

**M A S A R Y K
U N I V E R S I T Y**

FACULTY OF INFORMATICS

**Hardware Testbed for Cybersecurity
Education**

Master's Thesis

BC. ROBERT PRETSCHNER

Brno, Fall 2025

**M A S A R Y K
U N I V E R S I T Y**

FACULTY OF INFORMATICS

Hardware Testbed for Cybersecurity Education

Master's Thesis

BC. ROBERT PRETSCHNER

Advisor: Mgr. Ing. Jakub Čegan

Department of Computer Systems and Communications

Brno, Fall 2025



Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

In the course of preparing my thesis, I have used the following AI tools:

- Grammarly for grammar correction,
- Google Gemini to improve my writing and code style,
- Anthropic Claude Opus 4.5 for Readme documentation.

I declare that I have used these tools in accordance with the principles of academic integrity. I have reviewed the content generated and take full responsibility for it.

Bc. Robert Pretschner

Advisor: Mgr. Ing. Jakub Čegan

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Mgr. Ing. Jakub Čegan for his mentorship and expert guidance. I also wish to thank Mgr. Tomáš Sapák for his helpful discussions and technical insights. Finally, I would like to thank my partner and my family for their love, patience, and unwavering belief in me.

Abstract

The aim of this thesis is to develop a hardware testbed that simulates real-world operational technologies, which will later be used for cybersecurity education. The testbed utilizes open-source technologies, such as OpenPLC and Node-RED, to simulate SCADA systems and is constructed from inexpensive and readily available hardware, thereby minimizing operational costs. A cyberattack will be carried out on the testbed, leveraging insecure implementation of Modbus protocol. Most of the deployment will be automated using Ansible and then tested in a proof-of-concept topology on the CyberRange^{CZ} Platform.

Keywords

Cybersecurity, OT, OpenPLC, Raspberry Pi, Testbed, ICS, Wind energy, Hardware, SCADA, CyberRangeCZ Platform

Contents

Introduction	1
1 Introduction to Operational Technology (OT)	3
1.1 Industrial Control Systems and SCADA	3
1.2 Programmable Logic Controller	10
1.3 OpenPLC	13
2 Cyberattacks on ICS	15
2.1 Cyberattack fundamentals	15
2.2 Attack frameworks and methodologies	16
2.3 Threat landscape for wind energy infrastructure	19
2.4 Advanced Persistent Threats to the energy sector	23
3 Testbed design and architecture	25
3.1 Design requirements and criteria	25
3.2 System architecture overview	26
3.3 Hardware component selection	27
3.4 Electrical connections and wiring	30
3.5 Runtime environment deployment	33
3.6 Ladder diagram implementation	34
3.7 HMI development	40
3.8 Limitations and constraints	44
4 Integration with CyberRange^{CZ}	46
4.1 Sandbox definition	46
4.2 Hardware connection to the sandbox network	48
5 Cyberattack on OT testbed	50
5.1 Stage 1: Cyber intrusion and preparation	51
5.1.1 Initial access and lateral movement	51
5.1.2 Exploitation of HMI	51
5.2 Stage 2: ICS attack development and execution	52
5.2.1 Develop and Deliver	52
5.2.2 Execute ICS attack	54

6 Conclusion	60
6.1 Future work	62
A Archive content	63
Bibliography	64

List of Tables

3.1	GPIO pins allocation and OpenPLC addresses mapping from RPI perspective	32
-----	--	----

List of Figures

1.1	A diagram of generic SCADA architecture. [10]	5
1.2	Modbus TCP frame architecture. [13]	7
1.3	A high-level overview of wind farm SCADA architecture. [16]	9
1.4	A diagram of the PLC scan cycle. [21]	11
1.5	Basic components of the Ladder Diagram.	12
2.1	ICS Cyber Kill Chain roadmap. [36]	17
3.1	Chosen components for testbed, including a button, DC motor, Raspberry Pi 4B, LEDs, buzzer and L298N motor driver.	27
3.2	A wiring diagram of the testbed made in Fritzing using L298N community model. [59]	33
3.3	Monitoring tab in OpenPLC web server.	34
3.4	Rung for handling safety logic.	38
3.5	Fault logic rung in detail.	38
3.6	First prototype used for the Ladder Diagram testing.	39
3.7	HMI developed using Node-RED.	41
3.8	A flow that handles wind data logic for HMI.	42
3.9	A PLC healthcheck app showing status of coils and holding registers.	43
3.10	A final prototype of the wind farm.	45
4.1	A network topology diagram automatically generated in CyberRange ^{CZ} .	48
5.1	An exec node used to spawn a reverse shell.	52

Introduction

Cyberattacks on operational technology (OT) are increasing in both frequency and sophistication as industrial control systems become more interconnected and remotely accessible. A recent industry report from Dragos describes OT as a mainstream target and documents a sharp rise in ransomware targeting OT. [1]

A trend that is perhaps most starkly illustrated by the continued targeting of the Ukrainian power grid. Since 2015, Russian state-sponsored groups, for instance APT-40 (Sandworm), have repeatedly demonstrated the ability to penetrate utility networks and manipulate industrial control systems to cause blackouts, using Industroyer malware families. [2]

Parallel to this evolving threat landscape, the global energy transition is driving a shift where renewable generation plays more central role. Wind power in particular has become a key component of many national energy mixes. In the United States, wind energy now generates 10.3% of total electricity production. [3] In the Czech Republic, total electricity production in 2024 reached approximately 68.7 TWh, of which wind accounted for 0.7 TWh, which is around 1% of generation. [4] [5] The current installed wind capacity is approximately 0.3 GW, but according to the analysis by the Ministry of Industry and Trade of the Czech Republic, supportive policy measures could increase this to roughly 1.5 GW in the year 2030 and would be able to match the criteria from the European Union. [6] This anticipated expansion will significantly increase the number and size of wind assets connected to the grid, and with them, the reliance on networked OT and turbine control systems. Experience from the United States, where a much larger, geographically dispersed wind fleet is already tightly coupled to Supervisory Control and Data Acquisition (SCADA), remote monitoring, and automated control, shows that such growth in installed capacity is accompanied by a corresponding growth in the cyber attack surface of wind farms and their control infrastructure.

For the Czech Republic, this means that the planned increase of wind power is not only a technical and regulatory challenge, but also a cybersecurity challenge. As more turbines, substations, and meteorological systems are connected via industrial networks and

controlled through SCADA platforms, operators will face the same classes of threats that have already emerged in countries with more developed wind sectors.

To address these emerging risks, there is a need for practical OT cybersecurity education. As new wind projects are planned, built, and commissioned, operators, integrators, and vendors will need realistic environments in which they can rehearse the defense of their future control systems. A hardware testbed that resembles a small wind farm, built with affordable components and open technologies, can serve as such an environment. By utilizing open-source OT and SCADA software, such as OpenPLC and Node-RED, to emulate the control structures, communication patterns, and operator interfaces found in real wind farms, the testbed enables participants to experience both normal operation and cyber-physical attack scenarios in a controlled environment. Integrated into the CyberRange^{CZ} platform, it enables users to practice monitoring, incident response, and recovery on a realistic yet non-critical wind farm control system.

The thesis is structured as follows. The first chapter introduces the fundamentals of Operational Technology, with an emphasis on Industrial Control Systems, SCADA architectures, and Programmable Logic Controllers in the context of wind energy. The subsequent chapter surveys cyberattacks on ICS, including relevant threat actors, frameworks such as MITRE ATT&CK for ICS and the Industrial Control Cyber Kill Chain, and documented incidents affecting the wind sector. Chapter 3 presents the design and implementation of the hardware testbed, including component selection, wiring, PLC programming with OpenPLC, and HMI development in Node-RED. The following chapter describes the integration of the physical testbed with the CyberRange^{CZ} platform, focusing on network topology, sandbox provisioning, and VPN connectivity. Chapter 5 then develops and analyzes a proof-of-concept cyberattack against the testbed, mapping the adversary's actions to established ICS attack frameworks. The thesis concludes with a summary of the completed work, potential areas for improvement, and directions for future research.

1 Introduction to Operational Technology (OT)

Operational technology, as defined by the National Institute of Standards and Technology (NIST), "*encompasses a broad range of programmable systems or devices that interact with the physical environment or manage devices that interact with the physical environment.*" [7] This broad category encompasses the Industrial Control Systems (ICS) at the heart of these operations, including SCADA systems that provide high-level process supervision and PLCs that execute real-time, low-level control logic, forming the operational backbone of critical infrastructure.

Unlike traditional Information Technology (IT), which is data-centric and prioritizes the confidentiality of information, OT is physically oriented, placing the highest value on process availability and safety. This fundamental distinction means that while an IT system failure might lead to data loss, an OT system failure can result in physical damage, production shutdown, or even a threat to human life. Consequently, these systems have become increasingly frequent and high-value targets for specialized cybersecurity attacks. [1] The following section will focus on the fundamental terminology used in this work.

1.1 Industrial Control Systems and SCADA

To monitor and control OT systems, one of the key acronyms used throughout the operational technology field needs to be defined: SCADA. NIST defines this framework as a "*generic name for a computerized system that is capable of gathering and processing data and applying operational controls over long distances. Typical uses include power transmission and distribution and pipeline systems. SCADA was designed for the unique communication challenges (e.g., delays, data integrity) posed by the various media that must be used, such as phone lines, microwave, and satellite. Usually shared rather than dedicated.*" [8] SCADA systems are used in various industries and applications, including public transport, water and sewage systems, oil and gas industries, and power generation and distribution, which will be the main focus for this thesis.

SCADA architecture and components

A generic SCADA system consists of five hierarchical levels. These levels do not have a unified naming convention, and the naming may differ across the field. To use the most standardized naming, I refer to the PERA¹ architecture model, which lays the foundation for industry standards such as ISA-95 [9]. The following part introduces each layer.

Level 4 - Business logistics systems: Business operations and enterprise-level planning occur at this level, where ERP (Enterprise Resource Planning) systems are used for business and financial decision-making. In SCADA contexts, this level hosts corporate databases that aggregate operational data for long-term analysis, business intelligence reporting, and strategic planning. These systems receive summarized production data from lower levels but do not directly interact with control processes.

Level 3 - Manufacturing operations systems: Those systems manage manufacturing operations at this level, handling production scheduling, order processing, and coordinating activities across multiple systems. The SCADA Master Terminal Unit (MTU) typically resides here, serving as the central supervisory station that collects data from distributed field sites, processes alarms, maintains historical data-bases, and presents information to operators through centralized dashboards.

Level 2 - Control systems: The supervision of physical processes occurs at this level through SCADA software, Human-Machine Interfaces (HMIs), Distributed Control Systems (DCS), and supervisory control systems. The HMI provides the graphical interface that allows operators to visualize process states, acknowledge alarms, and issue control commands. SCADA servers at this level perform data logging, trend analysis, and alarm management, while communicating with field controllers through various industrial protocols.

Level 1 - Intelligent devices: Data collection and manipulation of the physical processes happen through sensors, smart devices, actuators, and field devices at this level. Remote Terminal Units

1. Purdue Enterprise Reference Architecture

(RTUs) and Programmable Logic Controllers (PLCs) serve as local controllers, executing real-time control logic, interfacing directly with field instrumentation, and communicating process data to supervisory systems. These devices translate high-level commands from Level 2 into specific actions at the field level.

Level 0 - Physical process: The physical processes include machinery and assets such as motors, pumps, sensors, valves, and actuators. In wind energy applications, this encompasses the turbine itself, including rotor blades, gearbox, generator, pitch control mechanisms, yaw systems, and environmental sensors such as anemometers, wind vanes, or temperature sensors. These components generate the raw data that flows upward through the SCADA hierarchy and respond to control signals flowing downward.

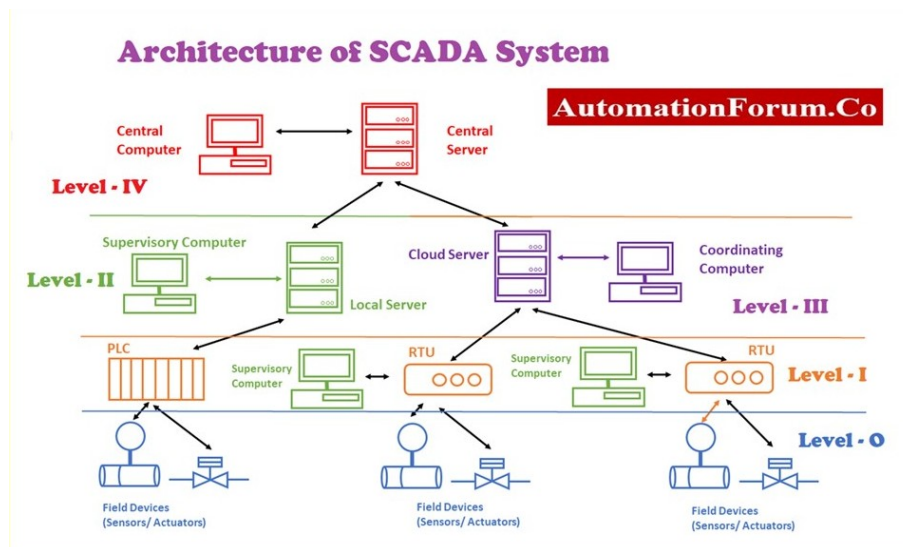


Figure 1.1: A diagram of generic SCADA architecture. [10]

SCADA communication protocols

Effective communication between the supervisory level and field devices relies on robust industrial protocols. While the market features various proprietary protocols, such as Siemens S7 Communication and Mitsubishi MELSEC Communication, this research focuses on open standards. [11] Among open protocols (e.g., DNP3, Ethernet/IP), Modbus TCP/IP is the industry standard for interoperability and was selected for this thesis due to its transparent data structure and lack of built-in security mechanisms², Modbus TCP provides an ideal environment for analyzing protocol-based attacks and network vulnerabilities.

Modbus TCP/IP Protocol Architecture

Modbus is a request-response protocol implemented at the Application Layer (Layer 7) of the OSI model. The protocol encapsulates Modbus frames within TCP/IP packets for transmission over Ethernet networks.

The Modbus TCP frame, known as the Application Data Unit (ADU), consists of the Modbus Application Protocol (MBAP) Header and the Protocol Data Unit (PDU). [12] The MBAP header (7 bytes) handles routing and transaction identification:

Transaction ID (2 Bytes): Used for transaction pairing. The Modbus server copies this ID from the request to the response, allowing the client to match replies to commands.

Protocol ID (2 Bytes): Always set to 0x0000 for Modbus TCP.

Length (2 Bytes): Indicates the number of following bytes (including the Unit ID and the PDU).

Unit ID (1 Byte): Identifies the remote server or specific slave device connected via a gateway. In pure TCP/IP networks, this is often set to 0xFF or 0x01.

