Physical Lab

End-to-End Wind Energy Farm Architecture:
ODU team initiated the testbed development by following the wind energy form design by utilizing the following detailed architecture as shown in the following Figure 1. They started the testbed development by designing the scaled down 3D printed wind turbines and the emulated wind turbine models. The operations of these wind turbines were controlled by microcontroller-based Arduino boards. Additionally, the real-time weather information that was generated by the small weather station was also interfaced with the turbine controller mimic the operations of the real wind turbines. Figure 2 shows the end-to-end testbed device interconnectivity architecture.

Physical Model development and integration :
During this project period, the ODU team successfully achieved several key milestones related to the establishment of the scaled-down turbine model. The first major milestone was the establishment of prototype architecture. This model was designed to be flexible and scalable, accommodating small, medium, and large-scale models of wind turbines. The scaled-down model was carefully planned to replicate the essential functionalities and control mechanisms, while remaining adaptable to the constraints of a laboratory environment.

This period the team built a larger partnership with the 3D printing assets at VMASC and at Hampton Roads Biomedical Research Consortium where better and stronger materials exist for the scaled down prototype. From this partnership, 3d models and designs of typical wind turbines were shared and modified for the equipment available. Parts were 3d printed and assembly has begun resulting in a physical working prototype that can be used for initial testing and data gathering.

Motors and sensors are currently being controlled with Raspberry Pi microcontrollers, but the team is preparing to utilize commercial PLCs for future controller hardware. This will provide more options and opportunities for the team test and measure critical aspects of varying configurations and controls of the wind turbine. The ODU team has met regularly with the local company building the wind turbines off the east coast to better understand how our test environment can support the security of the future system being built. Information exchange with that company has helped inspire the architecture of the prototype which is greatly based on standard critical infrastructure command control-based systems. Below is the current architecture the team has set up in a physical prototype.

PLC/RTU Options, Open Source and Commercial :
The wind turbine models developed in this project were integrated into a sophisticated supervisory control system through an array of field interfacing devices, encompassing both industrial-grade programmable logic controllers (PLCs) and remote terminal units (RTUs) as well as custom-emulated PLCs. The industrial PLCs included the Siemens Simatic S7-1200, Schneider Electric TM221CE16T, and Omron CP1H-XA40DT-D, each bringing distinct capabilities to the system, while the emulated PLCs were implemented using Raspberry Pi units running OpenPLC software to manage both scaled-down and model turbines. The Siemens Simatic S7-1200, specifically the CPU 1214C model, operates with a compact design featuring onboard input/output configurations of 14 digital inputs at 24 V DC, 10 digital outputs at 24 V DC, and 2 analog inputs ranging from 0 to 10 V DC, powered by a 20.4-28.8 V DC supply with a program and data memory capacity of 100 KB. This PLC supports a variety of communication protocols such as PROFINET IO Controller and Device, SIMATIC communication, and open internet communication (optionally encrypted), alongside a web server function, with Ethernet protocols including TCP/IP, SNMP,

DCP, and LLDP, achieving a maximum transmission rate of 100 Mbit/s. Meanwhile, the Raspberry Pi-based OpenPLC setup provided a cost-effective, open-source alternative, programmed to oversee turbine operations, demonstrating a practical bridge between industrial standards and educational prototyping.

At the supervisory control level, the integration extended to a SCADA (Supervisory Control and Data Acquisition) system, enhancing oversight and management of the turbine network. The Omron CP1H-XA40DT-D PLC offered extensive communication versatility, supporting options such as CAN, CompoBus/S (Master and Slave), CompoNet Master, DeviceNet (Master and Slave), EtherCAT Slave, EtherNet/IP, Ethernet TCP/IP, MODBUS (Master and Slave), PROFIBUS DP (Master and Slave), PROFINET Master, and various serial communications (RS-232C, RS-422, RS-485), making it a highly adaptable component for interfacing with diverse turbine models. Similarly, the Schneider Electric TM221CE16T contributed robust communication services, functioning as a Modbus TCP client and server, DHCP client, and Ethernet/IP adapter, ensuring seamless data exchange within the network. This combination of industrial PLCs and the emulated Raspberry Pi PLCs facilitated a comprehensive control architecture, where the SCADA system could monitor real-time turbine parameters, issue commands, and ensure operational safety, such as initiating shutdowns when RPM thresholds were exceeded. The synergy of these devices underscored the project's aim to replicate real-world wind energy systems, blending established industrial technologies with innovative, accessible solutions to create a functional and educational supervisory framework.

