OULUN AMMATTIKORKEAKOULU

Henri Hänninen

IMPLEMENTING ADS INTERFACE FOR INTEGRATION OF PLC AND MACHINE VISION APPLICATION

IMPLEMENTING ADS INTERFACE FOR INTEGRATION OF PLC AND MACHINE VISION APPLICATION

Henri Hänninen
Opinnäytetyö
Kevät 2023
Sähkö- ja automaatiotekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Sähkö- ja automaatiotekniikka, Automaatiotekniikka

Tekijä: Henri Hänninen
Opinnäytetyön nimi: Implementing ADS Interface for Integration of PLC and Machine Vision Application
Työn ohjaaja: Timo Heikkinen
Työn valmistumislukukausi ja -vuosi: Kevät 2023
Sivumäärä: 59 + 4 liitettä

Opinnäytetyön tarkoituksena oli toteuttaa kommunikaatiorajapinta Beckhoffin PLC:n (Programmable Logic Controller) ja konenäkösovelluksen välille käyttäen ADS-protokollaa. ADS (Automation Device Specification) on Beckhoff Automationin kehittämä tiedonsiirtoprotokolla, jota Beckhoffin ohjelmointiympäristö TwinCAT käyttää kommunikointiin eri ohjelmamoduulien välillä. Sitä voidaan käyttää myös ulkoiseen kommunikointiin automaatiolaitteiden kanssa, jotka tukevat protokollaa. ADS-rajapinta toteutettiin LabVIEW:n graafisella ohjelmointikielellä hyödyntäen Beckhoffin TF3710 TwinCAT 3 interface for LabVIEW -kirjastoa, joka tarjoaa mahdollisuuden kyseisen rajapinnan luomiseen.

Tiedonsiirtorajapinnan testaamista varten kehitettiin konenäkösovellus. Konenäkösovelluksen tarkoitus oli tunnistaa ja tarkastella kuusikulmiomuttereita. Konenäkösovellukselle rakennettiin testausympäristö, joka sisälsi kameralaitteiston. Opinnäytetyö sisältää kameralaitteiston konfiguroimisen ja konenäkösovelluksen ohjelmoinnin. Konenäkösovellus ohjelmoitiin käyttäen National Instrumentsin Vision Builder AI -ohjelmistoa.

Opinnäytetyössä ADS-rajapintaa vertaillaan TCP/IP client-rajapintaan. TCP/IP client on TCP/IP-protokollaan perustuva tiedonsiirtorajapinta, jota käytetään laitteiden väliseen kommunikointiin erilaisissa automaatiosovelluksissa ja sen on kehittänyt JOT Automation. Opinnäytetyössä tutkitaan rajapintojen suorituskykyä kommunikaationopeuden, vakauden sekä käytettävyyden osalta. Vertailu toteutettiin käyttäen PLC-ohjelmaa, joka laskee kommunikaatioajan viestin lähettämisen ja vastaanoton välillä. PLC-toiminnallisuus ohjelmoitiin TwinCAT 3 -ympäristössä.

Tuloksena opinnäytetyössä saatiin toimiva kommunikaatiorajapinta Beckhoffin PLC:n ja konenäkösovelluksen välille. ADS-rajapinta osoittautui suorituskyvyltään paremmaksi TCP/IP client-rajapintaan verrattuna kommunikaationopeuden ja vakauden osalta. Lisäksi ADS-rajapinta on monipuolinen sekä yksinkertainen käyttää ja muokata. Sitä voidaan käyttää sellaisenaan muissa sovelluksissa, missä Beckhoffin PLC kommunikoi konenäkösovelluksen kanssa.

Opinnäytetyö tehtiin JOT Automation Oy:lle. Työ tehtiin talven 2022 ja kevään 2023 aikana ja se tehtiin pääosin toimeksiantajan tiloissa Oulussa.

ADS-protokolla, Konenäkö, Kommunikaatiorajapinta, PLC ohjelmointi

ABSTRACT

Oulu University of Applied Sciences
Electrical and Automation Engineering, Automation Engineering

---

Author: Henri Hänninen
Title of thesis: Implementing ADS Interface for Integration of PLC and Machine Vision Application
Supervisor: Timo Heikkinen
Term and year when the thesis was submitted: Spring 2023
Number of pages: 59 + 4 appendices

The main purpose of this thesis was to implement a communication interface between Beckhoff PLC (Programmable Logic Controller) and a machine vision application using ADS protocol. ADS (Automation Device Specification) is a communication protocol developed by Beckhoff Automation, and it is used within Beckhoff's TwinCAT automation software to communicate between different software components. ADS can also be used for external communication with automation devices that support the protocol. The ADS communication interface was implemented using LabVIEW graphical programming language and utilizing TF3710 TwinCAT 3 interface for LabVIEW library by Beckhoff, which provides tools for such an interface.

A machine vision application was developed to test the interface. The purpose of the machine vision application was to detect and examine hexagon nuts. A test station with a camera setup was also constructed for the machine vision application. The thesis consists of the hardware configuration of the camera setup as well as the programming of machine vision software. Machine vision was programmed using National Instrument's Vision Builder AI.

The thesis compares the ADS interface to a TCP/IP client. The TCP/IP client is an external software used for communication in various automation systems, including machine vision applications. It is based on the TCP/IP protocol and developed by JOT Automation. Both interfaces were examined in terms of speed and stability of the connection as well as usability of the interface. The comparison was established using a PLC application measuring the time between sending a command and receiving a response from the machine vision application. The PLC functionality was programmed in the TwinCAT 3 environment.

The result of the thesis was a functional communication interface between Beckhoff PLC and a machine vision application. The ADS interface proved to be more efficient in terms of speed and stability. Additionally, it is versatile and simple to use and modify. It can be utilized in similar applications where Beckhoff PLC communicates with a machine vision application.

The thesis was done for JOT Automation Ltd between winter 2022 and spring 2023. The work for the thesis was done mainly in their premises in Oulu, Finland.

Keywords:
ADS protocol, Machine vision, Communication interface, PLC programming

CONTENTS

1 INTRODUCTION

The thesis is comprehensive and consists of several segments. First, it presents components of a machine vision system in terms of hardware and software as well as theoretical background for TCP/IP protocol and ADS protocol. In the thesis, the ADS interface and the TCP/IP client are implemented for communication between a PLC and a machine vision application. A machine vision application is developed to test the interfaces, and a test station with a camera setup is constructed. The camera setup includes a rack, camera, lens, illumination, and PLC. The thesis describes the test station's hardware configuration, PLC programming, establishing of communication interface, and vision software programming. The performance of both interfaces is examined in terms of speed and stability of connection and usability. Lastly, the thesis presents results and discusses possible implementations of the interface.

1.1 Background

The initial objective was to implement a communication interface between PLC and machine vision application using an ADS interface established with LabVIEW. Machine vision software used in the company provides an option for running a LabVIEW VI (Virtual Instrument), so this could be possible. The performance of the interface was to be tested as well. Implementing the ADS interface between PLC and machine vision application could make communication faster and more stable since no additional applications are involved. Currently, communication in similar applications is established using external TCP/IP client software. A further objective was to compare the performance of the ADS interface and the TCP/IP client. Usability was also to be addressed.

1.2 JOT Automation

The thesis was commissioned by JOT Automation Ltd. JOT Automation has over 30 years of experience in the electronics industry. It provides solutions for assembly, testing, and process automation. The main products are standalone solutions for product handling and testing, but JOT Automation also provides custom automation solutions for customers. Originally JOT Automation de-

signed solutions mainly for the telecom industry, but since then has also expanded to other industries, including consumer and power electronics, renewable energy, automotive, and battery industries. (1.)

Due to growing demand for increased production capacity and quality standards in the industry, JOT Automation has integrated machine vision into its solutions. Machine vision is mainly used in JOT automation in product handling, quality control, and measurement applications.

In product handling applications, a machine vision system provides positioning data to an actuator, typically a robotic arm, which then moves the product according to the provided coordinates. For quality control applications, a machine vision system provides pass/fail data and other valuable information about the product, which is then managed according to the results obtained from the machine vision system. Other uses for machine vision in JOT Automation include measurement applications and visual reading of 2D codes.

FIGURE 1 presents JOT odd-shape 620 assembly cell, which can be equipped with a machine vision system to facilitate board and component inspection.

FIGURE 1. JOT odd-shape 620 assembly cell. (2.)

2 COMPONENTS OF THE MACHINE VISION SYSTEM

Machine vision refers to real-time imaging-based automatic inspection technology, and it is one of the key elements of Industry 4.0. It has limitless applications in manufacturing environments. The basic principle of machine vision systems is that a machine determines action based on the input of a camera. The difference between computer vision and machine vision is that computer vision acquires information from an image or a video. In contrast, a machine vision system uses a live image from the system's camera and makes rapid decisions based on the results. Machine vision systems are often used in industrial contexts, whereas computer vision can be used as a standalone system. (3.)

The utilization of machine vision systems in industrial applications began in the 1980s, but at first, they were expensive and required a lot of computer programming. Due to the development of camera and computer components and technologies, machine vision systems have become more common in manufacturing environments. Since then, they have drastically improved the quality, efficiency, and operations of industrial applications. Some common applications for machine vision systems include correcting production line defects, inventory control and management, product tracking, and traceability and measurement applications. (4.)

A machine vision system eliminates human errors as it is not dependent on the performance of the human. It does not get fatigued and can operate 24/7, thus, lowering production costs and saving time. It is more accurate and faster than manual labor performed by humans, improving the quality of the process. A machine vision system can also work in demanding industrial circumstances. Additionally, it can detect ultraviolet light because it operates in a larger wavelength spectrum than the human eye. (5.)

2.1 Machine Vision Hardware

A machine vision system, illustrated in FIGURE 2, consists of several components referred to as machine vision hardware. These physical components work together to acquire and process visual information. Machine vision hardware plays a critical role in the performance of the automation system. The choice of hardware components depends on the specific application requirements and

performance parameters. The key components of the machine vision system are described in this chapter.

FIGURE 2. Machine vision system. (6.)

The machine vision system consists of:

- Illumination
- Imaging lens
- Sensor
- Vision processing system
- Communications system

2.1.1 Illumination

Illumination is one of the most important components of any optical solution. That's why choosing the correct lighting system for machine vision is crucial. Illumination needs to be effective and timely. A comprehensive analysis of the inspection environment and how a sample interacts with light is required to achieve this. Factors like geometry, reflectivity, and color determine how much light is reflected to the camera. The lighting system should be selected according to the features to

be measured or observed, maximizing contrast for wanted features and minimizing unwanted features. This is achieved by adjusting the intensity of light and choosing the correct placement and type of lighting. (7.)

Directional lighting is commonly used for machine vision because it is inexpensive and easy to install. Even though it produces bright illumination and sharp shadows, it is sufficient for most machine vision cases. For more complex and highly reflective objects, diffused lighting is often used. A diffused lighting system illuminates an object from multiple directions reducing shadows. A dome lighting setup is one of the most common diffused lighting systems. (7.)

A backlighting setup is often used when measuring dimensions or the object's general shape. The backlighting setup illuminates the object from behind, outlining the edges of the object. The disadvantage of the backlighting setup is that the object's surface is not clearly visible. Additionally, for moving objects, a strobe light setup is often used. A correctly timed powerful flash of light and short exposure time enables examining moving objects and prevent an image from getting blurred. A strobe light setup is especially useful for objects moving at high velocity. (7.)

2.1.2 Imaging Lens

The purpose of the imaging lens is to collect and focus light for the image sensor. The imaging lens consists of multiple optical lenses with the correct shape and spacing between them. They have different abilities to refract light and different angles of view. The imaging lens forms an area of inspection called the field of view, which is then captured on the camera's imager. It also defines the depth of field, which refers to the distance between the nearest and the furthest objects that appear sharp in an image. Defining features for imaging lenses are aperture, focal length, and f-number. Focal length is an optical property determining the lens's ability to converge or diverge light. Aperture refers to the diameter of the opening through which light travels. The ratio between focal length and aperture is called the f-number. (8.)

Some form of distortion usually occurs in optical systems. In barrel distortion the center of the image appears to be expanded, whereas, in pincushion distortion, it appears to be shrunk. Barrel distortion usually occurs with wide-angle lenses, and pincushion distortion occurs with long focal lengths.

Distortion is countered by using a lens with a narrower angle of view or a vision software calibration algorithm. (9.)

Imaging lenses for machine vision are robust and designed to withstand challenging industrial environments. They can be interchangeable or integrated into the machine vision system. The selection of the imaging lens should be based on the function being performed by the machine vision system, the dimensions of the object under observation, and the distance between the camera and the object. For example, a macro lens with a long focal length is usually selected for examining small product parts. (10.)

2.1.3 Camera Sensor

A camera sensor is a component in digital cameras that captures light from the optical system and converts it into a digital image. It is measured by the number of pixels available in the digital image. The pixels are arranged into a matrix to form an image. The resolution of the sensor is defined by the number of pixels in the matrix. Resolutions range from 640x480 to over 10MP, depending on the application and image accuracy required.

Most cameras in machine vision systems use CCD (A charge-coupled device) or CMOS (complementary metal oxide semiconductor) sensors. CCD sensors are used due to their ability to produce high-quality image data. However, in recent years CMOS (complementary metal oxide semiconductor) sensors have been used more since they consume less power and are more economical. The sensor for the machine vision system should be selected based on the dimensions of the measurements being made and the tolerance of those measurements. Sensors with higher resolution can produce images with higher quality and details, increasing the accuracy of the machine vision system. (11.)

2.1.4 Vision Processing System

A vision processing system refers to a processing unit designed to perform vision software tasks. Said tasks include acquiring an image from a sensor, preprocessing, analyzing the image, and establishing results. Vision processing systems can be external (computer) or internal (stand-alone

machine vision systems.) External processors that are designed for running vision tasks called CVS (compact vision system) are also available.

Stand-alone machine vision systems like smart cameras have all the machine vision system components integrated into one unit. Smart cameras are easier to install than traditional machine vision systems making them more accessible. Smart cameras are also smaller and can be mounted on a moving object, making them suitable for robot applications. (12.)

2.1.5 Communications System

Machine vision communicates with other devices in the automation system, like PLC or HMI, based on the result of vision processing. Since machine vision makes decisions in real-time, communication protocol needs to be fast and reliable. Different communication systems are used depending on the device and manufacturer.

The most common communication solutions for machine vision systems are ethernet-based interfaces like GigE Vision. GigE vision is an interface standard that enables high-speed data transfer and control over Ethernet networks. GigE is based on the Internet protocol (IP) standard, and its goal is to unify protocols for industrial cameras. USB 3.0 camera solutions are also commonly used for communication in machine vision systems. With USB 3.0, it is possible to attain even higher data transfer speed, although cable length might become an issue. Other Communication solutions include older interfaces like FireWire and Camera Link. (13.)

2.2 Vision Software

Vision software refers to functions and methods applied to the image signal from the camera and executed by the processing system. There are lots of different vision software solutions available for various applications. Some companies provide license-based vision software that can be used as a standalone system, while others provide vision software included with other machine vision equipment. The most popular machine vision software solutions available include Vision Builder AI by National Instruments, VisionPro by Cognex, and MVTec by Halcon. Also, Keyence and Omron provide machine vision solutions. Additionally, OpenCV (Open Source Computer Vision Library)

vision libraries can be used in a machine vision application. This chapter describes common functions performed by vision software.

2.2.1 Preprocessing

Preprocessing is the first vision function performed by the processing unit. A raw camera image is not optimal for machine vision applications; thus, some preprocessing is usually applied. These include digital filtering tools and effects like removing noise, adjusting contrast, and thresholding. Vision processing aims to retain as much information about the object being examined as possible. FIGURE 3 illustrates image processing with an image sharpening filter in Photoshop. (14.)

FIGURE 3. Image before (left) and after (right) applying an image sharpening filter. (15.)

2.2.2 Region of Interest

Applying a region of interest (ROI) is a form of cropping the image. A Region of interest is an image region that defines the boundaries of the relevant image processing area. It is typically rectangular, although it can be any shape. Processing made inside the region of interest does not affect the outside image, and it can be reset to the entire image at any time. A suitable region of interest should always be applied for better performance and fewer disruptions. It also decreases processing time and resources since the image region to be processed is smaller. FIGURE 4 illustrates applying region of interest in TwinCAT vision. (16.)