2. This stands for the original Modbus protocol. The Modbus Organization has developed the Modbus Security Protocol, which has built-in Transport Layer Security (TLS). [12]

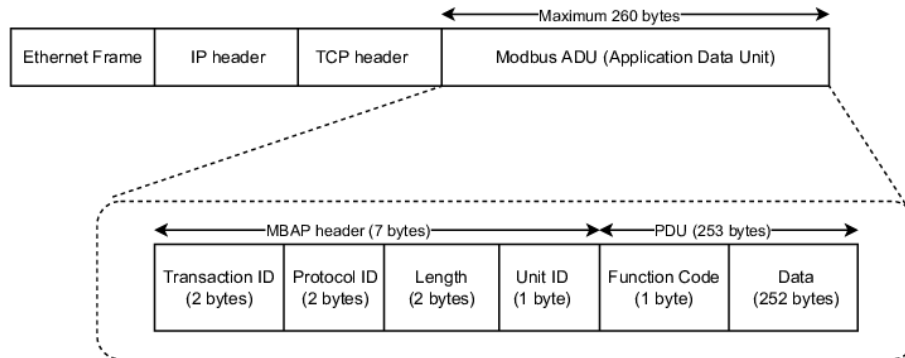


Figure 1.2: Modbus TCP frame architecture. [13]

Function Codes (FC)

Inside the PDU, the Function Code (byte 7 of TCP payload) dictates the action the server must perform. The most common codes used in SCADA attacks and monitoring are as follows.

FC 01 – Read Coils: Reads the status of binary outputs (ON/OFF).

FC 02 – Read Discrete Inputs: Reads the status of physical binary inputs (Sensors).

FC 03 – Read Holding Registers: Reads the content of analog output registers.

FC 04 – Read Input Registers: Reads the content of analog input registers.

FC 05 – Write Single Coil: Writes a single binary value to an actuator.

FC 06 – Write Single Register: Writes a specific integer value to one register.

FC 15 – Write Multiple Coils: Forces a block of coils to a specific sequence.

FC 16 – Write Multiple Registers: Writes values to a block of contiguous registers. [12]

Subsequently, there is a data field that contains data required by the specific FC. From the Modbus request perspective, the first 2 bytes³ contains the information about the starting address for the FC, followed by the 2 bytes specifying the quantity of inputs. From the Modbus response perspective, the first byte after FC specifies the number of bytes reflecting the quantity of inputs from the request. Then, the mentioned bytes with the status are sent. [12]

The Modbus data model

Modbus subsequently organizes data into four distinct object types based on their bit width and access permissions. These will be utilized in the PLC programming section later on.

Discrete inputs (read-only, 1 bit): Represents physical hardware status, such as limit switches or proximity sensors.

Coils (read-write, 1 bit): Represents binary actuators, such as relays, LEDs, or motor starters.

Input registers (read-only, 16 bit): Used for analog measurements like temperature sensors, wind speed, or pressure readings.

Holding registers (read-write, 16 bit): Used for adjustable parameters, such as PID setpoints or alarm thresholds. [12]

SCADA in Wind Energy Context

To apply general SCADA theory to the wind energy sector, it is necessary to define the primary components of a wind farm. A standard wind farm consists of a complex of Wind Turbine Generators (WTGs) and a substation, which is a facility responsible for converting generated electricity from high to low voltage (or vice versa) for grid transmission. [14, 15] Supporting these core components are meteorological stations (Met Masts) used for wind forecasting, all of which are interconnected via a dedicated communication network. These components map directly to the physical process layer (Level 0) and

3. Modbus uses Big-Endian data representation for 16-bit values.

intelligent device layer (Level 1) of the SCADA architecture defined previously.

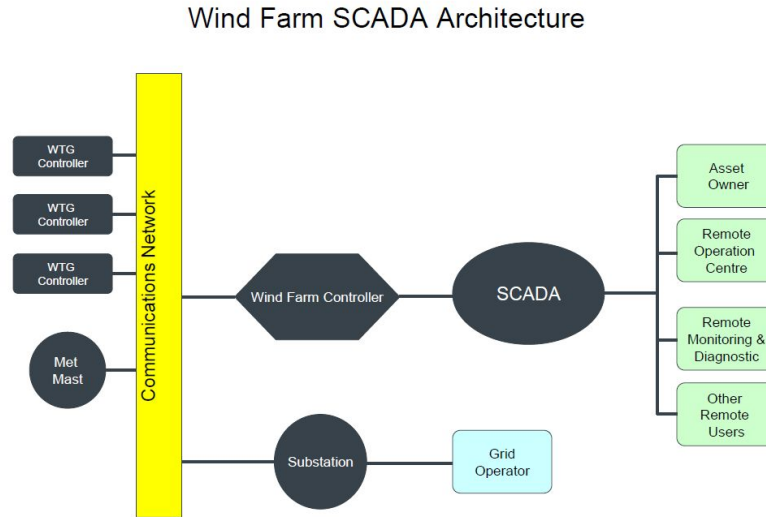


Figure 1.3: A high-level overview of wind farm SCADA architecture. [16]

Functional requirements of wind farm SCADA

To define the specific functional requirements for this domain, this thesis references the standards outlined by Miceli. [16] The fundamental objective of a wind farm SCADA system is the unified integration of data from WTGs, substations, and meteorological towers into a single system. This architecture must ensure uninterrupted data access for both local operators, typically stationed within the substation building, and remote control centers. Crucially, the system must handle diverse communication protocols to ensure compatibility between disparate hardware components.

Regarding data visualization, Miceli notes that the system must monitor critical parameters, including turbine status (working, paused, stopped), electrical output (power, voltage, current), mechanical metrics (rotor/generator speed), and environmental data (wind speed

and direction). [16] Beyond passive monitoring, the system requires active control capabilities, enabling operators to modify control parameters, initiate turbine start-up and shutdown sequences, and manage substation switching. Finally, a strict user hierarchy is mandated to define access levels, ensuring the physical safety of personnel working on the machinery by preventing unauthorized remote activation during maintenance.

1.2 Programmable Logic Controller

To realize the monitoring and control requirements described in the previous section, specifically, the real-time management of WTG status and electrical parameters, industrial hardware is required at the field level. While SCADA provides the high-level supervisory view, the actual execution of control logic and data acquisition occurs locally at the turbine. This role is fulfilled by the Programmable Logic Controller (PLC), which is defined as *"an industrial-grade computer that is capable of being programmed to perform control functions."* [17] This PLC then serves as the interface between the physical machinery of the wind farm and the digital network of the SCADA system.

Basic architecture

The hardware architecture of a standard PLC consists of three main components: the Central Processing Unit (CPU), memory, and Input/Output (I/O) modules. The CPU acts as the brain, executing control instructions to direct the computer's operation. Memory stores the operating system, user programs, and data. The I/O modules provide the physical interface to the outside world, converting high-voltage signals from sensors into digital logic levels for the CPU and vice versa for actuators. [18]

Scan cycle operation

Unlike general-purpose computers that are event-driven, PLCs operate on a cyclic execution model known as the Scan Cycle. This cycle is deterministic and consists of four distinct phases. Starting with reading inputs, the PLC takes a snapshot of all input devices and checks if the

voltage is present. Based on the presence, the CPU assigns a binary value of either 1 or 0 and stores the result in an input image table. [19]

After the initial phase is complete, the CPU proceeds to execute the user-defined logic in the PLC program instruction by instruction, typically from the first rung to the last. During this phase, the CPU references the data in the input image table and, as it solves the logic, outputs its results to an output image table. [19]

Immediately following program execution, the values stored in the image output table are transferred to the physical output modules. These are either energized or de-energized to match the logic results, thereby physically switching the connected actuators, motors, or indicators. This ensures that all field devices change state simultaneously rather than sequentially during logic solving. [19]

In the final phase, also known as the housekeeping phase, the CPU performs checks on the integrity of memory and hardware. [19] Additionally, this is the time when the PLC handles communication requests, such as Modbus TCP queries from SCADA HMI, or updates network variables before the cycle restarts.

Whole cycle typically runs every 10-20 milliseconds. If a change occurs faster than the time window of the scan cycle, the system experiences the input/output response lag phenomenon, which may lead to, for example, inaccurate sensor readings. [20]

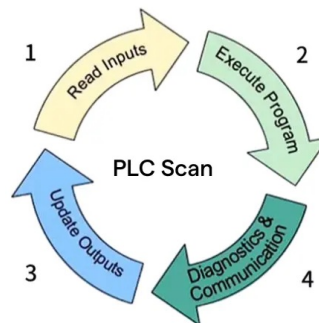


Figure 1.4: A diagram of the PLC scan cycle. [21]

PLC programming

The PLC programming can be done in one of the languages supported by the IEC⁴ 61131-3 standard. Those languages are Ladder Diagram (LD), Function Block Diagram (FBD), Structured Text (ST), Instruction List (IL), and Sequential Function Chart (SFC). [22] The language of focus for this thesis will be LD⁵, a graphical programming language with standardized symbols that recall relay circuit logic, which is not done without cause. Those similarities help to overcome the differences between electrical engineers and programmers.

To describe a general Ladder diagram, it would consist of five components. Starting with power rails, these components represent the power and ground connections. In a well-formatted LD, they should be from the start of the diagram to the bottom of the page. The following component is rung. Rungs are the wires that connect the left side of the power rail to the right side of the power rail. Between those rails, we have another category of components that are connected by rungs, and those are inputs⁶ and outputs⁷. Inputs are represented by contact symbols, whereas outputs are represented by coils. Subsequently, there are blocks. They are categorized into many types, but primarily allow you to perform operations such as numerical, arithmetic, time, bit-shift, bitwise, and counting. Lastly, we have comments that are used to describe more closely what each rung is doing.

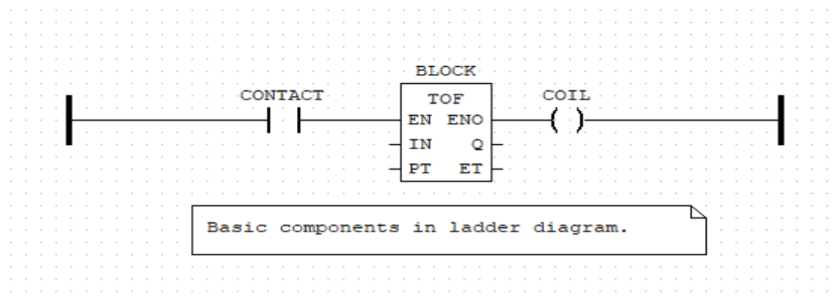


Figure 1.5: Basic components of the Ladder Diagram.

4. International Electrotechnical Commission
5. Sometimes also called Ladder Logic.
6. External control actions, such as a button press . [23]
7. For example, a device that can be turned on or off, such as an LED or motor.

1.3 OpenPLC

Proprietary PLC platforms dominate the industrial automation market, often with their closed, proprietary ecosystems. [24] These systems may limit user access to internal logic, source code, or proprietary hardware, which can restrict flexibility and increase vendor dependency.

OpenPLC represents an open-source implementation of PLC functionality, providing runtime and editor components for industrial control applications. Developed as an alternative to proprietary commercial PLC platforms, OpenPLC aims to democratize access to industrial automation technology while providing a transparent platform for education, research, and security analysis. [25]

The platform adheres to the IEC 61131-3 standard for PLC programming languages, ensuring compatibility with established industrial automation practices. [22] This standard's compliance enables users familiar with commercial PLC programming to transfer their knowledge directly to the OpenPLC environment, facilitating adoption in educational contexts and professional development scenarios.

Runtime and editor

The OpenPLC runtime executes IEC 61131-3 compliant programs and is designed to operate on various hardware platforms, primarily covering Windows, Linux, and MCUs⁸.

From a runtime perspective, the previously defined LD is later compiled into a structured text representation, which provides a higher-level, textual abstraction of the logic. [26] In OpenPLC, this ST code is then transpiled into C code using the MatIEC compiler⁹

Communication capabilities of the OpenPLC runtime include support for Modbus TCP/RTU and other standardized protocols, enabling integration with standard industrial equipment and SCADA systems [22]. Additional protocols can be implemented through the extensible architecture, allowing customization for specific application requirements. The runtime also provides web-based monitoring and config-

8. Microcontroller, also known as a Microcontroller unit.

9. OpenPLC uses the MatIEC compiler to convert Structured Text to C code for efficient execution on the target controller, which runs on the PLC's CPU. [27] [28]

uration interfaces, facilitating remote management and diagnostics [22].

The OpenPLC project features its own proprietary editor, which serves as an IEC 61131-3 compliant development environment. Within this editor, it is also possible to upload your PLC programs directly into the microcontroller via the serial COM port. All projects created in the editor contain a main XML configuration file that adheres to the PLCopen TC6-XML standard, facilitating interoperability across platforms. [29]

2 Cyberattacks on ICS

In this chapter, I will introduce the concept of cyberattacks on Industrial Control Systems (ICS), specifically in the domain of wind farms, and key terms associated with them. I will also introduce attack frameworks and methodologies that will provide a more detailed description of the attacker's perspective.

2.1 Cyberattack fundamentals

Understanding the security posture of an ICS requires a clear definition of the actors involved, the avenues they exploit, and the methods used to simulate these events.

Threat landscape: *"The threat landscape in Computer Science refers to the array of cyber threats, vulnerabilities, and attack vectors that impact computing systems, networks, and data. Its scope and sophistication continually expand due to technological advancements and the proliferation of connected devices."* [30]

Threat actors: Threat actors are entities that attempt to exploit vulnerabilities to compromise systems. In the context of the SCADA threat landscape, actors are typically categorized by their capability and motivation. Generally, we have three types of motivation: political, personal, and war/peacekeeping, which are dedicated to nation-states. Within politically motivated are also categorized nation-states, continuing with terrorist groups and hacktivists. The last two, which are politically motivated, overlap with personal motivations (as well as those of hackers with various skills and insiders). [31]

Attack surface: The attack surface represents the sum of all exposed points in a system that are accessible to a threat actor. [32] This could be represented by remote maintenance ports, HMI web interfaces, or wireless sensors.

Attack vector: The attack vector is the specific path or method used by the adversary to breach an attack surface. Common vectors in

OT environments include spear-phishing emails, compromised supply chain software, and exposed internet-facing ICS devices, which are among the most exploited attack vectors in 2024. [1]

Cyber Defense Exercise (CDX): A CDX is a simulated scenario designed to train personnel and test the organization's ability to put up resistance against cyber threats. [33] In a CDX, contestants and organizers are assigned to teams. Typically, the contestants of a CDX are members of a Blue team, who are the defenders of a fictitious organization. Against them stands the Red team, which resembles cyber attackers and typically performs a form of attack campaign. Subsequently, the CDX attends White team, which organizes the exercise, followed by the Green team responsible for the virtualized and physical infrastructure, and the Yellow team, which works as the connection between the teams and informs them about the progress and status of the exercise. [33]

While this term is not inherently related to the cyberattack fundamentals, it is a key term that will be used throughout this thesis. In OT research, a CDX can typically utilize a testbed to safely replicate attack vectors without risking actual physical infrastructure.

2.2 Attack frameworks and methodologies

To systematically analyze attacks against wind energy infrastructure, standardized frameworks and methodologies are required to map adversary behaviors.

MITRE ATT&CK for ICS

The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) for ICS is a knowledge base that describes the actions an adversary may take while operating within an industrial network. Unlike the enterprise version, which focuses on data confidentiality and integrity, the ICS matrix emphasizes impact on the physical process [34]. The framework is structured into 12 tactics, each of which houses

multiple techniques. This framework will be used to categorize the chosen attack on a testbed in Chapter 5.

