# DIY Wind Turbines: A Low-Cost Smart ICPS for Educational Research

Matthew M. Y. Kuo*, Roopak Sinha†, Ramon Lewis‡, Charlie Cumming§
Auckland University of Technology, Auckland, New Zealand
{*matthew.kuo, †roopak.sinha, ‡ramon.lewis}@aut.ac.nz, §vzc1836@autuni.ac.nz

Robin Alarcon
IRT Antoine de Saint Exupery, Toulouse, France
alarconrobin02@gmail.com

Chandan Sharma
INRIA, Grenoble, France
chandan.sharma@inria.fr

***Abstract*—Industrial cyber-physical systems (ICPS) research combines many fields, including mechatronics, electrical, electronic, network, systems and software engineering. ICPS testbeds are often prohibitively expensive to purchase, store and maintain, and tied to vendor-specific tool chains, making them infeasible for small-scale experimental and student-led research.**

**With the rising popularity of 3D printing and ubiquitous computing, it is now possible to rapidly and inexpensively build model ICPS for lab research. This paper presents the "from-scratch" construction of an open-source 3D-printed smart wind turbine as a case study. We cover 3D modelling and printing of the wind turbine parts, the hardware-software interfacing and controller design using Arduino and the ESP8266 embedded WiFi controller. Interested readers can build a similar smart wind turbine by downloading the guidance and resource files linked to this document within days.**

**The smart wind turbine also serves as a rich ICPS research and education testbed to enable next-generation learning. In addition to discipline-specific tasks like incrementally optimising the 3D model, the interfacing and/or controller design, students and researchers can also use it for interdisciplinary research exploration. As an example, we show how the design and development of the smart wind turbine can be used to study, learn, and research requirements traceability. We employ graph databases to effectively model, represent, manage and trace requirements throughout the development of the smart wind turbine. The key outcome is a light-weight and efficient method for managing requirements using the Neo4j graph database implementation.**

***Index Terms*—ICPS, CPS, IoT, control software, machines, 3D printing, requirements traceability, wind turbine.**

## I. Introduction

Industrial cyber-physical systems (ICPS) contain multiple components, exist and operate on distinct spatial and temporal scales, as well as the physical and cyber planes [1]. They richly combine multiple fields of engineering and are seen as a driver of growth and the solution for many contemporary problems in society. They are being widely researched and taught across the globe. However, the availability of real ICPS in educational and small-scale research environments is limited, thanks mainly to cost and issues like vendor-specificity and lack of complete control required for research. Virtualisation and simulation-based technologies are useful but can be equally difficult to access in non-specialised settings or are less teaching-friendly because of their non-physical nature [2], [3].

This article presents rapid and cost-effective "from-scratch" construction of a model ICPS for education and interdisciplinary research. The model ICPS is a smart wind turbine that orients itself via a controller to optimise energy production. Typically, we use a wind vane for direction or Anemometer for wind speed. In this case, the wind turbine model has a DC motor generator for power generation and a stepper motor that can be controlled to change orientation. System construction had several parts: (a) 3D modelling, printing and assembly of the turbine, (b) interfacing the physical system with an Arduino controller through appropriate circuitry, (c) writing a PID loop-based controller where the Arduino controls the orientation of the wind turbine by reading the power output of the generator and (d) extending the controller by adding WiFi capabilities through ESP8266 and designing a communication schema combining multiple protocols to allow the controller to interact with other devices. The wind turbine, including a ubiquitous controller, was designed entirely by undergraduate students at Auckland University of Technology within a total duration of about 2 man-months. The overall cost of the commissioned model is less than $200 USD, with the controller (Arduino) being the most expensive component. All resources and guidance needed to commission the smart wind turbine within days are available freely on https://github.com/emsoftaut/DIY-Wind-Turbines.

We also use the development of the smart wind turbine to study and apply requirements traceability in ICPS. Requirements traceability is an interdisciplinary research topic for ICPS with requirements relating to one or more of the physical, electrical, electronic, networking and software aspects of a system. Requirements traceability in ICPS is becoming increasingly necessary as the reach of such systems is expanding rapidly, especially as they grow larger and more pervasive through the widespread adoption of IoT and CPS [4].

Traceability also involves linking initial requirements with more detailed sub-requirements. These details are well-captured using a graph structure. In an ideal situation, the traceability between requirements and design/development artefacts is completely automated, and comprehensive visualisation and analysis support exists. In the ICPS context, there is a need to develop appropriate standards to enable

the sharing and transfer of traceability information and design/development artefacts. In [4], virtual verification of designs or models against requirements is carried out to report on the extent to which a system design satisfies given requirements. This approach uses a light-weight RESTful protocol defined by the Open Services for Life-cycle Collaboration (OSLC) initiative. In [5], a graph database-enabled traceability framework linking security standards and design/development artefacts to requirements is presented.

