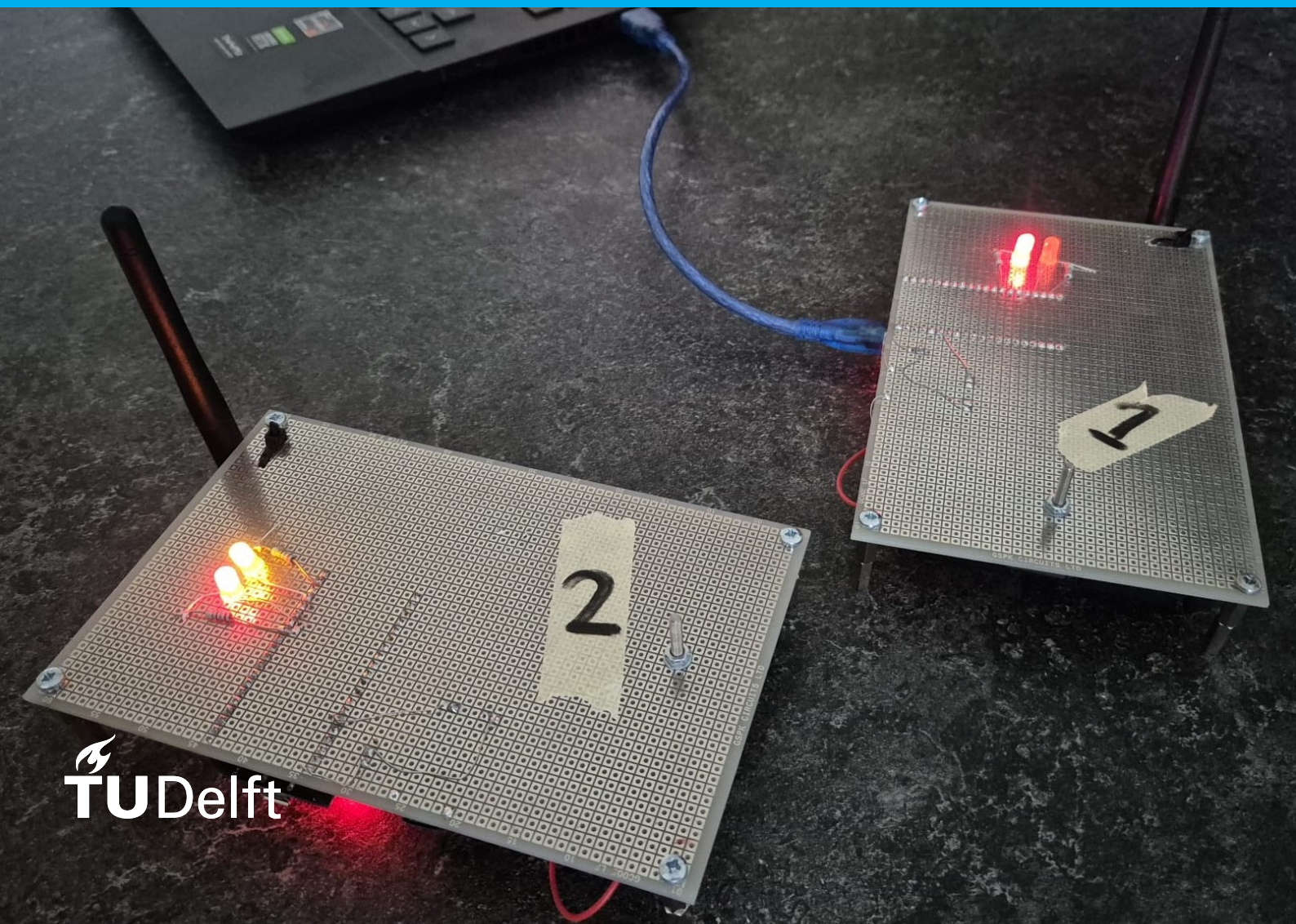
Visualizing Processes on Networks with Handheld Controllers

BSc Thesis

J. Van Ingen & T. Lipschart



Visualizing Processes on Networks with Handheld Controllers

BSc Thesis

by

J. Van Ingen & T. Lipschart

Project duration:	April 20, 2026 – June 26, 2026	
Thesis committee:	Prof. dr. ir. R.E. Kooij,	TU Delft, supervisor
	Dr. D. Cavallo,	TU Delft
	A.J. van Genderen,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

Networks are widely used in modern communication, social, and technological systems, but their structure and dynamic processes can be difficult to understand through static visualizations alone.
This thesis presents the design and implementation of a set of handheld wireless controllers that function as physical nodes in a network. By moving the controllers, users can create and modify network topologies interactively, making abstract network concepts more tangible for educational use.
The system consists of ESP32-based microcontroller nodes that communicate using the ESP-NOW protocol. Each node determines its connections to other nodes using received signal strength indicator measurements, which are smoothed and interpreted through a double-boundary method to improve connection stability. A coordinator node communicates with a host computer, where a Python-based user interface visualizes the network topology and node states in real time.
In addition to forming networks, the controllers can be used to run a dynamic process on the network: the Non-Consensus Opinion model. In this model, each node holds one of two opinions and updates its state based on the majority opinion in its local neighbourhood. The implementation was tested on several network structures, including line, circle, and island configurations. The experimental results show that the physical controller network can reproduce the expected behaviour of the opinion model while providing both local LED feedback and on-screen visualization.
The project demonstrates that handheld wireless controllers can provide an interactive and intuitive platform for teaching network topology and dynamic processes on networks. Future improvements may further increase communication efficiency, positioning accuracy, and scalability.

Preface

This thesis was written as part of our Bachelor's programme at Delft University of Technology. The project, titled Visualizing Processes on Networks with Handheld Controllers, was carried out between April and June 2026. Its goal was to design and develop a set of physical controllers that can represent nodes in a network and make abstract network concepts more interactive and accessible.

During this project, we worked on both the hardware and software aspects of the system. This included selecting suitable components, designing and building the handheld controllers, implementing wireless communication using ESP-NOW, developing a user interface, and testing the system with different network structures and the Non-Consensus Opinion model. The combination of theory, implementation, and experimentation made this project both challenging and rewarding.

We would like to thank our supervisor, Prof. dr. ir. R.E. Kooij, for proposing this project and for his guidance throughout the process.

We hope that this thesis provides a useful foundation for future work on interactive network visualization and that the developed system can contribute to a more intuitive way of teaching network theory and dynamic processes.

J. Van Ingen & T. Lipschart
Delft, June 2026

Contents

1 Introduction

1.1. Problem Definition

Networks play an important role in many modern technologies, such as communication systems, social media platforms, and transportation networks. Because of their widespread use and importance, network theory is taught in many university courses. Understanding how networks are structured and how processes behave on them is an essential part of this field.

To explain these concepts, lecturers often use slides, figures, and animations. While these visualizations help students understand network structures and dynamics, they can still feel abstract. Students can see what is happening, but they cannot directly interact with the network itself.

To make network education more interactive, Professor Robert Kooij proposed the development of a set of handheld controllers that can represent nodes in a network. The controllers communicate with each other using wireless technology and can form different network topologies. Connections between nodes are based on the distance between the controllers. This allows users to physically create and modify networks by moving the controllers around.

Besides representing network topologies, the controllers can also simulate dynamic processes that take place on networks. Examples of such processes include the spread of viruses, information, and opinions. This project focuses on implementing the Non-Consensus Opinion (NCO) model [17], which examines networks where multiple competing viewpoints coexist. In this model, opinions evolve as individual nodes dynamically update their states based on the opinions of their immediate neighbours. The model is mostly used by researchers, policymakers, and system designers. By visualizing this process on physical devices, students can gain a more intuitive understanding of opinion dynamics and the influence of network structure.

The goal of this project is to design and develop a set of controllers that can act as network nodes, establish wireless connections, and run the NCO opinion model. The resulting system aims to provide a more engaging and interactive way of teaching network-related concepts.

1.2. Structure of Thesis

The thesis is divided into ten chapters which all serve their own goals .Chapter 2 presents a state-of-the-art analysis of existing technologies relevant to building the wireless controller network.. In chapter 3 a program of requirements will be provided. The design choices made for the hardware of this goal will be explained in chapter 4. In chapter 5 the system architecture of the microcontroller network will be explained. Chapter 6 details the RSSI methods used by the nodes to estimate distances and establish wireless connections with each other. Chapter 7 will explain how the network is visualized. In chapter 8 the network stability will be tested. Chapter 9 presents the implementation and experimental results of running the dynamic NCO model on the system. In chapter 10 there will be a discussion, future work and a conclusion.

1.3. AI statement

For this project, AI was used responsibly to support the researchers throughout the process. It was used to improve the readability and structure of the report, as well as to assist with programming tasks during the project. AI was also used to generate some visual materials, such as images representing the network topologies. The use of AI was permitted in accordance with the BAP manual, as is readable in its chapter 1.5.

2

State of the Art

2.1. Communication protocols

Many communication protocols exist, but only a few are considered to be a choice when making a wireless sensing network with microcontroller nodes. In this state of the art analysis 5 communication protocols will be stated. The protocols are compared based on their suitability for a wireless microcontroller sensing network. Important criteria are power consumption, scalability, supported network topology, data rate, implementation complexity, and hardware compatibility. Since the nodes are battery powered, low power consumption is important. Since multiple nodes must communicate in the same network, scalability and topology are also important. A protocol that requires unnecessary infrastructure or adds too much complexity is less suitable for this project.

2.1.1. Bluetooth

Bluetooth is one of the most used communication protocols for wireless communication. Bluetooth is designed for continuous large data rates [19]. This comes with a degree of complexity which is not suited for the simplicity of a microcontroller sensing network. Bluetooth also does not have great scalability due to the connections being point-to-point. The strength of Bluetooth lies in it being implemented in all sorts of devices and platforms which makes it very accessible for all kinds of uses. A Bluetooth network building cell is also only able to have a network of 8 devices, of which one is the master node and 7 are slaves [15]. This would not be possible for a network of 10 nodes, where 1 node is the master node and 9 are slave nodes.

2.1.2. Wi-Fi

Wi-Fi is used for a lot of different appliances. In a home network Wi-Fi is used to connect internet using appliances to the router which is connected to the internet. Wi-Fi can be used for very high amounts of data [15]. This means there is a degree of complexity which is not required for simple sensing networks. Wi-Fi has a higher normalized power consumption than Bluetooth and Zigbee [15]. For a micro sensing node power consumption is important as it is not connected to a power grid.

2.1.3. BLE

BLE is a network protocol that can reuse a lot from the older Bluetooth network protocol. It is designed for low power consumption, which would make it perfect for sensing nodes that want to limit its total power consumption. BLE is limited to single point-to-point applications. It was designed to follow the star network topology, which is different from a mesh network [3]. The star network topology only allows communication between the master node and the slave nodes. It does not allow communication between slave nodes. There are certain solutions to support a mesh network using BLE as described in [3].

2.1.4. Zigbee

Zigbee is a network protocol that is widely used in home automation systems. The network can easily be expanded by adding new nodes, which can act as routers that increase the range of the network. The protocol distinguishes two nodes. A router node and a sensor node. The role of the router node is to always be on and able to expand the network range while the sensor node can be turned off until needed. Such a sensor node can be switched on when its sensor picks up information. Zigbee does have a longer transmission time than Bluetooth and Wi-Fi, due to a lower data rate [15]. According to [14], Zigbee will be a dominant force in the home automation market but will be used less in consumer electronics due to Zigbee providing very little advantage over its competitors.

2.1.5. ESP-NOW

ESP-NOW is a wireless communication protocol developed by Espressif Systems for ESP-series microcontrollers. The protocol is designed to be low-cost and low-power, making it well suited for systems that are not highly complex and where energy efficiency is important [13].
Unlike traditional Wi-Fi communication, ESP-NOW simplifies the communication process by merging several layers of the OSI model. As a result, communication takes place directly over the IEEE 802.11 physical layer without requiring a complete Wi-Fi network stack. Figure 2.1 illustrates how ESP-NOW simplifies the traditional OSI model.
This simplified architecture reduces communication overhead and allows devices to communicate directly using their MAC addresses. As a result, ESP-NOW is particularly suitable for simple microcontroller communication and wireless sensor network applications [13].
Research by [21] shows that ESP-NOW can achieve reliable indoor communication, while maintaining a high packet delivery ratio and good penetration performance. In addition, ESP-NOW has been successfully applied in mesh-based home automation systems, demonstrating its potential for scalable wireless sensor networks [12].