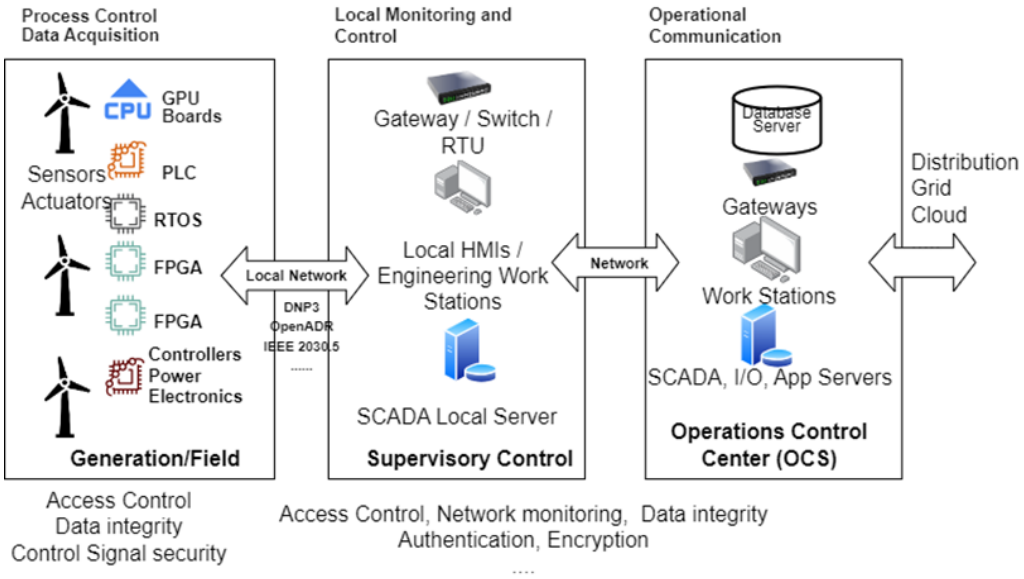


Figure 1. Overall Wind Energy Farm Testbed architecture.

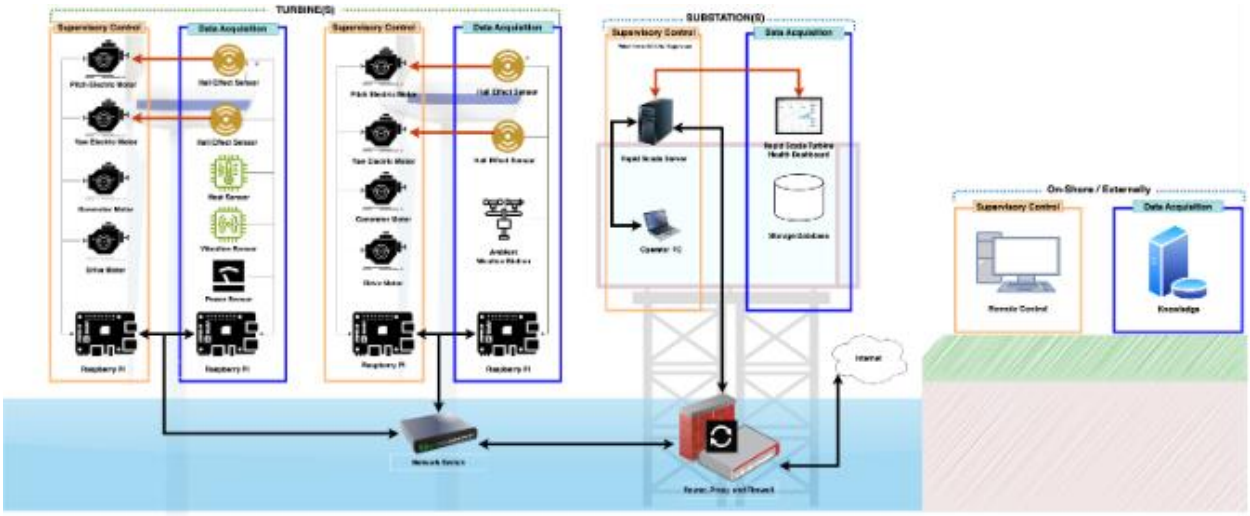


Figure 2: *Wind Energy Testbed SCADA System – Integrated turbine control, data acquisition, and secure remote monitoring via Rapid SCADA.*

Scada Server

Overview of Rapid SCADA

Rapid SCADA is an open-source industrial automation platform designed to monitor and control a wide range of industrial processes. It offers comprehensive features such as real-time data acquisition from various sensors and controllers, multi-protocol device communication including Modbus, OPC UA, MQTT, and SNMP, and a web-based human-machine interface (HMI) for system monitoring. The platform also supports configurable alarms and notifications for critical conditions, historical data logging for reporting, and extensive extensibility through custom drivers, scripts, and third-party integrations. Rapid SCADA is composed of three core applications: the Administrator for system configuration, the Communicator for device polling, and the Server for data storage, processing, and visualization.