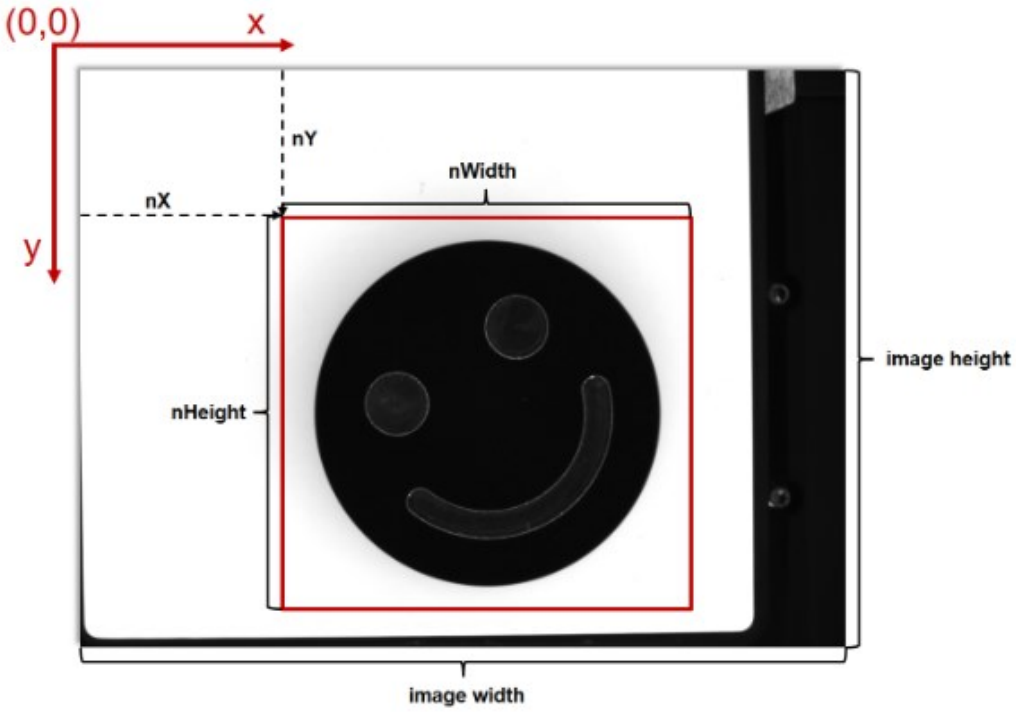


FIGURE 4. Applying a region of interest in TwinCAT vision. (16.)

2.2.3 Object Detection

Object detection is the process of detecting and locating an object from the background using software tools. It is part of almost any machine vision application. In manufacturing applications, the object is usually a product being produced. An object detection tool locates an object and returns the X and Y coordinates of the object's pixels. If the system is calibrated, object detection returns X and Y coordinates in real-world units instead.

An object can be detected using various vision software tools. One of the most common methods is called blob detection. Blob detection detects points and regions that differ from the background in properties like brightness or color. Image processing filters like greyscale and binarizing with certain threshold is usually applied before blob detection to make the objects appear bright, and the surroundings appear dark. A blob is any object that can be clearly distinguished from the background. Blob detection is commonly used in machine vision applications because it uses less CPU power than pattern matching. A method called edge detection has a lot of similarities with blob detection, but whereas blob detection examines regions of connected pixels, edge detection examines the intensity of transitions in an image. FIGURE 5 illustrates a blob detection software tool for detecting round shapes in TwinCAT vision. (17.)

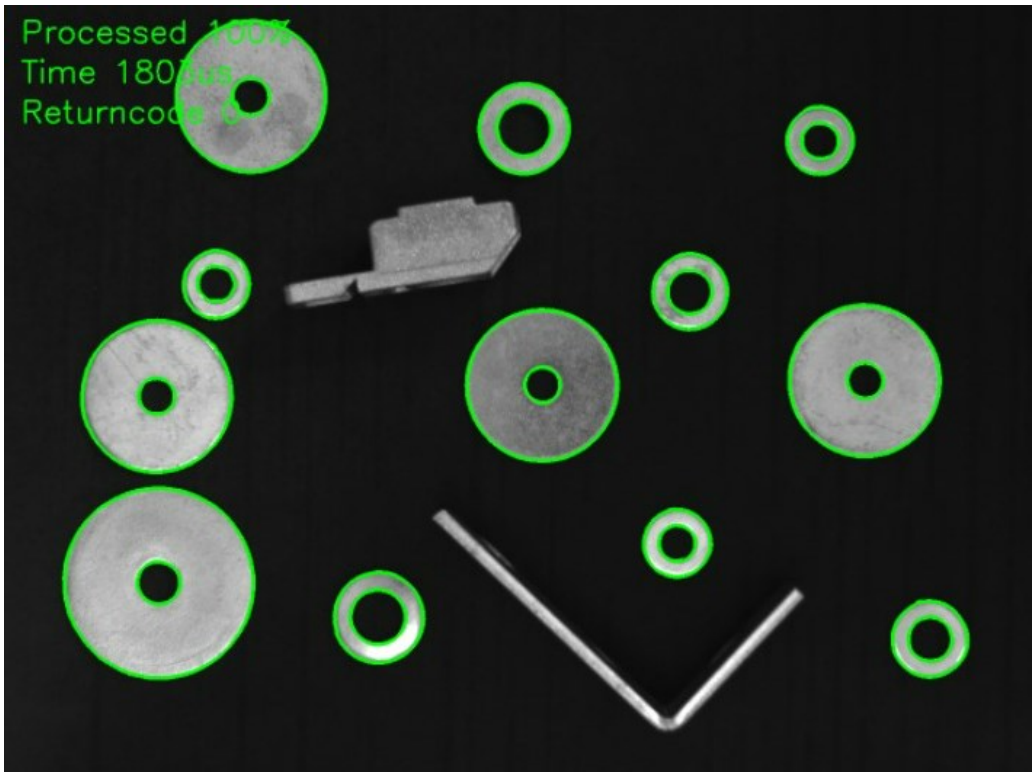


FIGURE 5. Blob detection with TwinCAT 3 Vision. (18.)

2.2.4 Measurement Tools

Machine vision is also a powerful measurement tool. Vision software measurement tools are highly accurate and can measure the object's dimensions and positioning. Also, features like diameter and roundness can be measured. Machine vision systems are often used for quality inspection applications in manufacturing. In quality inspection, a product is accepted or rejected based on the results of a machine vision measurement tool. Using machine vision as a measurement tool has advantages compared to traditional methods. It is fast, highly accurate, and can measure complex or even moving objects. For measuring in real-world units, a calibration of the machine vision system is required. Calibrating is usually executed by using a special calibrating tool or grid. The accuracy of the measurement depends on how well the calibration is executed.

2.2.5 OCR and Code Reading

OCR (Optical Character Recognition) is a software tool that can recognize characters on printed text. Early versions of OCR could only read specific fonts, but advanced OCR systems used nowadays can read most common fonts and even handwritten text. It is mostly used in machine vision

applications to identify parts and products by reading the serial number printed on them. The printed text might also include expiration dates and other important data that can be used in production.

Code reading in machine vision refers to the process of analyzing and interpreting codes using machine vision tools. The tools include reading 2D codes, such as data matrix, QR, and bar codes. The code reading function is useful in manufacturing environments since 2D codes can have more data in a smaller space than printed text. In addition, it is possible to read codes with machine vision even if they are damaged or partly removed due to the error correction data included in the code. (19.)

3 TCP/IP PROTOCOL

TCP/IP consists of two separate network protocols, TCP and IP, but they can be referred to as one network protocol since they usually work in unison. The TCP/IP protocol is the default method of data transmission on the Internet. Together TCP/IP enables highly reliable data transfer between devices over the Internet regardless of the hardware or the operating system. (20.)

TCP (Transmission Control Protocol) is responsible for sending packets over the network and ensuring the data is delivered successfully. It breaks large amounts of data into smaller packets and reorganizes them, so they can be transmitted while ensuring data integrity during the process. The TCP protocol makes communication faster and more reliable since each packet can take its own route to the destination avoiding the congestion of the network. (20.)

IP stands for Internet Protocol, and its primary purpose is delivering data packets between a source device and its destination. Each device connected to the Internet is uniquely identified by an IP address. This enables them to communicate and exchange data through the Internet. The IP protocol is also responsible for data encapsulation. In data encapsulation, header data is added to a packet as it travels to its destination through the TCP/IP protocol stack. (20.)

3.1 TCP/IP Model

The working principle of TCP/IP is defined by the TCP/IP model, which is based on the OSI model. The TCP/IP model represents how data is exchanged and organized over networks. Data is packed and unpacked on different layers of the TCP/IP stack for a continuous stream of communication. Each layer will add its own bit of information. After sending the data, the receiving device will then dissemble the data in the opposite order. The updated TCP/IP model, illustrated in FIGURE 6, has five layers: application, transport, network, data link, and physical layer. (21.)

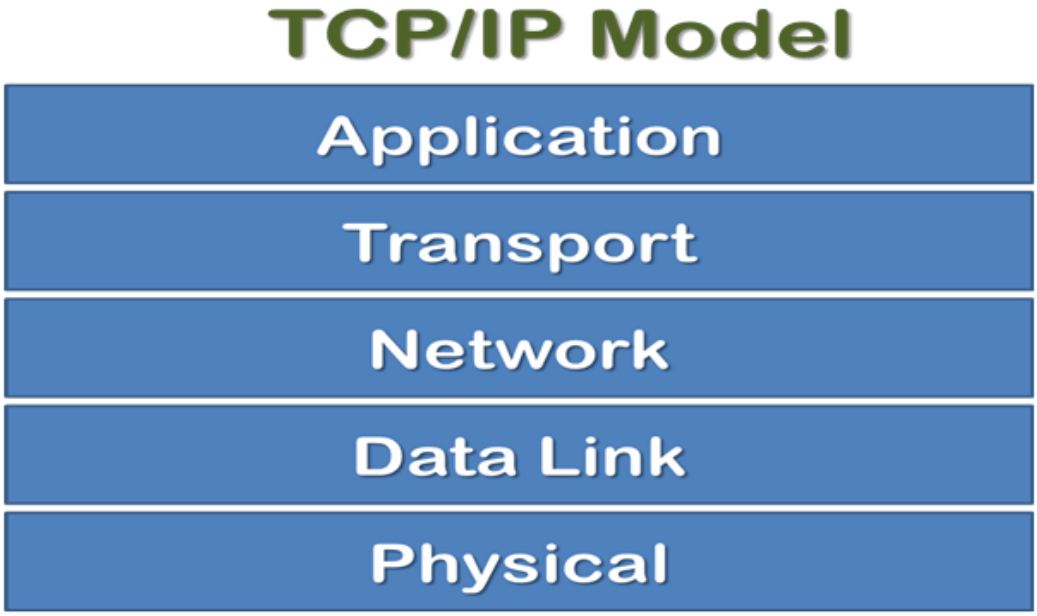


FIGURE 6. The TCP/IP model. (22.)

3.1.1 Application Layer

The application layer is the topmost layer of the TCP/IP model. It combines the application, presentation, and session layers of the OSI model. The application layer can be perceived as the interface which the user interacts with. It includes programs and software which use TCP/IP for communication. Protocols like HTTP, HTTPS, and FTP are present in this layer. (21.)

3.1.2 Transport Layer

The purpose of the transport layer is to provide reliable data transmission for applications. In the connection establishment phase, the client requests for connection, the server sends an acknowledgment to the client, and lastly, the client sends an acknowledgment to the server. This method is called a three-way TCP connection handshake. In the transport layer, TCP divides the data into smaller packets and adds a header to form a TCP segment, which is sent to the receiver. This allows TCP to have features like error handling, ordered data transfer, retransmission of lost data, discarding duplicate packets, and congestion throttling. TCP and UDP protocols are present in this layer. The UDP protocol is faster but does not support segmentation and is more unreliable. (21.)

3.1.3 Network Layer

The Network layer provides the functions and procedures for transferring data across networks. It assigns the sender's and the receiver's IP address to each TCP segment to form an IP packet. An IP address is assigned to ensure the IP packet reaches the correct destination. The IP packet is then sent to other networks through routers. Routing is unnecessary if the source and destination are in the same network. The network layer also determines the best path from the source to the destination, although it does not check for errors. (21.)

3.1.4 Datalink Layer

The datalink layer is the second lowest layer of the TCP/IP model. It is responsible for sending and receiving data through transmission devices, such as Ethernet cables and network interface cards. The datalink layer adds a header and a trailer to the IP packet received from the network layer. The header contains the MAC addresses of the sender and the receiver. The trailer contains 4 bytes of error-handling data used for detecting errors in the transmission. Once the header and the trailer are added, the data unit is called an ethernet frame. The datalink layer also has similar functionalities as the transport layer, like flow control and retransmission of data. (21.)

3.1.5 Physical Layer

The physical layer converts communication data into signals and transmits them to local media. The physical layer is where actual communication occurs. A signal can be electrical, in the case of cables, light, in case of optical fiber, or a radio signal. Most of the communication in the physical layer uses the Ethernet protocol. The physical layer also determines the types of cables that can be used for transmission. (21.)

3.2 TCP/IP CLIENT

Currently, a TCP/IP client is used for communication in JOT Automation. It is an external software used for data transmission between the server and the client in various automation systems, including machine vision applications. The TCP/IP client is based on the TCP/IP protocol, and it is developed by JOT Automation.

4 ADS PROTOCOL

ADS (Automation Device Specification) is a communication protocol and interface technology developed by Beckhoff Automation based on the TCP/IP standard. Within Beckhoff's TwinCAT automation software, it is used for communicating between different software components, for instance, the communication between the NC and the PLC. In addition, ADS can also be used for external communication with automation devices that supports the protocol. ADS communication between different software modules within the TwinCAT system and with other automation systems is illustrated in FIGURE 7. (23.)

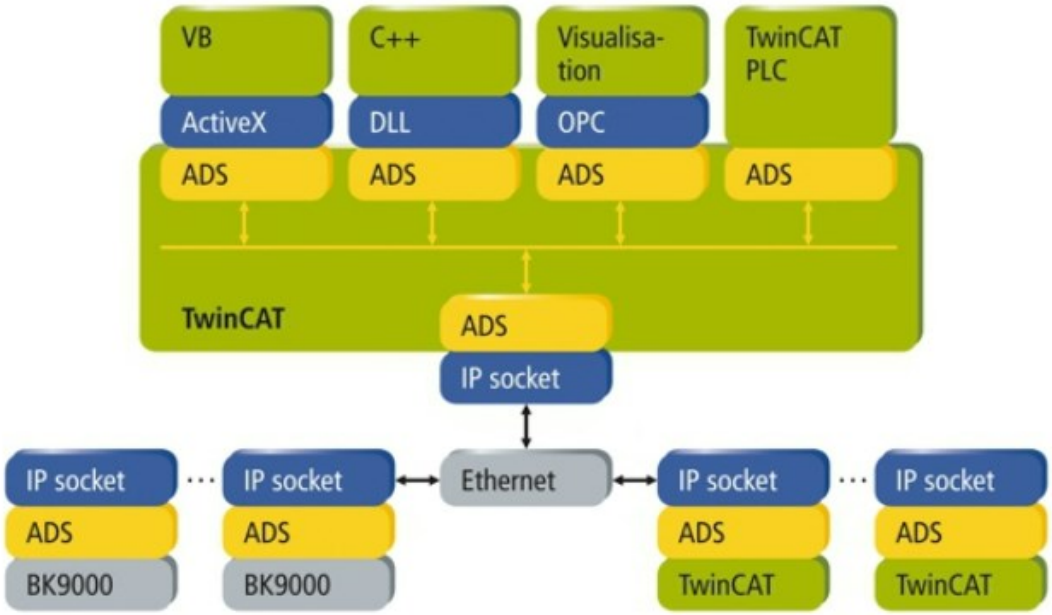


FIGURE 7. ADS communication within the TwinCAT system and for external communication. (23.)

4.1 Message Router

ADS enables communication between UserMode and RealTime. UserMode is the engineering environment of TwinCAT, whereas RealTime refers to the different operating modules of the TwinCAT system, such as IO tasks. Messages in the system are exchanged through a consistent ADS interface by the message router, illustrated in FIGURE 8. For every task, there is a software module called "server" or "client." The servers in the system are the executing working devices in the form of software. The clients are programs that request the services of the servers in the form of a

program. Within the TwinCAT system, individual software modules are treated as independent devices. (24.)

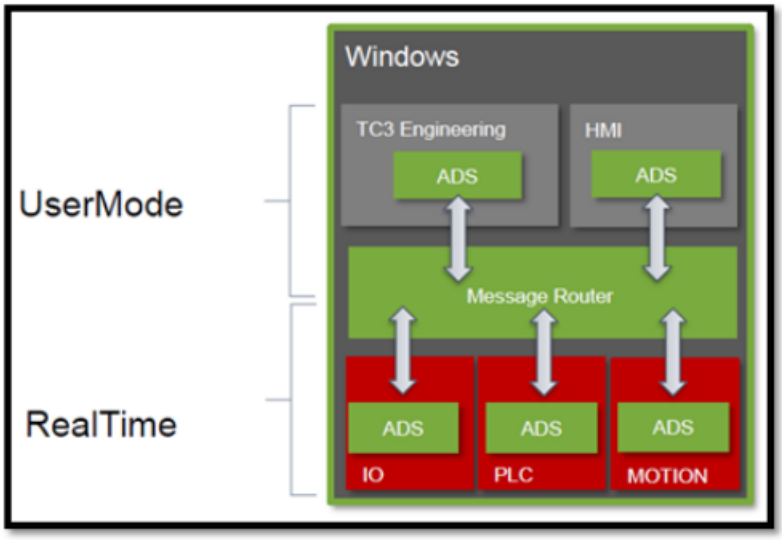


FIGURE 8. Message router of the TwinCAT system. (24.)