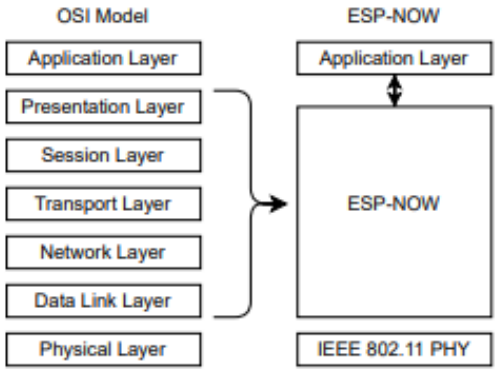


Figure 2.1: The ESP-NOW protocol stack as compared to the traditional OSI model.

2.2. Network topologies

In Wireless Sensor Networks (WSN), the topology dictates how nodes communicate and route data. For this project, the critical constraint is that all field data must eventually be routed to a single node connected to a PC for visualization. This constraint narrows the architectural choices to a few topologies:
The first option is a so called star topology. This is where every node is connected with one node in the middle. This offers a simple structure and low latency when sending a message. The bottleneck is that everything has to get through that one master node. This limits the range of the network to the range of the master node. Also the master node can be connected to a limited amount of nodes making this topology difficult to scale.
The other standard option is a mesh. In this topology nodes act as endpoints and routers. Although this decentralization increases communication latency due to the delays introduced by multi-hop routing, it significantly improves scalability. Data is relayed through neighbouring nodes, meaning no single

device is required to handle direct connections for the entire network, effectively avoiding the hard connection limits of a centralized master node.
Modern systems that have a similar constraint where data needs to be extracted from one node in the system mostly handle this with a hybrid of the two topologies above [7].

2.3. Distance measurement

For distance measuring where every node's location is unknown beforehand methods like measuring with light or sound are already off the table because the direction to aim is unknown. Two methods which can be used in such conditions are Ultra-Wideband(UWB) and received signal strength indicator(RSSI). UWB measures the distance between 2 points by sending a signal and measuring the time between the sending and receiving a signal back. The Accuracy is very high but it is expensive and complicated to implement [2].
RSSI is measured as the signal strength of the received signal. Using this power the distance can be calculated. The internal antenna's RSSI might be sensitive to the angle the boards are turned. This can be solved with an external upright dipole antenna. RSSI is cheap and easy to implement but less accurate than UWB. The difference in accuracy is however not very big. According to a test using mobile phones [20], UWB was only 0.75 % more accurate than RSSI. For this thesis this difference in accuracy is negligible. However, it is worth noting that RSSI accuracy degrades significantly more than UWB when devices are placed directly on or near the human body, due to signal interference. RSSI can be used inside as shown in [16].

2.4. Visualization methods

The network connections and NCO states are shown on the microcontroller nodes, but they also need to be visualized on a computer screen. The coordinator node sends the node IDs, connection status, and NCO states to the computer. The computer can then display the network, where each microcontroller is represented as a node and each connection as an edge.
A suitable option for this visualisation is Python with NetworkX. NetworkX is designed for creating, analysing, and visualising graph structures [8]. This fits the project well because the microcontroller network can naturally be represented as a graph. The NCO state can be stored as a node attribute, while connection information or RSSI values can be stored as edge attributes.
An alternative option is Node-RED, which is often used for IoT dashboards [5]. However, NetworkX gives more control over the graph structure and makes it easier to analyse and visualize changing network connections. This makes Python with NetworkX a strong option for displaying the live microcontroller network .

3

Program of Requirements

3.1. Controller Network

- The system consists of 4 to 10 controllers.
- All controllers must be identical in hardware and functionality.
- Controllers must be able to establish a connection when within 1 to 1.5 meters.
- A successful connection must be indicated locally on the microcontroller node.
- Connection data must also be transmitted to a central system.
- The central system must display which controllers are connected on a screen.
- The communication technology used to establish connections must be ESP-NOW.
- The controllers must not cost more than 25 euros due to the target group budget.

3.2. Experimental Functionality

- The controllers must work for different network setups. This must be tested and documented.

3.3. Dynamic Processes

- The microcontroller network must be able to run a dynamic process.
- The dynamic process tested in this project must be the NCO model.
- Opinion status must be visible on each controller and on the computer display.
- Data from the experiment should be accessible after the experiment.

3.4. Final Deliverables

- A set of 4 to 10 controllers.
- A complete user manual must be provided.
- The manual should enable users to operate the network independently, without assistance from the developers.

4

Hardware design

4.1. Hardware requirements

- **Microcontroller** The brain of our nodes must be a microcontroller. This microcontroller must be able to run a process on it
- **Antenna** To communicate the boards need to have an antenna which must be able to be oriented in a stable way on the microcontroller node.
- **Power supply** The microcontroller nodes must have their own power supply due to them being wireless nodes. They can not be connected to a wall outlet due to them being able to move around.
- **Visualisation** The microcontroller nodes will have a status of them being connected and having a certain status in the NCO opinion model. This needs to be visual on the nodes themselves. It needs to be visible in a classroom/lecture room setting.
- **Physical design** The microcontroller nodes need to be able to be handheld but also stand on their own in a stable way. All the components must combine into one complete circuit.

4.2. Component choices

4.2.1. Microcontroller

The chosen microcontroller is the ESP32-WROOM-32U [6]. The ESP module is a part of a ESP32 Devkit C4, which is perfect for a project like this due to including a lot of hardware that makes programming and powering the microcontroller easier. This devkit was chosen because of the cost and the ability to have some pins available for the LEDs and the power supply. Because there are not a lot of extra components there was no need for a to complex board with a lot of functionalities. The board has enough pins for the needed functionalities and has pins left for further expansions when needed. The devkit was also chosen to be able to have an external antenna mounted on it. This devkit and antenna combo is also able to communicate with the ESP-NOW protocol, which is talked more about in chapter 5.

4.2.2. Antenna

Several antenna options were considered for the wireless microcontroller node. In [1], four different antenna types are compared based on their RSSI values at different distances. The test uses a different ESP board but the antenna topology remains approximately the same. The results show that the dipole antenna achieved the strongest RSSI values compared with the other tested antenna types. Based on this comparison, a 2.4 GHz dipole antenna was selected for this design.
The chosen antenna is compatible with the selected ESP32 board and can be connected directly to its external antenna connector. Besides its good signal performance, the dipole antenna is also a simple and low-cost solution, which makes it suitable for a prototype consisting of multiple wireless nodes. A dipole antenna has an approximately doughnut-shaped radiation pattern. When the antenna

is mounted vertically, the strongest radiation is directed horizontally around the node. This is suitable for a classroom or lecture-room setup, where the nodes are expected to communicate with other nodes placed around them at approximately the same height.

4.2.3. Powersupply

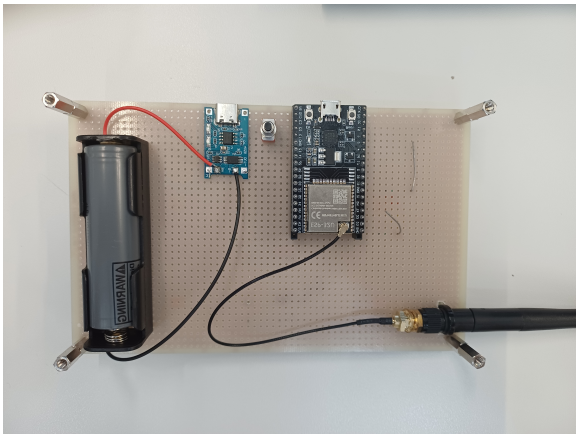
The board had multiple options to be charged. Namely through the micro-USB port, the 5 volt pin and ground pin, and the 3.3 volt pin and ground pin. The micro-USB port and the 5v pin both regulate the voltage that is supplied so that the voltage that is finally delivered to the ESP32 chip is perfect. The 3.3 volt pin needs to be supplied with exactly 3.3 volt, which would mean a more regulated power supply. Due to the controllers needing to be handheld, the power can not be supplied from the wall, which meant a battery which has a changing output voltage based on whether the battery is fully charged or not. This made the choice of the 3.3 volt pin harder and going from a battery to micro-USB is possible but not the cheapest and easiest option. That is why the choice was made for a battery connected to the 5 v pin. The voltage that is needed for the chip is 3.3 v. This makes it possible to supply the 5v pin with 3.7 volt, which makes the 18650 battery a perfect fit because it can also be charged, which makes usability for users easy. During testing the microcontroller node stayed operational as long as the battery stayed above the 3.6 volt threshold. The charging module chosen to power the battery is the TP4056. This module is the most commonly used option for 18650 batteries and is an affordable choice. It uses a USB-C port to charge the battery. The TP4056 was able to charge the battery up until 3.9 volts. Further more the power supply is turned on and off using a push button switch. Which is the most affordable option for the simple task it needs to fullfill.

4.2.4. Visualisation

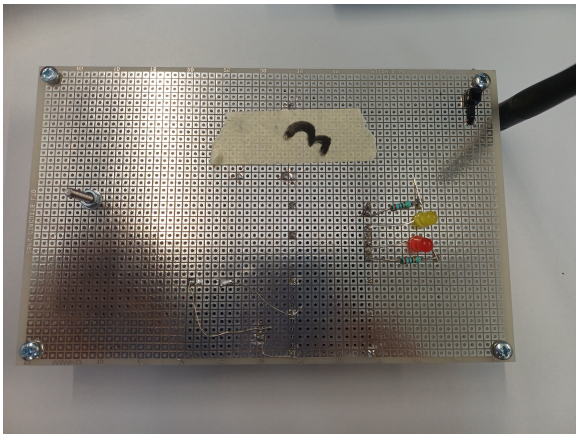
To visualise the connection status and the dynamic process status the microcontroller node has 2 LEDs, both connected to the ESP32 board with a 270 ohm resistor to limit current from the pins. For the connection status a red LED will turn on when the microcontroller node is wirelessly connected to another node and turn off when it is not connected. For the NCO model a yellow LED will turn on for option 1 and turn of for option 2. These different LEDs were chosen to be able to see from a classroom distance which LEDs are on to show the connection status and the NCO state. These two colour LEDs were the brightest colour options given to the research group. They also required a lower voltage than other colour options.

4.2.5. Physical design

In figure 4.1 the design of the microcontroller node is visible. All the components are mounted on a perfboard. This was the cheapest and most simple option for this project. A smaller perfboard would have been possible to fit all the components but such a board was not already available for the project. It consists of the 18650 battery in its holder. This battery is connected to the TP4056 charging module. The charging module is connected to the switch button and a ground pin of the ESP32 module. The ESP32 module has the external antenna mounted on the connector. The antenna is attached to the perfboard via a zip tie through 2 drilled holes. The red LED is mounted at pin 21 and a ground pin of the ESP32. The yellow pin is mounted at pin 22 and a ground pin of the ESP32. The entire board is lifted by using screws and two coupling nuts through 4 drilled holes at the corners of the perfboard. This choice was made so that the board would not lean on the battery and that it would be stable. An enclosure around the microcontroller node would improve its safety, robustness and appearance. To design an enclosure was not possible given the timeframe and the size of the research group. In figure 4.1 the microcontroller node is visible and in figure 4.2 the circuit that it corresponds to is visible.