The Industrial Control Cyber Kill Chain

The Cyber Kill Chain, originally developed by Lockheed Martin and now adopted by SANS Institute¹, outlines the sequential stages of a cyberattack. This framework divides the ICS attack into a two-stage process. [35]

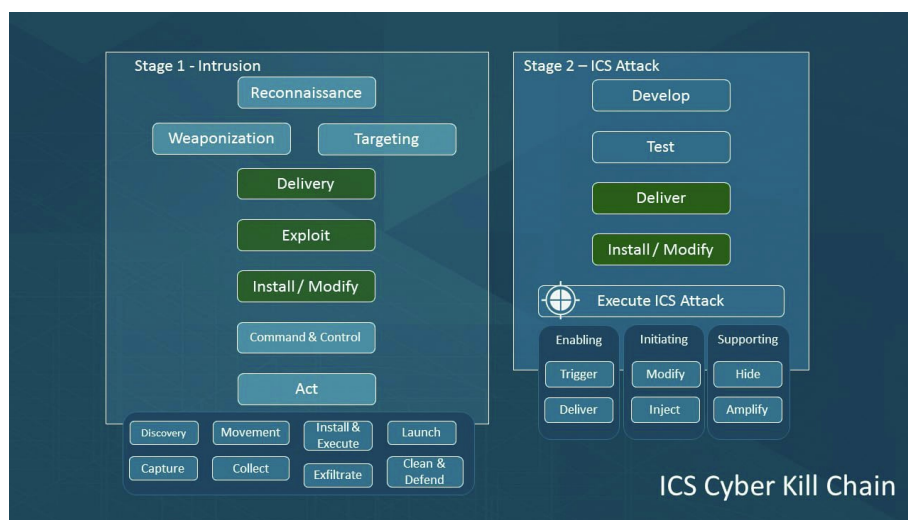


Figure 2.1: ICS Cyber Kill Chain roadmap. [36]

Stage 1: Cyber intrusion preparation and execution

The first stage mirrors a traditional IT attack. The adversary's goal is to gain a foothold in the corporate or control network to conduct espionage or prepare for the next stage.

Reconnaissance: The adversary performs reconnaissance, which is a process that identifies targets and gathers intelligence. In wind energy, this might involve identifying specific turbine

1. SysAdmin, Audit, Network, Security

manufacturers, SCADA protocols, and internet-facing devices via search engines like Shodan.

Weaponization and delivery: At this phase, the attacker creates a payload coupled with an exploit—such as a malicious firmware update or a phishing email containing malware—and transmits it to the target environment.

Exploitation: The malicious code exploits a vulnerability (e.g., an unpatched HMI or weak credentials) to gain unauthorized access to the target system.

Installation: The adversary installs software such as a backdoor or Remote Access Trojan (RAT) with the idea of maintaining persistence.

Command and Conquer (C2): The compromised system establishes a connection back to the attacker's owned machine. In OT environments, attackers may pivot from the IT network to the OT network at this stage.

Act: At the last phase of Stage 1, the adversary achieves their Stage 1 goal. In an OT attack, this goal is not the final impact; rather, internal reconnaissance and lateral movement are required to bridge the gap into the critical ICS network. [35]

Stage 2: ICS attack development and execution

Once established in the OT network, the attacker must execute a new kill chain specific to the industrial process.

Develop: The adversary analyzes the specific engineering logic of the victim system. They must develop a capability to modify the process (e.g., a script to write specific values to Modbus holding registers) without triggering safety shutdowns.

Test: The adversary validates the capability of the developed solution. Sophisticated actors, such as those behind Industroyer, [37] often test their payloads on identical hardware or simulations to ensure the attack will succeed.

Deliver: The ICS-specific weapon is transferred to the target control device, such as the engineering workstation or the PLC itself.

Install/Modify: The adversary modifies the controller logic, firmware, or runtime parameters. This could involve overwriting a PLC program or injecting FC commands.

Execute ICS Attack: The attack is launched, resulting in a physical consequence. For a wind turbine, this corresponds to the *Actions on Objectives* resulting in equipment damage, safety bypass, or cessation of power generation. [35]

2.3 Threat landscape for wind energy infrastructure

The threat landscape facing wind energy infrastructure is multifaceted, driven by the increasing interconnectedness of modern power generation systems. According to Staggs et al., the attack vectors leveraged against wind farm facilities can be broadly categorized into two primary domains: physical attacks and cyberattacks. [15]

Physical attack vectors

Physical attack vectors exploit the geographical distribution and isolation inherent to wind farm deployments. Turbines are frequently situated in remote environments—such as agricultural fields, deserts, or offshore locations—where continuous surveillance is difficult and response times to security incidents are prolonged. Staggs et al. highlight that while substations often employ robust perimeter defenses, such as fencing and surveillance systems, individual wind turbines often rely on rudimentary physical controls, including standard padlocks, to secure the tower base. [15]

The critical risk of a physical breach extends beyond simple vandalism or theft; it serves as a gateway to the OT network. An adversary with physical access to a turbine tower or unmanned substation can interact directly with the internal control network. This proximity allows for the implantation of rogue hardware, such as single-board computers (e.g., Raspberry Pi) or network taps, directly into network

switches. These devices can bridge air-gapped systems, bypass perimeter firewalls, and facilitate Man-in-the-Middle (MitM) attacks against the internal SCADA traffic. [15]

These types of attacks will not be further covered in this thesis.

Cyberattack vectors

While physical vectors require proximity, cyberattack vectors leverage digital connectivity to compromise control systems remotely. These vectors exploit vulnerabilities in the communication protocols and network architecture to manipulate the operation of the wind farm. [35]

I have conducted research into incidents over the past decade, which reveals distinct categories of cyber threats facing the wind energy sector. These range from financially motivated extortion that indirectly impacts operations to direct tampering with control logic. These incidents will be further used as an inspiration for the proof-of-concept attack against the hardware testbed. The following sections will present real-world examples of breaches.

Ransomware campaigns

Ransomware is primarily an IT-centric threat, yet it represents the most frequent cause of operational disruption in the wind sector to date. The common pattern in these attacks is not the direct encryption of PLCs, but rather the loss of supervisory control over them. When IT networks are compromised, operators often perform a precautionary shutdown of OT bridges to prevent lateral movement. This results in a loss of visibility (SCADA blinding), forcing manual operation or temporary termination of power generation.

Vestas (2021): A major Danish wind turbine manufacturer, Vestas, suffered a ransomware attack that compromised its internal file share systems, leading to a data leak. While the company stated that the OT systems controlling the turbines were not directly impacted, the incident forced a global shutdown of the IT infrastructure. This highlights the risk of IT-to-OT crossover, where the interdependency of systems means an IT breach can paralyze operational management. [38]

Nordex SE (2022): The Conti ransomware group², a prolific operator known for targeting industrial organizations, claimed responsibility for an attack against this German wind turbine manufacturer. The incident compelled Nordex to shut down its IT systems across multiple locations and disable remote access to the turbines it manages, severely disrupting business and operational functions. [40]

Deutsche Windtechnik (2022): A German wind turbine servicing company was targeted by a cyberattack that disrupted its IT systems. As a precautionary measure, the company switched off remote monitoring connections for over 2,000 wind turbines. While the turbines continued to operate automatically, the loss of real-time monitoring represents a significant safety risk. Intelligence suggests the threat actor may have been the Black Basta ransomware group, indicating a financially motivated target selection. [41, 42]

Denial-of-Service (DoS) attacks

Unlike ransomware, which disables systems as a side effect of extortion, disruptive attacks target the availability of the control system itself. These incidents demonstrate how fragility in communication dependencies, whether due to software bugs or reliance on satellites, can lead to a loss of operational control.

sPower (2019): In the year 2019, the USA experienced the first cybersecurity incident that caused a disruption in its power industry. It was a denial-of-service attack, exploited a known vulnerability in a Cisco firewall at a Utah-based wind and solar operator. The attack was conducted by an unknown attacker, which caused intermittent loss of communication between the company's control center and multiple generation sites. This case demonstrates that attack vectors may also include improper patch management for edge devices. [43]

2. A group, which is also a cyber threat actor in the Russian invasion of Ukraine. [39]

Enercon (2022): This incident serves as an example of collateral damage in cyber warfare. A cyberattack targeting Viasat's KA-SAT satellite network, intended to disrupt Ukrainian military communications at the onset of the Russian invasion, had a cascading effect on European infrastructure. The attack impacted the satellite modems used for the remote monitoring and control of 5,800 Enercon wind turbines in Germany. This incident demonstrates that wind farms rely on complex supply chains, such as those involving Internet Service Providers (ISPs) or satellites, which are themselves potential attack surfaces. [44]

SCADA and control system vulnerabilities

The most critical category of threats involves vulnerabilities that enable direct manipulation of the physical process, corresponding to Stage 2 of the ICS Kill Chain. Unlike IT-centric attacks that primarily affect data availability, vulnerabilities in the SCADA layer, such as authentication bypasses in web interfaces or weak input validation in HMIs, can grant adversaries unauthorized administrative control. Exploiting these flaws enables attackers to bypass safety logic, inject malicious commands, or alter operational setpoints, thereby directly compromising the physical integrity of the wind energy infrastructure.

Nordex (2014-2015): Multiple cross-site scripting (XSS) vulnerabilities (CVE-2014-5408 and CVE-2015-6477) were identified in the Nordex Control 2 (NC2) SCADA system. An attacker could exploit these flaws to execute arbitrary code within an operator's browser session, potentially stealing credentials or tricking the operator into executing malicious commands. [45]

XZERES (2015): A critical vulnerability (CVE-2015-0985) was discovered in the XZERES 442SR wind turbine's web-based interface. The flaw allowed a remote attacker to retrieve administrative usernames and passwords without authentication. This simple authentication bypass would have led to a complete system takeover, enabling an adversary to manipulate turbine operations or cause a shutdown, potentially resulting in a loss of power generation. Thankfully, there were no known exploits of this vulnerability. [46]

2.4 Advanced Persistent Threats to the energy sector

Advanced Persistent Threats (APTs) represent sophisticated, well-resourced adversaries that pursue prolonged network access rather than opportunistic, short-term strikes. According to the Cybersecurity and Infrastructure Security Agency (CISA), these actors conduct targeted campaigns with objectives ranging from espionage and data theft to the potential disruption or destruction of critical infrastructure systems. [47]

Within the specific context of this thesis, it is critical to note that APT groups have increasingly shifted focus toward OT. While actors such as APT44 (Sandworm) have demonstrated the capability to disrupt power grids through direct ICS manipulation, other groups, like APT40 (Leviathan), have been observed specifically targeting the supply chains and maintenance operations of wind energy infrastructure to obtain sensitive technical intelligence.

APT44 – Sandworm

APT44, widely known as Sandworm, is a Russian state-sponsored threat group attributed to the Main Centre for Special Technologies (GTsST) of the GRU. Recognized by intelligence agencies as the Kremlin's preeminent cyber sabotage unit, APT44 specializes in aggressive operations against Critical Infrastructure and Key Resources (CIKR), frequently coordinating its cyber campaigns to support conventional military objectives. [37]

The group's most notorious capability is the Industroyer malware, which represents a paradigm shift in OT cyber warfare. First deployed in December 2016 against Ukraine's power grid, Industroyer was the first malware designed to automate the physical disruption of electrical substations. By natively speaking industrial communication protocols, the malware successfully opened circuit breakers, causing a significant blackout in Kyiv. [37]

Coinciding with the full-scale invasion of Ukraine in 2022, APT44 deployed a successor variant, Industroyer2. This modernized toolkit was hardcoded to target specific high-voltage substations by abusing native MicroSCADA binaries to manipulate control logic. Unlike the 2016 incident, the Industroyer2 attack was successfully detected and

mitigated by defenders before it could disrupt the power supply. Nevertheless, the deployment confirms APT44's persistent capability and intent to weaponize OT environments. [37]

APT40 – Leviathan

Attributed to the Chinese Ministry of State Security, APT40 (also tracked as TA423 or Leviathan) conducts cyber espionage operations primarily in the South China Sea. While historically focused on government and naval targets, recent intelligence reveals a strategic shift toward the energy sector. In 2022, the group utilized the ScanBox reconnaissance framework to target heavy industry manufacturers and maintenance firms responsible for offshore wind turbines. This targeting suggests a deliberate effort to gather intelligence on critical maritime infrastructure and energy assets situated in disputed geopolitical waters. [48]

YoroTrooper

While not attributed to any specific nation-state or actor, YoroTrooper targets government and state-owned energy organizations, primarily operating within the Commonwealth of Independent States (CIS). Active since 2022, this group utilizes phishing campaigns to deploy commodity malware and open-source information stealers (e.g., LodaRAT). While their observed toolset is IT-centric, researchers highlight significant tactical overlaps between YoroTrooper and PoetRAT, an actor that has been previously confirmed to target SCADA systems in Azerbaijan explicitly. This connection implies that YoroTrooper's espionage against energy utilities serves as the critical reconnaissance phase required to facilitate future operational disruption. [49]

3 Testbed design and architecture

The previous chapters established the critical vulnerability of wind energy infrastructure and the theoretical underpinnings of the OT and SCADA systems that control it. As highlighted in the threat landscape analysis, the convergence of IT and OT networks has exposed these physical systems to sophisticated cyber threats. However, studying these threats on live critical infrastructure is operationally and financially dangerous. To bridge the gap between theoretical vulnerability analysis and practical defense, a dedicated experimental environment—a testbed—is required. The result of this chapter will be a functional prototype of a wind plant.

3.1 Design requirements and criteria

The primary objective of the testbed, in terms of both appearance and functionality, is to resemble a simplified version of a wind farm, containing one WTG, a substation, and a meteorological station.

In terms of hardware components, the testbed will consist of three primary hardware parts: controller, motor, and motor driver. To house these components, a simple 3D model will be used for the motor to resemble an actual wind plant. [50] The motor driver, controller, and breadboard will resemble a substation. The HMI will be hosted in a virtual machine in a CyberRange^{CZ} platform.

From the SCADA perspective, the testbed system will cover the first three levels, namely Level 0 (wind turbine and sensors), Level 1 (PLC), and Level 2 (HMI). The testbed will abstract from levels 3 and 4. The reason for the abstraction of Level 3 is that the aim of this thesis is to create a single functional prototype of a wind plant. In a future expansion of multiple WTGs, it will be possible to create one master terminal unit that will coordinate and collect data from all of the WTGs. The Level 4's ERP system is also not defined in the scope of this thesis, but will definitely be a point of interest in a possible future work when developing a CDX scenario.

More information about each component will be discussed in a dedicated section.

3.2 System architecture overview

This section translates the testbed’s functional requirements into a concrete architectural design. It evaluates two potential control configurations: a monolithic architecture, where the PLC runtime and control services share a single computational unit, and a distributed architecture, where these roles are separated across multiple devices.

Monolithic architecture

In a monolithic approach, the PLC runtime environment and control server components coexist on a single embedded platform. This configuration offers several distinct advantages. By eliminating the network infrastructure typically required between a PLC and a control server, the setup process is streamlined, and troubleshooting is simplified as all software components are accessible via a single interface.

From a cost perspective, this architecture maximizes the utilization of the computational unit’s resources, removing the need for redundant hardware. This efficiency lowers power consumption and physical space requirements, making it an ideal strategy for educational environments where budget constraints may require the deployment of multiple testbed instances.

However, the monolithic approach has certain limitations. Scalability is restricted as all components compete for the same computational and memory resources, which can limit the complexity of control programs or the number of simultaneous connections that can be supported. Additionally, this architecture diverges from industrial practices, where the separation of concerns typically distributes PLC and control server functions across dedicated hardware platforms.