Rapid SCADA Configuration for Wind Turbine Control : In the current implementation of Rapid SCADA for wind turbine systems, several Modbus device connections are configured to interface with programmable logic controllers (PLCs) that manage turbine operations. The setup enables the control of wind turbine functions by sending specific Modbus commands, such as 0 to stop and 1 to start a turbine. It also allows monitoring of turbine status by reading various operational parameters. The system maintains stable communication with multiple PLCs using Modbus TCP/IP, ensuring reliability and scalability across the wind farm infrastructure.

Modbus Integration in Rapid SCADA : Rapid SCADA integrates Modbus protocol support via the built-in Communicator module. This allows communication with both serial (Modbus RTU) and network-based (Modbus TCP) devices. Data mapping is flexible and can be configured to match the register layout of the PLCs. For advanced use cases, custom scripts written in C# can be added to custom drivers or calculated channels to manipulate and process Modbus data as needed.

Modbus Device Polling Options : To enhance communication efficiency, Rapid SCADA offers configurable polling options. These include read/write intervals to control how frequently data is updated, polling rates to avoid network congestion, and batch read/write operations to reduce the

number of Modbus transactions. These capabilities enable optimization of device communication while ensuring up-to-date data acquisition.

Modbus TCP Settings with Stunnel Encryption Gateway : For secure communication, Rapid SCADA can route Modbus TCP traffic through an encrypted Stunnel gateway. Communicator settings allow specification of the gateway's IP address and port, ensuring data is transmitted securely. Session timeout and reconnection parameters are also configurable, which helps maintain a stable and secure connection between devices and the SCADA system.

Connection Sequencing to Avoid Persistent Port Connection : To avoid issues with persistent TCP connections, Rapid SCADA supports sequential connection handling. This ensures that connections are not held open unnecessarily, reducing conflicts with other networked devices. Additionally, connection timeout management ensures that idle connections are properly closed and re-established as needed, and redundant connection support provides failover capabilities if the primary Modbus TCP connection is lost.

OPC UA Communication Protocol Integration in Rapid SCADA : Rapid SCADA supports OPC UA through a dedicated communication line and device configuration using the Administrator application. The OPC UA Client Module enables secure read and write access to OPC UA nodes. It supports secure connections using X.509 certificates, polling-based reads, writing via command interfaces, and browsing of server nodes during configuration.

OPC UA Security Features : OPC UA communication in Rapid SCADA supports multiple layers of security. Transport security is achieved through TLS, supporting security policies such as Basic256Sha256, Basic128Rsa15, and None for unsecured connections. Authentication using X.509 certificates is also supported, where both the client and server must trust each other's certificates. These certificates must be placed in the appropriate trusted directories on both systems. Additionally, OPC UA servers can require user authentication through usernames and passwords, which can be configured in Rapid SCADA's device settings.

OPC UA Polling Settings in Rapid SCADA : Polling frequency in OPC UA communication is configurable through parameters like poll interval, timeout, and retry count. The poll interval defines how often the system queries the server, while timeout and retry settings manage connection resilience. Signal mapping is done by associating Rapid SCADA channel numbers with OPC UA node IDs, providing a seamless link between server data and SCADA visualization.

OPC UA Command Settings in Rapid SCADA :

Besides polling, Rapid SCADA can issue commands to OPC UA servers. These include standard commands that write numeric or string values to specific nodes, as well as binary commands for sending raw byte data. This enables full control over OPC UA-enabled devices directly from the SCADA interface.