(a) Bottom of microcontroller node



(b) Top of microcontroller node

Figure 4.1: Microcontroller node

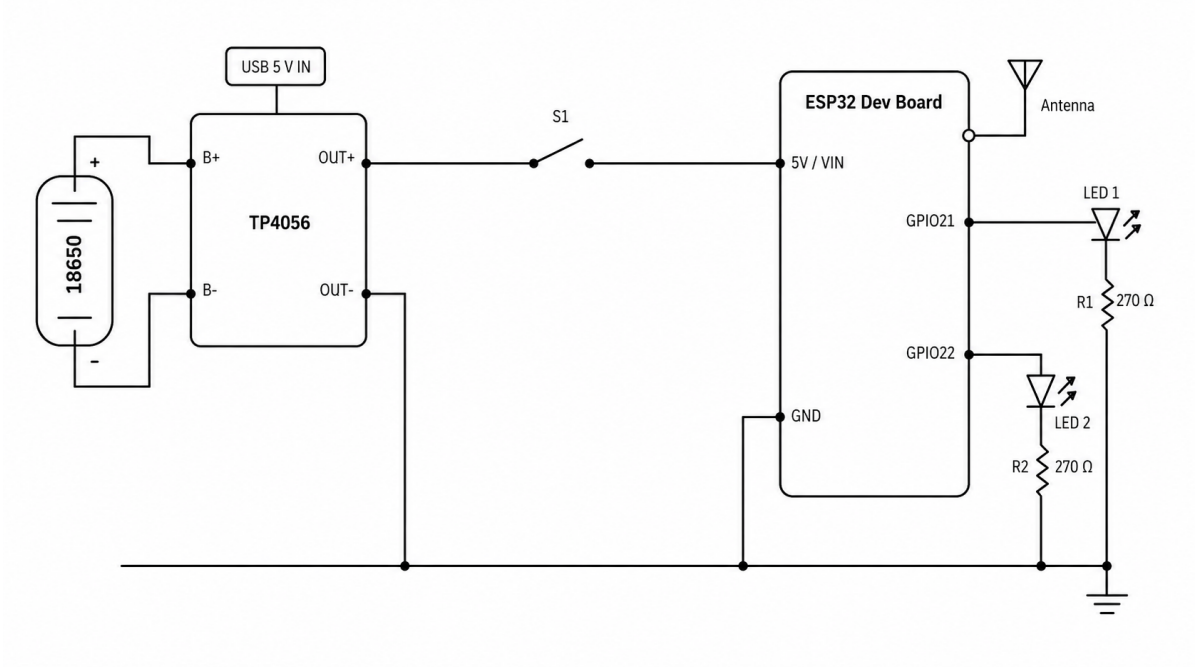


Figure 4.2: Circuit

4.2.6. Costs

A requirement of this project was to keep the cost under 25 euros per microcontroller node. With this hardware design the total cost per microcontroller node has been kept under 15 euros. This means that given the budget, the boards could be improved with for example a 3D-printed enclosure or other extra functionalities given more time.

5

System Architecture and Communication

Following the physical hardware design of the nodes detailed in Chapter 4, this chapter outlines the embedded software architecture and the communication protocols that enable the system to function. To create an interactive, real-time experience, the nodes must communicate efficiently, establish their relative spatial positions, and send this aggregated data back to the coordinator node. This chapter details the selection of the ESP-NOW protocol, the implementation of a hybrid network topology, and the synchronized three-stage communication pipeline that drives the system.

5.1. Communication Protocol: ESP-NOW

From the presented options for a protocol in section 2.1 ESP-NOW is chosen. This is because of the simplicity and freedom of implementation this communication protocol offers. In the state of the art section of this thesis the other communication protocols are shown to be less compatible with the program of requirements. The limitations that ESP-NOW brings, for example only 250 bytes message length limit, will not cause any problems for our product design.
A major advantage of using ESP-NOW is that it is largely implemented at the hardware level. Since it bypasses the overhead of a traditional Wi-Fi stack, the built-in hardware handles the low-level data transmission natively. As a result, the software implementation requires very little overhead, needing only basic setup functions to link nodes via their MAC addresses.

5.2. Network Topologies

Based on the theoretical topologies discussed in Section 2.2, a hybrid network design combining both mesh and star topologies was implemented to fulfill the system's functional requirements.
For the dynamic processes such as the distance estimation and node status sharing a **mesh topology** is utilized. This allows every node to independently estimate its distance to all surrounding nodes via Received Signal Strength Indicator (RSSI) measurements (which will be described in Chapter 6). This mesh structure enables users to physically interact with the nodes and dynamically shape the network connections based purely on physical proximity.
Conversely, for communication between the overall network and the central host, a **star topology** is employed. One specific node acts as a central coordinator. This star formation serves a dual purpose: first, it aggregates all the distributed mesh data to a single point for the host computer to visualize (described in Chapter 7), and second, it allows the host to uniformly broadcast instructions and user feedback back to all nodes.

5.3. System Workflow

For the system to operate efficiently and deliver a smooth user experience, the network operations must occur in a strictly ordered, synchronized loop. After initializing the ESP-NOW protocol and enabling the transceivers, the system enters a continuous three-stage pipeline. Because each stage has different routing and data aggregation requirements, the functional topology shifts between the stages. Figure 5.1 illustrates the specific time slots allocated to prevent packet collisions, while Figure 5.2 provides a comprehensive flowchart of the logic driving the host, the coordinator node, and the other nodes.

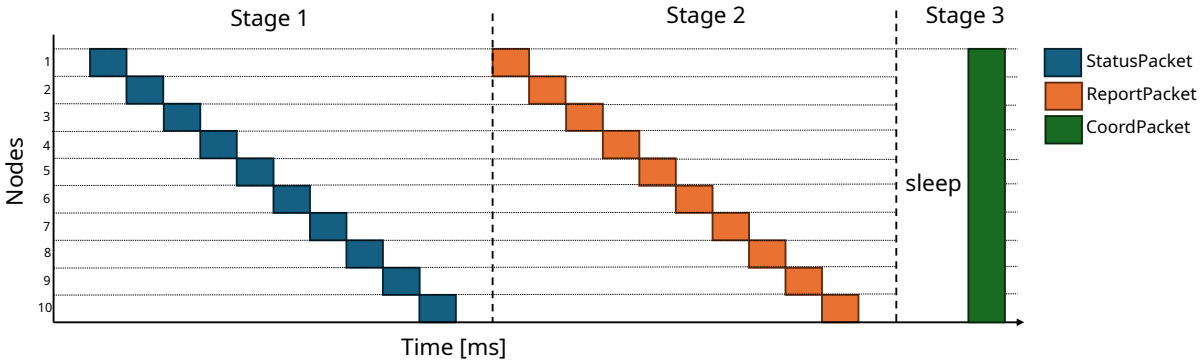


Figure 5.1: Timing diagram of the system

5.3.1. Stage 1: Status Broadcasting

In the first stage, every node broadcasts its current state. This payload includes its estimated 2D coordinates and its current status within the Non-Consensus Opinion (NCO) process (the implementation of this will be shown in Chapter 9).
To ensure network stability, each node is assigned a specific time slot for transmission[4]. Each timeslot is 20 milliseconds. This timeslot approach guarantees that no two signals collide. During this cycle, every node actively listens and saves the broadcasted messages from its peers. Once all nodes have transmitted, each node evaluates the received data to determine which peers are within its physical range, subsequently updating its own spatial coordinates(Section 6.3) and NCO status based on this new information.

5.3.2. Stage 2: Data Aggregation

At the beginning of Stage 2, all nodes possess an updated local state that is currently unknown to the central host. The newly calculated coordinates, NCO status, and local connection maps must therefore be transmitted to the coordinator node.
Using the same collision avoidant timeslot mechanism as Stage 1, each node sends its data directly to the coordinator. Once the coordinator aggregates the entire network state, it transmits this collective payload to the host computer via UART. The specifics of this Host-Network communication and the subsequent data visualization logic are detailed in Section 7.1.

5.3.3. Stage 3: Verification, Instruction and Synchronization

The final stage ensures the network cycle is completed successfully and synchronizes the next cycle. First, the coordinator cross-verifies bidirectional connections. For instance, if Node 2 reports a connection to Node 3, the coordinator verifies that Node 3 also reports a connection to Node 2. If a discrepancy is detected, the network step has to be redone.
The Coordinator node waits until the total time of the step is half a second. This sleep time will be approximately 60 ms. Following verification, the coordinator node broadcasts a final instruction packet (`CoordPacket`) to all nodes. This message dictates whether the previous step must be restarted or if the network should proceed. Furthermore, it delivers any incoming feedback or parameter changes initiated by the user from the host interface. Once this synchronization message is received, the nodes simultaneously transition back to Stage 1, maintaining the synchronous loop.

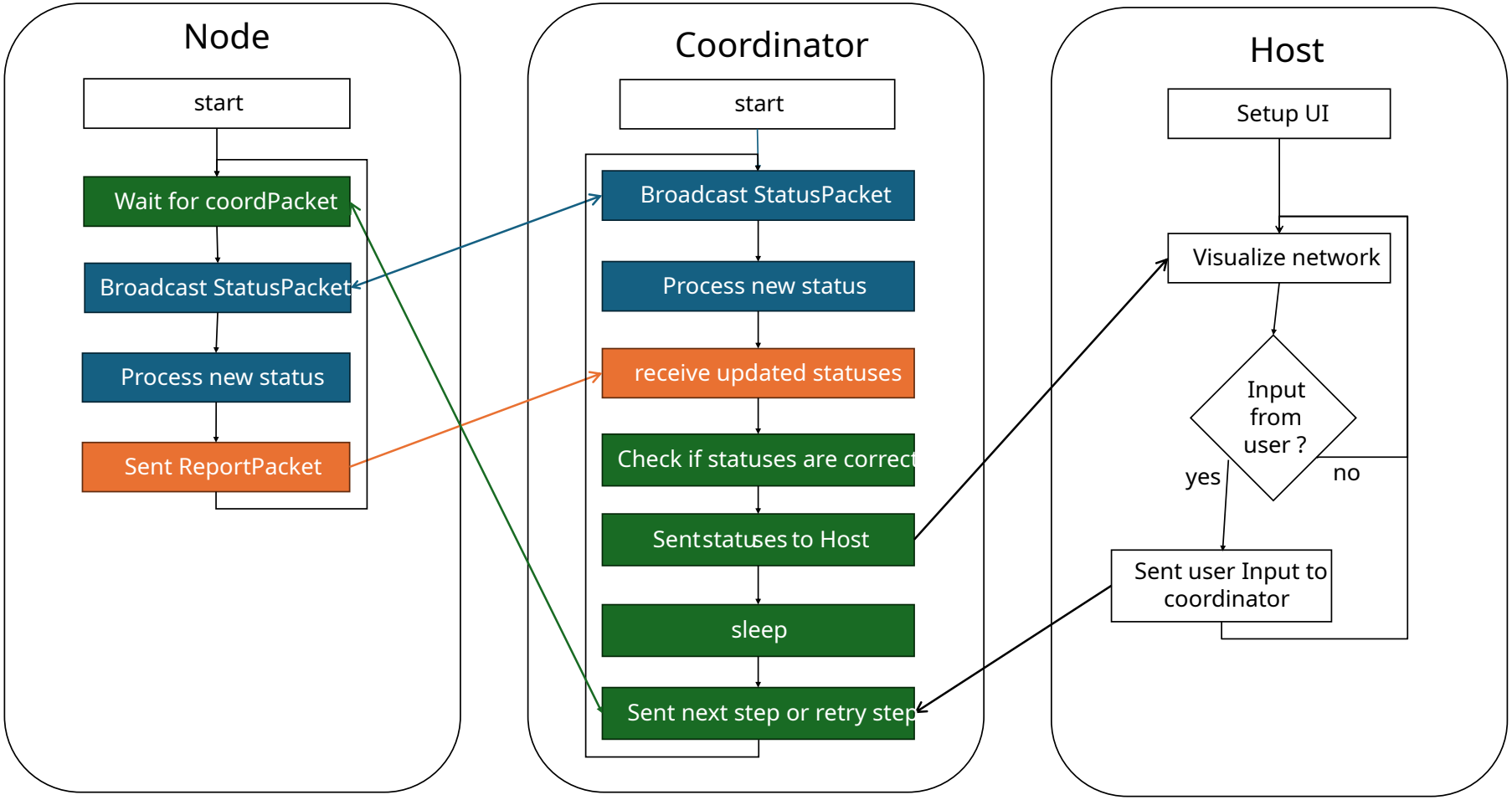


Figure 5.2: Pipeline of the Host, coordinator and the nodes. The coordinator controls the other nodes. The nodes only start upon receiving the Stage 3 message from the coordinator. The Coordinator and Host are asynchronous. The Host immediately acts upon receiving updates from the coordinator whereas when the coordinator receives an input from the Host it deploys the change in the system initiated by the Host at the end of stage 3.

6

RSSI based Network Formation

To establish physical connections and map the network topology, the system relies on the Received Signal Strength Indicator (RSSI). While Ultra-Wideband (UWB) offers higher precision, RSSI was selected to minimize implementation complexity and hardware costs(Section 4.2.6).