The ADS protocol also applies to external communication. When communicating with other PLCs or devices, the message router of a TwinCAT system exchanges data with the message router of another TwinCAT system using ADS as an extension of the TCP/IP protocol. The communication between TwinCAT systems is illustrated in FIGURE 9. (24.)

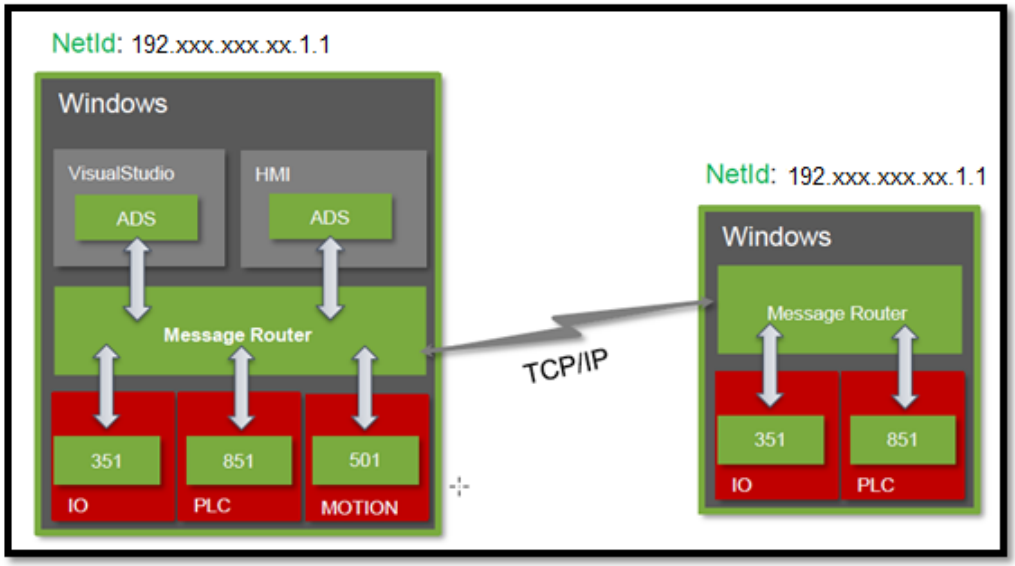


FIGURE 9. Communication between TwinCAT systems. (24.)

4.2 Structure of ADS Communication

Individual ADS devices in the network can be addressed by their NetId and PortNr. For internal communication, only PortNr is needed. ADS data is furthermore specified by its index group and index offset. The structure of ADS communication is illustrated in FIGURE 10. (25.)

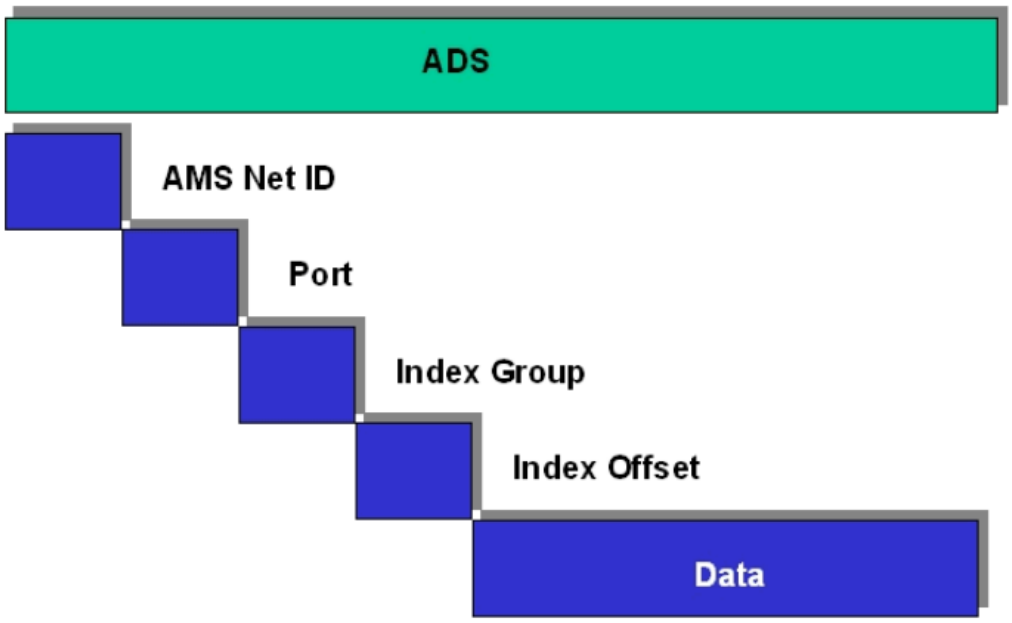


FIGURE 10. Structure of ADS communication. (25.)

4.2.1 NetId

The AMS NetId identifies individual TwinCAT systems. It consists of six octets, and it is initially an extension of the IP address. The last two octets serve as a subnet mask for field buses or further devices. The AMS NetId must be unique for all communication partners. The AMS NetId of a local or remote TwinCAT device can be set in SYSTEM\Routes\NetId Management of a TwinCAT project, illustrated in FIGURE 11. (25.)

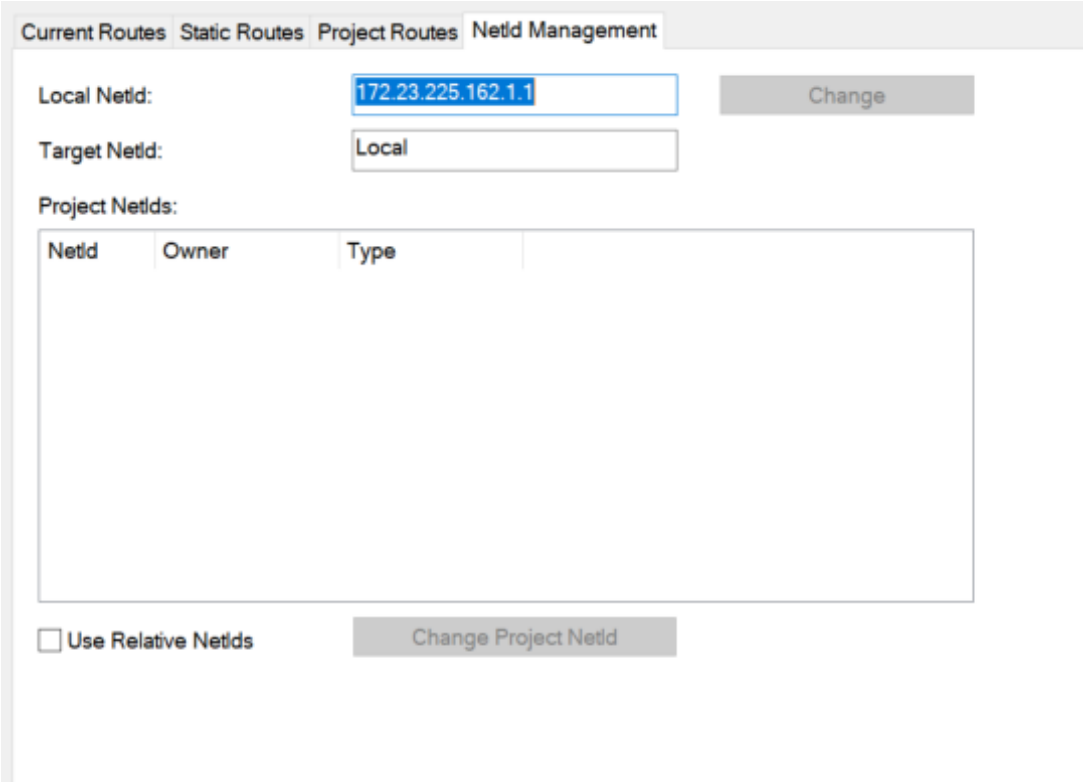


FIGURE 11. Setting NetId in TwinCAT system.

4.2.2 PortNr

ADS devices in a TwinCAT message router are uniquely identified by a number referred to as the PortNr. It also states the device's role in the system. For ADS devices, this has a fixed specification, whereas pure ADS client applications are allocated a variable port number when they first access the message router. Default AMS PLC Runtime ports start from 851 (800 for TwinCAT 2). Other default AMS ports are 500 for NC and 300 for I/O servers. (26.)

4.2.3 Index Group and Index Offset

Index group and index offset are categorized as subclasses for PortNr. The length of the index group is 16 bits, and it separates data from the PortNr into groups. The index offset indicates the offset from which reading or writing the byte is to start. The length of the index offset is 32 bits. (26.)

4.3 ADS Services

The client sends an ADS request at the start of ADS communication. Next, the ADS request arrives at the ADS server, where it is viewed as an ADS indication. Then the ADS server replies with an ADS response, which the ADS client views as an ADS confirmation. Server-client relation of ADS communication is illustrated in FIGURE 12. (27.)

ADS-Client	ADS-Server
Request ->	-> Indication
Confirmation <-	<- Response

FIGURE 12. Server-client relation of ADS communication. (27.)

Generally, ADS communication services are classified into synchronous, asynchronous, and event-based services. Other methods for communicating with the ADS server are direct access via address and indirect access via handles. (28.)

4.3.1 Synchronous Communication

In synchronous communication, client execution stops if an ADS request is sent. The client continues the code execution once a response from the server is received. The synchronous communication method is simple to implement between the client and the server. Additionally, this method is not prone to errors because the client waits for a response from the server. An example scenario of synchronous communication is the communication between an HMI and a PLC. Synchronous communication is illustrated in FIGURE 13. (28.)

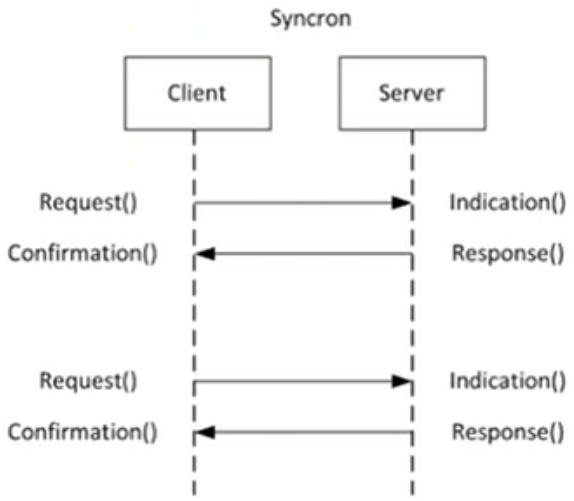


FIGURE 13. Synchronous communication. (28.)

4.3.2 Asynchronous Communication

In asynchronous communication, the client continues the code execution after an ADS request is sent. Therefore, the client can send several ADS requests in one cycle. Responses are assigned accordingly by a unique ID. The asynchronous communication method is considered more unstable than the synchronous communication method. It is used if it is not possible to stop the code execution while waiting for the response. An example scenario of asynchronous communication is the communication between two PLCs. Asynchronous communication is illustrated in FIGURE 14. (28.)

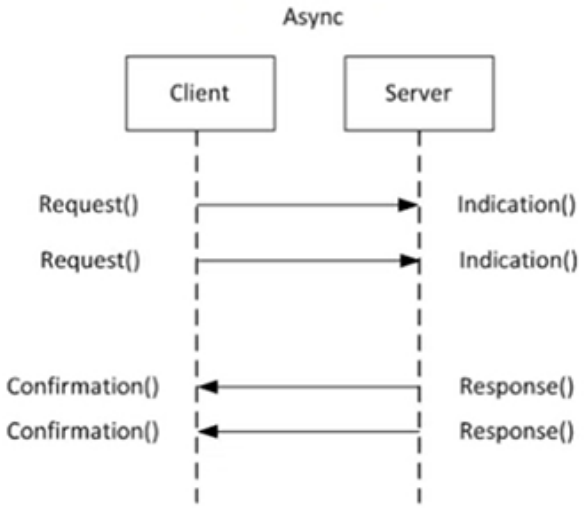


FIGURE 14. Asynchronous communication. (28.)

4.3.3 Event-based Communication

The concept in event-based communication is that the client sets an event where a threshold for values can be configured. If the threshold is reached, a response message from the server is generated. This method reduces the number of requests. It also provides an accurate timestamp for measurement and charting. In event-based communication, notifications are received “cyclic” or “OnChange.” Cyclic implies that the client gets a value after each cycle. The OnChange method sends a value only after the value changes. The event-based communication methods are illustrated in FIGURE 15. (28.)

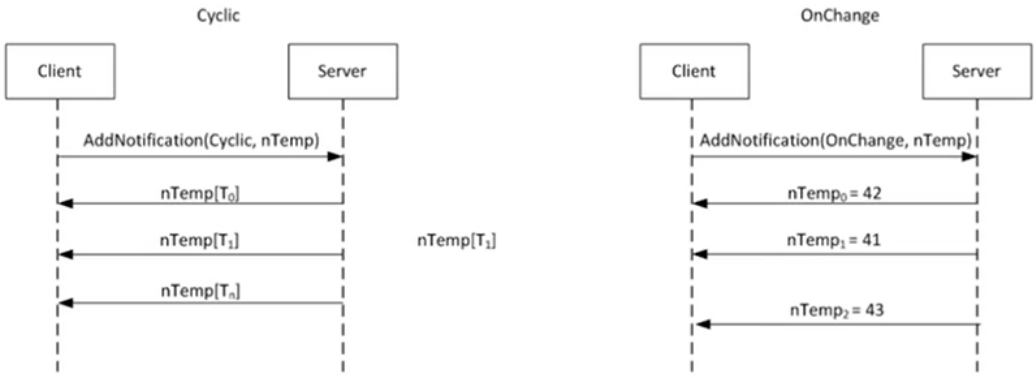


FIGURE 15. Event-based communication methods. (28.)

5 ADS INTERFACE

ADS interface between a Beckhoff PLC and a machine vision application was established using LabVIEW. LabVIEW is a graphical programming software developed by National Instruments. LabVIEW was originally developed as an automatic measurement tool for laboratory applications, but nowadays, it is a flexible programming environment for various applications like equipment control, data acquisition applications, and user interface design. Some benefits of using LabVIEW are its adaptive graphical design and compatibility with 3rd party applications. LabVIEW version 2019 was used for this thesis. FIGURE 16 illustrates the user interface of LabVIEW. (29.)

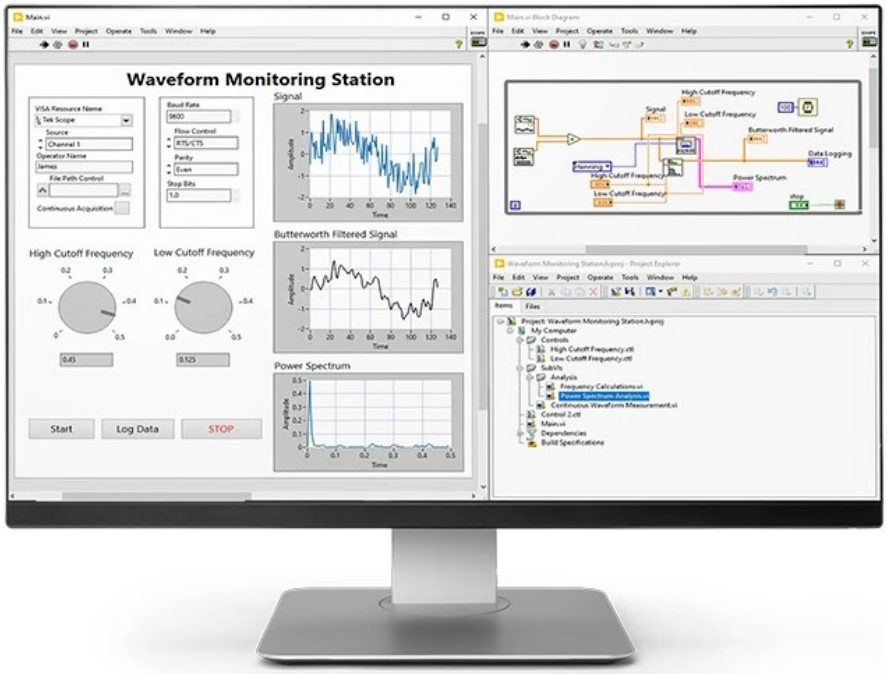


FIGURE 16. LabVIEW graphical programming software. (29.)

5.1 TF3710 TwinCAT 3 Interface for LabVIEW

Beckhoff provides user libraries and interfaces for 3rd party applications. The ADS interface was programmed utilizing TF3710 TwinCAT 3 Interface for LabVIEW. TF3710 is a library extension for LabVIEW that enables LabVIEW to exchange data with TwinCAT runtime. It provides functions necessary to write and read values to and from TwinCAT. Other similar tools available were also

considered but the TF3710 library was chosen because it is developed by Beckhoff and seemed the most suitable for this application. It can be downloaded from Beckhoff's website free of charge. FIGURE 17 shows the TF3710 menu in LabVIEW. (30.)