Rapid SCADA Device Authentication Workflow Using Proxy and SSI-Signed Credential: A secure authentication workflow has been implemented for Rapid SCADA using a custom Python-based proxy server and a device-side authentication client. This system enforces trusted device communication by using X.509 certificates and an SSI (Self-Sovereign Identity) signed credential. Communication with Rapid SCADA is only allowed after the device is successfully authenticated. In this system, a trusted Certificate Authority (CA) issues certificates to devices. Each device is provided with a private key and a certificate signed by the CA. To authenticate, the device creates a signed message containing the integer value 0. This message, along with the device certificate,

CA certificate, and SSI credential, is sent to the proxy. The proxy verifies that the integer is 0, ensures that the certificate chain is valid, checks that the device certificate is properly signed by the CA, verifies the signature of the integer using the device's public key, and confirms the validity of the SSI credential. Upon successful verification, the device IP is authorized, and subsequent Modbus TCP traffic from the device is forwarded to Rapid SCADA. The proxy server, written in Python, listens on port 1502 for incoming device connections. It parses incoming authentication packets, performs the necessary cryptographic verifications, and only forwards Modbus traffic to the SCADA server (typically on port 502) if the device passes all checks. This architecture enforces a robust security perimeter using identity-based access control.

Rapid SCADA Turbine Monitoring Dashboard :

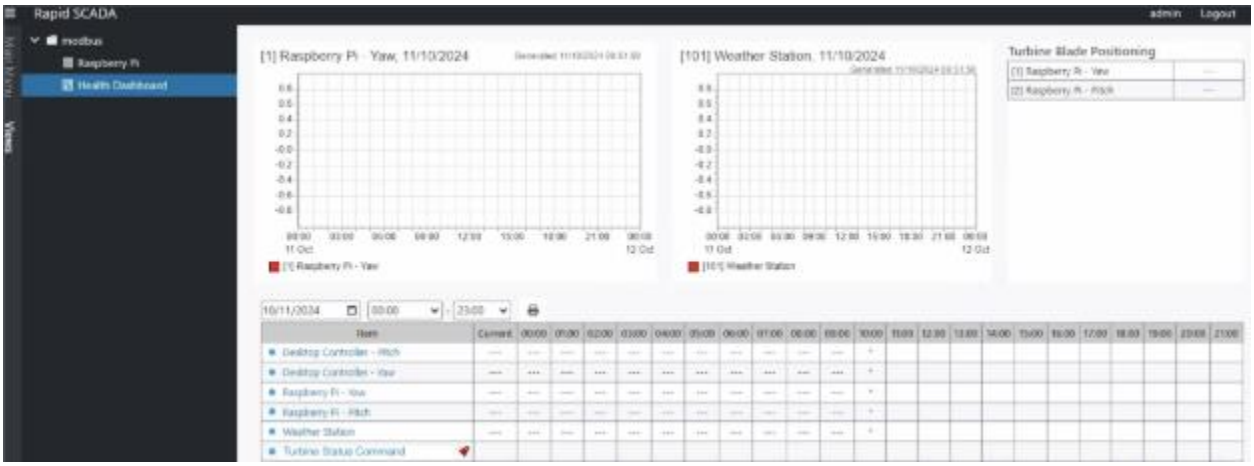


Figure 3. *Turbine Health Dashboard – Monitoring of PLCs, weather data, and turbine control commands via Rapid SCADA.*

Rapid SCADA features a Turbine Health Dashboard that displays turbine operational states such as Running, Stopped, or Faulted. It shows real-time values like pitch, yaw, wind speed from the weather station, and includes turbine status commands for the PLC. Graphs display real-time and historical performance trends, while alarms and notifications alert operators to abnormal conditions or maintenance needs.

Rapid SCADA Final Configurations : The Wind Energy Testbed Rapid SCADA system has been configured with multiple communication lines utilizing various protocols, including Modbus TCP, OPC UA, MQTT, and a REST API, all polling different devices and storing data simultaneously. Among the two Modbus TCP lines, one is connected to a locally hosted proxy server, which receives data from a client authentication sender script running on a separate device, while the other communicates with a Raspberry Pi that transmits simulated sine wave data. The OPC UA and MQTT communication lines operate as clients connected to locally hosted simulation servers, using ProSYS and MQTT Explorer respectively. Additionally, a REST API line connects to a custom driver module that queries an Ambient Weather Station. Each communication line in the system is associated with a specific channel, device, view, logs, and object within the Rapid SCADA configuration database.

The deployment is set to host the Webstation locally, although it has been tested successfully with hosting on a remote machine. To enhance functionality, the Automatic Control Module—used to trigger commands based on configured data thresholds—and the Dashboard plugin—used to build the Turbine Health Dashboard—were installed via the Rapid SCADA Store.