RSSI is a physical layer measurement representing the received power in decibel-milliwatts (dBm). Because the ESP32 calculates this as a snapshot of total energy capturing both the intended signal and environmental reflections the raw data is highly susceptible to noise. This chapter details how this noisy estimation is translated into stable connections and a two-dimensional network map.

6.1. Signal Processing and Distance Estimation

To mitigate the high fluctuation of raw RSSI signals, a common practice is to average RSSI signals over time [9]. Our system applies an Exponential Moving Average (EMA). This ensures that the physical movement of nodes is registered quickly, while anomalous spikes are filtered out to prevent erratic distance estimations. The smoothed signal is calculated as:

$$RSSI_{\text{smooth},t} = \alpha \cdot RSSI_{\text{raw},t} + (1 - \alpha) \cdot RSSI_{\text{smooth},t-1} \tag{6.1}$$

α is chosen to be 0.25 which smoothed the raw RSSI data enough to where there were no massive spikes but is not too low where it would cause the signal to react slowly to changes in the network.
During the network's Status phase, nodes exchange these smoothed values. The signal strength is then mapped to an estimated physical distance d_m using the log-distance path loss model:

$$d_m = 10^{\frac{RSSI_{1m} - RSSI}{10 \cdot n}} \tag{6.2}$$

In this model, $RSSI_{1m}$ represents the calibrated signal strength at exactly one meter (-23 dBm), and n is the path loss exponent (2.2), which accounts for standard indoor environmental interference[16]. To prevent the topology from scaling infinitely due to signal drops, d_m is strictly bounded between 0.1 and 20.0 meters.

6.2. Connection Stability

Even with EMA smoothing, signal fluctuations around the exact connection threshold can cause network instability. If a single boundary were used, two nodes could easily disagree on their connection status—one node might read the signal slightly above the threshold, while the other reads it below. This disagreement would disrupt dynamic processes running on the network.
To solve this asymmetry a double-boundary connection is established depicted in Figure 6.1. The nodes form a connection when RSSI is above the upper boundary of -23 dBm and disconnect when below the lower boundary of -29 dBm. These values are chosen because -26 dBm is at the desired distance of 1 meter(Section 8.2). RSSI can fluctuate 3 dBm based on the rotation of the board. Therefore, the

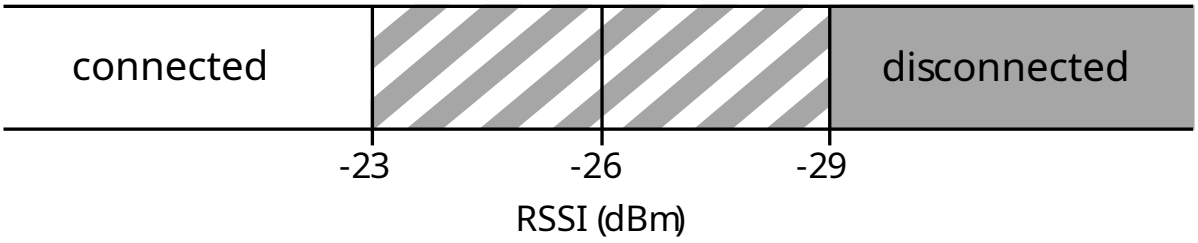


Figure 6.1: Double-boundary connection. Nodes connect when RSSI is higher than upper boundary and disconnect when RSSI is lower than lower boundary

bounds are 3 dBm away from the middle point.
In the case one of the nodes goes over a boundary but the other node does not this gives a similar problem as in the beginning of this chapter where the nodes are disagreeing on whether they are connected.
This is resolved by falling back on the middle point as a boundary. Because the variance between two communicating nodes is rarely larger than 3 dBm, this midpoint acts as a forced consensus, ensuring both nodes always agree on their connection state.

6.3. Spatial Derivation

To effectively visualize the network topology(Visualization is shown in Chapter 7), each node must derive a two-dimensional coordinate relative to its peers. Due to the lack of absolute positioning hardware, the system relies on a decentralized, iterative algorithm based on RSSI distance approximations.

6.3.1. System Initialization

To prevent mathematical singularities during the initial spatial calculation (e.g., division by zero when calculating directional vectors), the nodes cannot start at the exact same coordinate. The system initializes the network in a deterministic circular distribution.
The coordinator node acts as the static origin point to anchor the graph:

$$v_x^{(1)} = 0, \quad v_y^{(1)} = 0, \quad v_e^{(1)} = 0 \tag{6.3}$$

where v_e represents the spatial confidence error.
For all non-coordinator nodes, the initial positions are distributed along a circle with a radius of 2.0 meters. The angle θ_i for node i out of N total nodes is derived as:

$$\theta_i = i \cdot \left(\frac{2\pi}{N}\right) \tag{6.4}$$

$$v_x^{(i)} = 2.0 \cdot \cos(\theta_i), \quad v_y^{(i)} = 2.0 \cdot \sin(\theta_i) \tag{6.5}$$

Non-coordinator nodes are assigned an initial maximum error estimate of $v_e^{(i)} = 1.0$. This circular initialization provides a symmetric starting topology, encouraging the subsequent relaxation algorithm to converge smoothly.

6.3.2. Iterative Position Approximation

When node i receives a position broadcast from node j, it calculates the predicted distance d_p between them based on their current internal coordinates. The error between the predicted distance and the measured RSSI distance is defined as $e = d_p - d_m$.
To resolve this error, node i adjusts its position along the unit vector $\hat{u}$ pointing toward node j [11]. The magnitude of this adjustment is dictated by a confidence weight w_i, which compares the node's own error estimate against the peer's error estimate:

$$w_i = \frac{v_{e,i}}{v_{e,i} + v_{e,j}} \tag{6.6}$$

The position of node i is then updated using a proportional step size defined by a tuning constant C_c:

$$\begin{bmatrix} v_x^{(i)} \\ v_y^{(i)} \end{bmatrix} \leftarrow \begin{bmatrix} v_x^{(i)} \\ v_y^{(i)} \end{bmatrix} - (C_c \cdot w_i \cdot e)\hat{u} \tag{6.7}$$

The value $C_c = 0.15$ acts as the spatial learning rate of the network. This value was determined based on the 500 ms network cycle. If C_c is set too large, nodes over correct for every minor signal fluctuation, causing the visual graph to violently oscillate or without ever converging into a stable shape. Conversely, if C_c is too small, the system becomes overly sluggish; if a user physically walks a controller across the room, the visual map would lag unacceptably behind reality. A value of 0.15 provides enough responsiveness to track physical movement while damping out overshoot.
Finally, the node updates its own error estimate v_e using an exponential moving average (EMA) of the relative error $\frac{|e|}{d_m}$, scaled by an error constant $C_e = 0.15$.

Using the relative error rather than the absolute error (e) is crucial for stability. An absolute distance error of 1.0 meter is highly significant if two nodes are only 2.0 meters apart, but it is negligible if they are 20.0 meters apart. Dividing by the measured distance (d_m) normalizes the error, preventing distant, noisy connections from violently dragging the rest of the network topology around.

Because raw RSSI signals contain inherent noise that spikes from cycle to cycle, updating the node's confidence based on a single raw reading would cause the entire graph to wobble. Therefore an EMA is applied (Equation 6.1). The tuning constant $C_e = 0.15$ dictates how quickly a node loses or gains this confidence. If C_e were too large (e.g., 0.8), a single bad signal would lower the node's confidence extremely, making it highly sensitive to being pushed out of place. At 0.15, nodes with stable physical connections gradually lock into their correct relative geometry, while still remaining flexible enough to adapt if a user picks up the controller and moves it to a new location.

6.3.3. Limitations and Topological Uncertainty

While this iterative approach successfully groups highly connected nodes visually, it is subject to specific systemic limitations.
Because the system relies entirely on relative distances without multiple absolute anchors, the derived topology exhibits unconstrained degrees of freedom. Specifically, the entire graph can freely rotate around the coordinator node, and the network can logically "flip" (reflect across an axis). Furthermore, in a highly constrained environment with severe multi-path fading, the algorithm may settle into local minima, resulting in a folded graph representation. Mitigating these topological uncertainties requires either physical hardware anchors or a more rigid coordinate protocol, which is proposed as future extensions to this framework.
Because of these limitations the spatial derivations are only used for visualizing the network in the UI.The physical distance estimations and network logic rely purely on relative connections. Using relative connections is also often used in other wireless sensing networks[22]. In these cases the exact location of nodes is also not needed, which is the same for our system.

7

Host Application and User Interface

To visualize the network for the user and enable centralized control over the distributed nodes, the coordinator node is connected to a host computer. A Python-based application runs on this host, acting as the bridge between the physical microcontroller network and the user. This chapter details the communication protocol between the host and the coordinator, the logic driving the real-time graph visualization, and the layout of the user interface.

7.1. Host-Network Communication

Communication between the network and the host computer is done via a UART serial connection to the coordinator node. Throughout the system's operational cycle, the coordinator continuously sends network updates to the host.
The host application listens asynchronously to the UART data stream. Regular expressions (Regex) are utilized to filter and extract values from these text messages, such as active node IDs, peer to peer connections, and current NCO statuses.
The host processes incoming messages immediately. If the message contains structural or state changes, the visualization is updated instantly.
In contrast to the constant automatic messages from the coordinator, the host only transmits data when initiated by user input. The messages from the host to the coordinator are not in text in contrast to the coordinator messages. The first character indicates the instruction type, followed by a specific parameter value. Upon receiving this input, the coordinator modifies its internal variables and injects the new configuration into the CoordPacket during Stage 3 of the network cycle, deploying the user's changes across the entire network.

7.2. Interface Layout

The user interface is designed to provide control over the dynamic processes while offering insights into the physical network. The dashboard is divided into several functional blocks:

- Status and Cycle Tracking: The top-left corner displays the current step number and the active stage of the communication pipeline. An adjacent history bar tracks the success of recent cycles, displaying OK for successful iterations, RETRY or RESET_BOUNDS when connection verification fails, and RESTART when the NCO parameters are reset.
- Network Controls: This panel allows users to initiate or halt the NCO process. It features a "Step Time" adjustment, which allows the user to slow down the visual updates by executing the NCO logic only once every few network cycles (which inherently take 0.5 seconds each).
- Node List: Located on the left side of the UI, this panel lists all theoretically available nodes. When a physical node comes online, its indicator turns green. This section displays real-time telemetry, including the RSSI received by the coordinator and the node's projected coordinates. Users can also interact with toggle switches here to define the initial opinion of each node before starting an NCO experiment.

- Analysis Tools: To analyze the network's behavior over time, users can access a Serial Monitor to read raw coordinator messages or view a consensus graph that plots the distribution of opinions across the network throughout an experiment.
- Spatial Network: On the central and largest portion of the user interface, the map of the network spatial localization(described in section 6.3). Nodes are shown as points positioned exactly at their estimated (v_x, v_y) coordinates. This block is to visually confirm the networks topology by drawing lines between nodes that are connected. As users physically move the controllers, the nodes recalculate their position and the digital nodes are seen moving across the screen to their new positions, providing immediate feedback to the changing network's structure.