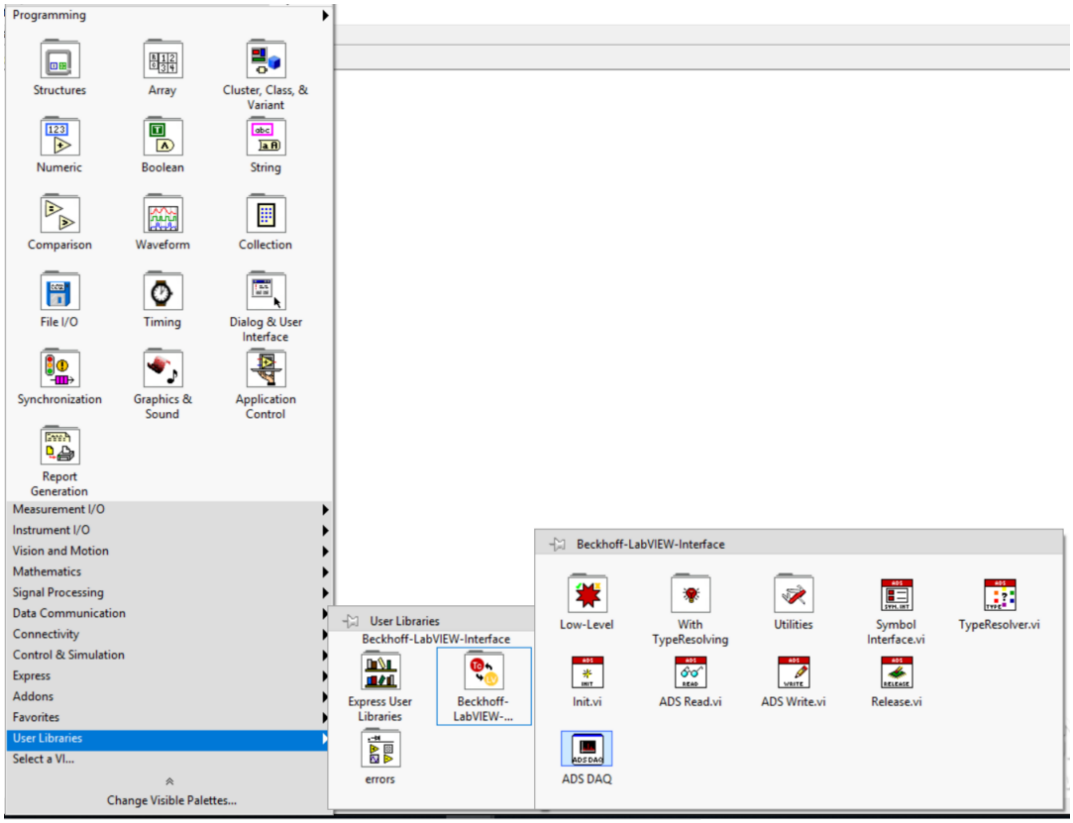


FIGURE 17. TF3710 TwinCAT 3 Interface for LabVIEW user library. (30.)

5.2 Structure of Read and Write Virtual Instruments

The ADS interface was programmed as a LabVIEW VI. A VI (Virtual Instrument) is a basic building block of programs written in a graphical programming language. It consists of two software components: a front panel and a block diagram. The front panel can be perceived as an HMI containing controls and indicators for the software, whereas the block diagram contains the software's functionality. (30.)

For two-way communication, read and write virtual instruments are required. The read and write virtual instruments contain a symbol interface, initialization of the ADS client, reading/writing of values, and type resolver and release functions. Data is exchanged between the functions via handles. Handles are identifiers that are used for keeping track of and managing data objects. A basic structure of read and write virtual instruments using the TF3710 library is illustrated in FIGURE 18. (30.)

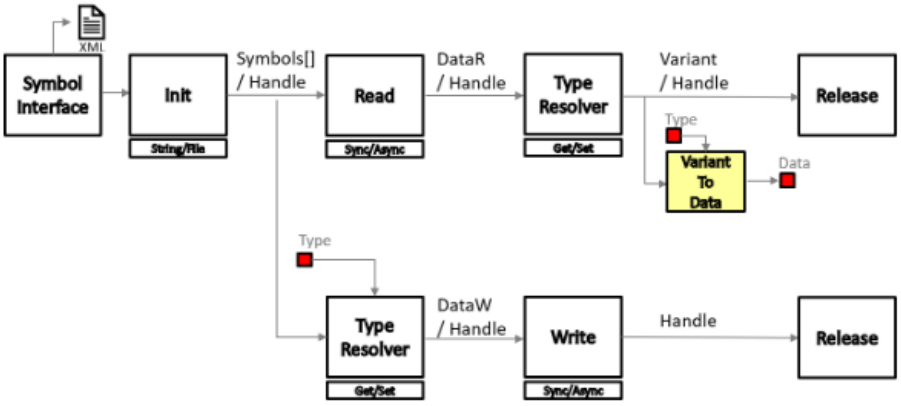


FIGURE 18. Structure of read and write virtual instruments using the TF3710 library. (30.)

5.3 Symbol List

The first step in establishing the ADS interface is creating a symbol list. This can be achieved by using a symbol interface function from the TF3710 library or reading symbols from an XML file. The latter method was used. The symbol list XML file, shown in FIGURE 19, was created using an ADS DAQ function from the TF3710 library. The ADS DAQ function reads variables from TwinCAT's runtime and creates a symbol list in XML format. ADS read and ADS write symbols are defined accordingly in the XML file.

```
<?xml version="1.0"?>
- <PortInfo>
  - <Inputs>
      <SymbolControlItems/>
    </Inputs>
  - <Outputs>
    - <SymbolControlItems>
      - <SymbolControlItem GroupDescription="Parameters for ADS Read command" GroupName="ADS Read">
        - <Parameters>
            <Parameter Name="Name">VisionSequence.bStart</Parameter>
            <Parameter Name="SymbolName">VisionSequence.bStart</Parameter>
            <Parameter Name="SymbolType">BOOL</Parameter>
            <Parameter Name="SymbolGuid"/>
            <Parameter Name="AmsNetId">172.23.225.162.1.1</Parameter>
            <Parameter Name="Port">851</Parameter>
            <Parameter Name="IndexGroup">#x0</Parameter>
            <Parameter Name="IndexOffset">#x0</Parameter>
            <Parameter Name="BitSize">8</Parameter>
            <Parameter Name="CycleTime [ms]">1</Parameter>
            <Parameter Name="Transmode">3</Parameter>
            <Parameter Name="Timeout [ms]">1000</Parameter>
            <Parameter Name="TCBufferSize [Samples]">10</Parameter>
            <Parameter Name="LVBufferSize [Samples]">50</Parameter>
            <Parameter Name="Expected min delay [ms]">50</Parameter>
          </Parameters>
        </SymbolControlItem>
      - <SymbolControlItem GroupDescription="Parameters for ADS Write command" GroupName="ADS Write">
        - <Parameters>
            <Parameter Name="Name">VisionSequence.Diameter</Parameter>
            <Parameter Name="SymbolName">VisionSequence.Diameter</Parameter>
            <Parameter Name="SymbolType">DINT</Parameter>
            <Parameter Name="SymbolGuid"/>
            <Parameter Name="AmsNetId">172.23.225.162.1.1</Parameter>
            <Parameter Name="Port">851</Parameter>
            <Parameter Name="IndexGroup">#x0</Parameter>
            <Parameter Name="IndexOffset">#x0</Parameter>
            <Parameter Name="BitSize">32</Parameter>
            <Parameter Name="CycleTime [ms]">1</Parameter>
            <Parameter Name="Transmode">3</Parameter>
            <Parameter Name="Timeout [ms]">1000</Parameter>
            <Parameter Name="TCBufferSize [Samples]">10</Parameter>
            <Parameter Name="LVBufferSize [Samples]">50</Parameter>
            <Parameter Name="Expected min delay [ms]">50</Parameter>
          </Parameters>
        </SymbolControlItem>
      - <SymbolControlItem GroupDescription="Parameters for ADS Write command" GroupName="ADS Write">
        - <Parameters>
            <Parameter Name="Name">VisionSequence.X_Coordinate</Parameter>
            <Parameter Name="SymbolName">VisionSequence.X_Coordinate</Parameter>
            <Parameter Name="SymbolType">DINT</Parameter>
            <Parameter Name="SymbolGuid"/>
            <Parameter Name="AmsNetId">172.23.225.162.1.1</Parameter>
            <Parameter Name="Port">851</Parameter>
            <Parameter Name="IndexGroup">#x0</Parameter>
            <Parameter Name="IndexOffset">#x0</Parameter>
            <Parameter Name="BitSize">32</Parameter>
```

FIGURE 19. Symbol list XML file.

5.4 ADS Read Virtual Instrument

The purpose of the ADS read virtual instrument is to read variables from PLC to vision software. The symbol list is read using a basic Read Text File function and the previously configured XML file. Then an Init function from the TF3710 library is used to initialize variables. The Init function passes the handles of the ADS symbols to a read function. A Read Sync Single TypeResolved function was used for reading values. It reads variables from the ADS server synchronously and converts them to LabVIEW data type. It also releases the ADS client from memory. The variables are then converted to an appropriate data type using a Variant to Data function. The read function is placed inside a for loop for reading multiple values. The output of the ADS read virtual instrument is in array format. ADS read virtual instrument is shown in FIGURE 20.

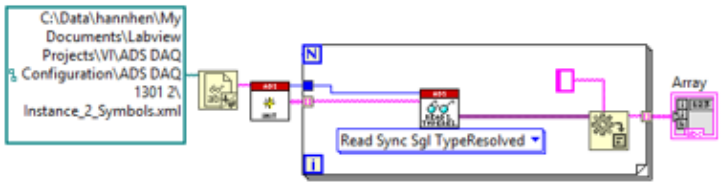


FIGURE 20. ADS read virtual instrument.

5.5 ADS Write Virtual Instrument

An ADS write virtual instrument consists of similar functions but in a different order. At first, it reads the symbol list and initializes the variables. The input of the ADS write virtual instrument is in array format. A Variant to Data function converts the variables to the appropriate data type, and a To TC function converts them to the TwinCAT data type. Handles and raw ADS write data are then passed to a write function. ADS Write Sync Single function was used to write values to the server synchronously. The writer is automatically released after the data packet has been successfully sent to TwinCAT. The write function is placed inside a for loop for writing multiple values. The ADS write virtual instrument is shown in FIGURE 21.

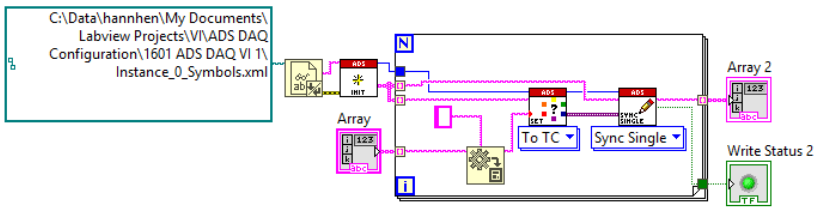


FIGURE 21. ADS write virtual instrument.

5.6 Library File

After programming the read and write virtual instruments, they must be exported into separate LLB files. A LLB (library file) is a container file for multiple non-LLB files. In this case, the library file combines all the sub virtual instruments and the TF3710 library data into a single file, which can then be imported into vision software. The library file is created by selecting source distribution from the project item menu of LabVIEW and configuring source file settings and other specifications. FIGURE 22 illustrates creating a library file in LabVIEW. (30.)

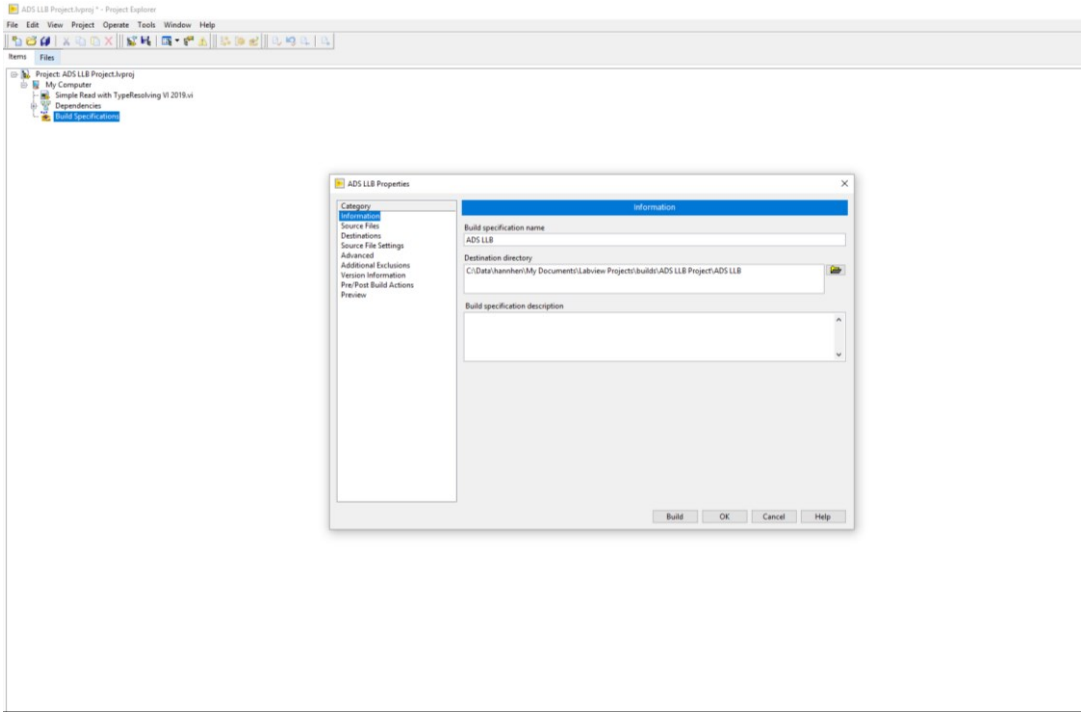


FIGURE 22. Creating a library file in LabVIEW.

6 PLC APPLICATION

The PLC functionality was programmed in TwinCAT 3 environment. TwinCAT (The Windows Control and Automation Technology) is a complete automation system for PC-compatible computers, and it is developed by Beckhoff. The TwinCAT system consists of multiple components. The programming environment is called eXtended Automation Engineering (XAE.) It allows hardware configuration, programming, and debugging in a single system, and it is integrated into Microsoft's Visual Studio. It supports programming languages such as IEC 61131-3 programming languages, C/C++, and MATLAB/Simulink. (31.)

The real-time capable component of the TwinCAT system is called eXtended Automation Runtime (XAR.) It allows its components to be loaded, executed, and administrated in real time, making it a real-time control system. The program code can be simulated in the local target system. The TwinCAT system also supports core isolation for multi-core systems making it possible to assign single cores for specific tasks, which helps with managing RAM usage. For example, I/O tasks can be assigned to an isolated core. The real-time execution of the PLC code is illustrated in FIGURE 23. ADS communication takes place after writing outputs in the TwinCAT PLC cycle. (31.)

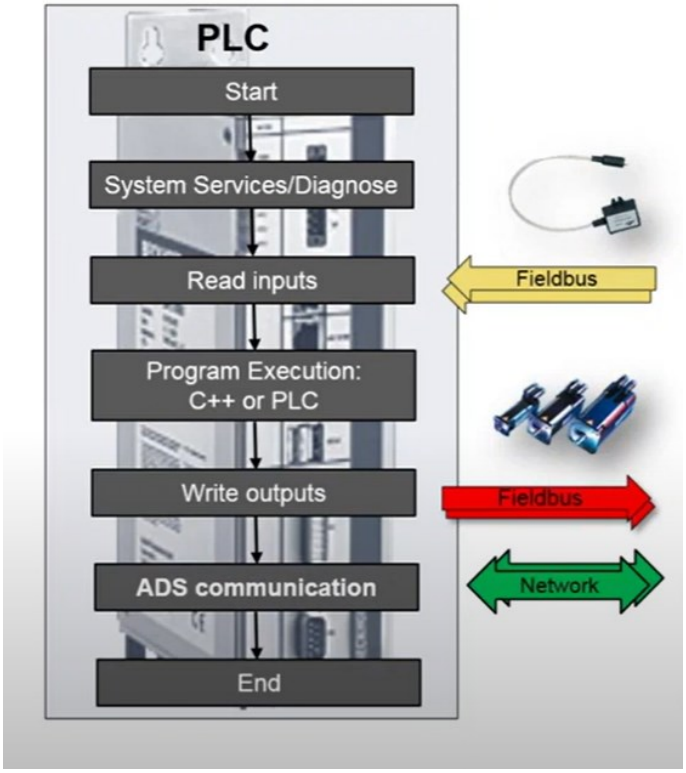


FIGURE 23. Structure of TwinCAT PLC cycle. (31.)

