAccData.iDataSize         := SIZEOF(nRdData);
o                   stAccData.pData                     := ADR(nRdData);
o                   bExe := TRUE;
o                   nSubState := nSubState + 1;
o           END_IF
o   1:
o           IF NOT rfid_read.bBusy THEN
o                   bExe := FALSE;
o                   IF NOT rfid_read.bError THEN
o                           IF  rfid_read.bResponseRcv THEN
o                                   IF          rfid_read.eResponse          =
  eRFR_Data_ReadData THEN
o                                           nWrData := nRdData + 1;
    (* increase the read data by one and write it back to tag *)
o                                           nSubState := nSubState + 1;
o                                   ELSIF        rfid_read.eResponse        =
  eRFR_NoTransponder THEN
o                                           nSubState   :=   nSubState   -   1;
    (* try again *)
o                                   END_IF
o                           END_IF
o                   END_IF
o           END_IF
o   2:
o           IF NOT rfid_read.bBusy THEN
o                   eCmd := eRFC_WriteBlock;
o                   stCtrl.iHeadNumber := 1;
o                   stAccData.iBlockCount       := 1;
```

```
o                       stAccData.iBlockSize := 4;
o                       stAccData.iStartBlock        := 0;
o                       stAccData.iDataSize          := SIZEOF(nWrData);
o                       stAccData.pData                        := ADR(nWrData);
o                       bExe := TRUE;
o                       nSubState := nSubState + 1;
o               END_IF
o       3:
o               IF NOT rfid_read.bBusy THEN
o                       bExe := FALSE;
o                       IF NOT rfid_read.bError THEN
o                               IF  rfid_read.bResponseRcv THEN
o                                       IF          rfid_read.eResponse           =
    eRFR_WriteCmdSucceded THEN
o                                               nSubState := 0;
o                                               nState := 1;  (*  set  buffered  cmd
    again *)
o                                       ELSIF        rfid_read.eResponse         =
    eRFR_NoTransponder THEN
o                                               nSubState   :=   nSubState   -   1;
       (* try again *)
o                                       END_IF
o                               END_IF
o                       END_IF
o               END_IF
o       END_CASE
o   5: (* action at head2 *)
o       CASE nSubState OF
o       0:
o               IF NOT rfid_read.bBusy THEN
o                       eCmd := eRFC_ReadBlock;
o                       stCtrl.iHeadNumber := 2;
o                       stAccData.iBlockCount       := 1;
o                       stAccData.iBlockSize := 4;
o                       stAccData.iStartBlock        := 0;
o                       stAccData.iDataSize          := SIZEOF(nRdData);
o                       stAccData.pData                        := ADR(nRdData);
o                       bExe := TRUE;
o                       nSubState := nSubState + 1;
o               END_IF
o       1:
o               IF NOT rfid_read.bBusy THEN
o                       bExe := FALSE;
o                       IF NOT rfid_read.bError THEN
o                               IF  rfid_read.bResponseRcv THEN
o                                       IF          rfid_read.eResponse           =
    eRFR_Data_ReadData THEN
o                                               nWrData := nRdData - 1;
       (* decrease the read data by one and write it back to tag *)
o                                               nSubState := nSubState + 1;
o                                       ELSIF        rfid_read.eResponse         =
    eRFR_NoTransponder THEN
```