Since the system uses the free tier of these modules, a new access token must be acquired every 24 hours.

Lessons Learned :

- The Linux version of Rapid SCADA lacks the Administrator application, which complicated the configuration of communication lines. The Windows version is more user-friendly due to the inclusion of this tool.
- ModRSsim2 (Modbus Simulator) : Attempts to install ModRSsim2 may result in .dll errors. To resolve this, download and install the Microsoft Visual C++ 2010 Service Pack 1 Redistributable Package (MFC Security Update), then restart the computer.
- Rapid SCADA Troubleshooting Notes
- An "ILLEGAL DATA ADDRESS" error in the communication line log typically indicates an issue with the device template configuration. Verify that the correct element parameters are being used, that the data type is appropriate (default is ushort), and that the start element address is accurate.
- If a Modbus device refuses connection, you may need to adjust the request sequencing to increase the delay between requests or use "On Command Polling" to prevent overload. Ensure that Modbus TCP is configured on the correct port.
- If you encounter a "Resolver could not find assembly 'Newtonsoft.Json'" error in the communicator log, follow steps 6 and 7 in the Custom Driver Development section.
- If the communication line log file is not found, this may indicate the communication line has not yet been started. Refer to steps 8 through 11 in the Custom Driver Development section.
- For debugging custom C# drivers, use Log.WriteLine() in the Admin application. If uploading a configuration results in a "Service control command failed" error, restart the SCADA website in IIS by opening the IIS management console, selecting the SCADA website, and pressing the Restart button.

References

"Rapid SCADA Docs." *Introduction - Rapid SCADA*, rapidscada.net/docs/en/latest/software-overview/introduction. Accessed 10 Apr. 2025.

"Rapid Scada." *GitHub*, github.com/RapidScada. Accessed 10 Apr. 2025.

Wind Turbine Model Simulation: Our Simulink wind turbine simulation provides a model of the steady-state power characteristics of variable pitch wind turbines, as shown in Figure 5. This simulation captures the essential dynamics between wind speed, blade pitch angle, generator speed, and resulting mechanical output power without requiring users to navigate the complex underlying mathematical equations.

The simulation serves as a virtual testbed where users can analyze turbine behavior under various environmental conditions, test control algorithms, and get insight into their performance characteristics under different scenarios including different types of cyber-attacks. Users can input different generator speeds, pitch angles, and wind speeds to observe the resulting mechanical torque output. This capability allows for testing of wind turbine designs across their entire operational envelope-from startup mode to generating mode to pitch brake mode-within a safe virtual environment. Furthermore, it allows us to simulate the effect of various cyberattacks on the wind turbine system.

When integrated with our wind turbine testbed, these models enable real-time monitoring and control optimization based on actual operating conditions. The simulation can be extended to include electrical components, grid connections, and control systems, creating a complete

end-to-end model of wind energy conversion systems. The coefficients c_1 to c_6 are: c_1 = 0.5176, c_2 = 116, c_3 = 0.4, c_4 = 5, c_5 = 21 and c_6 = 0.0068. The *cp*-*λ* characteristics, for different values of the pitch angle β, are illustrated below. The maximum value of *cp* (cp_{max}= 0.48) is achieved for β = 0 degrees and for λ = 8.1. This particular value of *λ* is defined as the nominal value (λ_{nom}).