An undergraduate software engineering student adopted the approach in [5] and used it for the complete end-to-end development of the smart wind turbine. The system requirements are managed through a Neo4j graph database [6] and recorded changes as the development progressed. A graph database allows structuring and manipulating data into a graph model [7]. Graph databases can handle highly interrelated and large data sets more efficiently than relational databases, enabling an intuitive grasp of the relationships. Graph query languages and database implementations are typically based on graph theory and algorithms, allowing for systematic/formal data manipulation. These features make graph databases attractive for traceability-related research [8]. Our brief exploration of ICPS requirements traceability showed that the windmill offers an exciting opportunity for learning, applying, and delving into interdisciplinary education and research.

The primary contributions of this paper are:

1) Resources to allow researchers and educators to commission the smart wind turbine ICPS for their work rapidly. The details of the system and its construction process are covered in Sec. II.
2) A case study on ICPS requirements traceability using graph databases based on the smart wind turbine ICPS. This case study is covered in Sec. III and shows how the system can be explored for interdisciplinary education and research.

## II. Construction of the Smart Wind Turbine ICPS

The smart wind turbine is a simple, student-driven model ICPS for use in educational and research projects. The key considerations in building this system were cost, time for commissioning, and the ability to use the system for interdisciplinary research. The project team decided to model and 3D print the physical part of the system and then control it via a ubiquitous controller. Details of this design are presented in the following sections.

### A. Physical Design - 3D Modelling and Printing

There are two main types of wind turbines: vertical axis wind turbines (VAWT) and horizontal axis wind turbines (HAWT). VAWTs feature blades with propellers facing the sky, rotating along the vertical axis. This design allows unidirectional wind power generation on the horizontal plane without steering. On the other hand, HAWTs have propellers that rotate along the horizontal axis, similar to a household fan. HAWTs require steering to face the wind directly for optimal power generation, which optimises energy production. Generally, HAWTs are more efficient in energy production compared to VAWTs when they are positioned directly facing the wind [9]. They are also usually cheaper to produce and maintain. They also present an intriguing control challenge and serve as a valuable testing ground for large-scale wind farm research, which is ideal for education and research. The physical part of the smart wind turbine system follows a HAWT design.

The HAWT for the smart wind turbine was modelled using SOLIDWORKS by an intern student with negligible initial training in 3D modelling and printing. Through support from the project team and 3D printing technicians, the student could generate the complete design and 3D print it within the equivalent of three weeks of full-time work. The overall 3D model is shown in Fig. 1.

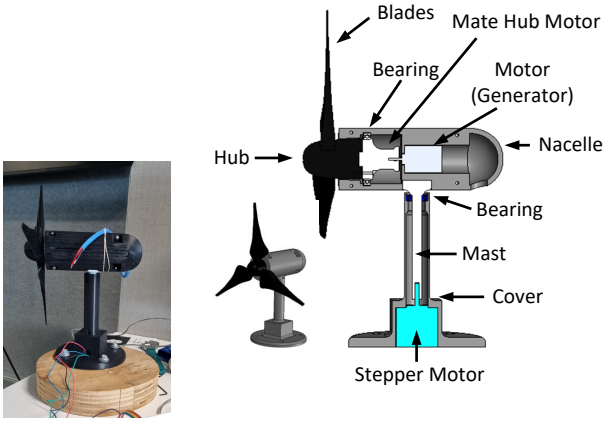


Fig. 1. Wind turbine

### B. System Integration

Figure 2 illustrates the integrated smart wind turbine system. The chosen technology stack has been widely embraced for instructing and grasping concepts related to embedded and web development. The wind turbine is controlled by an Arduino Uno, which transmits system information and receives commands through the ESP8266 Wi-Fi module. The ESP8266 Wi-Fi module utilises the MQ Telemetry Transport (MQTT) protocol to facilitate communication with the backend web server, implemented in Node.js. MQTT is a lightweight communication protocol commonly employed in various applications, particularly in industrial automation and the Internet of Things (IoT).

The front-end dashboard and user interface are developed using ReactJS, enabling seamless web browser communication through real-time streaming via HTTP WebSockets. Design details of each block will be presented in subsequent sections.