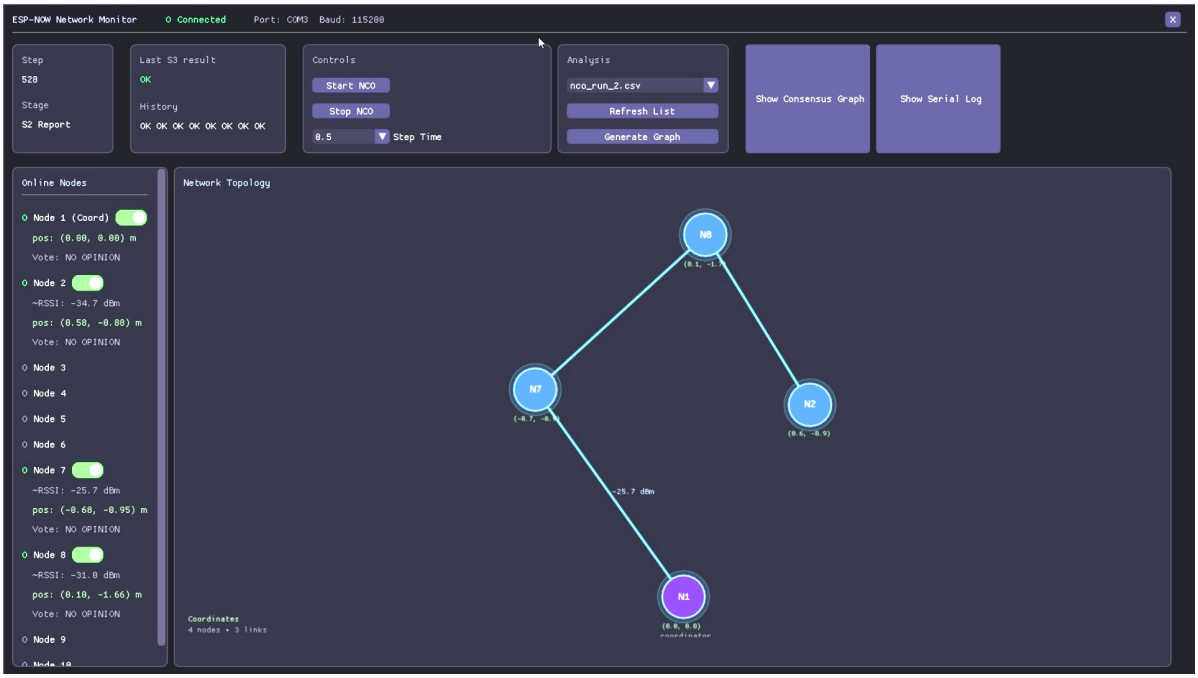


Figure 7.1: UI with 4 online nodes

8

Network Testing

To test the system on its robustness two main categories of tests will be done. First the nodes will be tested on physical displacement with a single connection. Secondly, The nodes will be used in a network for which they are intended and tested for the response to external factors.

8.1. RSSI measurements and calibration

To accurately determine the distance between two nodes, the system relies on the Received Signal Strength Indicator (RSSI)(Chapter 6). Since the nodes connect via an external antenna, the physical displacement of the nodes will influence the RSSI.
The nodes will be tested on relative distance, rotation and tilting.

8.1.1. Distance test

This test evaluates how signal strength changes as the physical distance between two nodes increases. The measurement setup involved placing two nodes on the ground and incrementally increasing the distance between them by 15 cm, as shown in Figure 8.1.

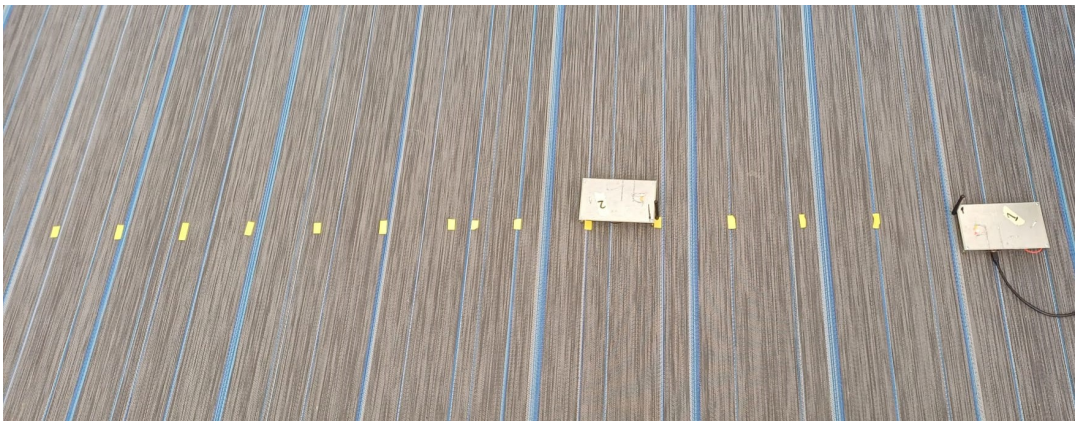


Figure 8.1: Measurement setup for distance and rotation test

As seen in Figure 8.2, the signal strength generally decreases as distance increases. This corresponds to results from similar tests, such as the one from [10]. The signal strength gradually decreases up to 100 cm. Beyond this point, the signal strength occasionally plateaus despite further increases in distance. As the distance between the nodes increases each step the relative size of the step distance decreases in comparison to the total distance. Also, at bigger distances the amount of reflections that contribute to the signal strength becomes bigger. Because of these reasons the signal strength does not increase as much as before 100 cm. Consequently, 1 meter is established as the maximum reliable cut-off point for distance estimation; above 1 meter, RSSI measurements become inconsistent.

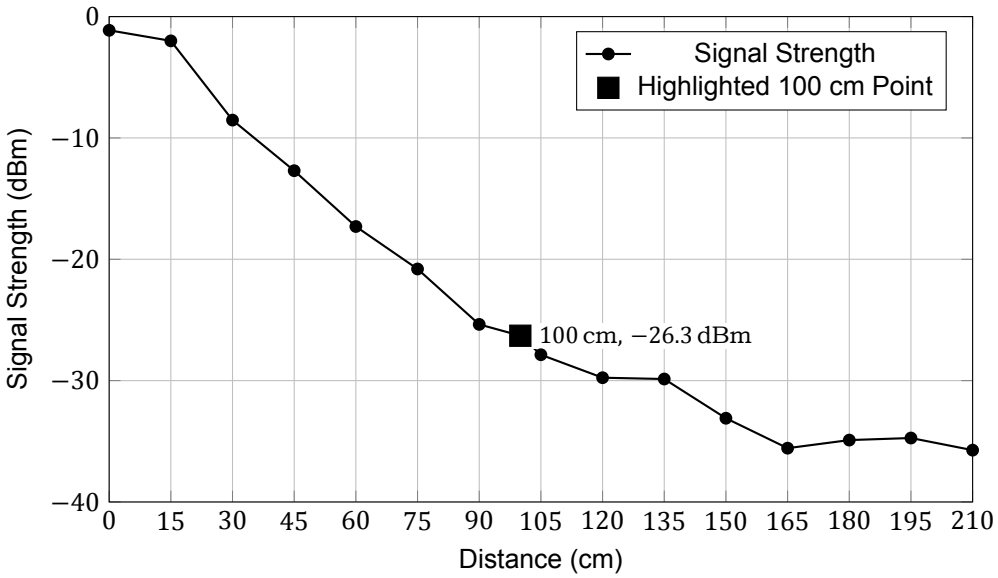


Figure 8.2: Signal Strength vs Distance with the 100 cm Measurement Highlighted

8.1.2. Rotation test

In a practical network, nodes are rarely oriented perfectly toward each other. This test determines the impact of rotation on signal strength. Two nodes were placed 1 meter apart, and the node not connected to the PC was rotated clockwise from an initial, pointing towards each other, orientation (see Figure 8.1).

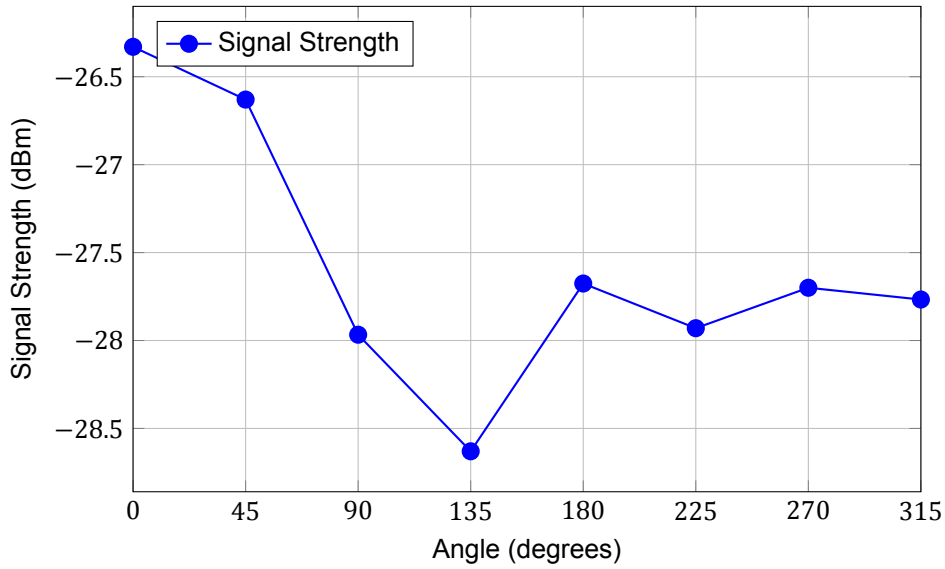


Figure 8.3: Signal Strength vs Angle (reversed y-axis), at distance of 1 meter

As shown in Figure 8.3, any deviation from a directly facing orientation decreases the signal strength. The maximum observed loss was a 2.3 dB drop at an angle of 135 degrees. While standard antennas typically yield symmetric results, the asymmetry observed here is likely caused by interference from components mounted on the perfboard. The primary takeaway from this test is identifying that despite the omnidirectional dipole antenna the nodes are still dependent on rotation. This may influence the physical placement when aiming for a certain topology.

8.1.3. Tilt test

Although the antenna itself is mounted straight up on the board, users holding the devices might tilt them during operation. Therefore, the influence of board tilt on signal strength was tested. Measurements were taken at a 1-meter distance while tilting the boards between 0 degrees (flat on the ground) and 90 degrees (held completely vertically) as seen in Figure 8.4.

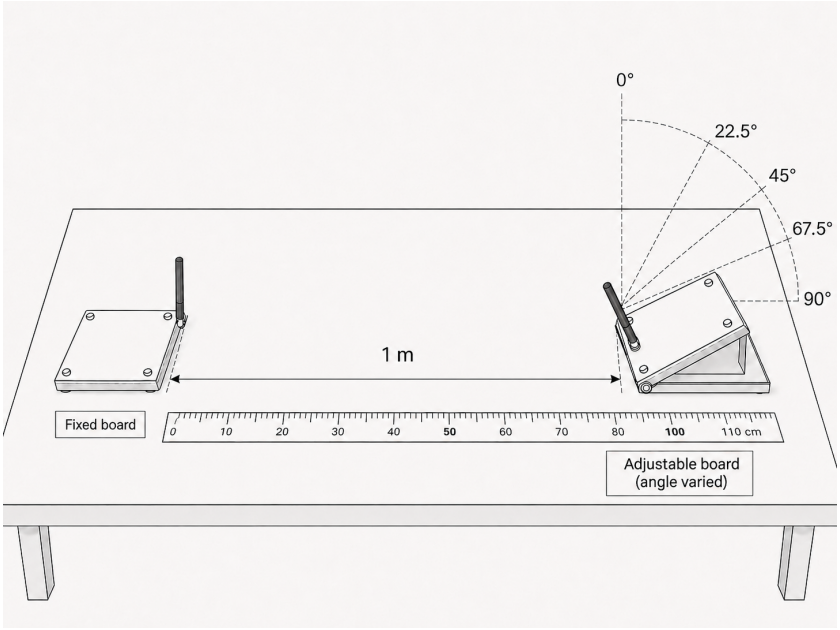


Figure 8.4: Tilt test measurement setup

Angle (degrees)	Average dBm value
0°	-25.43
22.5°	-27.67
45°	-35.87
67.5°	-38.03
90°	-35.00

Table 8.1: Tilt test results

The results, documented in Table 8.1, demonstrate that the highest signal strength is achieved at a 0-degree tilt. This represents the standard use case when the microcontroller node rests flat on its four legs. Tilting the board causes the signal strength to decrease significantly. This drop occurs too rapidly to reliably account for it mathematically in advance. Thus, it is highly recommended that users avoid tilting the microcontroller nodes during usage.

8.2. Network Stability

To verify the system's reliability and determine if the project's goals were met, two different network topologies were tested for stability against physical disturbances.

8.2.1. 4-Node Stability

First, a simple diamond-shaped network of four nodes was set up on the ground (see Figure 8.5).