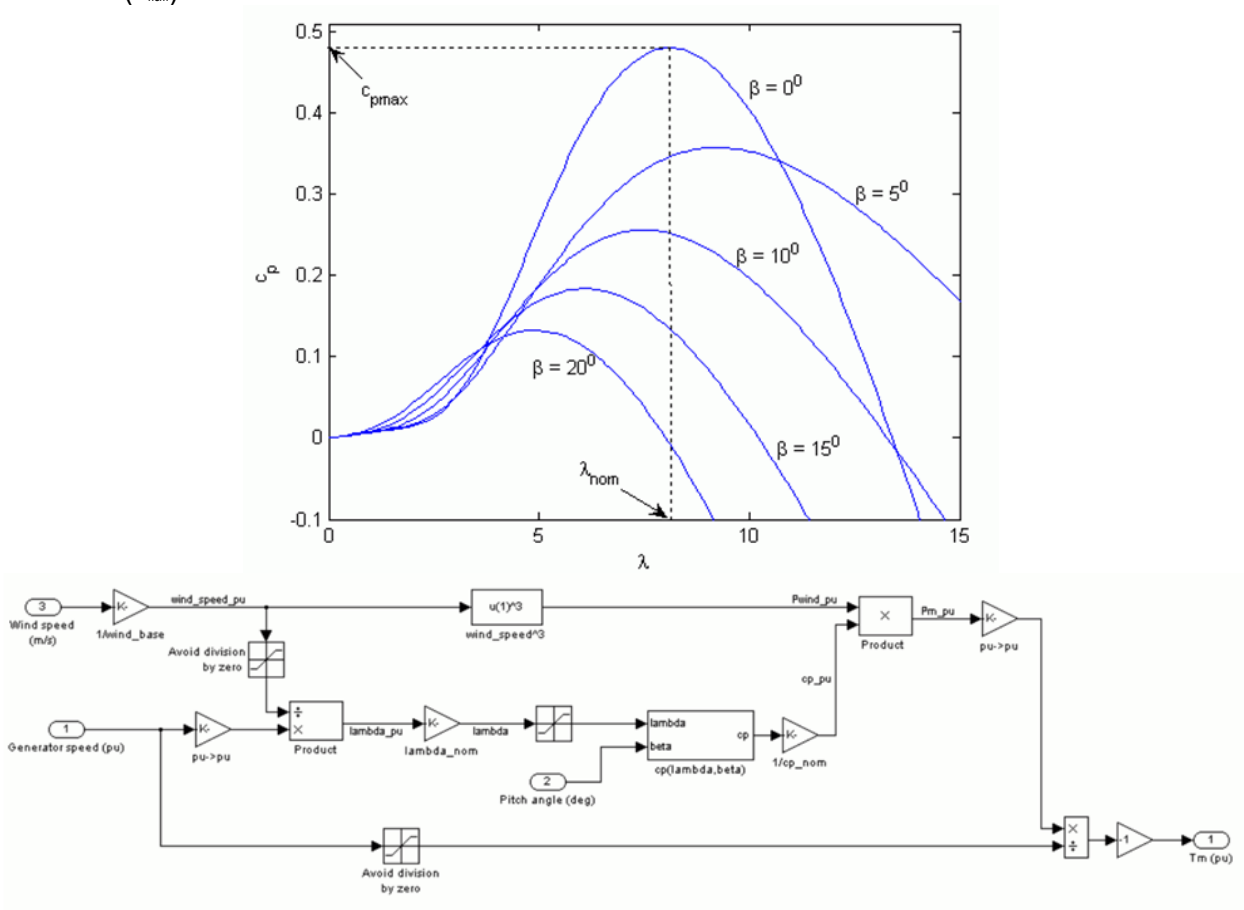


Figure 5: Implementation of our steady state model in Simulink and the constants used.

The Simulink approach provided a robust framework for preliminary analysis, allowing parametric studies of turbine behavior under diverse wind conditions. However, its limitations became apparent when attempting to interface with OpenPLC, an open-source programmable logic controller software hosted on a Raspberry Pi. The constrained communication capabilities between MATLAB, Arduino, and Raspberry Pi precluded seamless cross-platform integration. This necessitated a reevaluation of the implementation strategy, leading to the adoption of a hardware-centric approach in the subsequent phase.

Step 2: Rewriting the Model in Arduino Code : In order to overcome the interoperability challenges, the Simulink model was re-engineered into raw C code and executed over the Arduino. This transition to a microcontroller-based platform enhanced system flexibility and facilitated integration with external hardware, notably the Raspberry Pi. The Arduino setup also enabled real-time data visualization through an LCD display, providing immediate feedback on turbine parameters.

The Arduino-based model was architected to replicate key operational characteristics of industrial wind turbines. Wind speed, a primary input, was simulated using a potentiometer,

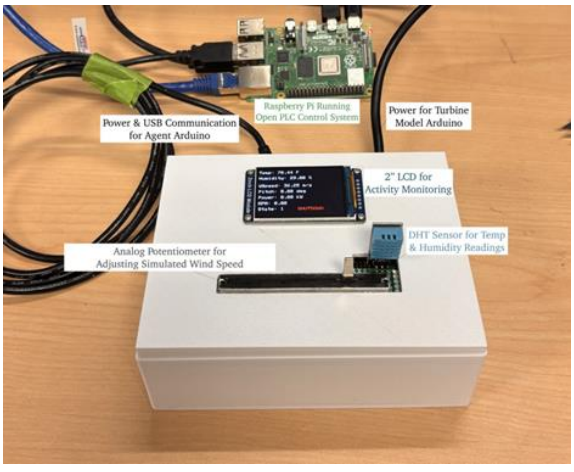
yielding a variable range of 0 to 37 meters per second (m/s). Environmental realism was further enhanced by incorporating a DHT sensor to measure ambient temperature and humidity, parameters that influence turbine performance in real-world scenarios. These sensor inputs were processed within the model to adjust turbine behavior dynamically.