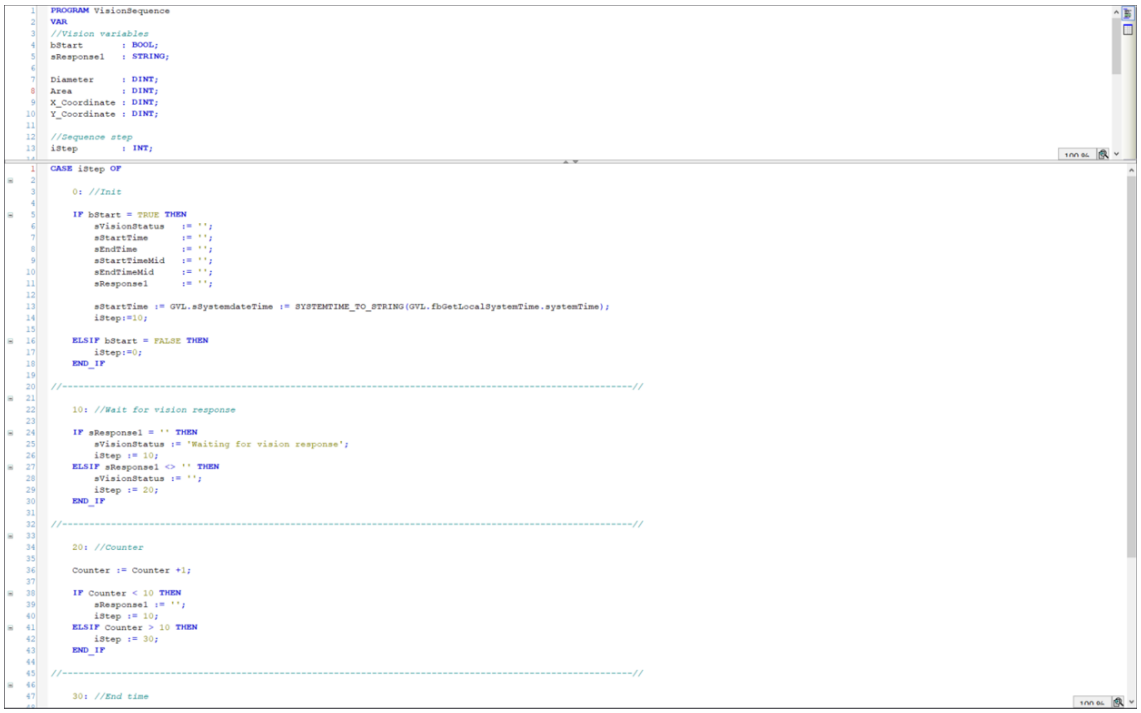
6.1 PLC Functionality

In the thesis, PLC is responsible for sending a start command and receiving a response message from the vision application. Two separate PLC applications were programmed for the ADS interface and the TCP/IP client. These programs are then called in the main program. In the ADS interface PLC application command and response messages are sent using the ADS protocol. Accordingly, in the TCP/IP client PLC application, messages are sent using the TCP/IP protocol. Command and response messages are also timed using a TwinCAT function.

6.2 ADS Interface PLC Application

The ADS interface PLC application is programmed using a case structure. Since the communication is symbol-based, the TwinCAT symbol list must be equal to the symbol list used in the ADS interface. The symbol list is shown in APPENDIX 1. The PLC sends a Boolean start command to start the vision application. The start time is acquired using the FB_LocalSystemTime function from the TcUtilities library. After sending a start command, the PLC waits for a response message from the vision application. The response message from the vision application contains measurement values in the Dint data type. Vision status is updated accordingly. Lastly, the PLC application parses response data and returns the correct nut size.

For sending commands continuously, a counter is programmed. If the counter value is below the setpoint, the PLC sends another command and waits for results. Once the counter limit is reached, the end time is set, and the PLC cycle ends. The code execution time can be calculated from the start and end time values. It is also possible to send and receive multiple messages at once using the PLC application. The ADS interface PLC application is shown in FIGURE 24.



```
PROGRAM VisionSequence
VAR
//Vision variables
bStart       : BOOL;
sResponse1   : STRING;

Diameter     : DINT;
Area         : DINT;
X_Coordinate : DINT;
Y_Coordinate : DINT;

//Sequence step
iStep        : INT;
```

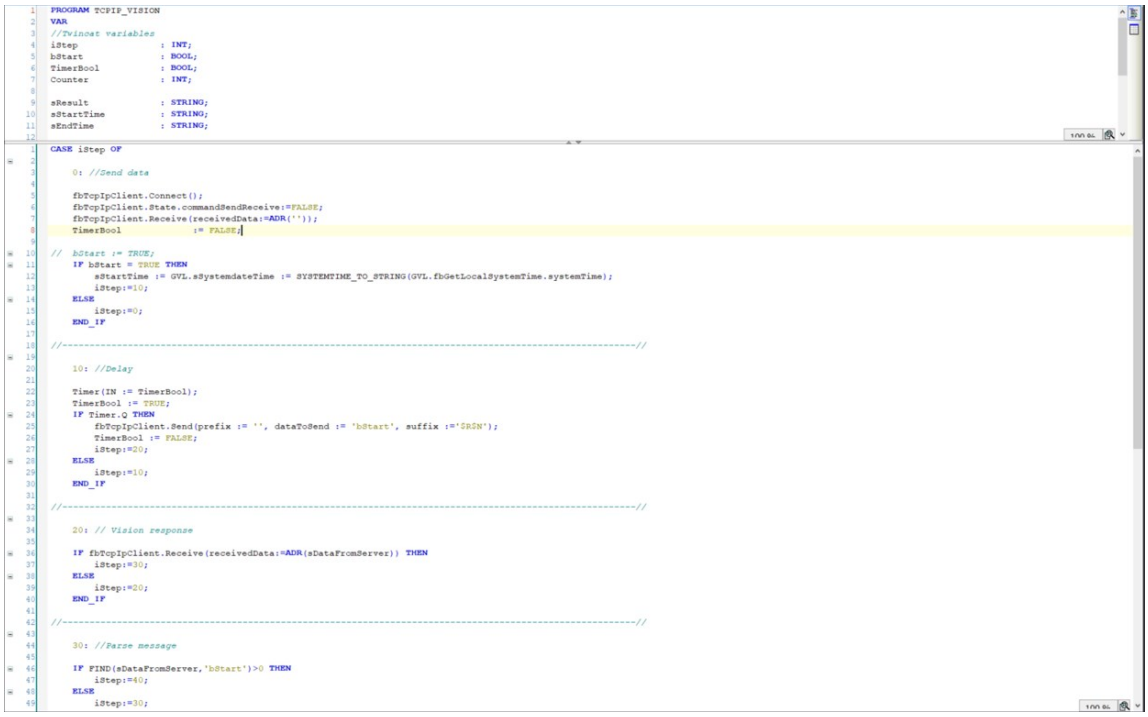
```
CASE iStep OF

    0: //Init

    IF bStart = TRUE THEN
        sVisionStatus   := '';
        sStartTime      := '';
        sEndTime        := '';
        sStartTimeMid   := '';
        sEndTimeMid     := '';
        sResponse1      := '';

        sStartTime := GVL.sSystemdateTime := SYSTEMTIME_TO_STRING(GVL.fbGetLocalSystemTime.systemTime);
        iStep:=10;

    ELSIF bStart = FALSE THEN
        iStep:=0;
    END_IF

//-------------------------------------------------------------------------------------------//

    10: //Wait for vision response

    IF sResponse1 = '' THEN
        sVisionStatus := 'Waiting for vision response';
        iStep := 10;
    ELSIF sResponse1 <> '' THEN
        sVisionStatus := '';
        iStep := 20;
    END_IF

//-------------------------------------------------------------------------------------------//

    20: //Counter

    Counter := Counter +1;

    IF Counter < 10 THEN
        sResponse1 := '';
        iStep := 10;
    ELSIF Counter > 10 THEN
        iStep := 30;
    END_IF

//-------------------------------------------------------------------------------------------//

    30: //End time
```

FIGURE 24. ADS interface PLC application.

6.3 TCP/IP Client PLC Application

TCP/IP client PLC application has similar functionality to the ADS interface PLC application. The symbol list for the TCP/IP client PLC application is shown in APPENDIX 2. However, communication with the TCP/IP client is established using a specific function block. Data unit types are defined for the function block. Sending and receiving messages with the vision application are executed using this function block. Command and response messages are string variables and require a suffix for the TCP/IP client.

An additional delay in the PLC cycle is required to send commands continuously with the TCP/IP client. This is because the TCP/IP client is event-based communication, and thus, it cannot process consecutive commands in the network. Communication error occurs when commands are sent faster than a response is received. Therefore, a timer is programmed as a delay in the PLC application. The code execution time is calculated from the start and end time variables. The TCP client can also send and receive multiple messages at once using a parser in the PLC application for the response message, which separates different measurement data from a single string. The TCP/IP client PLC application is shown in FIGURE 25.



```
PROGRAM TCPIP_VISION
VAR
//Twincat variables
iStep               : INT;
bStart              : BOOL;
TimerBool           : BOOL;
Counter             : INT;

sResult             : STRING;
sStartTime          : STRING;
sEndTime            : STRING;
```

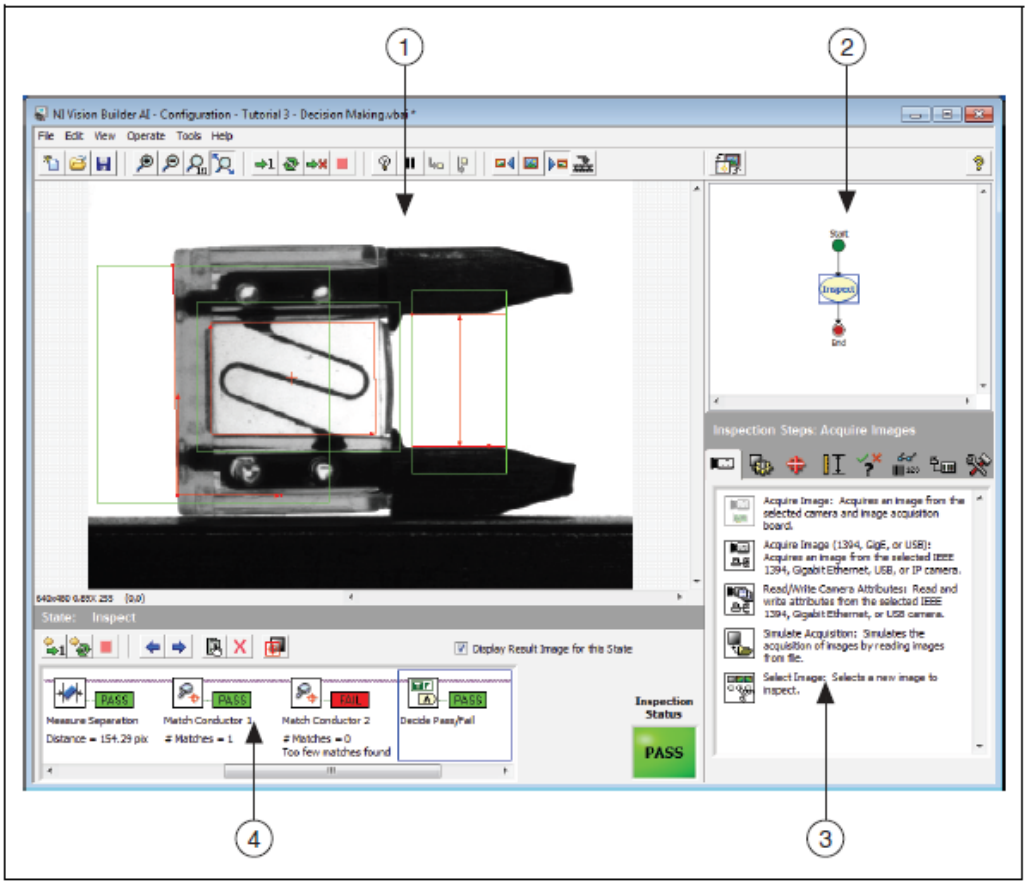
```
CASE iStep OF

    0: //Send data

    fbTcpIpClient.Connect();
    fbTcpIpClient.State.commandSendReceive:=FALSE;
    fbTcpIpClient.Receive(receivedData:=ADR(''));
    TimerBool           := FALSE;

//  bStart := TRUE;
    IF bStart = TRUE THEN
        sStartTime := GVL.sSystemdateTime := SYSTEMTIME_TO_STRING(GVL.fbGetLocalSystemTime.systemTime);
        iStep:=10;
    ELSE
        iStep:=0;
    END_IF

//------------------------------------------------------------------------------------------------//

    10: //Delay

    Timer(IN := TimerBool);
    TimerBool := TRUE;
    IF Timer.Q THEN
        fbTcpIpClient.Send(prefix := '', dataToSend := 'bStart', suffix :='SRSN');
        TimerBool := FALSE;
        iStep:=20;
    ELSE
        iStep:=10;
    END_IF

//------------------------------------------------------------------------------------------------//

    20: // Vision response

    IF fbTcpIpClient.Receive(receivedData:=ADR(sDataFromServer)) THEN
        iStep:=30;
    ELSE
        iStep:=20;
    END_IF

//------------------------------------------------------------------------------------------------//

    30: //Parse message

    IF FIND(sDataFromServer,'bStart')>0 THEN
        iStep:=40;
    ELSE
        iStep:=30;
```

FIGURE 25. TCP/IP client PLC application.

7 VISION APPLICATION

Vision application was programmed using Vision Builder AI by National Instruments. Vision Builder AI has hundreds of vision tools for image processing and inspection. It also includes a vision acquisition software tool that enables connecting and configuring Camera Link, GigE, and USB 3.0 cameras. It allows communication with PLCs and other industrial hardware with interfaces such as TCP/IP, Modbus, and Ethernet/IP. (32.)

Machine vision functionality is programmed within states in Vision Builder AI. A state contains various functions and tools for the visual inspection of an object. The transition between states can be defined for a certain condition. States are programmed between the start and end in the state diagram. Usually, multiple states and different transition conditions are required in vision programming. An overview of the inspection can be seen in the inspection window. The user interface of NI Vision Builder is shown in FIGURE 26. (32.)



1	Main Window	3	Inspection Steps Palette
2	Overview Window	4	State Configuration Window

FIGURE 26. NI Vision Builder AI user interface. (33.)

7.1 Vision Application

A vision application is developed to test communication interfaces between PLC and machine vision practically. The purpose of the vision application is to detect and examine standard hexagon nuts. The vision application detects and measures nuts using vision tools and returns various data. It can detect nuts from size M3 to size M8, as shown in FIGURE 27, and it also works for detecting multiple nuts simultaneously.

FIGURE 27. Standard hexagon nuts.

Two separate versions of vision application were programmed, one for the ADS interface and the second for the TCP/IP client. The basic functionality and state machine of the software is similar in both cases. The state machine consists of Idle, Init, Detect Objects, Find Circular Edge, No Objects Found, and Results states. FIGURE 28 shows the state diagram of the vision application.

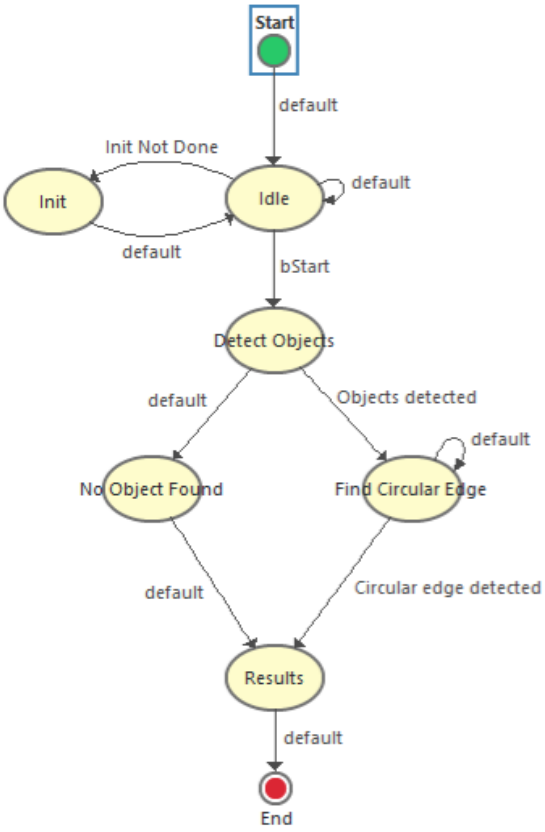


FIGURE 28. State diagram of vision application.

The first state in the vision application is the Init state, which resets counters and variables. In the Idle state, the vision application waits for a start command from the PLC. The next state is Detect Objects, which consists of multiple functions. It acquires an image, detects objects, and stores object count into a variable. The image acquiring can be simulated by reading an image from a file or acquiring a live image from a camera. Finally, vision start time data is written into a log file using a Data Logging function. FIGURE 29 illustrates the detect object function in NI Vision Builder AI.