Distributed architecture

In contrast, a distributed architectural approach separates the PLC runtime environment onto a distinct controller node, while the control server and HMI components operate on a separate platform. This configuration more accurately mirrors industrial deployment patterns, including master–slave PLC setups, and offers robustness.

By distributing tasks, computationally intensive operations, such as data logging and complex visualization, can be offloaded to more powerful hardware, allowing the primary PLC controller to focus strictly on logic execution with deterministic performance. Moreover, this architecture improves fault isolation; a failure in the supervisory server does not necessarily disrupt the critical control loop.

The trade-off for this approach is increased complexity in network synchronization and higher initial hardware costs. Given this thesis's focus on affordability and compactness, the monolithic architecture based on a single computational unit was selected for the prototype.

3.3 Hardware component selection

Having defined the system architecture as monolithic, the next step is to identify the specific hardware category suitable for acting as the system's central computational unit.

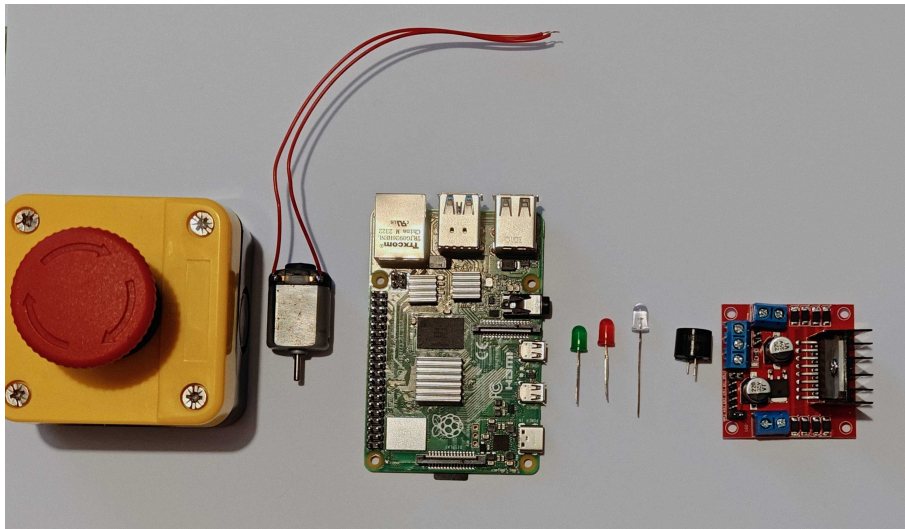


Figure 3.1: Chosen components for testbed, including a button, DC motor, Raspberry Pi 4B, LEDs, buzzer and L298N motor driver.

Controller

The primary hardware decision involved choosing between a Microcontroller Unit (MCU) and a Single-Board Computer (SBC).

MCUs are integrated circuits that contain a microprocessor core, memory, and peripherals all on a single chip, designed for use in embedded systems and for specific functions. Typically low-power devices with limited processing power, they are ideal for applications where real-time control, low cost, and low power consumption are critical. [51] On the other hand, SBCs come with more processing power and the ability to run an operating system, enabling them to handle complex tasks in applications such as multimedia processing, robotics, and network servers. [51]

Considering the software requirements of OpenPLC, I have chosen an SBC for the purpose of this thesis.

Furthermore, I have opted for a Raspberry Pi¹ as my SBC of choice. This choice was made in accordance with the philosophy of this thesis, which is to create a testbed using affordable and widely available components. While other SBC alternatives exist (e.g., Banana Pi or Orange Pi), the Raspberry Pi was chosen for its superior global availability and extensive community support. This model has already pre-soldered GPIO² pins that are used to interact with external sensors.

Motor

Essentially, an electrical motor needed to be selected that would replicate the wind turbine's rotation. To ensure smooth movement, I decided against using a DC stepper motor. Instead, I chose a brushed DC motor³. This type of motor operates within a voltage range of 6V to 14.8V and can reach speeds of up to 14,000 RPM⁴, which is more than sufficient for this application. The motor will be controlled using PWM⁵, but to achieve that, an appropriate motor driver needs to be selected.

-
1. Raspberry Pi 4 Model B – 1GB RAM
 2. General-purpose input/output pins
 3. MIG 2025 12V electromotor
 4. Revolutions per minute
 5. Pulse-width modulation, operating in duty cycles. The cycle can be regulated in percentages.

Motor driver

To select the appropriate motor driver, I needed to meet several criteria. The first requirement, mentioned in the previous chapter, was that the driver's control mechanism must be PWM, as OpenPLC does not support protocols, such as SPI⁶, by default. However, there is a way to write drivers for such components that require similar communication protocols with the Python Sub Module (PSM). [52] The next requirement was to ensure that the operating voltage matched that of the DC motor. Lastly, it was essential for the circuit's internal logic to be powered by no more than 5V, since the Raspberry Pi can provide either 3.3V or 5V on its GPIO pins. The L298N Dual H-Bridge motor driver meets these criteria, featuring an operational voltage range of 5V to 35V and the capability to control up to two DC motors.

Power supply

The L298N requires an external power source to provide a regulated voltage for the motor. A generic power source was chosen, which provides 12V. This is perfectly sufficient, for the simulation of the turbine rotation will typically be at low RPMs. This is to ensure a level of realism and also maintain a level of safety for wind turbine operators in a possible CDX scenario. To achieve this, I have implemented a software lock in the LD that will be discussed later in the PLC programming chapter.

Physical feedback mechanisms

I have gathered feedback from the previous OT-focused CDX runs and incorporated it into the vision for the new testbed. [53] The participants were enthusiastic about the interactive features of the OT elements that they could control⁷. To address this feedback, I have introduced several new elements, starting with LEDs. To reflect on this feedback, I have introduced numerous elements, starting with LEDs.

6. Serial Peripheral Interface

7. For example, a physical button they needed to push to restart a device.

In this scenario, I'm using three LEDs, each representing a different state of the wind plant's operational capabilities. The first LED is white, symbolizing a light siren for any incoming air traffic. The second LED is red, which indicates that the wind plant is in a critical state due to either an internal or external factor.⁸ The last LED is green, and when it lights up, it signifies that the wind plant is generating electrical power for the grid.

Additionally, I have implemented an active sound siren, which functions as a buzzer operating on 3.3V, making it suitable for the Raspberry Pi's GPIO pins. The buzzer emits a sound when the wind plant enters an undesirable state, working in conjunction with the red signaling LED. Lastly, there is a large red button used to restart the wind plant's operational state.

3.4 Electrical connections and wiring

Before starting the wiring process, I had to adjust to the new mapping of the GPIO pins. When installing the OpenPLC runtime with the hardware layer for RPI, the GPIO pins are assigned a fixed memory address, allowing most pins to be utilized in a PLC program. In total, OpenPLC allows you to use 14 digital inputs, 11 digital outputs, and one analog output address. [54] This means that the OpenPLC utilizes 65% of the Raspberry Pi's available GPIO pins. Since I will be using Modbus to communicate with the RPI, I need to focus on output memory addresses. The input memory addresses are of only read type and are used to receive signals from input devices exclusively. [55] Below, I have created a table that summarizes the wiring between the main components and their mapping from physical pins to GPIO pins to OpenPLC addresses.

This section provides a detailed description of the wiring for all components depicted in Figure 3.2.

DC motor: The wiring of this component is straightforward. The motor has only two contacts, each of which needs to be soldered to a wire, as indicated by the cyan one in the wiring diagram in the Figure 3.2. Those wires are connected to a terminal block

8. This may be a situation when, for example, an emergency brake is turned on.

labeled OUT1 and OUT2, which control the rotation of Motor A.

L298N motor driver: Apart from the connection with the DC motor, a power needs to be provided using the generic 12V power supply to the terminal blocks labeled +12V and GND⁹ labeled with blue wires. L298N comes with a jumper¹⁰ placed on the 5V Enable pin used for the 78M05 5V regulator, which can power up an MCU together with pins ENA and ENB used for the PWM signal to the motors. [57] I have left the jumper only at ENB, for I will not be using the second motor, and there is no need to use the 5V for the controller.

The remaining connections of this component are targeting RPI and are labeled with purple wires. Initially, the internal circuit logic of the L298N chip requires voltage to function, so the RPI supplies 5V to the appropriately named pin, using Pin 4. RPI unifies grounds with the power supply and has its Pin 6 connected to the controller's pin GND. The remaining connections for this component are already listed in Table 3.1. To clarify its functionality, IN1 and IN2 are used to specify the spinning direction, and the ENA pin controls the rotation intensity of the motor.

Feedback mechanisms: The last wiring section regards feedback mechanisms. Starting with LEDs, each is connected to its anode with the RPI's GPIO pin, which will be used in the PLC program logic. The cathode of the LEDs is connected to the ground via a 330 Ω resistor. The ground is distributed from the RPI's Pin 14 through the breadboard using black wires, as shown in the diagram.

Although the active buzzer operates at the RPI's native 3.3V logic level, it was not connected directly to a GPIO pin due to current constraints. A standard GPIO pin can safely source approximately 16 mA [58], whereas the buzzer requires an operating current up to 25 mA. To prevent potential hardware

9. Ground

10. A small conductor that can modify a component's settings by connecting or disconnecting it. [56]

damage of the RPI pins, a MOSFET transistor¹¹ was deployed as a low-side switch. As a voltage-controlled device, it allows the weak GPIO signal to toggle the high-current load without drawing significant current from the processor itself. In this specific configuration, the MOSFET's gate pin is connected to the control signal at GPIO 24. The drain pin is wired to the negative terminal of the buzzer, while the source pin is connected to the common ground at Pin 14. Consequently, when the GPIO signal is high, the operating current flows directly from the 3.3V power rail through the buzzer and the MOSFET. Last but not least, the emergency button is connected to the ground on the breadboard and with the GPIO 2, which is in the OpenPLC addresses reserved for input devices.

Table 3.1: GPIO pins allocation and OpenPLC addresses mapping from RPI perspective

Component	RPI	GPIO	OpenPLC
Emergency button	Pin 3	GPIO 2	%IX0.0
IN1	Pin 8	GPIO 14	%QX0.0
IN2	Pin 10	GPIO 15	Not mapped ^a
Motor PWM (ENA)	Pin 12	GPIO 18	%QW0
White LED	Pin 16	GPIO 23	%QX0.2
Buzzer	Pin 18	GPIO 24	%QX0.3
Green LED	Pin 22	GPIO 25	%QX0.4
Red LED	Pin 24	GPIO 8	%QX0.5

^a. This pin was not included in the final program version, as it has no significant use in a wind turbine model.

11. BS170 N-channel type

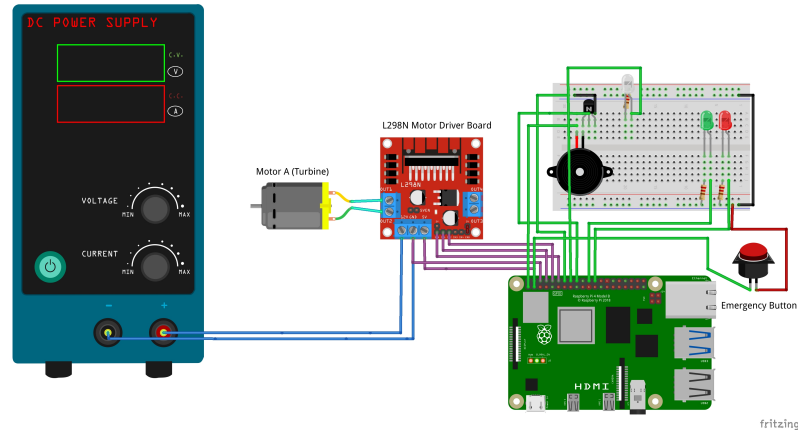


Figure 3.2: A wiring diagram of the testbed made in Fritzing using L298N community model. [59]

3.5 Runtime environment deployment

The deployment of the control logic relies on a stable and deterministic runtime environment. The OpenPLC v3 runtime was deployed on the Raspberry Pi, serving as the central processing unit for the testbed.

The device was configured with the Raspberry Pi OS Lite (64-bit, version Bullseye). The Lite version was selected to minimize resource overhead by excluding the desktop environment and unnecessary background services, ensuring that maximum CPU cycles are available for the PLC scan cycle.

OpenPLC runtime setup

The OpenPLC runtime was compiled from source for the Raspberry Pi platform using the installation script.

Upon installation, the OpenPLC runtime operates as a system service (openplc.service), configured to start automatically on boot. This installation then enables the OpenPLC web server at port 8080, using the default credentials specified in the documentation. This web user interface is used to transfer and run your PLC programs on the target device. However, to successfully run and compile programs in

structured text, it is essential to select the correct OpenPLC hardware layer, specifically the Raspberry Pi.

Assuming that the structured text compiled successfully, we may proceed to run the program. At this point, port 502, typically used for Modbus TCP communication, changes its state to open and is ready to accept incoming connections. Subsequently, we can monitor the status of coils and registers on the hardware tab.

At this stage, the PLC is ready to be connected with its HMI. But before that, I will introduce the LD that supplies the main control logic for the wind turbine.

Point Name	Type	Location	Write	Value
IN1	BOOL	%QX0.0	Write Read	FALSE
LED	BOOL	%QX0.2	Write Read	TRUE
BUZZER	BOOL	%QX0.3	Write Read	FALSE
LED_POW	BOOL	%QX0.4	Write Read	FALSE
LED_ERR	BOOL	%QX0.5	Write Read	FALSE
START	BOOL	%QX1.3	Write Read	FALSE
ESTOP	BOOL	%QX1.4	Write Read	FALSE
MANUAL_MODE	BOOL	%QX1.5	Write Read	FALSE
WIND_SPEED_FAULT	BOOL	%QX1.6	Write Read	FALSE
DESTRUCT_SIGNAL	BOOL	%QX1.7	Write Read	FALSE
ENA_PWM	INT	%QW0		0

Figure 3.3: Monitoring tab in OpenPLC web server.

3.6 Ladder diagram implementation

The control logic for the wind turbine simulation was implemented using LD. The program is structured into distinct sections that handle speed regulation, power simulation, safety interlocks, and state indication. The following subsections detail the specific logic implemented in the OpenPLC environment.

PLC in wind turbine control

The central PLC program continuously monitors inputs from the SCADA interface (e.g., wind speed setpoints, manual overrides) and physical safety switches (E-STOP). Based on these inputs, the PLC calculates the required actuator outputs (motor PWM) and updates the internal status registers read by the HMI.

The logic flow is hierarchical: safety conditions are evaluated first. If the system is in a healthy state, control and computation blocks are executed. Finally, the outputs are mapped to the physical GPIO pins.

Speed control

The core function of the testbed is to translate a simulated wind speed into a physical rotation of the DC motor. The speed control logic, visualized in the main program block, operates in two distinct modes:

1. **Automatic mode:** The system reads a simulated WIND_SPEED value from the HMI. This integer is converted to a REAL data type and scaled. The logic implements a simplified power curve:

Cut-in speed: If the wind speed is below 10.0 m/s, the target RPM is set to 0.

Operational speed: Between 10.0 m/s and 45.0 m/s, the RPM increases linearly. This relationship is defined by a scaling factor derived from the boundary conditions of the simulation: 0 RPM at 10.0 m/s and a maximum of 500 RPM at 45.0 m/s. This logic is implemented in the PLC using a SUB and MUL block configuration: $RPM = (WindSpeed - 10.0) \times 14.28$.