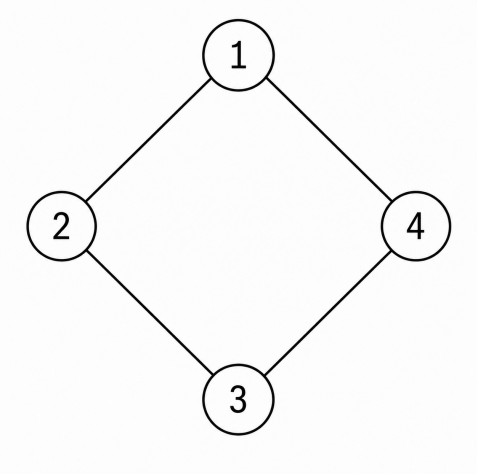
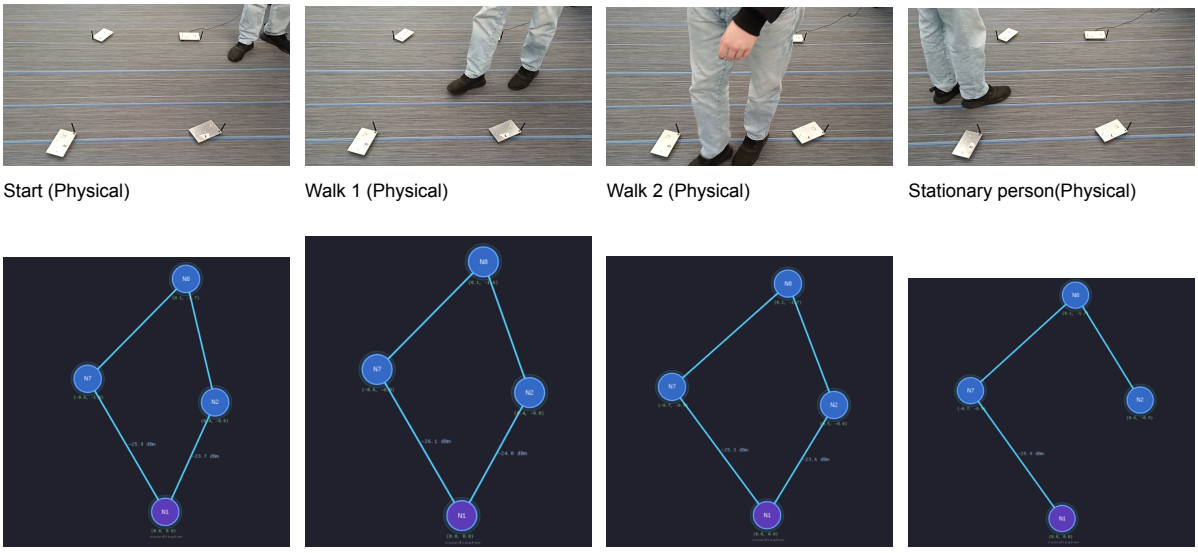


Figure 8.5: Diamond network consisting of 4 nodes

To simulate real-world interference, a person walked directly through the network and then stood stationary between two of the nodes (illustrated in Figure 8.6).



Start (Physical) | Walk 1 (Physical) | Walk 2 (Physical) | Stationary person(Physical)

Start (UI) | Walk 1 (UI) | Walk 2 (UI) | Stationary person(UI)

Figure 8.6: Testing stability within a 4-node network. The top row shows the physical disturbances introduced, while the bottom row displays the corresponding network topology captured in the User Interface.

The test revealed that walking between the nodes did not cause an immediate loss of connection, which means the system methods to filter out sudden changes worked sufficiently. However, when a person stood completely still between two nodes, the signal strength dropped too low, resulting in a lost connection. This demonstrates that dynamic movement between nodes is acceptable, but stationary blockages should be avoided. The LEDs on the microcontrollers accurately reflected these changing connection states in real-time, matching the behaviour expected from the software.

8.2.2. 10-Node Stability

A more complex network consisting of all 10 nodes was then randomly distributed on the ground (see Figure 8.7).

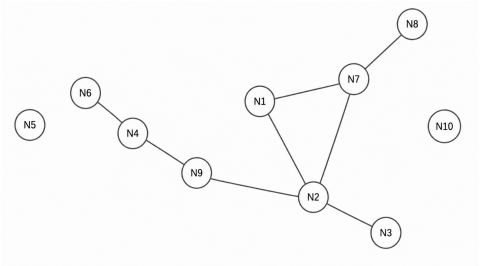


Figure 8.7: Network consisting of 10 nodes

The same physical disturbance test was applied as with the 4 node network, with a person walking through the nodes.

Start (Physical) | Walk 1 (Physical) | Walk 2 (Physical) | Stationary Person(Physical)

Start (UI) | Walk 1 (UI) | Walk 2 (UI) | Stationary Person(UI)

Figure 8.8: Testing stability within a 10-node network. The top row shows the physical disturbances introduced, while the bottom row displays the corresponding network topology captured in the User Interface.

During this test, the network remained highly resilient. All nodes stayed connected while a person walked through the setup. Moreover, unlike the 4-node network, connections remained intact even when the person stood stationary among the nodes. The User Interface registered only minor fluctuations in signal strength—none large enough to disrupt the network topology—and the hardware LEDs continuously confirmed the stable connections.

9

Dynamic Opinion Model

9.1. What is a Dynamic Opinion Model

The previous chapters explain how a network is constructed and tested. The goal of this network is to let a dynamic process run on it. A dynamic process in networking is an automated, adaptive operation that continuously adjusts network behavior based on changing conditions, topology, traffic patterns, or device status.

The dynamic process selected for this network is the nonconsensus opinion (NCO) model [18]. As proposed by the original authors, this is a ”nonconsensus opinion model that allows for stable coexistence of two opinions by forming clusters of agents holding the same opinion.” In this model, each node represents an individual holding one of two opinions. At every update step, an individual considers both its own opinion and the opinions of its neighbors. The node adopts the majority opinion within its local neighborhood (which includes itself), and ties result in no change of opinion. Because of the clustering and community support of agents holding the same opinion, ”these clusters cannot be invaded by the other opinion.” This mechanism allows multiple opinions to coexist in stable clusters. The following sections detail how this model is implemented and present examples of its execution on the network.

9.2. NCO implementation

To let the NCO model run on the network this process has to be integrated in all 3 stages of the system. In the first stage, the node broadcasts its NCO status to all nodes. It also receives the broadcasts of all nodes and decides whether the received signals are within the neighborhood of the node. After that the node updates its status to the majority opinion in its neighborhood. In the second stage the node sends its updated status to the coordinator and in the third stage the node receives from the coordinator whether the step was successful. If the state was not successful the node does not accept its previously decided status and goes back to the previous status. If the step has successfully been completed this cycle starts again.

9.3. Physical testing NCO

To evaluate the NCO model on the network, three different networks and initial conditions were selected.

9.3.1. Line NCO

The first test consists of a line network where each node is connected to one node on the left and one node on the right, except for the nodes at the ends of the line. This test demonstrates how a steady state is achieved in four time steps, as shown in Figure 9.1.

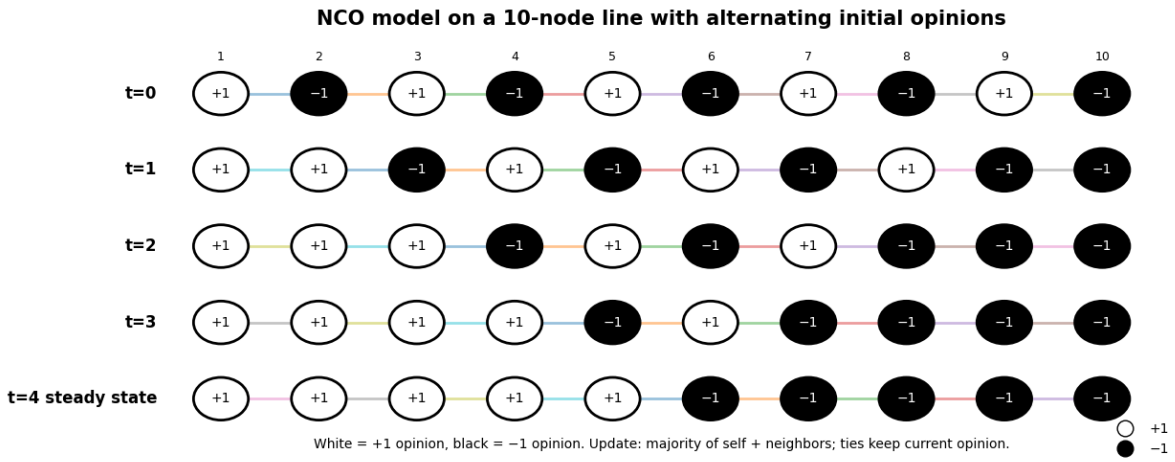
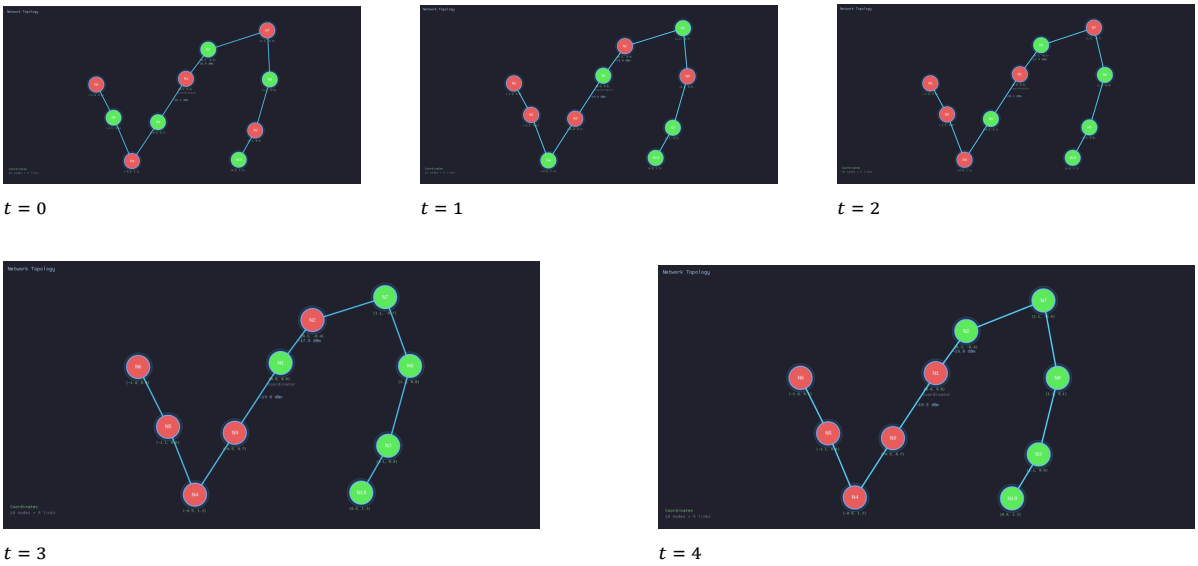


Figure 9.1: Line NCO

Figure 9.2: Line NCO setup

Figure 9.2 displays the physical test setup. In this scenario, node 1 is positioned in the middle, serving as the coordinator node.



$t = 0$ $t = 1$ $t = 2$

$t = 3$ $t = 4$

Figure 9.3: State of the line network at different timesteps.

Figure 9.3 displays the five time steps as represented in the UI, corresponding to the stages outlined in Figure 9.1. The result aligns perfectly with the theoretical NCO model. Furthermore, the LEDs on the microcontroller nodes reflected the exact same states.

9.3.2. Circle NCO

The second test for the NCO model utilizes a 10-node network in the form of a circle. The network and its expected oscillatory behavior (alternating opinions) are illustrated in Figure 9.4. The physical test setup is shown in Figure 9.5.