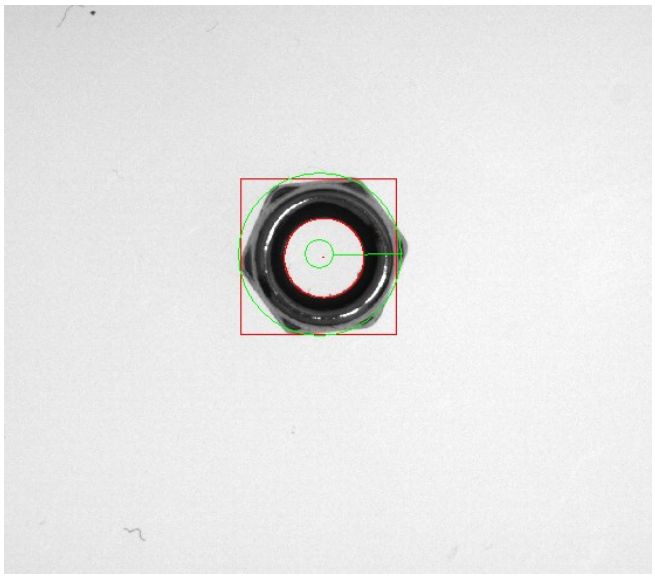


FIGURE 29. Detect objects function in NI Vision Builder AI.

The next state after Detect Objects depends on the result of the state. If no objects were detected, the next state is No Objects Found. This state overlays an error and sends an error message to the PLC. If objects are detected, the next state is called Find Circular Edge. In the Find Circular Edge state, an ROI (Region of Interest) location is defined according to the object's X and Y coordinates. The circular edge is then detected using this ROI. Figure 30 illustrates the find circular edge function in Vision Builder AI.

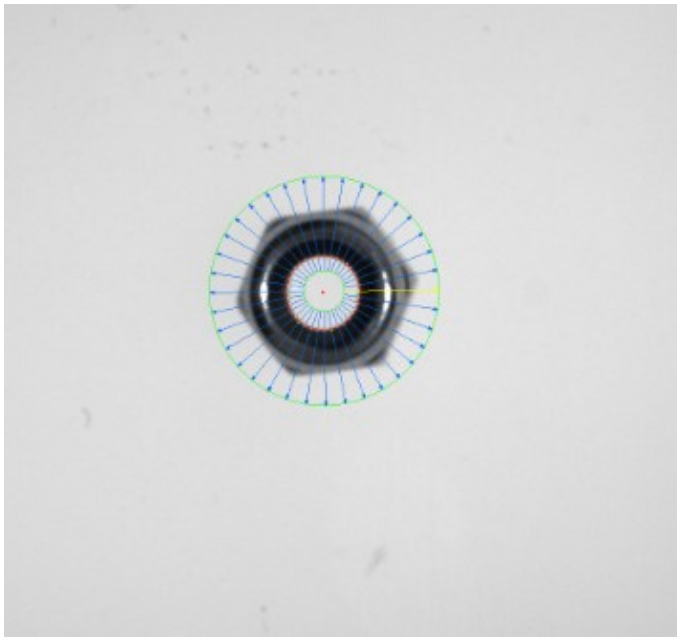


FIGURE 30. Find circular edge function in Vision Builder AI.

The Result state sends vision results to the PLC. The result consists of measurement data, including diameter, X and Y coordinates, and area of the nut. The PLC parses the results and returns the

correct nut size. An error message is sent to the PLC if no objects were detected. Vision end time data is acquired using a data logging function. The vision run time is calculated from the difference between the start and end time variables.

7.2 Vision Application Communication Protocol

The vision application versions have different communication protocols. The ADS interface vision application communicates with the PLC using a LabVIEW virtual instrument. In the Idle state, the vision application waits for a start command from the PLC using a RUN LabVIEW VI function and the previously created ADS read virtual instrument. The ADS read virtual instrument is imported to the vision application with the corresponding library file. Accordingly, in the Results state, the vision application sends results using the ADS write virtual instrument. The Run LabVIEW VI function is shown in FIGURE 31.

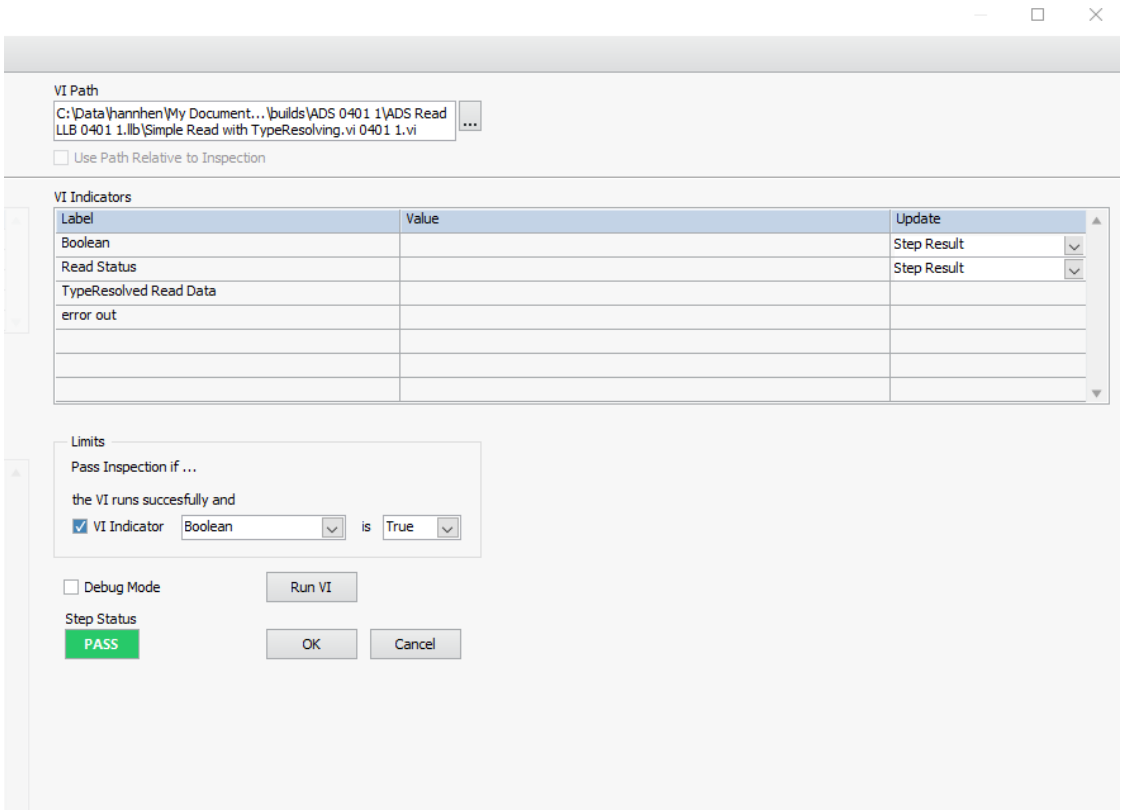


FIGURE 31. Run LabVIEW VI function in NI Vision Builder AI.

The TCP/IP client vision application communicates with the PLC using a TCP I/O function, shown in FIGURE 32. A TCP/IP server is defined in System Resource Manager with a correct port number. The same port number is set in the TCP/IP client. In the Idle state, the TCP/IP server waits for a

command message from the PLC. Accordingly, the TCP/IP server sends a response message consisting of the result data in the Result state.

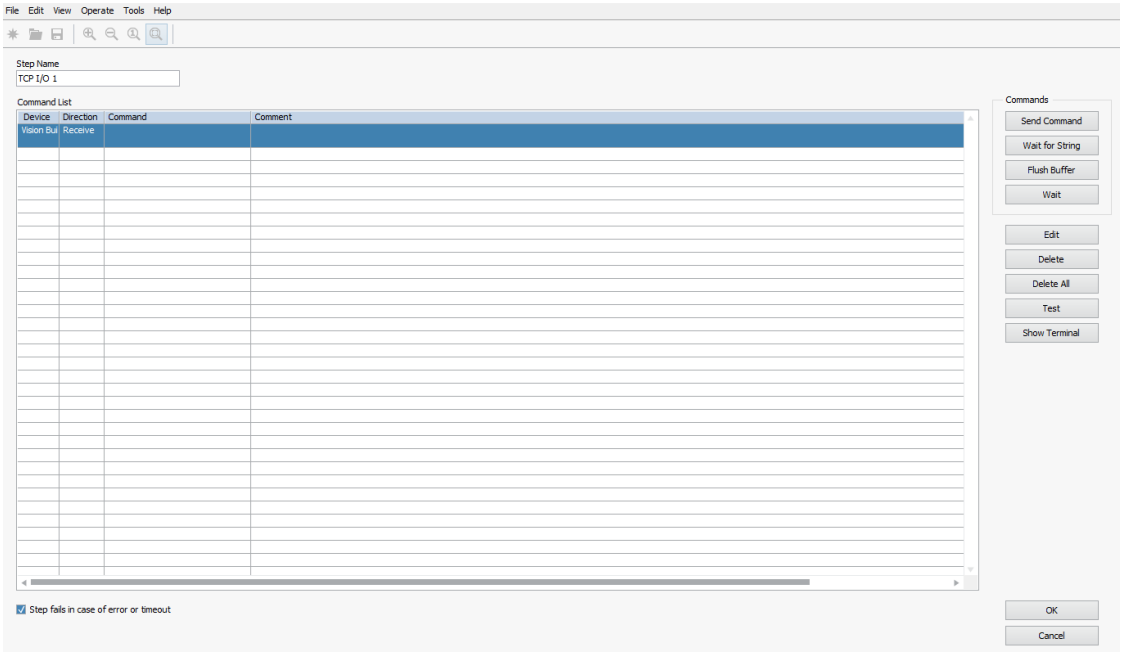


FIGURE 32. TCP I/O-function in NI Vision Builder AI.

8 TEST STATION

A test station was constructed for the thesis. The purpose of the test station was to demonstrate a real machine vision environment so that the performance of the ADS interface and the TCPIP client could be compared. The main components for the test station include a rack, camera, lens, illumination, and PLC. The test station also includes a power supply for the illumination and communication interfaces. The test station is shown in FIGURE 33.

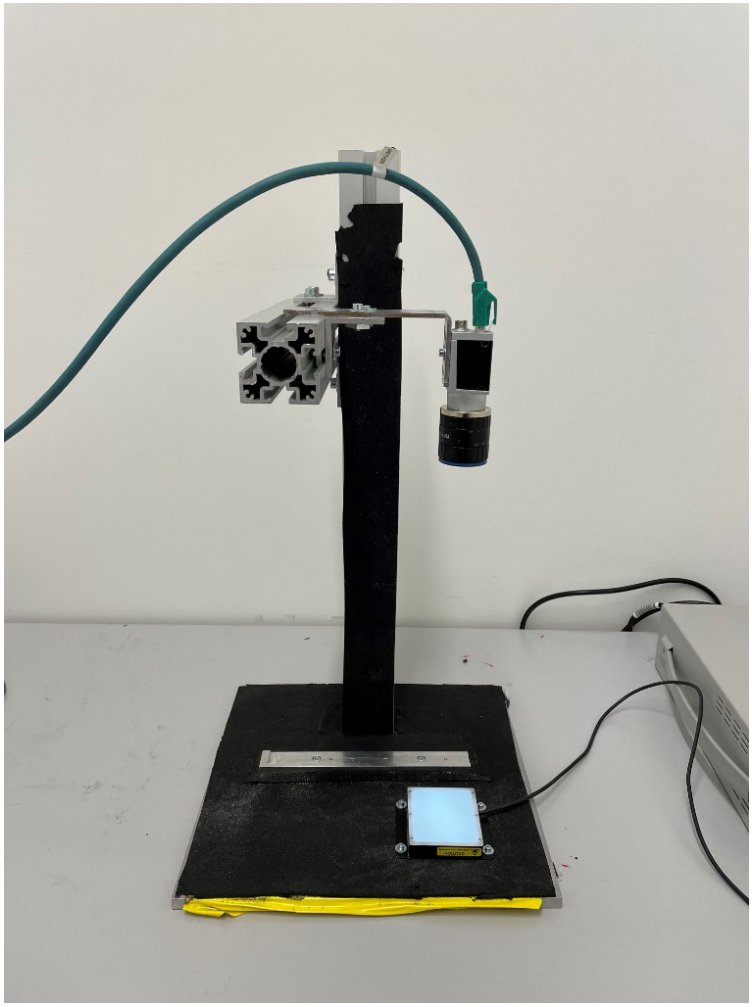


FIGURE 33. Test station.

Requirements for the test station were to have a rigid test setup with fixed positions for the camera and the object and with adjustable height for the camera. The rack for the test station was constructed from a metallic base plate and aluminum profiles. The camera for the test station needs to have a Gigabit ethernet connection interface and be fast enough for machine vision applications. Additionally, illumination with adjustable intensity control is required.

8.1 Camera

Basler acA2500-14gm GigE camera, shown in FIGURE 34, was chosen for the test station. It has a 5 MP resolution and 14 fps frame rate, and it uses a 5.7 mm x 4.3 mm monochrome CMOS sensor. Basler acA2500-14gm uses GigE vision for communication, which enables high-speed data transfer. The camera was connected to the PLC by a PoE (Power over Ethernet) adapter. The PoE adapter enables data transfer as well as provides power for the device through an Ethernet cable.

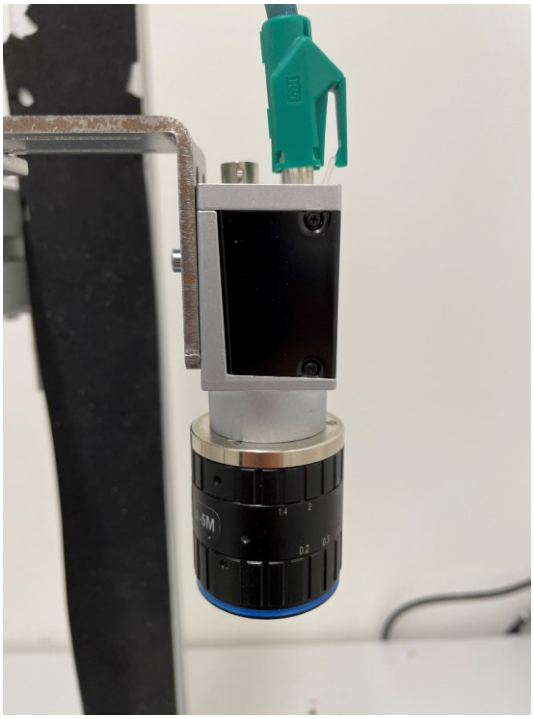


FIGURE 34. Basler acA2500-14gm.

8.2 Lens

OPT-AC2514-5M lens was chosen for the camera based on the requirements for performance and focal length. It is suitable for C-mount 5 MP cameras and has a 25 mm focal length, which is efficient for relatively short working distances. The aperture was set to about four, and the lens was precisely focused on the target.

8.3 Illumination

Different lighting solutions were considered for the test station. A light table solution was chosen as it proved to be the most efficient for this type of vision application. The light table is a backlighting

solution that illuminates the object from behind, displaying the shape of the object and making it easier to detect. Additionally, it is not too sensitive to interfering ambient light. The light table was mounted to the base plate with screws.

Adjustable intensity control was achieved by using a power supply with adjustable voltage control, Agilent U8001A, in this case. It is a single-output DC power supply with adjustable 0-30 V voltage control. By using adjustable intensity control, illumination can be set according to the vision application. The light table is shown in FIGURE 35.

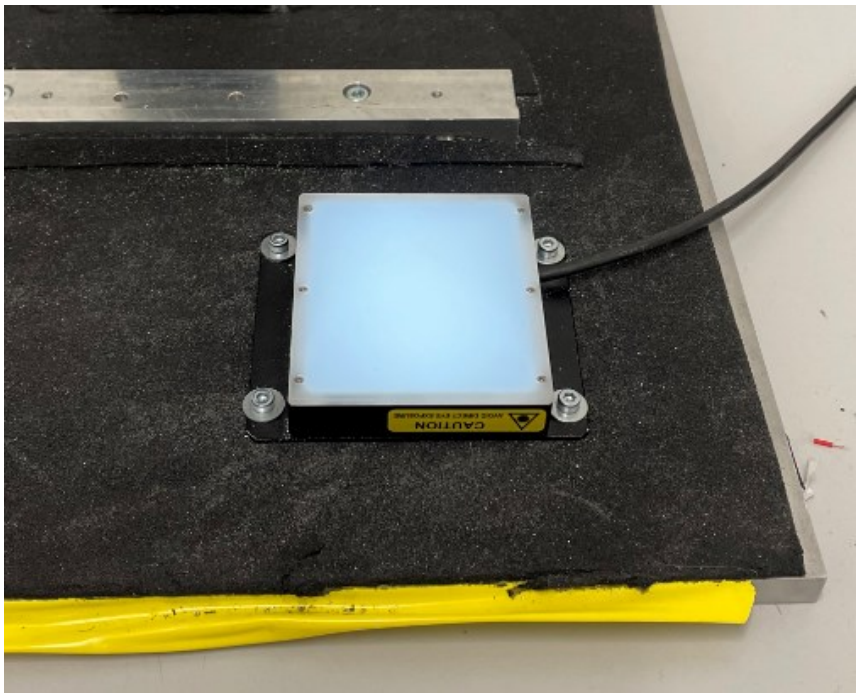


FIGURE 35. Light table.