Fig 3 shows examples of the ReactJS-based panel for the smart wind turbine controller. In Fig. 3, the top tab displays the metrics of all connected turbine controllers, providing a summary of their current states. The metrics can be refreshed using the blue refresh button at the top left. Adjacent to

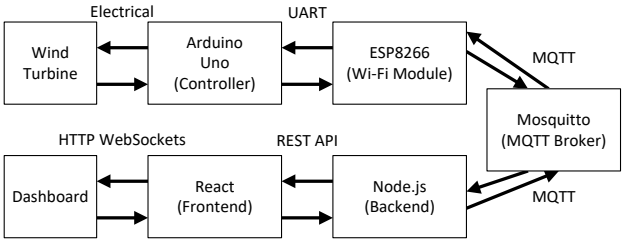


Fig. 2. System block diagram

this button is a dropdown menu where users can select a specific turbine by its ID. This selection is necessary for the functionality of the stepper and PID tabs, as they require control over a particular turbine.


Fig. 3. Web interface

The middle section of Fig. 3 presents the orientation stepper control tab. This tab offers various options for controlling the stepper motor's orientation. Users can choose continuous motion, per-step movement, or the PID-enabled algorithm to detect the optimal wind direction. The bottom section of Fig. 3 includes real-time graphics to track and visualise the turbine rotation and PID error, giving students and researchers an intuitive understanding of the internal operations of system.

The challenge is to ensure the wind turbine is aligned to the wind. The example shows that after several oscillations, an average centre point was calculated to determine the optimal direction value as shown in Fig 3.

Overall, the complete design and construction of the smart wind turbine met the target success criteria: it took 2 months to build, and can now be easily replicated within days (provided sufficient infrastructure) at a cost of less than $200 USD per installation.

## III. Case Study: ICPS Requirements Traceability in the Smart Wind Turbine

The smart wind turbine ICPS presented in Sec. II offers a foundation to explore various research problems. Researchers and students can leverage this prototype to investigate novel methods for control algorithms and physical performance optimisation. It can be used as a component in larger compositional systems like smart grids for power distribution to study interdisciplinary problems like self-adaptability, self-reconfigurability and reliability. This section shows how we can study ICPS requirements traceability using the construction of the smart wind turbine in detail.

### A. Requirements management

Requirements gathering and analysis is the key first step to development. Requirements are guided by the IEC 61400 standard for wind turbine design, manufacturing, and testing [10] and are recorded using a graph database implementation using Neo4J [6]. Requirements originating through the refinement or elaboration of initial requirements are added as sub-requirements and linked back to parent requirements via graph edges. Neo4j allows using the graph query language cypher to carry out traceability-related operations (like identifying links between requirements and system artefacts).

Fig. 4 presents the database schema employed by the student to represent wind turbine requirements, standards, and implementations. The diagram illustrates the following entities and relationship types.

- **Standard** represents industrial standards that provide the associated requirements. It encompasses three attributes: ID, Journal, and Description. The ID attribute provides a unique identifier for each standard, allowing for unambiguous referencing. The Journal attribute denotes the publication source where the standard is documented, enabling proper citation. The Description attribute provides a concise summary of the content encompassed by the standard.
- **Requirement** encompasses the individual requirements of the wind turbine system. It is characterized by two attributes: ID and Description. The ID attribute is a distinct identifier for each requirement, ensuring its uniqueness within the database. The Description attribute contains a comprehensive written statement that defines the specific requirement.
- **Implementation** represents the software implementations associated with the identified requirements. It comprises two attributes: ID and Commit Hash. The ID attribute provides a unique identifier for each implementation, facilitating its identification and differentiation. The Commit Hash attribute refers to the specific software implementation linked to the requirement, enabling traceability and version control. It is important to note that multiple

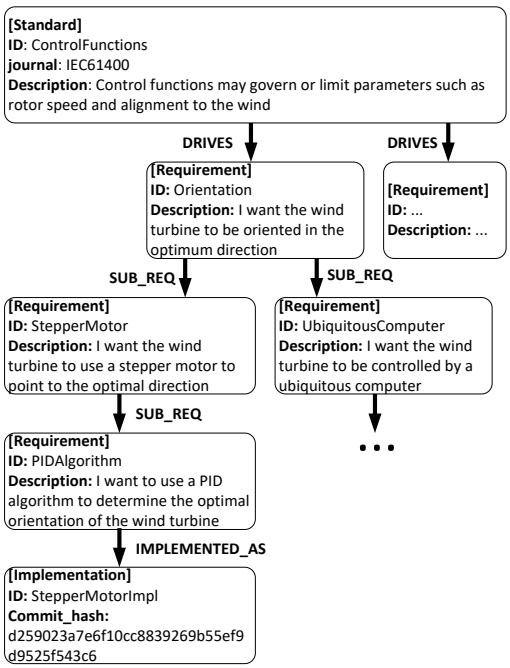


Fig. 4. Requirement representation in graph database

implementations may share the same Commit Hash, as a single software commit may address multiple requirements.