Cut-out speed: If wind speed exceeds 45.0 m/s, the system limits the RPM to protect the turbine (simulated by clamping the value to 500.0). This maximum RPM value was derived from a small scaled WTG. [60]

2. **Manual Mode:** For maintenance and testing, a MANUAL_MODE contact allows an operator to bypass the wind simulation and write a direct RPM setpoint to the controller. This override capability also functions as a manual brake.

To control the physical L298N motor driver via the Raspberry Pi's GPIO, the calculated RPM must be converted into a PWM duty cycle. Since the OpenPLC hardware layer utilizes a 16-bit integer (0–65535) for PWM output, a scaling formula was implemented to map the target RPM (0–500) to the effective voltage range of the motor. The logic uses an ADD and MUL block to apply a base offset:

$$PWM_{output} = (RPM \times 5.0) + 19500.0 \quad (3.1)$$

The offset of 19,500 ensures that the voltage supplied to the motor is sufficient to overcome static friction (dead zone) and start rotation immediately, while the multiplier scales the speed up to a maximum value of approximately 22,000. These values were determined by empirical testing of the motor's PWM cycle. Subsequently, this software limiter prevents the wind turbine model from physical damage.

Power calculation

As the testbed lacks a physical generator to measure electrical output, power generation is simulated mathematically within the PLC logic. The program implements a realistic, non-linear power curve that approximates the relationship between the turbine's rotational speed and its power output. While the theoretically available power from wind is proportional to the cube of the wind speed ($P \propto v^3$), this simulation models the generated power as being proportional to the cube of the rotor's RPM.

The model is designed around a hypothetical small-sized turbine with a maximum power (Rated Power) of 100kW. [61] This maximum output is achieved at a maximum speed (Rated Speed) of 500 RPM. The power output follows the cubic formula:

$$P_{kW} = k \times (RPM)^3 \quad (3.2)$$

where P_{kW} is the power produced in kilowatts and k is a scaling constant. By calibrating the model against the rated operating point (100kW at 500 RPM), the scaling constant k is determined to be 0.0000008.

The implementation in the LD consists of three cascaded MUL (Multiply) blocks, calculating:

$$KW_{PRODUCED} = RPM \times RPM \times RPM \times k \quad (3.3)$$

The result is stored in a register exposed to the SCADA system for visualization.

Safety logic

Safety is the highest priority in the control hierarchy. The safety logic monitors three critical fault conditions:

SYSTEM_FAULT: An internal error state that causes the wind turbine to stop producing electricity. At this state, the red LED is turned on, the motor is turned off, and a buzzer starts the alarm. For the turbine to start working again, the operator needs to press the physical restart button.

ESTOP: A variable representing emergency stop/brake, that sets the state of the turbine to SYSTEM_FAULT. The operator needs to press the physical button to restore the turbine's functionality.

WIND_SPEED_FAULT: A condition triggered if the wind speed exceeds the absolute maximum rating (80.0 m/s). Under these conditions, a SYSTEM_FAULT condition is triggered, and the turbine ceases to produce electricity.

At Figure 3.4, the program utilizes a *Normally Closed (NC)* design. If any of the safety contacts open (logic FALSE), the main power rung is broken. This immediately sets the ENA_PWM output to 0, cutting power to the motor driver via hardware, regardless of the software speed setpoint.

To recover from a fault, the system implements a latching mechanism. Even if the fault condition clears (e.g., the E-STOP is released), the system remains in a halted state until a specific RESTART_BUTTON signal is received.

As mentioned, the physical restart button is interfaced via GPIO 2 on the Raspberry Pi. This pin is configured with an internal pull-up resistor, which maintains the signal at a logic High when the circuit is open (idle state). [58] Pressing the button connects the pin to ground, closing the circuit and pulling the signal to logic Low. Consequently, the PLC runtime interprets the idle state as a boolean TRUE. To correctly detect the button press, the control logic must be inverted to trigger on the logic Low state or a boolean FALSE.

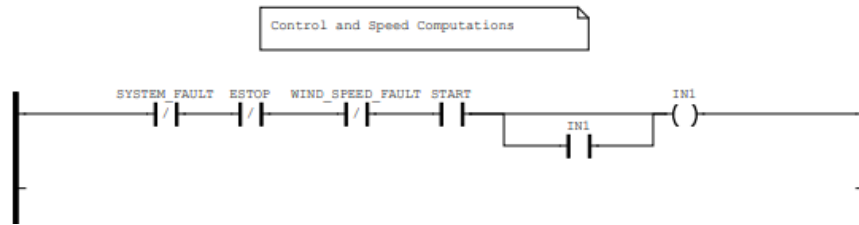


Figure 3.4: Rung for handling safety logic.

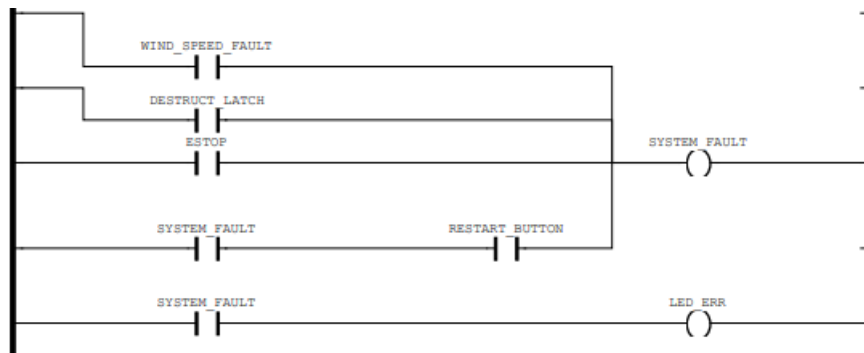


Figure 3.5: Fault logic rung in detail.

Alarm control

To provide feedback to the operator, the PLC controls a visual and auditory alarm system based on the `SYSTEM_FAULT` variable.

To prevent operator fatigue, the buzzer logic is more complex than the LED, which only lights up when the `SYSTEM_FAULT` variable is set to `TRUE`.

It utilizes a Counter Up (CTU) block and a long-duration timer. When a fault occurs, the buzzer pulses and the counter increments with each pulse. Once the counter reaches a preset of 3, the buzzer is silenced for 15 seconds by the `TON_PAUSE` timer. This creates a distinct pattern, which represents industrial alarm systems.

Destruction

To support the cybersecurity training scenario, the program includes a "Destruction" subroutine designed to simulate catastrophic mechanical failure due to overspeeding.

This logic monitors the `READ_RPM_REAL` value. If the RPM remains above the critical threshold of 500.0 for a continuous duration of 10 minutes (monitored by the `DESTRUCT_TON` timer), the system triggers the `DESTRUCT_LATCH`. This latch permanently sets the `SYSTEM_FAULT` and disables the turbine. In a training scenario, this state simulates physical damage to the gearbox or rotor that cannot be fixed by a simple software restart, representing the impact of a successful cyber-physical attack.

The entire ladder diagram was tested using a hardware prototype shown in Figure 3.6.

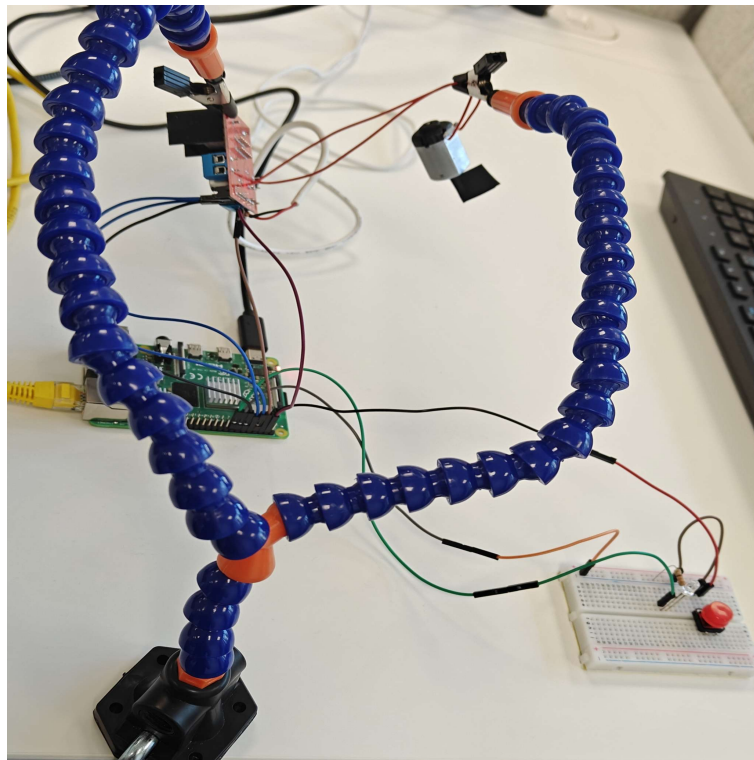


Figure 3.6: First prototype used for the Ladder Diagram testing.

3.7 HMI development

The Human-Machine Interface serves as the primary layer for visualizing and controlling the wind turbine testbed. It was developed using Node-RED, a flow-based development tool used in IoT and OT prototyping. The flow is composed of nodes, each providing different functionality. The visual interface was constructed using the `@flowfuse/node-red-dashboard` package (Dashboard 2.0), while communication with the PLC is handled by the `node-red-contrib-modbus` library. The connection with the PLC was a straightforward process, as each of the Modbus nodes has a section to configure a Modbus server, making the HMI the master in the Modbus TCP communication.

Interface architecture

The dashboard is organized into four logical groups to assist the operator in maintaining situational awareness:

Control panel: Allows the operator to manually start the turbine, set the target RPM via a slider, and trigger the Emergency Brake.

Status panel: Visualizes real-time telemetry, including current RPM, wind speed, and wind direction, using semi-circular gauges and a compass widget.

Production panel: Displays the generated power output over time using a line chart and calculates the total cumulative energy (kWh).

Event logs: A custom-coded Vue.js template that parses status codes from the PLC to display a color-coded log of system events with timestamps.

To simulate realistic environmental conditions, the Node-RED flow ingests a CSV dataset, `wind.csv`, inspired by data from CHMI¹², containing wind speed and wind direction in azimuth degrees. This data is injected into the PLC's holding registers at 10-second intervals to drive the simulation when the system is in automatic mode. A 3-second delay is introduced between the change in wind speed and

12. Czech Hydrometeorological Institute

direction to provide a more realistic simulation. The wind directions are then converted from degrees to a wind direction description by the node-red-contrib-compass module. By this implementation, I have met the wind farm's requirements for a meteorological station.

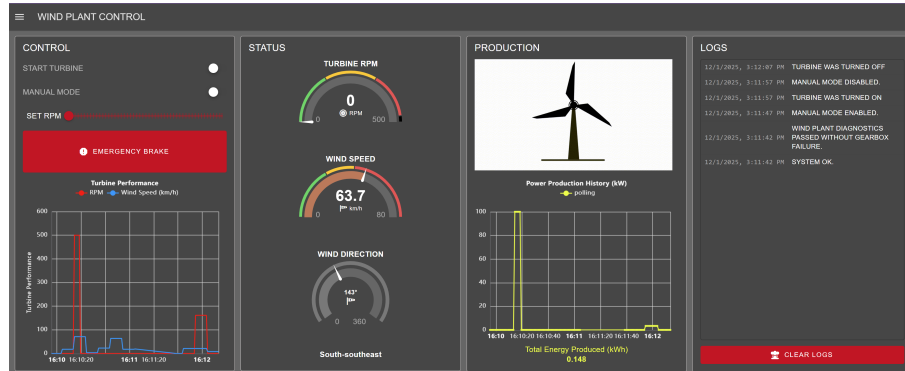


Figure 3.7: HMI developed using Node-RED.

Data type handling

A challenge was encountered during the development and configuration of Modbus communication. There was a data type mismatch between the Modbus plugin read holding register operation and the required precision of the wind speed data. Standard Modbus holding registers are 16-bit integers (0–65535). However, physical measurements such as wind speed (e.g., 12.5 m/s) or power output (e.g., 45.5 kW) require floating-point precision, meaning the use of a 32-bit floating-point data type. Although Modbus can handle 32-bit values by combining two contiguous 16-bit registers, this approach introduces complexity regarding byte alignment¹³. To ensure robustness and simplify the educational implementation, a fixed-point arithmetic approach using standard 16-bit integers was selected.

13. Endianness

The scaling logic is distributed between the PLC and the HMI as follows:

PLC side (encoding): The logic multiplies the real value by a scaling factor (typically 10 or 100) and casts it to an integer.

$$Val_{PLC} = \text{Round}(Val_{Real} \times 10) \quad (3.4)$$

For example, a wind speed of 12.5 m/s is stored in the register as the integer 125.

HMI side (decoding): Upon receiving the integer via Modbus, a JavaScript function node in Node-RED divides the value by the same factor to restore the floating-point precision before passing it to the dashboard widgets.

$$Val_{HMI} = \frac{Val_{PLC}}{10.0} \quad (3.5)$$

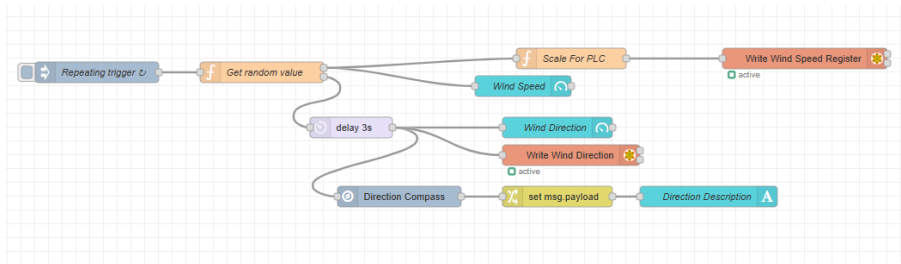


Figure 3.8: A flow that handles wind data logic for HMI.

HMI usage

From the perspective of wind plant control, there is a separation of duties into two roles: wind plant operator and wind plant technician. The idea behind this separation is the potential for later adaptation in a CDX scenario. The wind plant operator will be a member of the defender's team, having access to the HMI and the ability to control the wind plant's functionality. On the other hand, the wind farm technician will be a member of a Green team, possibly shielded by a

WTG's vendor role. This role will have access to the OpenPLC web server, unlike the operator, who will be forbidden from using it.

The primary reason for this precaution is that if the operator fails to defend the turbine against an attacker, they will be required to communicate with the vendor about a potential repair process, which involves stopping the runtime program of the wind turbine and restarting it. If an operator has access to this OpenPLC web server, they can evade the need to physically press the emergency button to restart the plant's functionality.

To clearly describe the wind plant operator's role, they will monitor the wind turbine's RPM in relation to the wind speed. They should be aware of the situations when the turbine is not producing any electricity, even though the wind is strong enough to produce and send power to the grid. To help operators view the packets flowing into the turbine, I have created a simple Python application that periodically checks the status of coils and holding registers, supplementing the *Monitoring* section in the OpenPLC web server. This application helps to clearly understand what is happening to the wind plant, for example, in a case where the attackers are performing a DoS attack on the HMI.

PLC Healthcheck			
Coils (0-19):			
Address	State	Address	State
0	False	10	False
1	False	11	False
2	True	12	False
3	False	13	False
4	False	14	False
5	False	15	False
6	False	16	False
7	False	17	False
8	False	18	False
9	False	19	False

Holding Registers (0-9):	
Address	Value
0	0
1	0
2	345
3	0
4	0
5	227
6	0
7	0
8	0
9	0

Figure 3.9: A PLC healthcheck app showing status of coils and holding registers.