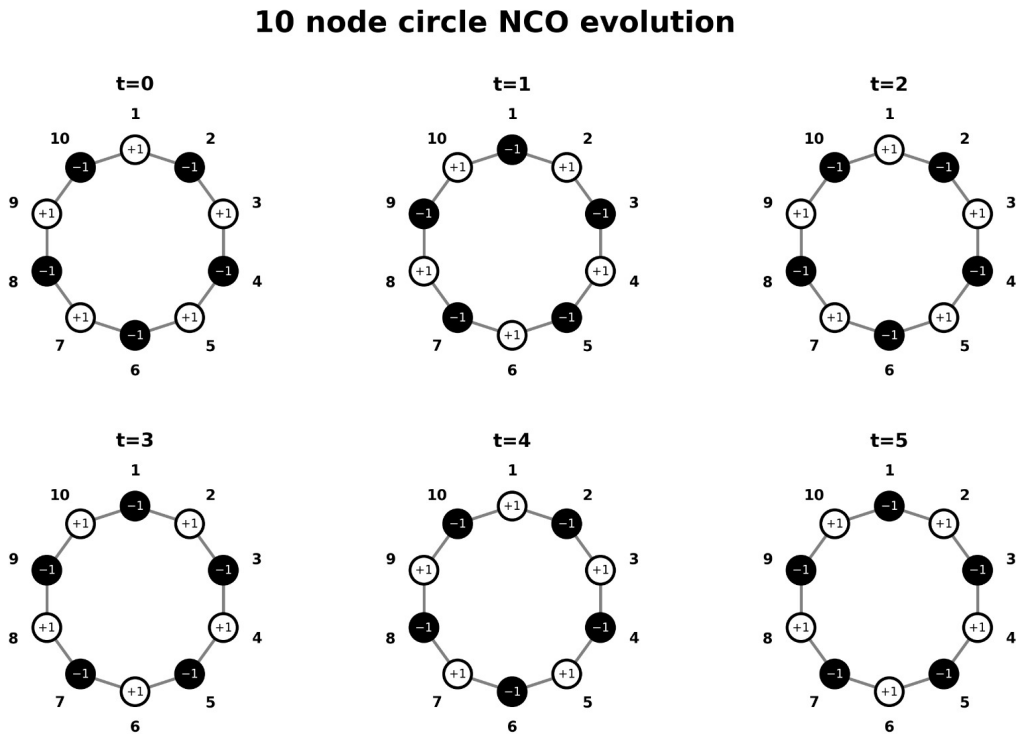


Figure 9.4: Circle NCO

Figure 9.5: Circle

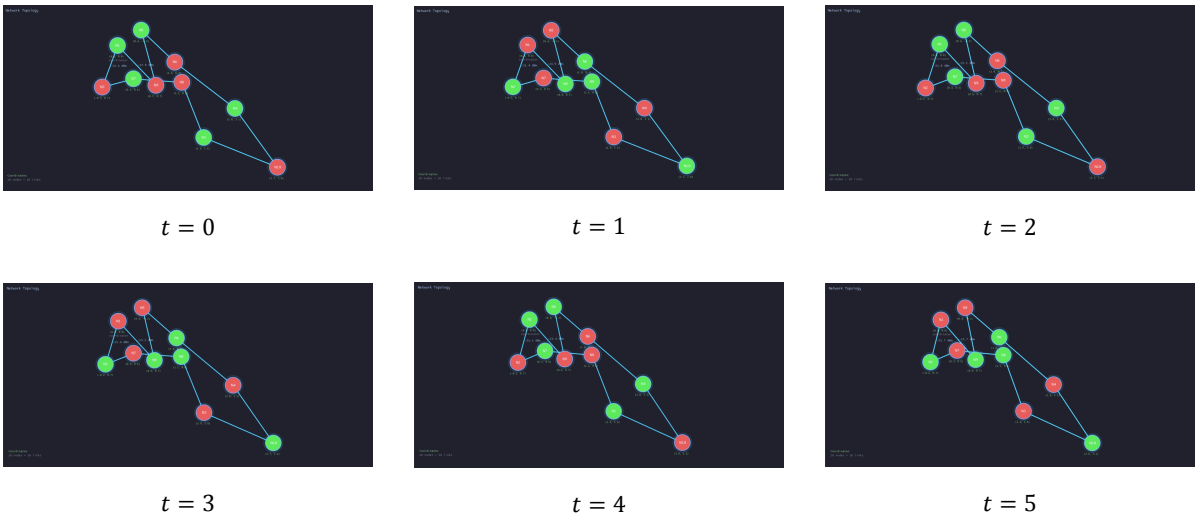


Figure 9.6: 10 node circle NCO at different time steps

In Figure 9.6, the UI results for five time steps of the NCO model running on the circle network are shown. They match the expected states from the theoretical model illustrated in Figure 9.4. While the UI does not spatially mirror the physical layout of the network on the ground, the topological connections are accurate. The LEDs on the microcontroller nodes also displayed identical states.

9.3.3. Island NCO

The final test simulates the network used in [17]. The network topology and the NCO stages are shown in Figure 9.7. This network demonstrates a steady state where three distinct opinion islands form. The physical setup is visible in Figure 9.8.

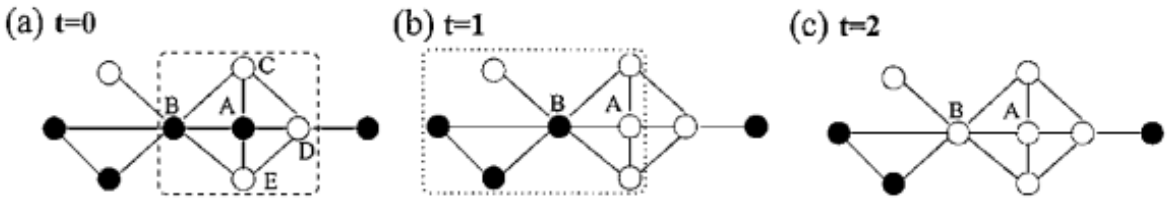


Figure 9.7: Cluster NCO [18]

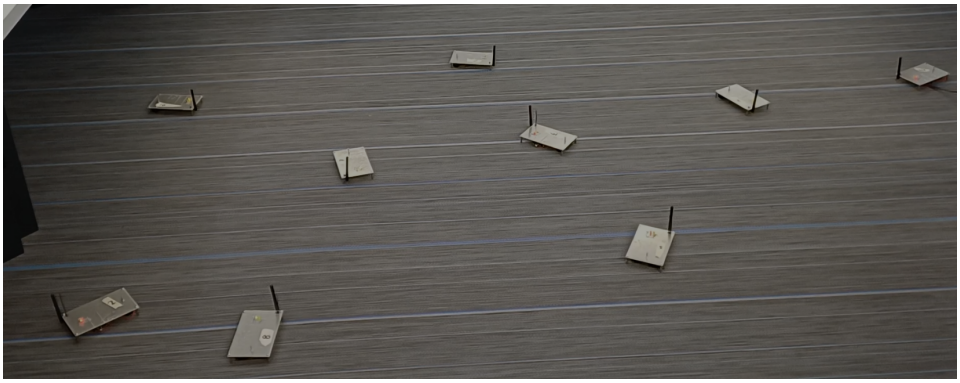
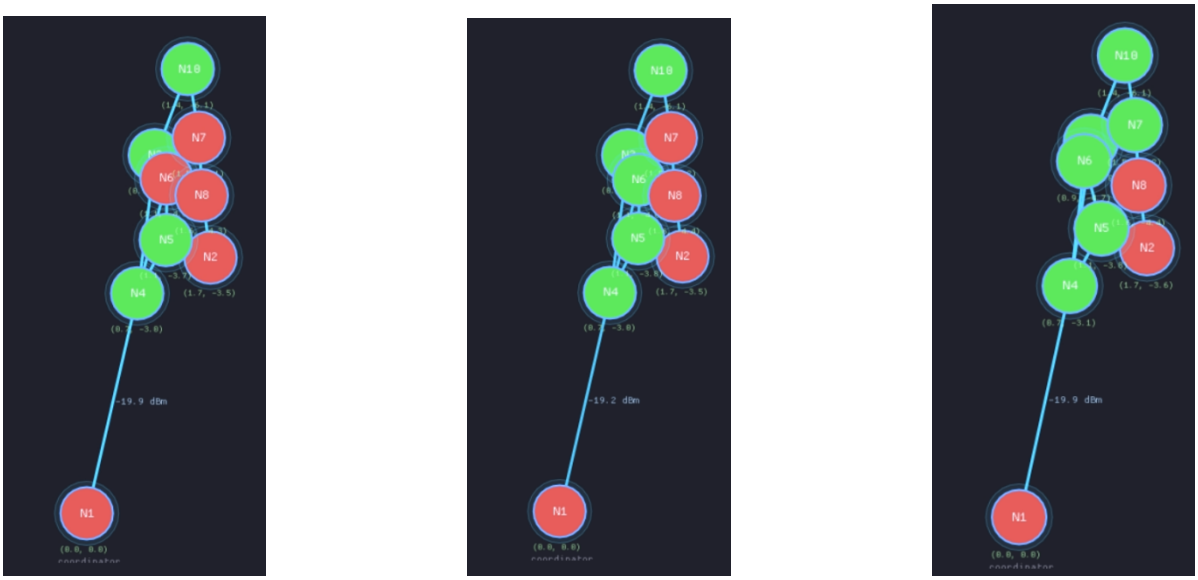


Figure 9.8: Cluster set-up



$t = 1$ $t = 2$ $t = 3$

Figure 9.9: State of the network at different timesteps.

The UI results of the simulation for the network are presented in Figure 9.9. The expected states are clearly observable in the UI. Once again, the UI does not reflect the actual physical positioning of the nodes on the ground, but it accurately represents the network connections. Due to the high number of nodes and the complexity of the network, the UI representation is slightly more difficult to interpret. As with previous tests, the LEDs on the microcontroller nodes showed the same states.

10

Conclusion and Discussion

10.1. Conclusion

The objectives of this thesis have successfully been met. The final system consists of 10 handheld controller nodes capable of forming a network. The communication between the nodes is done with ESP-NOW and the nodes approximate the distance to its peers with RSSI. A non-consensus model is able to run on the network in real time. The nodes show if they are connected to a peer and which consensus state they currently have using integrated LEDs.The network topology and consensus states are visualized in real-time on a dedicated interactive user interface.

10.2. Discussion

While the system meets the core requirements, several design trade-offs have been made which impact the system's performance and scalability.

10.2.1. RSSI

The network stability tests show that for the intended usage, the network stability is high enough. The choice made for including the double boundary for the distance approximation and using an average greatly improved this stability. For complex systems with a lot of connections the network setup is harder. This is due to the RSSI being quite susceptible to reflections. When a connection is lost the entire network topology changes which gives the NCO model a different outcome. This could be improved upon by using a UWB distance approximation. This was outside of the budget for this project.

10.2.2. Hardware

The final microcontroller nodes are stable and are rechargeable which is a nice feature. The size of the microcontroller could have been less. The choice for this size was mainly determined by the perfboards that were available for this project. Given more time and more focus on the hardware design, the microcontroller nodes could have been made smaller and more compact.

10.2.3. UI

The UI shows an accurate network with correct connections and a quick response to the network on the boards. The UI is also user friendly and gives the ability to implement the NCO in an easy way. The only concern with our UI is that it does not fully copy the network nodes location, but only its connections. With complex networks where there are loose nodes, the rest of the network can become harder to interpret. But for the goals of this thesis the, complexity of these networks is not often needed.

10.2.4. Testing

The microcontroller network has passed the tests which have been conducted. The tests were however done in controlled environments. For usage in high volatility environments with a lot of people moving around, more testing would be needed to be sure that the networks would also be stable in these situations. Although the tests were conducted in a controlled environment there were a substantial

number of metal surfaces present. These surfaces may have caused reflections which could have influenced the test result. These tests should preferably be conducted in environment without metal or other reflecting objects.

10.3. Future work

Although the current system is functional there are some areas of improvement. Both the hardware and the software implementation could be further optimized in future iterations:

- **Improve Message Organization**: The current system separates the status, report, and coordination stages. However, since the payloads are far below the maximum payload of ESP-NOW, which is 250 bytes, it would be possible to combine these stages into one combined stage. Benefits of this would be a faster cycle, reduction in amount of packages and thus less risk of package loss. Disadvantages are that it will increase the complexity of the system.
- **Multi Node Distance Estimation**: Various methods for stable RSSI distance approximation have been evaluated. Although the multi-node spatial derivation (Section 6.3) demonstrated practical limitations, the theoretical aggregation of total RSSI information across all nodes should outperform single-connection accuracy.
- **Dynamic TDMA Slotting**: Currently, the system loops through a static loop based on the total number of nodes. There is no time slot automatically bypassed for offline nodes, meaning airtime is wasted waiting for nodes that are offline. Implementing dynamic slotting and registration on which nodes are online would allow the system to improve its efficiency.

A

Ethical Implications of the Product and Engineering Process

A.1. Societal impacts

The primary purpose of this handheld controller network is to serve as an interactive educational tool. If fully implemented in academic settings, the main stakeholders affected would be students, educators, and educational institutions. A significant positive impact is making abstract network theories and dynamic processes, such as the Non-Consensus Opinion(NCO) model, easier to understand. By making these concepts tangible students can better understand how information, resources, or viruses spread across networks. This will ultimately cause a more informed society.