```
○                                         nSubState   :=   nSubState   -   1;
     (* try again *)
○                                  END_IF
○                         END_IF
○                 END_IF
○         END_IF
○    2:
○         IF NOT rfid_read.bBusy THEN
○                 eCmd := eRFC_WriteBlock;
○                 stCtrl.iHeadNumber := 2;
○                 stAccData.iBlockCount       := 1;
○                 stAccData.iBlockSize := 4;
○                 stAccData.iStartBlock       := 0;
○                 stAccData.iDataSize         := SIZEOF(nWrData);
○                 stAccData.pData                     := ADR(nWrData);
○                 bExe := TRUE;
○                 nSubState := nSubState + 1;
○         END_IF
○    3:
○         IF NOT rfid_read.bBusy THEN
○                 bExe := FALSE;
○                 IF NOT rfid_read.bError THEN
○                         IF  rfid_read.bResponseRcv THEN
○                                  IF          rfid_read.eResponse          =
    eRFR_WriteCmdSucceded THEN
○                                          nSubState := 0;
○                                          nState := 1;  (*  set  buffered  cmd
    again *)
○                                  ELSIF         rfid_read.eResponse         =
    eRFR_NoTransponder THEN
○                                          nSubState   :=   nSubState   -   1;
     (* try again *)
○                                  END_IF
○                         END_IF
○                 END_IF
○         END_IF
○    END_CASE
○
○ END_CASE
○
○ //Constructor to read rfid
○ rfid_read(
○    bExecute       := bExe,
○    eCommand       := eCmd,
○    stAccessData:= stAccData,
○    stCtrl         := stCtrl,
○    RxBuffer       := RxBufferEL6002,
○    TxBuffer       := TxBufferEL6002,
○    bBusy          => ,
○    bResponseRcv=> ,
○    eResponse      => ,
○    bError         => ,
○    iErrorID       => ,
o      iErrCodeRcv   => ,
o      stReaderCfg   => ,
o      stReaderInfo=> ,
o      stTranspInfo=> ,
o      stRawData     =>
o   );
o
o   eErrorID := rfid_read.iErrorID;
o   eErrCodeRcv := rfid_read.iErrCodeRcv;
o   IF rfid_read.bError THEN                    (* get raw data of response in case of
    error *)
o      MEMSET(ADR(arrRcvData), 0, SIZEOF(arrRcvData));
o      MEMCPY(ADR(arrRcvData),              rfid_read.stRawData.pReceivedRsp,
    MIN(SIZEOF(arrRcvData),rfid_read.stRawData.iReceivedRspLen));
o   END_IF
o   fbErrHist(bErr:=rfid_read.bError,                    nErrId:=rfid_read.iErrorID,
    bEdgeDetect:=TRUE);
o   bError := (fbErrHist.nErrSum > 0);
```

GVL (Global Variable List) It is Used to define Variable which is used globally in Program

Global_Variables (GVL FILE Name): variables to store the read values from the RFID head

- VAR_GLOBAL
- //Com Port Global Variables
- // RxBuffer   :ComBuffer;
- // TxBuffer   :ComBuffer;
- // ComInData AT %I* :PcComInData;
- // ComOutData AT %Q* :PcComOutData;
- // EL6002 Global Variables which is available to all programs
- RxBufferEL6002 : ComBuffer; (* Receive data buffer; used with all receive function blocks *)
- TxBufferEL6002 : ComBuffer; (* Transmit data buffer; used with all receive function blocks *)
-
- (* I/O variables for an EL6002 terminal*)
- In_EL6002 AT %I*   : EL6inData22B; (* linked to the EL6002 in the TwinCAT System Manager *)
- In_EL6002_2 AT %I*   : EL6inData22B; (* linked to the EL6002 in the TwinCAT System Manager *)
- Out_EL6002 AT %Q*   : EL6outData22B;(* linked to the EL6002 in the TwinCAT System Manager *)
- Out_EL6002_2 AT %Q*   : EL6outData22B;(* linked to the EL6002 in the TwinCAT System Manager *)
- END_VAR

DUT (Data Unit Type) It is used to make a Data Structure

Enum_DBMode   (DUT FILE Name): it is an Enum to set some number to the variable to specify some specific work used in DB like TableCreation = 1.

- TYPE Enum_DBMode :
- (
-    TableCreation := 2,
-    DataInsertion := 3,
-    ReadData := 4
-);
- END_TYPE

ST_ErrHist (DUT FILE Name): variables to store the ErrorId and ErrorCounts

- TYPE ST_ErrHist :
- STRUCT
-    nErrId   :UDINT;
-    NErrCount   :UDINT;
- END_STRUCT

- END_TYPE

ST_RFID (DUT FILE Name): variables to store the ID and RFID Tag

```
o   TYPE ST_RFID :
o   STRUCT
o       nID : INT;
o       sRfidtag : T_MaxString;
o   END_STRUCT
o   END_TYPE
```