3.8 Limitations and constraints

This section focuses on the differences and simplifications between a real-world wind plant and a hardware testbed model. It is important to clarify that the primary objective of this thesis was not the creation of a high-fidelity mechanical replica of a wind turbine, but rather the development of a functional cyber-physical system that authentically implements industrial control logic and communication standards.

The most fundamental difference from a real-world environment is the direction of energy flow. While a commercial wind turbine converts kinetic energy from wind into electricity, this testbed operates as a motor-driven system that consumes electrical power. Since the testbed is designed for indoor laboratory environments where consistent wind flow is unavailable, the DC motor acts as a proxy for the wind's force, spinning the rotor to visually simulate operation. Consequently, the generated power displayed on the SCADA system is purely a mathematical calculation derived within the PLC logic ($P \propto v^3$), rather than a physical measurement from a generator coil.

Mechanically, the testbed utilizes a static structure with fixed-geometry blades. Unlike industrial turbines that actively regulate aerodynamics through blade pitch actuation and optimize orientation via nacelle yaw control, this model's components are rigid. [62] As a result, the HMI and control logic are simplified; controls for pitch angle adjustments and yaw alignment are omitted, as there are no physical actuators to execute these commands.

To ensure user safety during physical exercises, the control logic implements a software-defined limiter via PWM scaling. By mapping the target RPM to a restricted duty cycle range (19500–22000), the system utilizes only approximately 30-35% of the motor's maximum voltage capacity.

Acknowledging these mechanical abstractions, a prototype was constructed to fulfill the thesis objective of providing a functional, hardware testbed environment. In Figure 3.10, from left to right, there is an assembled 3D-printed model of a wind turbine, followed by an open housing that contains RPI, breadboard, and L298N motor driver, which collectively resemble an electrical substation. Additionally, a restart button is included.



Figure 3.10: A final prototype of the wind farm.

4 Integration with CyberRange^{CZ}

At this phase, the hardware testbed is fully constructed, meaning that the next step in this thesis is the process of creating PoC¹ network topology in a CyberRange^{CZ} Platform. This involves generating a sandbox definition and Ansible roles, which will be used to set up and configure the machines defined in the sandbox. Followingly, a VPN server needs to be created and placed inside this topology to allow the connection of the physical hardware inside a virtualized network.

CyberRange^{CZ} Platform² is an open-source fork of the KYPO CRP³ project that can be categorized as a cyber range type of software. A cyber range is typically a platform that allows users to emulate network environments and, accordingly, adversary campaigns against them. This means that cyber ranges can also be used for cybersecurity training of varying sizes and types, ranging from CTFs⁴ to large and complex CDXs. CRCZP runs on top of OpenStack, which is a cloud operating system that acts as a hypervisor, creating compute, storage, and network resources. [63]

4.1 Sandbox definition

Sandboxes within the CRCZP are isolated network environments grouped into pools. To define what will be available in a sandbox environment, we need to create a sandbox definition, which consists of two parts: topology definition and sandbox provisioning. [64]

The topology definition contains information about the specific machines used, their routers, and high-level networking between them. To list the specific content of the topology.yml, there is a definition of two network segments: the OT segment with the CIDR of 10.0.20.0/24 for the OT devices, and the USR segment with the CIDR of 10.0.10.0/24 for the employees of a fictitious company. This separation of segments adheres to the practice of separating OT or IoT devices due to security constraints. Each of these segments contains a router.

-
1. Proof of concept
 2. Hereafter referred to as CRCZP.
 3. Kybernetický polygon, a registered trademark of Masaryk University.
 4. Capture the Flag competitions

In the OT segment, the `ot-hmi` machine is placed, which hosts the Node-RED HMI developed in Chapter 3. HMI will be available through the web interface at the `/hmi` endpoint on port 1880. The user segment contains an `usr-workstation` running Debian 12 OS. Foreshadowing the attack scenario, this workstation will be used to conduct the cyberattack, eliminating the need for a separate attacker machine, thereby creating a compact and presentable topology. The following listing displays the specific hardware configuration for `ot-hmi` and `ot-router`.

Listing 4.1: Topology specification in `topology.yml` focusing on hosts and routers.

```
hosts:
  - name: ot-hmi
    base_box:
      image: ubuntu-noble-x86_64
      man_user: ubuntu
      flavor: a2-2-35 # CPU cores, RAM, storage

routers:
  - name: ot-router
    base_box:
      image: debian-12-x86_64
      man_user: debian
      flavor: a1-2-20
```

Now that the topology and provisioning are defined, the sandbox is ready to be built. The CRCZP has three stages of the building process: *Terraform Stage*, *Networking Ansible Stage*, and *User Ansible Stage*. The first two stages are constructed based on the specifications outlined in the `topology.yml`, utilizing OpenTofu⁵ and Ansible. The last one, *User Ansible Stage*, uses provisioning, which is the last missing component to have a valid sandbox definition.

Provisioning requires at least one file, specifically `playbook.yml`. The main playbook imports machine-specific playbooks, which configure each machine in the sandbox topology. The machines `ot-hmi` and

5. An open-source fork of Terraform that allows for defining virtualized Infrastructures as Code (IaC).

usr-workstation have their specific Ansible roles to achieve reusability. Ansible roles are available in the archive of this thesis in the roles folder.

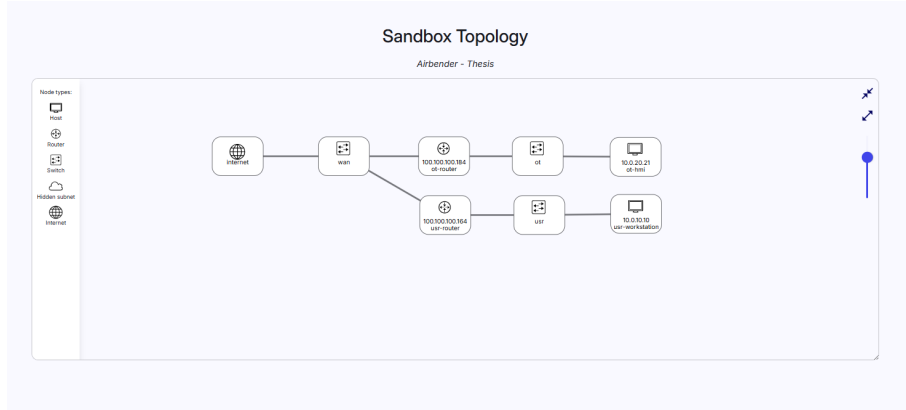


Figure 4.1: A network topology diagram automatically generated in CyberRange^{CZ}.

4.2 Hardware connection to the sandbox network

Virtualized machines are connected to sandbox networks specified in the topology.yml file; however, to achieve this with physical hardware, it needs to be added to this topology externally.

The most feasible solution to this problem is to create a VPN server, which will provide access to outside machines within an OT network segment. The implementation of this server varies depending on the physical area's networking and the deployment environment's situation.

To address my specific case, I needed to create a router instance in OpenStack that would be connected to the OT network segment in the sandbox. This router was configured to use the network interface of the OT segment and with the appropriate gateway 10.0.20.3.

Subsequently, I created a VM using a generic Ubuntu Linux image and assigned it a floating IP address that is accessible from the company's network segment. To achieve this with a fixed network address

located in the OT network segment, I had to create an OpenStack network port with the address 10.0.20.100.

The next step was to provide the network configuration to enable connection to the game network, more precisely, to the OT segment. Utilizing WireGuard VPN software, I set up the server instance to use a VPN network segment with 10.66.0.0/24 and generated VPN configuration files for all necessary machines with fixed IP addresses: `ot-hmi` (10.66.0.2), `usr-workstation` (10.66.0.3), and a Raspberry Pi (10.66.0.4). Those configuration files were applied by utilizing the `wg0-quick` system service, and the machines were connected to the OT-VPN segment.

5 Cyberattack on OT testbed

This chapter presents a detailed analysis of a simulated cyberattack against the wind energy testbed. The cyberattack scenario, codenamed *Airbender*, is structured according to the ICS Cyber Kill Chain methodology described in Chapter 2. It demonstrates how a threat actor pivots from a compromised IT environment (Stage 1) into the OT segment to execute physical effects (Stage 2). The analysis maps each step of the adversary's tradecraft to the corresponding MITRE ATT&CK for ICS tactics and techniques, highlighting the specific vulnerabilities exploited in the Modbus TCP protocol and the testbed's configuration.

The *Airbender* campaign simulates a realistic adversary by combining tactics from several known incidents. It adopts the DoS disruptions seen in the sPower attack, while also exploiting specific HMI vulnerabilities similar to the Nordex and XZERES breaches. The attack's overall strategy follows the Sandworm group's methodology, utilizing LotL techniques and manipulating SCADA protocols directly, much like the original Industroyer malware. However, these destructive techniques are adapted to target a wind farm rather than a transmission substation.

To achieve this, the attacker infiltrates a `usr-workstation` machine, gains possession of the VPN configuration file, and connects to the `ot` network segment. At this point, they initiate the reconnaissance process, identifying two other machines crucial to their goal: `ot-hmi` and `wind plant (RPI)`. With basic vulnerability scanning, it is discovered that the `ot-hmi` is running a vulnerable installation of Node-RED. Exploiting this vulnerability, the attacker gains root privileges on the machine. Subsequently, the cyberattack progresses from the Intrusion phase to the ICS attack phase, as the attacker leverages a MitM attack, placing itself between the HMI and RPI, intercepting and altering their communication, and preparing their strike. With the gained knowledge, the attacker constructs a TCP proxy that manipulates packets sent between the machines, leading to the mechanical destruction of the wind turbine.

5.1 Stage 1: Cyber intrusion and preparation

In this phase, the adversary's goal is to gain access to the OT network and compromise the supervisory station (HMI).

5.1.1 Initial access and lateral movement

Technique: *Remote Services* [T0886]

The attack lifecycle begins with a simulated social engineering campaign targeting administrative staff with a targeted phishing email. Detailed simulation of this initial access vector is omitted to maintain focus on ICS exploitation. As the testbed employs a lightweight, headless architecture to simulate the workstation, deploying the heavy infrastructure necessary for realistic phishing simulations, such as Windows domains and mail servers, offers little value to this specific OT research goal.

Instead, the scenario operates under an Assumed Breach¹ model and begins at the post-compromise stage, when the attacker discovers that the WireGuard VPN configuration file is present on the workstation. The `usr.conf` file resides unencrypted in the file system, allowing the attacker to establish a VPN tunnel directly into the OT network via VPN network segment `10.66.0.0/24`, thereby bypassing the perimeter firewall. Upon establishing network access, the attacker performed active reconnaissance using Nmap to identify active hosts. The important network ports for the attack scenario are as follows:

10.66.0.2: Identified as the HMI (TCP port 1880 – NodeRED).

10.66.0.4: Identified as the PLC (TCP port 502 – Modbus).

5.1.2 Exploitation of HMI

Technique: *Exploitation of Remote Services* [T0866]

The attacker targeted the operator's HMI, which was running Node-RED on port 1880. The installation was found to be vulnerable to CVE-2025-41656, a known vulnerability where the editor lacks

1. This model "assumes a threat has some level of access to a target at the initiation of the engagement." [65]

authentication by default. This allowed the attacker to access the flow editor directly via a web browser. With access to the logic flow, the adversary executed a LotL attack. By dragging a standard exec node onto the canvas, they created a reverse shell payload.

Since the Node-RED service was running with root privileges, violating the principle of least privilege, the execution of this node granted the attacker full control over the HMI device. At this phase, the attacker can maintain persistence by deploying a beacon² into the ot-hmi machine, which might be later used with C2 frameworks, such as Crowdstrike or Sliver.

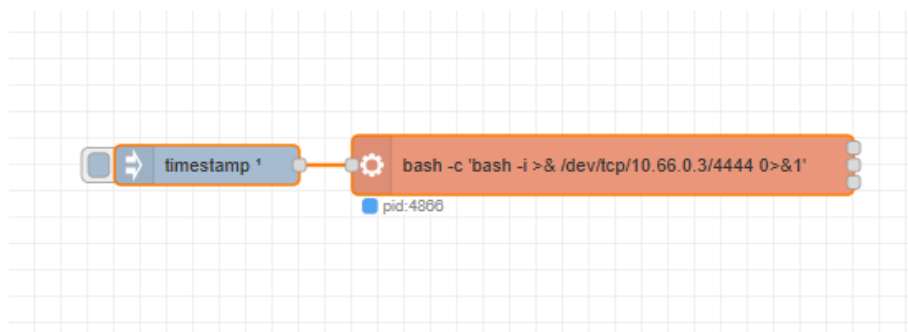


Figure 5.1: An exec node used to spawn a reverse shell.

5.2 Stage 2: ICS attack development and execution

Having secured a foothold on the HMI, the adversary transitioned to Stage 2. The objective shifted from network access to manipulating the physical process.

5.2.1 Develop and Deliver

Technique: *Adversary-in-the-Middle* [T0830]

Before diving into the implementation of the MitM attack, it is essential to address a specific topological limitation inherent in the testbed's design. The attack vector described below relies on Address

2. A program that allows the attacker to execute commands remotely on an infected system.

Resolution Protocol (ARP) spoofing, a technique that is fundamentally tied to the Data Link Layer (Layer 2) and relies on broadcast traffic.

However, the connection between the physical wind turbine and the CyberRange^{CZ} platform, as outlined in Chapter 4, utilizes a Wire-Guard VPN. Standard VPN configurations operate at the Network Layer (Layer 3) through routed interfaces, which do not support the local broadcast traffic necessary for ARP manipulation. As a result, a remote attacker relying solely on the routed VPN tunnel would be unable to execute ARP spoofing against the target devices.

Due to the current implementation and computational resource limitations of the CyberRange^{CZ} platform, the MitM attack was tested in a local development environment using the Proxmox VE hypervisor, allowing the ot-hmi, usr-workstation, and wind plant machines to operate on Layer 2. Full integration of this attack will be completed once the VPN limitation is addressed. A potential solution for future work is to use a different VPN software that supports a bridged configuration, such as OpenVPN. [66]

Continuing with the MitM, the attacker identified that the Modbus TCP protocol used between the HMI and PLC lacks encryption and integrity checking. To exploit this, the Bettercap tool, used for reconnaissance and network attacks, was deployed to perform ARP spoofing.

While still in the flow editor, the attacker tampered with modbus-client node configuration, setting larger timeouts from the default 1000ms to 30000ms. The purpose of this alteration is to ensure that Bettercap can reestablish network traffic interception and the ARP spoofing process. Utilizing this setup, they observe which coils and registers are being accessed and for what purpose, in order to develop and deliver a perfect exploit.

The following Bettercap configuration was used to redirect Modbus traffic (port 502) to a local TCP proxy running at port 8502. This script `bettercap.cap` firstly enables IPv4 forwarding on the attacker machine, so the intercepted packets are forwarded, followed by network probing that shows discovered devices after a three-second timeout. The discovered devices are targeted with the ARP spoofing commands, specifying with the `arp.spoof.full duplex`, that both the devices and the gateway will be targeted. [67] Then, with the use of iptables, the caplet specifies that any incoming and outgoing traffic for the wind

plant will be redirected to port 8502, which is used by the attacker's custom TCP proxy, handling the attacker-defined logic.