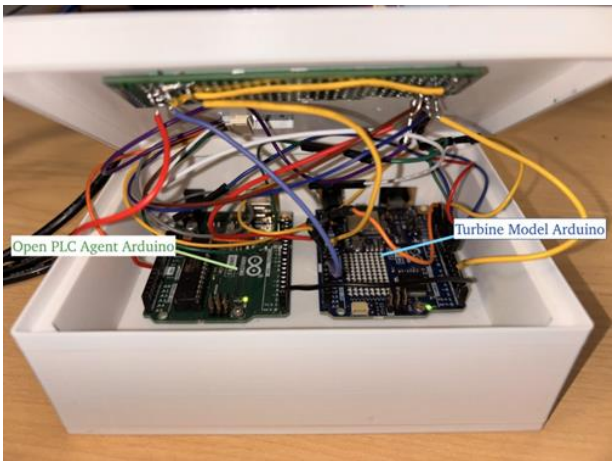
The design adhered to established wind energy standards, adopting a Tip Speed Ratio (TSR) of 7, which optimizes the relationship between rotor tip speed and wind velocity. A rotor diameter of 100 meters was assumed, reflecting a utility-scale turbine. Operational thresholds included a cut-in wind speed of 4 m/s, below which power generation ceases, and a rated wind speed of 12 m/s, at which the turbine achieves its maximum output of 2400 kilowatts (kW).
The blade pitch, or the angle of the turbine blades, was adjusted based on wind speed to keep the turbine running smoothly. Power output was calculated using basic physics equations, though they were kept simple. The focus wasn't on perfectly copying a real turbine's complex math but on showing how data could move between devices. This Arduino setup became the heart of the physical model, ready to connect to the next stage.

Step 3: Linking to Raspberry Pi and OpenPLC : The final step was to tie the Arduino model to a Raspberry Pi running OpenPLC, an open-source software that mimics industrial control systems (called PLCs, or Programmable Logic Controllers). This setup acted like a boss and worker team: the Raspberry Pi was the "Principal," giving orders, and the Arduino was the "Agent," following them. Together, they created a mini control system for the wind turbine.

The Raspberry Pi's job was to keep an eye on the turbine's RPM (revolutions per minute), which shows how fast the blades are spinning. If the RPM went above 35—a safety limit based on real turbine designs—the Raspberry Pi sent a shutdown signal to the Arduino. When this happened, the Arduino ran a "SHUTDOWN" routine, setting the RPM, blade pitch, and power output to zero, just like a real turbine stopping to avoid damage.

To make this communication work, the team used MODBUS, a popular method in factories and wind farms for devices to talk to each other. MODBUS let the Arduino and Raspberry Pi share data reliably, like RPM readings and control commands. It's also great because it can connect to bigger systems, like SCADA (Supervisory Control and Data Acquisition), which lets people monitor machines over the internet. This made the project feel like a small version of a real wind farm's control setup.For further details on OpenPLC configuration, please refer to the linked documentation: https://autonomylogic.com/docs/openplc-overview/





(a) External View (b) Internal View

Figure 6. Complete connectivity (external) and component wiring diagram (internal) of the emulated wind turbine prototype.

These physical and modeled generation side wind energy equipment (physical and Model turbines plus the PLCs and emulated PLCs/RTUs) are integrated with the SCADA server to offer a complete end-to-end Wind Energy Farm generation and control system. The ODU's testbed developed the Rapid SCADA system that is configured with multiple communication lines utilizing various protocols, including Modbus TCP, OPC UA, MQTT, and a REST API, all polling different devices and storing data simultaneously from the generation side control and communication devices. Among the two Modbus TCP lines, one is connected to a locally hosted proxy server, which receives data from a client authentication sender script running on a separate device, while the other communicates with a Raspberry Pi that transmits simulated sine wave data. The OPC UA and MQTT communication lines operate as clients connected to locally hosted simulation servers, using ProSYS and MQTT Explorer respectively. Additionally, a REST API line connects to a custom driver module that queries an Ambient Weather Station. Each communication line in the system is associated with a specific channel, device, view, logs, and object within the Rapid SCADA configuration database.