In addition to the entities, Figure 3 demonstrates three types of relationships:

- **`DRIVES`**: this relationship type signifies the influence of a particular standard on a requirement. It establishes the connection between a requirement and the standard that drives it.
- **`SUB_REQ`**: this relationship type represents the progression from an existing requirement to sub-requirements or new requirements that stem from it. It allows for the hierarchical structuring and tracking of requirement dependencies.
- **`IMPLEMENTED_AS`**: this relationship type denotes the association between a **Requirement** and its corresponding **Implementation**. When linked to a higher-level entity, this relationship implies the inclusion of lower-level sub-requirements within the same commit until the following **Implementation** entity is defined.

The windmill drove the student's interest to learn and adopt requirements engineering with the use of graph databases. Several traceability operations are automated, such as monitoring coverage of requirements and visualising links between requirements, standards and system artefacts for certification purposes.

## IV. Conclusions and Future Works

This paper presents the construction of an open-source smart wind turbine system that acts as a low-cost testbed for ICPS research. The system combines components such as electronics, PID controls, WiFi connectivity, and a web interface. The system is open to further refining, such as adding more sensors and utilising more advanced optimisation algorithms.

The smart wind turbine system can be rapidly constructed, costs significantly lower than commercial products, and can act as an ideal platform for education and research. Using a physical testbed not only ignites student curiosity but also empowers them to apply their acquired knowledge in diverse domains like requirements engineering. This exploration produced a graph database model of a subset of IEC 61400, captured system requirements for the smart wind turbine, and links all these requirements between each other and to system design/development artefacts.

Future design improvements include enhancing the graphical user interface on the control panel to provide intuitive visuals and effective monitoring of the system.

Subsequent endeavours will encompass extensive student projects focused on the windmill system. These projects will serve as a comprehensive trail, enabling us to meticulously evaluate and enhance the design based on the valuable feedback provided by the students.

Further research could include exploring control and optimisation strategies for large-scale wind farms within a smart, sustainable city by using multiple system installations. A collection of smart wind turbines can be used to explore and exploit synergies in a wind farm. This work may contribute to research to develop efficient and environmentally friendly energy solutions.

## References

[1] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: state of the art and future trends," *International journal of production research*, vol. 56, no. 8, pp. 2941–2962, 2018.

[2] S. I. Shafiq, C. Sanin, E. Szczerbicki, and C. Toro, "Virtual engineering object/virtual engineering process: a specialized form of cyber physical system for industrie 4.0," *Procedia Computer Science*, vol. 60, pp. 1146–1155, 2015.

[3] C. B. Vellaithurai, S. S. Biswas, and A. K. Srivastava, "Development and application of a real-time test bed for cyber–physical system," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2192–2203, 2015.

[4] A. Mengist, L. Buffoni, and A. Pop, "An integrated framework for traceability and impact analysis in requirements verification of cyber–physical systems," *Electronics*, vol. 10, no. 8, p. 983, 2021.

[5] A. Tanveer, C. Sharma, R. Sinha, and M. M. Kuo, "Tracing security requirements in industrial control systems using graph databases," *Software and Systems Modeling*, vol. 22, no. 3, pp. 851–870, 2023.

[6] Neo4j, "Neo4j - the world's leading graph database," 2012. [Online]. Available: http://neo4j.org/

[7] S.-T. Wang, J. Jin, P. Rivett, and A. Kitazawa, "Technical survey graph databases and applications," *International Journal of Semantic Computing*, vol. 9, no. 04, pp. 523–545, 2015.

[8] R. Sinha, B. Dowdeswell, G. Zhabelova, and V. Vyatkin, "Torus: Scalable requirements traceability for large-scale cyber-physical systems," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–25, 2018.

[9] M. K. Johari, M. Jalil, and M. F. M. Shariff, "Comparison of horizontal axis wind turbine (hawt) and vertical axis wind turbine (vawt)," *International Journal of Engineering and Technology*, vol. 7, no. 4.13, pp. 74–80, 2018.

[10] P. H. Madsen and D. Risø, "Introduction to the iec 61400-1 standard," *Risø National Laboratory, Technical University of Denmark*, 2008.