Listing 5.1: Bettercap caplet configuration for Modbus MitM.

```
!echo 1 > /proc/sys/net/ipv4/ip_forward
net.probe on
sleep 3
net.show
set arp.spoof.targets {{ hmi_ip }},{
    windplant_ip }}
set arp.spoof.internal true
set arp.spoof.full duplex true
arp.spoof on
!iptables -t nat -F
!iptables -t nat -A PREROUTING -p tcp --dport
    502 -d {{ windplant_ip }} -j REDIRECT --to-
    port 8502
!iptables -t nat -A PREROUTING -p tcp --sport
    502 -s {{ windplant_ip }} -j REDIRECT --to-
    port 8502
set tcp.address {{ windplant_ip }}
set tcp.port 502
set tcp.proxy.address 0.0.0.0
set tcp.proxy.port 8502
set tcp.proxy.script {{ proxy_script }}
tcp.proxy on
active
```

5.2.2 Execute ICS attack

The final phase involves the real-time injection and manipulation of Modbus TCP packets. The attacker's script is designed to exploit the periodic weather updates sent by the HMI (every 10 seconds). By hijacking these legitimate packets, the attacker can inject malicious commands without altering the traffic volume or frequency, making detection by traffic analysis tools difficult. The attack logic is divided into four synchronized vectors.

Packet transformation

Technique: *Unauthorized Command Message [T0855]*

The HMI periodically writes the wind direction to Register 5 every 10 seconds. The attacker utilizes this specific packet to covertly enable the Manual Mode required to unlock the RPM control registers.

The script firstly intercepts packets carrying FC 6 (Write register). When the script detects a write to Register 5, it completely restructures the packet. It changes the Function Code from 06 (Write Register) to 05 (Write Coil), retargets the address to Coil 13, and sets the value to TRUE. This keeps Manual Mode active without requiring the attacker to initiate a new TCP connection.

Listing 5.2: Manual mode injection logic in destruction.js.

```
if (functionCode == 6 && isFromHMI) {
  var regAddr = (data[8] << 8) | data[9];

  // TRIGGER 1: Write to Register 5
  if (regAddr === 5) {
    log("[ATTACK]" + time + "] Trigger 2: Write
      to Reg 5 - Transforming to write TRUE to
      Coil 13");
    data[7] = 5; // Change FC to Write Coil
    data[8] = 0x00; data[9] = 13; // Target Coil
    13
    data[10] = 0xFF; // Value TRUE (ON)
    data[11] = 0x00;
    return data;
  }
}
```

Write operation hijacking for registers)

Technique: *Modify Parameter [T0836]*

With Manual Mode enabled, the speed control register 1 becomes writable. The script waits for the periodic wind speed update, which the HMI sends to register 2 every 10 seconds. The script intercepts this packet and redirects it to a different destination. Instead of writing

the wind speed to register 2, it redirects the write to register 1 and forces the value to 5000 (corresponding to 500 RPM). The objective is to maintain this overspeed condition for a sustained period. The PLC logic states that if the turbine spins at 500 RPM for a defined duration (simulating 10 minutes of stress), a critical gearbox failure occurs. In this situation, the PLC runtime must be restarted for the wind plant to become functional again.

Listing 5.3: RPM injection logic in destruction.js.

```
// TRIGGER 2: Write to Register 2
else if (regAddr === 2) {
    log("[ATTACK]" + time + "] Trigger 1: Write
      to Reg 2 - Hijacking to write 5000 to Reg
      1");
    data[8] = 0x00; data[9] = 0x01; // Target
      Register 1
    data[10] = 0x13; data[11] = 0x88; // Value
      5000 (500 RPM)
    return data;
}
```

Coil status masquerading

Technique: *Spoof Reporting Message [T0856]*

As mentioned, to successfully manipulate the turbine's speed, the attacker must first enable Manual mode (Coil 13). However, the operator must not be aware that this mode is active. To achieve this, the script intercepts all Read Coils (FC 01) responses. When the HMI polls the status of Coil 13, the proxy modifies the response to always return FALSE. This ensures that the Manual Mode indicator on the operator's log section remains off, even when the mode is actively engaged by the attacker.

Listing 5.4: Coil 13 masquerade in destruction.js.

```
var responseId = (data[0] << 8) | data[1];
if (responseId === transactionToSpoofCoil13 &&
    transactionToSpoofCoil13 !== -1) {
```

```
log("[DECEIVED] [" + time + "] Coil 13 read
    falsified - Forcing response to FALSE");
data[9] = data[9] & 0xFE; //FALSE
transactionToSpoofCoil13 = -1;
return data;
```

Write operation blocking for registers

Technique: *Block Command Message [T0803]*

To ensure the destruction process proceeds uninterrupted, the adversary implements a DoS filter targeting critical control points. The proxy is configured to silently discard operator commands directed at coils 11, 12, and 13 by modifying those packets with an invalid length. This effectively isolates the system by preventing the turbine from stopping, disabling the emergency stop mechanism, and blocking any attempt to switch the system into manual mode to regain control.

Listing 5.5: Coil write blocking in destruction.js.

```
// Block critical coils
else if (functionCode == 5 && isFromHMI) {
    var coilAddr = (data[8] << 8) | data[9];
    var blocked_coils = [11, 12, 13];

    if (blocked_coils.indexOf(coilAddr) !==
        -1) {
        log("[BLOCKED] [" + time + "] Coil "
            + coilAddr + " write blocked ->
            Corrupting packet");
        // Corrupt the packet by setting
        invalid length
        data[4] = 0;
        data[5] = 0;
        return data;
    }
}
```

Setting up this proxy, the attacker only needs to wait for the preset 10-minute timer to activate the internal destruction logic, which will

disable the wind turbine's functionality. At this phase, the operator must contact the turbine technician or vendor to arrange for repairs.

Alternative attacks

This section details alternative cyberattack scenarios, such as a supply chain compromise, which targets the integrity of the control logic prior to deployment, and a suite of specialized Man-in-the-Middle payloads. These payloads leverage established network interception to execute stealthy DoS and deception tactics, specifically tailored for use in CDX.

Supply chain compromise [T0862]

In addition to the network-based attack, I have also created a supply chain compromise scenario. In this scenario, the adversary compromised the PLC logic file (.st) before it was uploaded to the OpenPLC runtime. The idea behind this could be a legitimate firmware update from a wind plant vendor. The following malicious additions were inserted into the ladder diagram:

Safety override (T0880 – Loss of Safety): A hidden normally closed contact labeled SABOTAGE_ESTOP was added to the safety rung. By writing to this Modbus coil at address %QX1.8, the attacker disables the emergency brake's functionality, leaving the wind plant without this critical feature.

System fault generator (T0813 – Denial of service): Newly included timer block triggers the SYSTEM_FAULT coil every 15 seconds, after certain conditions are met. The attacker created another coil named ENABLE_RANDOM_FAULTS. When set to true, the timer starts running, and when it elapses, it triggers the SYSTEM_FAULT. This creates intermittent, non-reproducible failures designed to fatigue operators and desensitize them to genuine alarms.

Additional DoS attacks

I have created additional proxy scripts `r_rpm.js` and `w_coils.js` for the attacker's armory that can be utilized in a possible CDX scenario.

Those are not as destructive, but can be used as threats to the Blue team.

Manipulating the RPM register: In the `r_rpm.js` proxy script, the attacker falsifies the read request (FC 3) to the holding register 3 called SPEED. When this specific request is detected in the Modbus traffic, the response value is falsified to 0. This tricks the operator into thinking that the wind turbine is turned off, even when the logs do not say so. Other traffic remains unchanged.

Disable write to coils: When the attacker uses the `w_coils.js` script, they manipulate the ability of the operator to be able to start the turbine, start the manual mode, or press the emergency brake. When the script finds FC 5, it completely drops the packet. The rest of the communication remains unchanged.

6 Conclusion

This thesis aimed to design and implement a hardware testbed utilizing open-source technologies, which supports cybersecurity education focused on operational technology. Motivated by the increasing reliance of wind energy on networked control systems and the scarcity of safe, realistic environments for hands-on training, the work combined theoretical analysis of OT security with the practical construction of a testbed and its integration into the CyberRange^{CZ} platform.

The first part of the thesis established the field foundations. It introduced the core elements of OT and ICS, with emphasis on SCADA architectures, the Purdue model, and the role of PLCs as the primary interface between digital control logic and the physical process. The chapter also discussed open protocols such as Modbus TCP and justified their use as a suitable vehicle for educational security analysis. This chapter further contextualized these concepts in the wind energy domain by mapping SCADA layers to wind farm components and operational requirements.

Chapter 2 then examined the cyber threat landscape facing ICS, with a focus on wind energy infrastructure. It defined key security concepts, introduced standardized frameworks such as MITRE ATT&CK for ICS and the Industrial Control Cyber Kill Chain, and researched real-world incidents ranging from IT-centric ransomware campaigns to OT-specific vulnerabilities and APT groups.

Building on this background, Chapter 3 presented the design and implementation of the hardware testbed. The testbed models a single wind turbine generator, a substation, and a meteorological station at SCADA Levels 0–2. A monolithic architecture based on a single SBC was selected to keep the system compact, reproducible, and affordable, while still reflecting typical ICS design patterns. The concrete implementation utilized a Raspberry Pi running the OpenPLC runtime to control a DC motor, representing the wind turbine, via an L298N motor driver. Physical feedback mechanisms (LEDs, buzzer, emergency button) were added to provide interaction and state indication, reflecting feedback from earlier OT-focused cyber defence exercises.

On the software side, the PLC control logic was implemented using Ladder Diagrams in IEC 61131-3 constructs. The program includes speed regulation based on a simplified wind power curve, simulated power calculation, safety interlocks, alarm handling, and a destruction routine that models catastrophic mechanical failure due to sustained overspeed. The HMI was developed using Node-RED, providing operators with real-time visualization and control, as well as a simulated meteorological station that injects wind data into the PLC via Modbus TCP. By this testbed design, the requirements stated in the first chapter were fulfilled.

Chapter 4 integrated this physical testbed into the CyberRange^{CZ} platform. A sandbox definition was created to model a small OT network, including an operator HMI host and a user workstation, and to automate their provisioning via Ansible. A VPN-based approach was then used to attach the physical hardware to the virtual OT segment, enabling the combination of real devices with emulated infrastructure. This integration demonstrates that the testbed can be embedded into larger training topologies and reused across multiple exercises with minimal manual configuration.

Finally, Chapter 5 developed and analyzed a proof-of-concept cyberattack scenario, codenamed *Airbender*. The attack followed the two-stage ICS Cyber Kill Chain: Stage 1 covered intrusion into the IT environment and pivoting into the OT segment via mismanaged VPN credentials and an unauthenticated Node-RED instance; Stage 2 demonstrated a Man-in-the-Middle attack against unencrypted Modbus TCP communication. Using Bettercap and custom JavaScript proxies, the adversary selectively manipulated Modbus requests and responses to covertly enable manual mode, force the turbine into sustained overspeed, conceal the true system state from the operator, and block safety commands, which compromised the wind turbine. Additional scenarios, including supply-chain-style logic modification and focused denial-of-service attacks against specific registers and coils, were outlined as further examples of how the testbed can support realistic OT attack narratives.

Overall, the thesis successfully achieved its primary objective: it developed a functional, documented, and reproducible hardware testbed that integrates physical hardware, open-source OT software, and cyber range capabilities. The testbed is sufficiently realistic to support the

study of genuine ICS attack techniques and defensive measures, yet sufficiently constrained and inexpensive to be deployed in educational settings and future CyberRange^{CZ}-based exercises.

6.1 Future work

For future work, the system's physical and logical scale can be expanded. The current design focuses on a single turbine and does not include SCADA Levels 3 and 4. By extending the hardware to a multi-turbine configuration, it would be possible to implement a dedicated wind farm controller at Level 3 and aggregate data from multiple PLC instances, potentially allowing for analysis with ERP systems. On the HMI side, Node-RED can easily scale to create a multi-unit dashboard, allowing operators to manage multiple turbines instead of just one asset.

The integration with CyberRange^{CZ} can be deepened. The current VPN-based connection between the physical testbed and the virtual OT network operates at Layer 3, which imposes limitations on certain Layer 2 attack techniques, such as ARP spoofing. Future work could explore bridged VPN configurations or alternative connectivity mechanisms that preserve Layer 2 semantics, enabling a wider range of network-level attack and detection techniques to be demonstrated. In parallel, a reusable training module could be developed to turn the testbed into a standardized component of OT-focused cyber defence exercises.

Finally, systematic evaluation and testing of the testbed in educational practice remains to be conducted. This is also tied to the development of the entire CDX scenario, focusing on creating a fictitious organization and its associated network.

By following these directions, the hardware testbed developed in this thesis can evolve from a prototype into a versatile platform that supports research, teaching, and training on the cybersecurity of wind energy and industrial control systems.

A Archive content

The archive containing the source codes of the practical part of this thesis is of the following structure:

healthcheck: Folder with PLC health check application source code, requirements file, and system service file,

openplc: folder containing the baseline and compromised (supply chain) PLC logic in Ladder Diagram (PDF/XML) and Structured Text formats,

proxy: folder with TCP proxy scripts used for cyberattacks,

roles: folder containing two Ansible roles that automatically configure machines in the sandbox definition,

sandbox: folder with sandbox definition containing provisioning and topology definition for CyberRange^{CZ},

wireguard: folder containing WireGuard configuration files used to access the VPN server in the OT network.