8.4 PLC

Beckhoff CX2020, shown in FIGURE 36, was chosen as a PLC for this setup. The PLC triggers the machine vision application, receives a response message, and monitors start and end time values. It has a 1.4 GHz Intel® Celeron® CPU and 2 GB RAM. Beckhoff CX2020 has an option for a CFast memory card, two independent Gbit Ethernet interfaces, 4 USB 2.0 interfaces, and a DVI-I interface for a monitor.

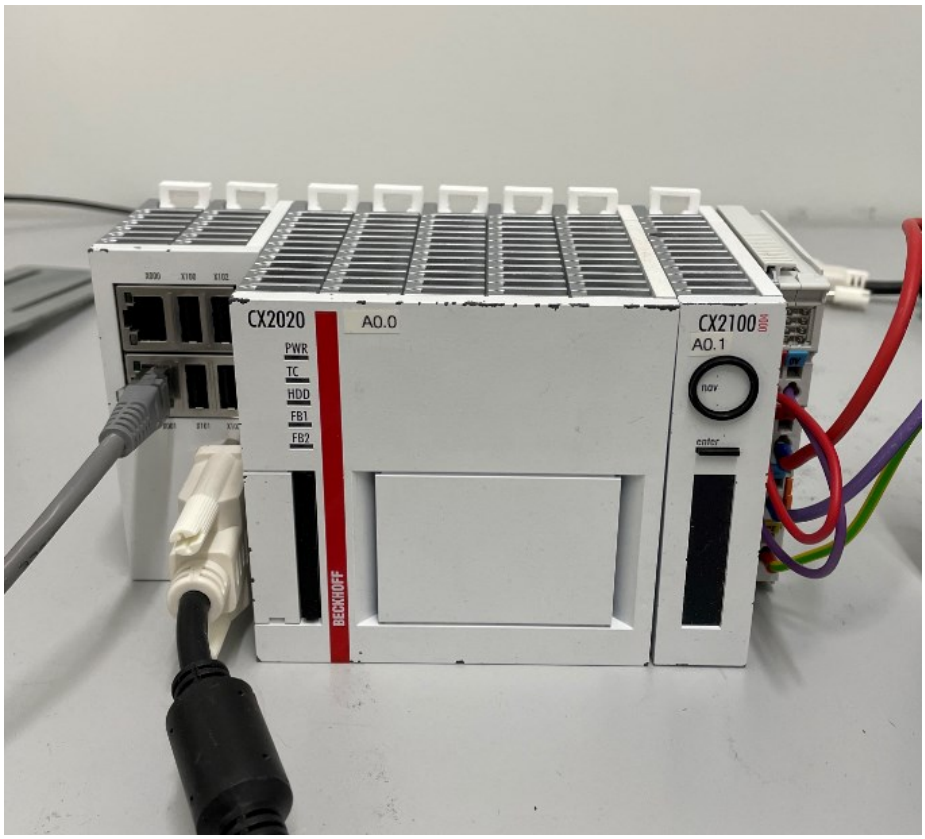


FIGURE 36. Beckhoff CX2020.

However, all the programming and testing of the communication interfaces were executed using an HP EliteBook laptop operating on a Windows platform. I/O tasks were assigned to an isolated core for TwinCAT to run in real time. The laptop has an 11th-generation Intel core processor and 16 GB of installed physical memory.

8.5 Camera Configuration

The camera was configurated using Pylon IP configurator software. Also, National Instruments camera drivers were installed. Finally, the proper functionality of the camera was confirmed using a live image in National Instruments NI Max software. FIGURE 37 illustrates configuring the camera in pylon IP configurator software.

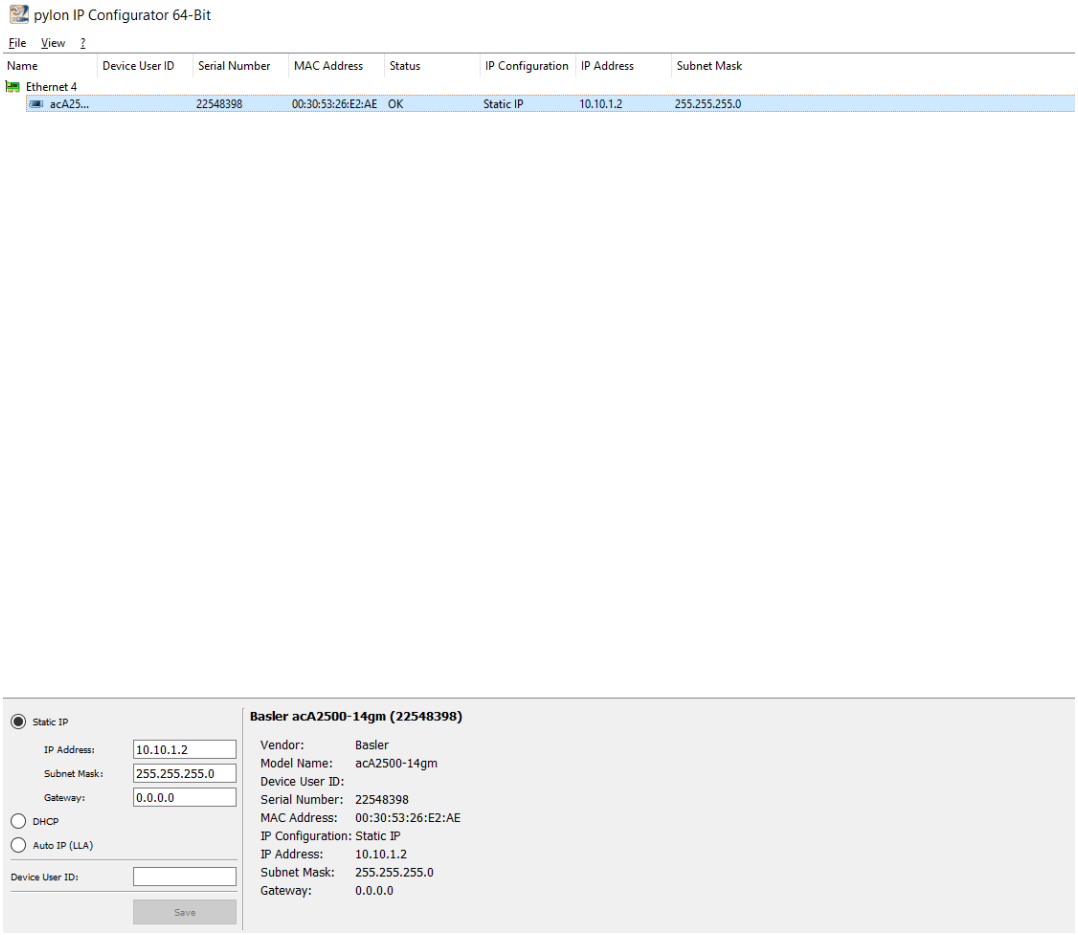


FIGURE 37. Configuring camera in pylon IP configurator software.

8.6 Camera Working Distance

Optimal camera working distance (WD) was calculated from the field of view (Y), focal length (f), and sensor size (X). The field of view is approximately the length of the light table (68 mm.) The focal length of the OPT-AC2514-5M lens is 25 mm, and the size of the CMOS sensor used is 5,7 mm. The equation for the camera height is shown in Equation 1.

$$WD = \frac{Yf}{X} = \frac{68\ mm * 25\ mm}{5{,}7\ mm} = 298{,}245\ mm \approx 30\ cm$$

EQUATION 1

8.7 Camera Calibration

For correct and accurate measurement, the camera needs to be calibrated. In the calibration, the pixel coordinate system of the camera is mapped into a real-world coordinate system. The camera is calibrated using a National Instruments Vision Builder AI software calibration sequence with a specific dot-patterned calibration grid. The distance between the dots of the calibration grid is 1mm. After the calibration is finished, the machine vision application can measure in real-world units. The calibration grid is shown in FIGURE 38.

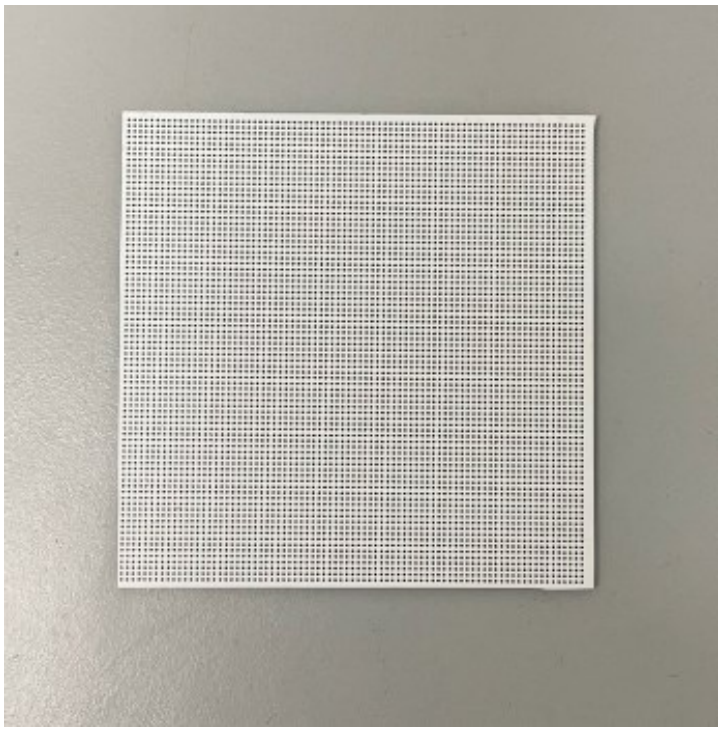


FIGURE 38. Calibration grid.

9 TEST RESULTS

In automation systems, data is transmitted continuously between different applications and devices. That is why the performance of the communication interface is crucial for the efficiency of the automation system. Latency in communication adds up and can increase cycle time significantly. The latency affects the productivity of the process and makes the automation system less effective. An unstable communication interface can also lead to errors and data loss. Thus, a fast and stable connection interface is required, especially in real-time control systems such as machine vision applications.

This chapter presents the results of test cases executed for both communication interfaces: the ADS interface and the TCP/IP client. The performance of the interfaces was tested in terms of speed and consistency of the connection. Command-response, continuous communication, and multiple commands test cases were executed for the interfaces, and jitter was calculated based on the results of those tests. Command and response messages were string data types for test purposes. Also, the usability of the interfaces was examined.

9.1 Command–Response Test

The communication speed of the ADS interface and the TCP/IP client was tested with the command–response test. The command–response test is a case of a single communication cycle between the PLC and the machine vision application. The PLC sends a command message and measures the time it takes to receive a response message from the vision application. The vision application used was identical in both cases, excluding the communication protocol. The content of the message is irrelevant since the connection's performance is being tested. This test was executed ten times to measure the average communication time for both interfaces. The results of the test are shown in Appendix 3 and Appendix 4.

9.2 Continuous Communication Test

Additionally, the ability to send commands continuously was tested. This test case is more similar to a real-time application where continuous communication between a PLC and a machine vision

application is required. In the continuous communication test case, the PLC sends a command message, and once it receives a response message from the vision application, a new command is sent. This set of command and response messages was executed ten times continuously as fast as possible. The continuous communication test was executed ten times to measure the average communication time for both interfaces. Start time is measured from sending the first command, and end time from receiving the last response. Results for the ADS interface are shown in Appendix 5.

For the TCP/IP client, a delay in the loop is required for a stable connection. Different delay values were tested for optimal performance of the interface. The delay was set to 200 milliseconds, and the test was executed ten times. In an actual application, the delay would need to be closer to 300 milliseconds to make sure no errors occur in the communication, and the connection will be stable. The results of the continuous communication test for the TCP/IP client are shown in Appendix 6.

9.3 Multiple Data Test

The previous tests discussed sending a single command message and receiving a response message from the machine vision application. The ability to send and receive multiple messages simultaneously is tested in multiple data test. The ADS interface achieves this by using the same read and write virtual instruments as in the previous tests. Since the output of the read virtual instrument is of array data type, the PLC can send multiple commands at once, which are then mapped individually in the vision application. The response messages from the vision application are received accordingly in array data type using the previously programmed write virtual instrument. Communication time in the multiple data test case was approximately the same as in the command–response test case.

Sending and receiving multiple data at once is also possible for the TCP/IP client. The PLC sends multiple data at once using the TCP/IP client. However, receiving and interpreting a response message from the vision application with multiple measurement data is more complex than with the ADS interface. First, the response message is formed in the vision application into a single string which is then sent to the PLC. The string is then parsed in the PLC application for each individual measurement data to be addressed. This method requires extra work in forming the response message and parsing it.

9.4 Vision Software Code Execution Time

Vision software code execution time is measured from the vision start time and end time variables acquired using the data logging function in NI Vision Builder AI. The vision software code execution time for the ADS interface vision application and the TCP/IP client vision application is shown in Appendix 7 and Appendix 8.

9.5 Jitter

Jitter affects the performance of the communication interface. Jitter in data transfer refers to the variation in the time delay between when a message is sent and when a response is received over the network. Jitter affects the connection negatively since it makes the communication interface act unpredictably. In an industrial application, an unstable data connection can lead to many issues, including unplanned process shutdown and data loss. (34.)

Jitter was calculated from the command–response test results. Jitter for the ADS interface is shown in Equation 2 and for the TCP/IP client in Equation 3.

$$\frac{(ADS_{T1} - AVG)^2 + (ADS_{T2} - AVG)^2 + \dots (ADS_{Tn} - AVG)^2}{n}$$

$$= \frac{(0{,}23\ s - 0{,}231\ s)^2 + (0{,}23\ s - 0{,}231\ s)^2 + \cdots (0{,}24\ s - 0{,}231\ s)^2}{10}$$

$$= 8{,}9 * 10^{-5}\ s\ \approx 0{,}09\ ms$$

EQUATION 2

$$\frac{(TCPIP_{T1} - AVG)^2 + (TCPIP_{T2} - AVG)^2 + \dots (TCPIP_{Tn} - AVG)^2}{n}$$

$$= \frac{(0{,}29\ s - 0{,}231\ s)^2 + (0{,}32\ s - 0{,}231\ s)^2 + \cdots (0{,}301\ s - 0{,}231\ s)^2}{10}$$

$$= 0{,}005485\ s \approx 5{,}5\ ms$$

EQUATION 3

9.6 Comparison

The ADS interface proved to be faster in all the test cases. In the command-response test, the average time for the ADS interface was 0,231 seconds, and for the TCP/IP client, the average time was 0,294 seconds. This result is not notable, but it can make a difference in automation systems where the speed of the communication interface is crucial for the process. The ADS interface was 21% faster in the command-response test, shown in Equation 4.

$$\left(1 - \frac{ADS}{TCPIP}\right) * 100 = \left(1 - \frac{0{,}231\ s}{0{,}2939\ s}\right) * 100 = 21{,}401\ \approx 21\%$$

EQUATION 4

Also, the jitter was measured from the results of the command-response test. Jitter for the ADS interface was 0,09 milliseconds, and for the TCP/IP client, 5,5 milliseconds, so there was also less jitter in ADS interface communication. However, the jitter was slight in both interfaces and can be seen as insignificant.

In the continuous communication test, the difference in the performance of the interfaces was more significant. This is due to the delay required in the TCP/IP client PLC application. The average execution time for the ADS interface was 3,483 seconds, while the average execution time for the TCP/IP client was 6,031 seconds. These extra seconds can make a notable difference in an industrial application requiring cycle time optimization. The difference between the ADS interface and the TCP/IP client in the continuous communication test case is calculated in Equation 5. The ADS interface proved to be 42% faster.

$$\left(1 - \frac{ADS}{TCPIP}\right) * 100 = \left(1 - \frac{3{,}4827\ s}{6{,}0309\ s}\right) * 100 = \ 42{,}252\ \approx 42\%$$

EQUATION 5

Test results of the vision software code execution time test were incoherent. The code execution time should be the same with both applications since they are identical besides the communication protocol. However, as seen in Appendix 7 and Appendix 8, there is some variation between the

interfaces. The average code execution time for the ADS interface was 0,119 seconds, and for the TCP/IP client, 0,179 seconds, so the TCP/IP vision software code execution time was slower than the ADS interface vision time.

9.7 Usability

The ADS interface does not have a user interface and is used by running the previously programmed read and write virtual instruments in the vision software. The communication is symbol-based, so the symbol list of the PLC application and the ADS interface must be compatible. New Variables can be added or removed by modifying the XML file used by the ADS interface.

The ADS interface is versatile as it supports various data types of variables. It supports LabVIEW data types, including Boolean, string, array, and integer. In addition, the LabVIEW front panel can be used for debugging purposes by using step status indicators. The ADS interface is simple to implement in applications where a Beckhoff PLC communicates with a machine vision application. It requires a modest amount of programming regarding PLC and machine vision software.