Capabilities and testing opportunities :

The end-to-end wind energy testbed offers the wind turbine models, integrating industrial-grade PLCs like the Siemens Simatic S7-1200, Schneider Electric TM221CE16T, and Omron CP1H-XA40DT-D with Raspberry Pi-based OpenPLC units, offers a robust platform for exploring cybersecurity capabilities and vulnerabilities in a controlled environment. By incorporating a variety of communication protocols such as PROFINET, Modbus TCP, EtherNet/IP, and TCP/IP, alongside serial options like RS-232C and RS-485, the system mirrors the connectivity found in real-world industrial control systems. This diversity allows researchers to investigate how data travels securely between the end devices, providing insights into potential weak points where unauthorized access might occur. The inclusion of a Rapid SCADA system further enhances this capability, enabling the simulation of remote monitoring and control scenarios that are common in modern energy infrastructures, thus offering a realistic setting to study how cyber threats could disrupt turbine operations or manipulate critical parameters like RPM and power output.

The testing opportunities within ODU's testbed are extensive, particularly in assessing the resilience of the wind turbine control network against cyber-attacks, as mentioned before. The students and researchers can simulate scenarios such as data interception or tampering by exploiting the Ethernet-based communications supported by the Siemens and Schneider PLCs, which handle high-speed data transfers up to 100 Mbit/s. The Omron PLC's wide range of protocol support, including PROFIBUS and DeviceNet, allows for experiments targeting specific communication pathways to see how an attacker might infiltrate the system through less-secured channels. Additionally, the Raspberry Pi units running OpenPLC provide a chance to test open-source software security, comparing its performance under attack to that of proprietary industrial systems. By integrating these elements with a SCADA layer, the testbed can replicate real-world incidents like unauthorized shutdown commands or false data injections, enabling the development and validation of protective measures to safeguard turbine operations against cyber threats in a practical, hands-on manner.

EM Testing (side channel)

Spectrum Analysis

The microcontrollers analyzed were the Arduino R3 and R4, along with the Siemens S7-1200 CPU-1214C AC/DC/Relay PLC. These 3 microcontrollers were compared side-by-side to see the differences in EM shielding and to compare similar programs emissions on different processors. The data was collected through the Rohde & Schwarz MXO-5 Oscilloscope which had the capability to collect near real-time spectrum data for ease of testing and analysis to view transient signals. The Arduinos were programmed utilizing the Arduino IDE, while the Siemens PLC utilized an educational license for Siemens TIA Portal.

Multiple experiments were conducted, measuring the EM radiation of both microcontrollers and motors to understand if any additional side channel information would be leaked through. For the Arduinos, a program which would repeatedly run two instructions alongside each other in rapid succession was programmed onto the boards. The output was measured utilizing the oscilloscope through a near-field probe. The Siemens PLC was programmed with a MODBUS server to simulate working as a PLC in a windmill which would take orders from an outside MODBUS client. The PLC was taken apart and had each component PCB moved away from each other and connected via wires. Each of these PCBs would be measured separately with a near-field probe. Two motors were tested. A large DC and small DC motor were tested one-by-one to view the radiation generated when powered.

The Arduinos tested leaked a significant amount of EM and was directly visible on an oscilloscope. Specifically, the loading and running of the program were able to produce varying spikes around the center clock frequency. These spikes would change depending on the types of instructions being run. When analyzing the PLC, it was clear to see that the local power electronics on the board were leaking a significant amount of EM. This would produce noise that would put the relevant signals below the noise floor. Due to this, direct CPU instructions were not able to be captured on the commercial PLC. The motor experiments showed some clear signals being generated by the wires carrying power.

In conclusion, consumer-grade microcontrollers such as the Arduino can produce EM signals that may be enough for understanding what may be happening inside the controller. Commercial grade PLCs have enough power electronics to handle the incoming power and convert it to a useful form to mask the signals being generated by the digital hardware on-board. However, changes to the surrounding environment could potentially be correlated with the changes in power to produce some results to understand how the internal system works on-board. Future work and research will focus on measuring the correlation between the connected voltage of the PLCs and the sensors and servos. The connected lines provide EM that is not always well shielded, providing opportunities for malicious attackers to extract information that can be used for targeted attacks.