Bibliography

1. DRAGOS INC. Year in Review [online]. 2025, pp. 5–8 [visited on 2025-12-13]. Available from: <https://www.dragos.com/ot-cybersecurity-year-in-review#download-the-report>.
2. CHEREPANOV, Anton. WIN32/INDUSTROYER: *A new threat for industrial control systems* [online]. 2017. [visited on 2025-11-09]. Available from: <https://autonomylogic.com/docs/2-4-physical-addressing/>.
3. U.S. DEPARTMENT OF ENERGY, WIND ENERGY TECHNOLOGIES OFFICE. *Protecting Wind Energy Systems From Cyberattacks* [online]. 2024. [visited on 2025-10-10]. Available from: <https://www.energy.gov/eere/wind/articles/protecting-wind-energy-systems-cyberattacks>.
4. MINISTRY OF INDUSTRY AND TRADE OF THE CZECH REPUBLIC. Podíl obnovitelných zdrojů energie na hrubé konečné spotřebě energie v letech 2010–2024 | MPO [online]. 2025 [visited on 2025-12-05]. Available from: https://mpo.gov.cz/cz/energetika/statistika/obnovitelne-zdroje-energie/podil-obnovitelnych-zdroju-energie-na-hrube-konecne-spotrebe-energie-v-letech-2010_2024--290765/.
5. HOŠEK, Petr; VOŘÍŠEK, Martin. *Elektroenergetika v ČR v roce 2024* [online]. 2024. [visited on 2025-12-05]. Available from: <https://oenergetice.cz/energostat/report/rocni/2024>. Section: Energetický report.
6. MINISTRY OF INDUSTRY AND TRADE OF THE CZECH REPUBLIC. Vnitrostátní plán České republiky v oblasti energetiky a klimatu | MPO [online]. 2024, pp. 2–4 [visited on 2025-12-05]. Available from: <https://mpo.gov.cz/cz/energetika/strategicke-a-koncepcni-dokumenty/vnitrostatni-plan-ceske-republiky-v-oblasti-energetiky-a-klimatu--285293/>.
7. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Operational Technology Security* [online]. 2025. [visited on 2025-08-03]. Available from: <https://www.nist.gov/programs->

- projects/operational-technology-security. Last Modified: 2025-03-26T16:41-04:00.
8. EDITOR, CSRC Content. *Supervisory Control and Data Acquisition* [online]. [N.d.]. [visited on 2025-11-17]. Available from: https://csrc.nist.gov/glossary/term/Supervisory_Control_and_Data_Acquisition.
 9. AUTOMATION, International Society of. *ISA-95 Standard: Enterprise-Control System Integration* [online]. [N.d.]. [visited on 2025-11-17]. Available from: <https://www.isa.org/standards-and-publications/isa-standards/isa-95-standard>.
 10. IYALUNAIDU, Sundareswaran. *Different Types of SCADA System Architecture* [online]. 2023. [visited on 2025-11-08]. Available from: <https://automationforum.co/different-types-of-scada-system-architecture/>.
 11. MORDOR INTELLIGENCE. *PLC Market - Share, Size & Growth - Programmable Logic Controller* [online]. 2025. [visited on 2025-12-09]. Available from: <https://www.mordorintelligence.com/industry-reports/programmable-logic-controller-plc-market>.
 12. ORGANIZATION, Modbus. *MODBUS APPLICATION PROTOCOL SPECIFICATION* [online]. 2012. [visited on 2025-11-17]. Available from: <https://www.modbus.org/file/secure/modbusprotocolspecification.pdf>.
 13. MOHAMMED, Abubakar; SAXENA, Neetesh; RANA, Omer. Wheels on the Modbus-Attacking Modbus TCP Communications. *WiSec '22: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2022, pp. 288–289. Available from doi: 10.1145/3507657.3529654.
 14. RENEWABLES, Green Wind. *About Wind Farms* [online]. 2023. [visited on 2025-12-02]. Available from: <https://greenwind.com.au/about-wind-farms/>.
 15. STAGGS, Jason; FERLEMANN, David; SHENOI, Sujeet. Wind farm security: attack surface, targets, scenarios and mitigation. *International Journal of Critical Infrastructure Protection* [online].

- 2017, vol. 17, pp. 3–14 [visited on 2025-10-05]. ISSN 1874-5482. Available from doi: 10.1016/j.ijcip.2017.03.001.
16. MICELI, Francesco. *Windfarm SCADA system characteristics - Wind farm BoP* [online]. 2012. [visited on 2025-10-28]. Available from: <https://www.windfarmbop.com/windfarm-scada-system-characteristics/>.
 17. PETRUZELLA, Frank D. Programmable Logic Controllers. In: *Programmable logic controllers*. Sixth edition. New York, NY: McGraw Hill Education, 2023, p. 2. ISBN 978-1-264-16334-2.
 18. PETRUZELLA, Frank D. Parts of a PLC. In: *Programmable logic controllers*. Sixth edition. New York, NY: McGraw Hill Education, 2023, pp. 4–6. ISBN 978-1-264-16334-2.
 19. PETRUZELLA, Frank D. Program Scan. In: *Programmable logic controllers*. Sixth edition. New York, NY: McGraw Hill Education, 2023, pp. 70–74. ISBN 978-1-264-16334-2.
 20. THE AUTOMIZATION.COM. *PLC Working Principle and PLC Scan Cycle* [online]. 2020. [visited on 2025-11-29]. Available from: <https://theautomation.com/plc-working-principle-and-plc-scan-cycle/>.
 21. KWOCO. *The Complete Guide to Programmable Logic Controllers (PLCs) - New Original HMI, Servo, PLC Supplier in China - KWOCO* [online]. 2024. [visited on 2025-10-28]. Available from: <https://kwoco-plc.com/complete-guide-to-plc/>. Section: Blog.
 22. ALVES, Thiago Rodrigues; BURATTO, Mario; SOUZA, Flavio Mauricio de; RODRIGUES, Thelma Virginia. OpenPLC: An open source alternative to automation. In: *IEEE Global Humanitarian Technology Conference (GHTC 2014)* [online]. 2014, pp. 585–589 [visited on 2025-10-11]. Available from doi: 10.1109/GHTC.2014.6970342.
 23. WORLD, Ladder Logic. *Ladder Logic Basics* [online]. 2021. [visited on 2025-10-12]. Available from: <https://ladderlogicworld.com/ladder-logic-basics/>.

24. INDUSTRY RESEARCH. *Programmable Logic Controller (PLC) Market Size* [online]. 2025. [visited on 2025-11-29]. Available from: <https://www.industryresearch.biz/market-reports/programmable-logic-controller-plc-market-110396>.
25. ALVES, Thiago; MORRIS, Thomas. OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research. *Computers & Security* [online]. 2018, vol. 78, pp. 364–379 [visited on 2025-10-11]. ISSN 0167-4048. Available from DOI: 10.1016/j.cose.2018.07.007.
26. PETER. *Structured Text Programming: A Step by Step Guide (With Examples)* [online]. 2015. [visited on 2025-10-28]. Available from: <https://www.plcademy.com/structured-text-tutorial/>.
27. ALVES, Thiago. *OpenPLC Editor* [online]. [N.d.]. [visited on 2025-10-28]. Available from: https://github.com/thiagoralves/OpenPLC_Editor/blob/master/README.md.
28. BEREMIZ. *MATIEC - IEC 61131-3 compiler* [online]. [N.d.]. [visited on 2025-10-28]. Available from: <https://github.com/beremiz/matiec/blob/default/readme>.
29. OPENPLC. *3.1 OpenPLC Editor Overview – Autonomy* [online]. [N.d.]. [visited on 2025-12-02]. Available from: <https://autonomylogic.com/docs/3-1-openplc-editor-overview/>.
30. SCIENCEDIRECT. *Threat Landscape - an overview | ScienceDirect Topics* [online]. [N.d.]. [visited on 2025-12-03]. Available from: <https://www.sciencedirect.com/topics/computer-science/threat-landscape>.
31. ROBINSON, Michael. The SCADA Threat Landscape | PDF | Security | Computer Security. In: *ICS-CSR 2013: Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research 2013* [online]. 2013 [visited on 2025-12-03]. Available from: <https://www.scribd.com/document/916752102/The-SCADA-Threat-Landscape>.
32. SENTINELONE. *Attack Surface vs Attack Vector: Key Differences* [online]. 2024. [visited on 2025-12-10]. Available from: <https://www.sentinelone.com/cybersecurity-101/cybersecurity/attack-surface-vs-attack-vector/>.

33. SEKER, Ensar; OZBENLI, Hasan. The Concept of Cyber Defence Exercises (CDX): Planning, Execution, Evaluation. In: *Research-Gate* [online]. 2018, pp. 1–9 [visited on 2025-12-03]. Available from doi: 10.1109/CyberSecPODS.2018.8560673.
34. MITRE ATT&CK. *ICS Matrix* [online]. [N.d.]. [visited on 2025-12-03]. Available from: <https://attack.mitre.org/matrices/ics/>.
35. ASSANTE, Michael J.; LEE, Robert M. *The Industrial Control System Cyber Kill Chain* [online]. [N.d.]. [visited on 2025-11-08]. Available from: <https://www.sans.org/white-papers/36297>.
36. POWER. *Malware in Modern ICS: Understanding Impact While Avoiding Hype* [online]. 2017. [visited on 2025-12-11]. Available from: <https://www.powermag.com/malware-in-modern-ics-understanding-impact-while-avoiding-hype/>.
37. RONCONE, Gabby; BLACK, Dan; WOLFRAM, John; SIMONIAN, Nick; MCLELLAN, Tyler; HALL, Ryan; PROKOPENKOV, Anton; JENKINS, Luke; PEREZ, Dan; AYLES, Lexie; WAHLSTROM, Alden. *Unearthing APT44: Russia's Notorious Cyber Sabotage Unit Sandworm* [online]. [N.d.]. [visited on 2025-11-08]. Available from: <https://services.google.com/fh/files/misc/apt44-unearthing-sandworm.pdf>.
38. JACOBSEN, Stine. Hackers make some Vestas' data public after ransomware attack. *Reuters* [online]. 2021 [visited on 2025-11-02]. Available from: <https://www.reuters.com/business/energy/hackers-make-some-vestas-data-public-after-ransomware-attack-2021-12-09/>.
39. BING, Christopher. Russia-based ransomware group Conti issues warning to Kremlin foes. *Reuters* [online]. 2022 [visited on 2025-11-02]. Available from: <https://www.reuters.com/technology/russia-based-ransomware-group-conti-issues-warning-kremlin-foes-2022-02-25/>.
40. ABRAMS, Lawrence. *Wind turbine firm Nordex hit by Conti ransomware attack* [online]. [N.d.]. [visited on 2025-11-02]. Available from: <https://www.bleepingcomputer.com/news/security/wind-turbine-firm-nordex-hit-by-conti-ransomware-attack/>.

BIBLIOGRAPHY

41. PETKAUSKAS, Vilius. *Deutsche Windtechnik hit with a cyberattack, a third on Germany's wind energy sector* [online]. 2022. [visited on 2025-11-02]. Available from: <https://cybernews.com/news/deutsche-windtechnik-hit-with-a-cyberattack-a-third-on-germanys-wind-energy-sector/>.
42. GREIG, Jonathan. *German wind farm operator confirms cybersecurity incident* [online]. [N.d.]. [visited on 2025-11-02]. Available from: <https://therecord.media/german-wind-farm-operator-confirms-cybersecurity-incident-after-ransomware-group>.
43. LYNKAAS, Sean. *Utah renewables company was hit by rare cyber-attack in March* [online]. 2019. [visited on 2025-11-02]. Available from: <https://cyberscoop.com/spower-power-grid-cyberattack-foia/>.
44. SHEAHAN, Maria; STEITZ, Christoph; RINKE, Andreas. *Satellite outage knocks out thousands of Enercon's wind turbines. Reuters* [online]. 2022 [visited on 2025-11-02]. Available from: <https://www.reuters.com/business/energy/satellite-outage-knocks-out-control-enercon-wind-turbines-2022-02-28/>.
45. CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY. *Nordex NC2 XSS Vulnerability | CISA* [online]. 2018. [visited on 2025-11-02]. Available from: <https://www.cisa.gov/news-events/ics-advisories/icsa-15-286-01>.
46. CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY. *XZERES 442SR Wind Turbine CSRF Vulnerability | CISA* [online]. 2018. [visited on 2025-11-02]. Available from: <https://www.cisa.gov/news-events/ics-advisories/icsa-15-155-01>.
47. AGENCY, Cybersecurity & Infrastructure Security. *Nation-State Threats | Cybersecurity and Infrastructure Security Agency CISA* [online]. [N.d.]. [visited on 2025-12-04]. Available from: <https://www.cisa.gov/topics/cyber-threats-and-advisories/nation-state-cyber-actors>.

48. GREIG, Jonathan. *China-linked hackers target organizations operating in South China Sea* [online]. [N.d.]. [visited on 2025-11-08]. Available from: <https://therecord.media/china-linked-hackers-target-organizations-operating-in-south-china-sea>.
49. MALHOTRA, Asheer; VENTURA, Vitor. *Talos uncovers espionage campaigns targeting CIS countries, embassies and EU health care agency* [online]. 2023. [visited on 2025-11-08]. Available from: <https://blog.talosintelligence.com/yourtrooper-espionage-campaign-cis-turkey-europe/>.
50. ME2SPACE. *Wind Turbine by me2space* [online]. [N.d.]. [visited on 2025-12-03]. Available from: <https://www.thingiverse.com/thing:1767153>.
51. VEMEKO. *The Differences between MCU and SBC* [online]. [N.d.]. [visited on 2025-10-11]. Available from: <https://www.vemeko.com/blog/the-differences-between-mcu-and-sbc.html>.
52. OPENPLC. *Writing OpenPLC drivers using PSM - DS18B20 on Raspberry Pi* [online]. 2023. [visited on 2025-12-11]. Available from: <https://www.youtube.com/watch?v=WYJM7Ss6SkA>.
53. CYBERRANGE CZ. *Designing ČEZ defense exercise: Preparation and planning (Part I) - CyberRangeCZ – Making Professionals Ready* [online]. 2025. [visited on 2025-11-07]. Available from: <https://www.cyberrange.cz/insights/cez-i/>.
54. OPENPLC. *2.4 Physical Addressing* [online]. [N.d.]. [visited on 2025-11-09]. Available from: <https://autonomylogic.com/docs/2-4-physical-addressing/>.
55. OPENPLC. *2.5 Modbus Addressing* [online]. [N.d.]. [visited on 2025-11-17]. Available from: <https://autonomylogic.com/docs/2-5-modbus-addressing/>.
56. LENOVO. *What is a Jumper and What Does It Do?* [online]. [N.d.]. [visited on 2025-12-02]. Available from: <https://www.lenovo.com/us/en/glossary/jumper/>.
57. COMPONENTS101. *L298N Motor Driver Module* [online]. 2021. [visited on 2025-12-02]. Available from: <https://components101.com/modules/l293n-motor-driver-module>.

58. RASPBERRY PI (TRADING) LTD. *Raspberry Pi Datasheets* [online]. 2024. [visited on 2025-12-02]. Available from: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>.
59. ROLAND_TMM. *L298N Driver Module for Arduino* [online]. [N.d.]. [visited on 2025-11-17]. Available from: <https://fritzing.org/projects/l298n-driver-module-for-arduino>.
60. EDGHILL, Lani. *What Rpm Is Required To Generate Electricity For Wind Energy* [online]. 2025. [visited on 2025-12-14]. Available from: <https://greenbusinessbarbados.com/what-rpm-is-required-to-generate-electricity-for.html>.
61. U.S. DEPARTMENT OF ENERGY, OFFICE OF ENERGY EFFICIENCY AND RENEWABLE ENERGY. *WINDExchange: Small Wind Guidebook* [online]. [N.d.]. [visited on 2025-12-11]. Available from: <https://windexchange.energy.gov/small-wind-guidebook#intro>.
62. WINDMILL TECH EDITOR. *Wind Turbine Components - Windmills Tech* [online]. 2022. [visited on 2025-12-11]. Available from: <https://windmillstech.com/wind-turbine-components/>. Section: Basics.
63. OPENSTACK. *What is OpenStack?* [online]. [N.d.]. [visited on 2025-12-05]. Available from: <https://www.openstack.org/software/>.
64. CYBERRANGECZ. *Terminology - CyberRangeCZ Platform* [online]. [N.d.]. [visited on 2025-10-26]. Available from: <https://docs.platform.cyberrange.cz/basic-concepts/terminology/>.
65. VEST, Joe; TUBBERVILLE, James. *Definitions | Red Team Development and Operations* [online]. 2022. [visited on 2025-10-12]. Available from: <https://redteam.guide/docs/definitions/>.
66. OPENVPN. *Ethernet Bridging* [online]. [N.d.]. [visited on 2025-12-05]. Available from: <https://openvpn.net/community-docs/ethernet-bridging.html>.

BIBLIOGRAPHY

67. MARGARITELLI, Simone. *arp.spoof* [online]. 2025. [visited on 2025-12-12]. Available from: <https://www.bettercap.org/modules/ethernet/spoofers/arpspoof/>.