The TCP/IP client has a user interface, which can be used for monitoring communication. The user interface is well-designed and simple to use. However, there was a significant amount of lag during communication. The lag made the TCP/IP client challenging to use sometimes and required rebooting occasionally.

The TCP/IP client requires programming and configuring of the PLC, the machine vision application, and the client itself. PLC programming for the TCP/IP client was more complex and required a specific communication block and data unit types. A specific suffix in the command message was also needed for a successful data transfer. Vision application receives and sends messages by running a TCP/IP server and using a TCP IO function in the software. The TCP/IP client is less versatile since it supports only messages of string type. However, the string message can be parsed in the PLC application and converted into an integer or Boolean data type.

10 CONCLUSIONS

The thesis aimed to establish ADS communication between a Beckhoff PLC and a machine vision application using a LabVIEW interface. Another objective of the thesis was to make a comparison with the TCP/IP client. Several ADS libraries were examined. Communication was established using the TF3710 TwinCAT 3 Interface for the LabVIEW library. The performance of the interface was tested at the test station with a machine vision application. The Purpose of the machine vision application was to detect and examine hexagon nuts. Also, the performance of the TCP/IP client was tested.

Several performance test cases were executed for both interfaces. The ADS interface proved to perform better than the TCP/IP client in all test cases regarding the connection speed. Especially in the continuous communication test, the ADS interface proved to perform better. Additionally, the usability of both interfaces was addressed. The ADS interface proved to be simple to implement and use and more versatile regarding data types. It also required less PLC programming than the TCP/IP client.

Overall, the thesis was successful, and the goals of the thesis were achieved. The subject of the thesis was comprehensive, and the work for it required various skills, including PLC programming, establishing a communication interface, and vision software programming, as well as constructing and configuring a machine vision setup. Programming and configuring machine vision was a new challenge for me, requiring a lot of research. However, the subject proved to be interesting, and many new skills were acquired. In the future, these skills can be applied to software tasks concerning programming machine vision applications. Also, research about different communication protocols proved to be useful.

The schedule for the thesis was to finish it by the end of May, which was achieved. The work phase for the thesis was relatively straightforward and was completed in a few months. The guidance from my colleagues in JOT Automation during the work phase was beneficial and helped me reach my goal. The writing process took a little longer than expected and was sometimes challenging because I also worked full-time besides writing the thesis. Regardless, when looking back on this journey, I am satisfied with the result of the thesis and how it was executed.

10.1 Future Implementations

The established ADS interface is compatible with many applications since Beckhoff control devices and machine vision systems are commonly used in automation. It enables fast and reliable communication between a PLC and a machine vision system. Fast data transfer between devices decreases cycle time and makes automation systems more efficient. More importantly, based on the performance tests, the established ADS interface is also stable and highly reliable. Fewer disturbances reduce downtime and improve the efficiency of the system even further. Decreasing disturbances could save time and money in real-time control applications like machine vision applications.

Although, the TCP/IP client is currently used for communication with other devices in the automation system, it may be worth considering switching to the ADS interface. While implementing a new communication interface might not be practical, the benefits of using ADS could outweigh the costs. It is also noteworthy that it is possible to use both the ADS interface and the TCP/IP client simultaneously by using the ADS interface to communicate with a machine vision application and the TCP/IP client to communicate with other automation devices. Furthermore, it would be more reasonable to implement the ADS interface into new systems rather than replacing the TCP/IP client in existing systems. In any case, in machine vision applications where cycle time is crucial, the ADS interface is recommended for data transfer for its speed and stability.

Another method for achieving fast and reliable data transfer in similar applications would be using Beckhoff's TwinCAT vision software. Configuring and programming a machine vision application with the TwinCAT vision software was an alternative subject for the thesis. However, it was rejected since the original subject of the thesis proved comprehensive enough. The TwinCAT vision software integrates a machine vision system into the TwinCAT environment. Therefore, both PLC and machine vision functionality can be programmed within TwinCAT. Using the TwinCAT vision software, achieving even higher communication speed might be possible since no external vision software is involved. Even so, the performance of the software and how it compares to other communication interfaces would require more research. Also, the TwinCAT vision library would need to be studied further and compared to other vision software in the market to understand better the software and how it can be applied to machine vision applications.

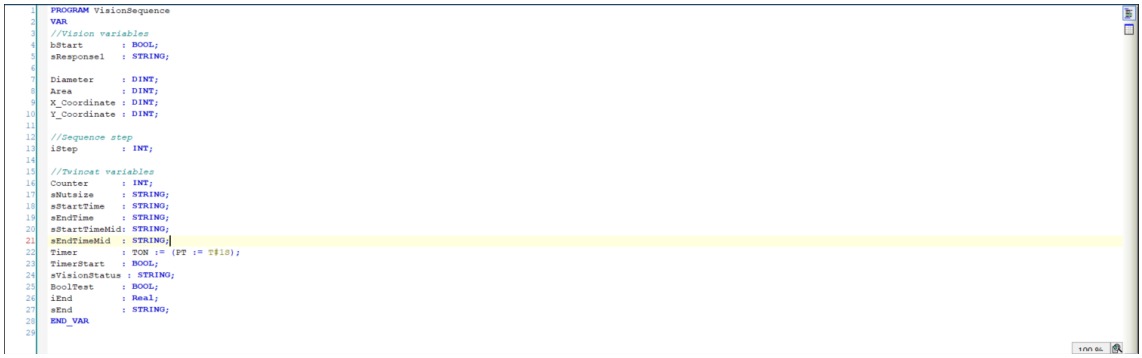
SOURCES

1. JOT Automation Ltd. home page. 2023. Date of retrieval 3.3.2023. https://www.jotautomation.com/about-us
2. JOT Automation Ltd. ODD-SHAPE XL ASSEMBLY CELL. 2023. Date of retrieval 22.3.2023. https://www.jotautomation.com/products/assembly/odd-shape-xl-assembly-cell
3. AI Multiple. Machine Vision in 2023: In-Depth Guide. 2023. Date of retrieval 3.3.2023. https://research.aimultiple.com/machine-vision/
4. Vision Systems Design. The history of machine vision. 2023. Date of retrieval 3.3.2023. https://www.vision-systems.com/knowledge-zone/article/14069209/the-history-of-machine-vision
5. SAAB RDS. 9 Benefits and Applications of Machine Vision Systems. 2020. Date of retrieval 3.3.2023. https://saabrds.com/9-benefits-and-applications-of-machine-vision-systems/
6. 4th Vector Technologies, LLC. Top 7 Contributors to a Robust Machine Vision System. 2022. 2023. Date of retrieval 22.3.2023. https://4thvectortech.com/tech-briefs/top-7-contributors-to-a-robust-machine-vision-system
7. National Instruments. A Practical Guide to Machine Vision Lighting. 2023. Date of retrieval 3.3.2023. https://www.ni.com/fi-fi/innovations/white-papers/12/a-practical-guide-to-machine-vision-lighting.html
8. Vision Systems Design. Understanding Focal Length and Field of View. 2023. Date of retrieval 3.3.2023. https://www.vision-systems.com/sponsored/edmund-optics/article/16752143/understanding-focal-length-and-field-of-view
9. Photography Life. What is Lens Distortion? 2023. Date of retrieval 3.3.2023. https://photographylife.com/what-is-distortion
10. Edmund Optics, Ltd. Types of Machine Vision Lenses. 2023. Date of retrieval 3.3.2023. https://www.edmundoptics.eu/knowledge-center/application-notes/imaging/imaging-lens-selection-guide/
11. PetaPixel. What is the Difference Between a CCD and CMOS Camera Sensor? 2021. Date of retrieval 3.3.2023. https://petapixel.com/what-is-ccd-cmos-sensor/
12. Quality Magazine. Smart Cameras: Yesterday, Today and Tomorrow. 2014. Date of retrieval 3.3.2023. https://www.qualitymag.com/articles/91857-smart-cameras-yesterday-today-and-tomorrow

13. Basler AG. From Gigabit Ethernet and GigE Vision to 5GigE. 2023. Date of retrieval 3.3.2023. https://www.baslerweb.com/en/vision-campus/interfaces-and-standards/gigabit-ethernet/
14. UKDiss.com. Colour vs Greyscale in Machine Vision. 2020. Date of retrieval 3.3.2023. https://ukdiss.com/examples/colour-vs-greyscale-in-machine-vision.php
15. Envira Gallery. How to Sharpen an Image in Photoshop. 2020. Date of retrieval 22.3.2023. https://enviragallery.com/how-to-sharpen-an-image-in-photoshop/
16. Beckhoff Information System. Region of Interest. 2023. Date of retrieval 3.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/tf7xxx_tc3_vision/7888560139.html&id
17. Medium.com. Blob Detection. 2019. Date of retrieval 3.3.2023. https://medium.com/image-processing-in-robotics/blob-detection-309226a3ea5b
18. Beckhoff Information System. Blob Detection with watchdog monitoring. 2023. Date of retrieval 22.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/tf7xxx_tc3_vision/4850677771.html&id=
19. Keyence Corporation. Character Inspection/OCR. 2023. Date of retrieval 3.3.2023. https://www.keyence.com/ss/products/vision/visionbasics/use/inspection05/
20. Fortinet, Inc. What is Transmission Control Protocol TCP/IP? 2022. Date of retrieval 3.3.2023. https://www.fortinet.com/resources/cyberglossary/tcp-ip
21. GeeksforGeeks. TCP/IP Model. 2023. Date of retrieval 3.3.2023. https://www.geeksforgeeks.org/tcp-ip-model/
22. Computer Networking Notes. TCP/IP Reference Model Explained. 2021. Date of retrieval 22.3.2023. https://www.computernetworkingnotes.com/ccna-study-guide/tcp-ip-reference-model-explained.html
23. Beckhoff Information System. ADS-Communication. 2023. Date of retrieval 3.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/cx8190_hw/5091854987.html&id
24. PLCCoder.com. Communicating between Beckhoff controllers part 2: ADS. 2020. Date of retrieval 3.3.2023. https://www.plccoder.com/communicating-between-beckhoff-controllers-part-2-ads/
25. Beckhoff Information System. ADS protocol. 2023. Date of retrieval 3.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/bc9xx0/2802214411.html&id
26. Beckhoff Information System. T_AmsPort. 2023. Date of retrieval 22.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_system/31064331.html&id=

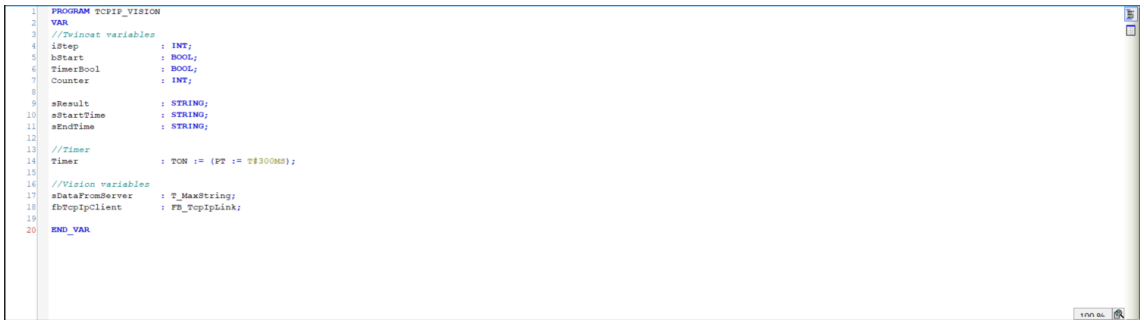
27. Beckhoff Information System. Client-server relationship. 2023. Date of retrieval 3.3.2023. https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_ads_intro/12902381579.html&id=2963750268128113040
28. Webinar. ADS-communication in TwinCAT: Connecting link for TwinCAT modules. Date of retrieval 3.3.2023. https://www.beckhoff.com/en-en/support/webinars/
29. LabVIEW. home page. 2023. Date of retrieval 3.3.2023. https://www.ni.com/fi-fi/shop/labview.html
30. Beckhoff manual. TF3710 TwinCAT 3 | Interface for LabVIEW™. 2022. Date of retrieval 3.3.2023.
31. Beckhoff Automation. Twincat Automation Software. 2023. Date of retrieval 3.3.2023. https://www.beckhoff.com/fi-fi/products/automation/twincat/
32. National Instruments. What Is Vision Builder for Automated Inspection? 2023. Date of retrieval 3.3.2023. https://www.ni.com/fi-fi/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-vision-builder-for-automated-inspection.html
33. National Instruments manual. NI Vision Builder for Automated Inspection Tutorial. 2012. 22.3.2023.
34. IR. Network Jitter - Common Causes and Best Solutions. 2023. Date of retrieval 3.3.2023. https://www.ir.com/guides/what-is-network-jitter

APPENDICES



```
PROGRAM VisionSequence
VAR
//Vision variables
bStart          : BOOL;
sResponse1      : STRING;

Diameter        : DINT;
Area            : DINT;
X_Coordinate    : DINT;
Y_Coordinate    : DINT;

//Sequence step
iStep           : INT;

//Twincat variables
Counter         : INT;
sNutsize        : STRING;
sStartTime      : STRING;
sEndTime        : STRING;
sStartTimeMid   : STRING;
sEndTimeMid     : STRING;
Timer           : TON := (PT := T#1S);
TimerStart      : BOOL;
sVisionStatus   : STRING;
BoolTest        : BOOL;
iEnd            : Real;
sEnd            : STRING;
END_VAR
```

APPENDIX 1. TwinCAT symbol list for ADS interface.



```
PROGRAM TCPIP_VISION
VAR
//Twincat variables
iStep               : INT;
bStart              : BOOL;
TimerBool           : BOOL;
Counter             : INT;

sResult             : STRING;
sStartTime          : STRING;
sEndTime            : STRING;

//Timer
Timer               : TON := (PT := T#300MS);

//Vision variables
sDataFromServer     : T_MaxString;
fbTcpIpClient       : FB_TcpIpLink;

END_VAR
```

APPENDIX 2. TwinCAT symbol list for TCP/IP interface.

ADS COM	Start	End	Result (S)
1.	34,226	34,456	0,23
2.	42,859	43,089	0,23
3.	5,312	5,312	0,23
4.	43,921	44,161	0,24
5.	33,388	33,618	0,23
6.	42,495	42,735	0,24
7.	14,451	14,691	0,24
8.	27,037	27,247	0,21
9.	50,262	50,482	0,22
10.	37,87	38,11	0,24
AVG			0,231

APPENDIX 3. Communication times for ADS interface in command – response test.

TCP/IP COM	Start	End	Result (S)
1.	41,656	41,946	0,29
2.	6,26	6,58	0,32
3.	51,248	51,467	0,219
4.	39,873	40,274	0,401
5.	22,522	22,921	0,399
6.	59,259	59,479	0,22
7.	42,854	43,084	0,23
8.	24,123	24,492	0,369
9.	0,45	0,64	0,19
10.	45,436	45,737	0,301
AVG			0,2939

APPENDIX 4. Communication times for TCP/IP client in command – response test.

ADS COM	Start	End	Result (S)
1.	30,532	34,143	3,611
2.	57,817	1,296	3,479
3.	51,451	54,84	3,389
4.	50,908	54,467	3,559
5.	56,703	0,073	3,37
6.	17,088	20,519	3,431
7.	2,526	6,505	3,979
8.	4,412	7,773	3,361
9.	51,02	54,328	3,308
10.	41,335	44,675	3,34
AVG			3,4827

APPENDIX 5. Communication times for ADS interface in continuous communication test.

TCP/IP COM	Start	End	Result (S)
1.	2,234	8,344	6,11
2.	13,768	19,808	6,04
3.	48,08	54,319	6,239
4.	38,926	44,916	5,99
5.	36,702	42,88	6,178
6.	31,006	36,867	5,861
7.	42,319	48,319	6
8.	32,325	38,064	5,739
9.	25,138	31,269	6,131
10.	15,473	21,494	6,021
AVG			6,0309

APPENDIX 6. Communication times for TCP/IP client in continuous communication test.

Vision ADS	Start	End	Result (S)
1.	34,313	34,433	0,12
2.	42,94	43,059	0,119
3.	5,393	5,515	0,122
4.	44,018	44,138	0,12
5.	33,471	33,591	0,12
6.	42,595	42,706	0,111
7.	14,536	14,658	0,122
8.	27,1	27,218	0,118
9.	50,337	50,455	0,118
10.	37,973	38,088	0,115
AVG			0,1185

APPENDIX 7. Vision software code execution time for the ADS interface vision application.

Vision TCP/IP	Start	End	Result (S)

1.	41,771	41,939	0,168
2.	6,327	6,574	0,247
3.	51,318	51,463	0,145
4.	40,115	40,27	0,155
5.	22,711	22,91	0,199
6.	59,307	59,475	0,168
7.	42,9	43,074	0,174
8.	24,314	24,486	0,172
9.	0,497	0,635	0,138
10.	45,504	45,729	0,225
AVG			0,1791

APPENDIX 8. Vision software code execution time for the TCP/IP client application.