A.2. Ethical aspects of development

During the engineering process, the most prominent ethically relevant aspect was the environmental impact associated with the hardware. Developing the physical nodes required sourcing ESP32 microcontrollers, dipole antennas, lithium-ion batteries, perfboards and other small electronic modules and components. The extraction of raw materials for these electronic components, energy consumed during manufacturing contributed to environmental degradation. Regarding the treatment of human participants the testing phase involved human individuals placing nodes and walking in between the nodes to test connection stability. This interaction posed no physical or psychological risk, meaning human testing concerns were effectively and ethically managed.

A.3. Ethical evaluation

The environmental impact of producing electronic hardware raises the most ethical concerns, especially if these node sets are mass produced for classrooms globally. However, this is counterbalanced by the praiseworthy educational value of the product. Future engineers with an intuitive understanding of network behavior is highly beneficial for modern technological development.
A major trade off was accessibility versus precision and material efficiency. To ensure the controllers remained accessible for educational budgets the cost was kept below 20 euros per node. Cheaper components and options like perfboards and RSSI distance estimation were chosen over more accurate but expensive alternatives like Ultra-Wideband (UWB) sensors. While this optimizes the products reach, it compromises on optimal material efficiency and system accuracy.

A.4. Reflection on design choices

Ethical considerations regarding sustainability and accessibility directly influenced several of the projects design choices. To address the environmental impact of powering wireless nodes, a rechargeable 18650 battery system paired with a TP4056 charging module was deliberately selected over the use of disposable batteries. This simple but significant concrete step significantly extends the products lifespan, reduces long term operational costs for educational institutions and prevents toxic battery waste.

Looking back, to better address the identified material waste issues, a custom PCB could have been designed instead of relying on oversized perfboards. This would have reduced the physical footprint of each node. This minimizes the raw material usage and makes the devices more durable for longterm use.

A.5. Forward-looking responsibility

If this concept demonstrator were to be developed into a real world commercial educational application, then several measures must be taken. First, the manufacturer should establish a recycling or return program for the microcontrollers and lithium batteries to ensure responsible waste management.
Additionally, the system's communication protocol must be secured. Currently, the network transmits unencrypted data over the air. While acceptable for a supervised demonstration, a commercial deployment leaves the system vulnerable to spoofing or external interference. Malicious actors could inject false node statuses or manipulate the network topology, undermining the tool's educational integrity. Future iterations should implement basic authentication and data validation protocols to ensure a stable and reliable learning environment.

B

User Manual

This guide outlines how to launch the visualization software and interact with the handheld controller network.

B.1. Before Starting

To allow your computer to communicate with the ESP32 Coordinator Node, you must have the correct USB-to-UART bridge drivers installed.
Download the driver here:
`https://www.silabs.com/software-and-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads`
If you have trouble installing please watch this short tutorial:
`https://www.youtube.com/watch?v=r_eMEXvt0v0`

B.2. Setup

1. Connect the Coordinator Node (Node 1) to your computer with a USB to micro-USB cable.
2. Power on the other controllers you want to use. This can be done by clicking the big button on the board to run on battery or by powering it via the micro-USB port.
3. Open the included file ”run_ui.exe” to start the application.

After opening the ”run_ui.exe” your screen will look similar to this:

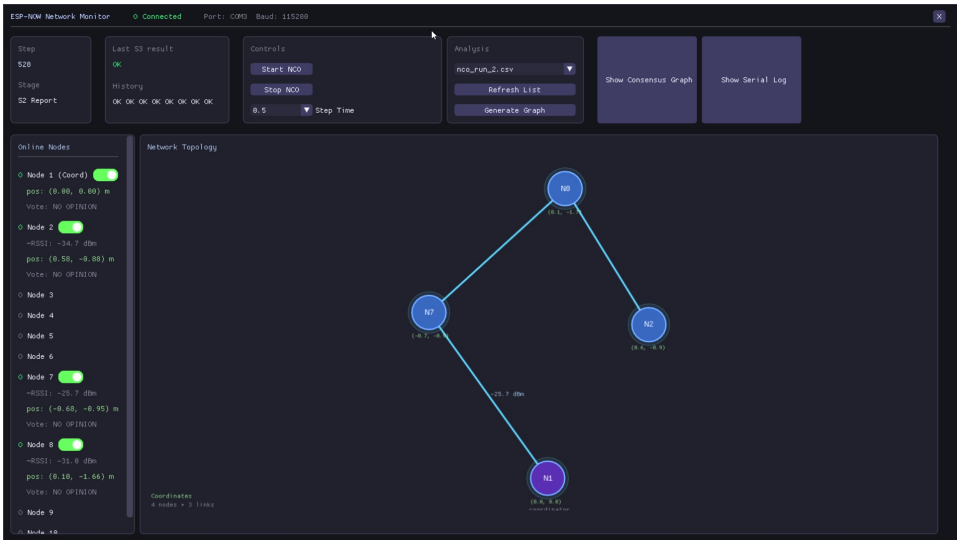


Figure B.1: UI with 4 active nodes

B.3. How to use the Dashboard

1. Before running the NCO model look at the Node list on the left side of the screen
    - verify nodes are communicating: The ”O” indicator will turn green when the node is online. You will also see its live RSSI and approximated coordinates.
    - Set initial opinions: Use the red and green toggle switches next to each online node to set what their starting opinion will be when the NCO process begins.
2. Once your nodes are online, connected to the right peers and initial opinions are set, look to the controls block.
    - Click ”Start NCO” to push the initial opinions to the network and begin the NCO process.
    - Adjust Step Time: The network is standardized on 0.5 seconds. If you want the process to go slower you can select another speed by clicking on the drop down bar.
    - Click ”Stop NCO” to stop the current process and set all nodes back to having no opinion.
3. Monitor Network
    - Top left shows the current cycle step and which stage the network is currently executing.
    - The history shows the result of past cycles.
        - OK : the cycle concluded successfully.
        - RETRY or RESET_BOUNDS: The network encountered a connection issue and is recovering.
        - RESTART: The NCO process was stopped and the parameters have been reset.
4. Analyze the data
    - Show consensus graph to watch a live plot of how many nodes have a particular opinion over time.
    - Show the Serial Monitor to read all the messages sent from the coordinator to the computer.
    - Generate graph if you want to analyze past runs for more insight.
5. To exit the dashboard click the ”X” in the top right corner.

C

Code

The complete source code for both the handheld microcontroller nodes and the host visualization application is available online. To ensure optimal readability the code is in a dedicated GitHub repository, which can be accessed and reviewed here:
`https://github.com/JanvanIngen1/BAP`

Bibliography

[1] Yoppy Chia et al. “RSSI Comparison of ESP8266 Modules”. In: (Oct. 2018), pp. 150–153. DOI: 10.1109/EECCIS.2018.8692892.

[2] Pablo Corbalán and Gian Pietro Picco. “Ultra-wideband Concurrent Ranging”. In: *CoRR* abs/2004.06324 (2020). arXiv: 2004.06324. URL: https://arxiv.org/abs/2004.06324.

[3] Seyed Mahdi Darroudi and Carles Gomez. “Bluetooth Low Energy Mesh Networks: A Survey”. In: *Sensors* 17.7 (2017). ISSN: 1424-8220. DOI: 10.3390/s17071467. URL: https://www.mdpi.com/1424-8220/17/7/1467.

[4] Ilker Demirkol, Cem Ersoy, and Fatih Alagöz. “MAC Protocols for Wireless Sensor Networks: A Survey”. In: *IEEE Communications Magazine* 44.4 (Apr. 2006), pp. 115–121. ISSN: 0163-6804. DOI: 10.1109/MCOM.2006.1632658.

[5] Anh-Vu Dinh-Duc, The-Hien Dang-Ha, and Ngoc-An Lam. “Nviz - A General Purpse Visualization tool for Wireless Sensor Networks”. In: *CoRR* abs/1703.02744 (2017). arXiv: 1703.02744. URL: http://arxiv.org/abs/1703.02744.

[6] Espressif Systems. *ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet*. Available: https://documentation.espressif.com/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf. Espressif Systems.

[7] Laura García et al. “Compliant and Seamless Hybrid (Star and Mesh) Network Topology Coexistence for LoRaWAN: A Proof of Concept”. In: *Applied Sciences* 15.7 (2025). ISSN: 2076-3417. DOI: 10.3390/app15073487. URL: https://www.mdpi.com/2076-3417/15/7/3487.

[8] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Python in Science Conference* (2008). DOI: 10.25080/TCWV9851. URL: https://doi.org/10.25080/TCWV9851.

[9] Sharly Halder, Paritosh Giri, and Wooju Kim. “Advanced Smoothing Approach of RSSI and LQI for Indoor Localization System”. In: *International Journal of Distributed Sensor Networks* (Nov. 2014). DOI: 10.1155/2015/195297.

[10] Xu Jiuqiang et al. “Distance measurement model based on RSSI in WSN”. In: *Wireless Sensor Network* 2 (Jan. 2010), pp. 606–611. DOI: 10.4236/wsn.2010.28072.

[11] Tomihisa Kamada and Satoru Kawai. “An algorithm for drawing general undirected graphs”. In: *Information Processing Letters* 31.1 (1989), pp. 7–15. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(89)90102-6. URL: https://www.sciencedirect.com/science/article/pii/0020019089901026.

[12] Joel Paguay et al. “Secure home automation system based on ESP-NOW mesh network, MQTT and Home Assistant platform”. In: *IEEE Latin America Transactions* 21 (July 2023), pp. 829–838. DOI: 10.1109/TLA.2023.10244182.

[13] Roberto Pasic, Ivo Kuzmanov, and Kokan Atanasovski. “ESP-NOW communication protocol with ESP32”. In: *Izzivi prihodnost* 6 (Mar. 2021). DOI: 10.37886/ip.2021.019.

[14] Yash Paul and Rajesh Singh. “ZigBee Technology: A Comprehensive Review of Protocol, Applications, and Advancements”. In: *IJRDO - Journal of Computer Science Engineering* 1.2 (2015), pp. 36–42. DOI: 10.53555/cse.v1i2.6309.

[15] Karunakar Pothuganti and Anusha Chitneni. “A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi”. In: Jan. 2014. URL: https://www.researchgate.net/publication/312471356_A_comparative_study_of_wireless_protocols_Bluetooth_UWB_ZigBee_and_Wi-Fi.

[16] Özlem Şeker et al. “A Physical Tracking of ESP32 IoT Devices with RSSI Based Indoor Position Calculation”. In: *Journal of Millimeterwave Communication, Optimization and Modelling* 4 (Feb. 2024), pp. 13–16. URL: https://www.jomcom.org/index.php/1/article/view/76.

[17] Jia Shao, Shlomo Havlin, and H. Eugene Stanley. “Dynamic Opinion Model and Invasion Percolation”. In: *Phys. Rev. Lett.* 103 (1 July 2009), p. 018701. DOI: 10.1103/PhysRevLett.103.018701. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.018701.

[18] Jia Shao, Shlomo Havlin, and H. Eugene Stanley. “Dynamic Opinion Model and Invasion Percolation”. In: *Phys. Rev. Lett.* 103 (1 July 2009), p. 018701. DOI: 10.1103/PhysRevLett.103.018701. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.018701.

[19] Yejin Shin and Soonuk Seol. “On-demand Connection Establishment Scheme to Increase the Capacity of BLE Networks”. In: *International Journal of Control and Automation* 10.12 (2017), pp. 99–108. DOI: 10.14257/ijca.2017.10.12.09.

[20] Zhuoran Su et al. “Proximity Detection During Epidemics: Direct UWB TOA Versus Machine Learning Based RSSI”. In: *International Journal of Wireless Information Networks* 29 (Oct. 2022), pp. 480–490. DOI: 10.1007/s10776-022-00577-4.

[21] Dnislam Urazayev et al. “Indoor Performance Evaluation of ESP-NOW”. In: *2023 IEEE International Conference on Smart Information Systems and Technologies (SIST)*. May 2023. DOI: 10.1109/SIST58284.2023.10223585. URL: https://doi.org/10.1109/SIST58284.2023.10223585.

[22] Nour Zaarour, Nadir Hakem, and Nahi Kandil. “Localization Context-Aware Models for Wireless Sensor Network”. In: (2022). Ed. by Venkata Krishna Parimala. DOI: 10.5772/intechopen.103893. URL: https://doi.org/10.5772/intechopen.103893.