MITOR: Jurnal Teknik Elektro

Implementation of Coordinated Auto-Pairing Design in Wireless Sensor Networks Using Gateway-to-Node Handshake Method

Nur Rohman Rosyid, Ronald Adrian, Wafiyyah Amalyris, Dzulfikar, Alif Subardono
Departemen Teknik Elektro dan Informatika/Sekolah Vokasi – Universitas Gadjah Mada Yogyakarta
Yogyakarta, Indonesia
*nrohmanr@ugm.ac.id

Abstract – *In the current and future era of automation, the widespread utilization of Wireless Sensor Network (WSN) technology becomes essential. The ease of installation, configuration, and management of WSN system devices involving sensor nodes is a crucial factor in the effective utilization of this system. The way sensor nodes connect to the gateway/controller in a large and widespread WSN system is a pivotal point. The experimentation in this research has successfully developed a coordinated auto-pairing protocol between the controller/gateway and sensor nodes using a handshake method approach. The results of the coordinated auto-pairing process between the controller and nodes indicate an average time of 435ms per node and a data transmission accuracy of 97%.*

Keywords – *Wireless Sensor Network (WSN); Auto-pairing protocol; Gateway-to-node handshake method; Sensor node configuration; Data transmission accuracy.*

I. Introduction

THE Wireless Sensor Network (WSN) technology provides numerous applications used across various industrial sectors. The ease of configuration in operating a large number of sensor nodes will affect the overall operational efficiency of the system. Additionally, in the current and future eras, as the development of IoT technology accelerates, people will become increasingly dependent on WSN [1]. The implementation of wireless sensor network and Internet of Things (IoT) in smart homes facilitates residents in automatically monitoring and controlling household objects. Thus, a system that is easy to install is required [2]. How sensor nodes connect with a gateway wirelessly and automatically has become a crucial requirement for massive utilization. Several studies on communication protocols in Wireless Sensor Networks (WSN) have been conducted to obtain efficient and secure protocols. The Centralized Sensor Pairing Strategy (CSPS) is a method used in Wireless Sensor Networks (WSN) to establish connections or pairing between sensors within the network. This method is used to enhance the efficiency and quality of communication between nodes in a WSN [3], [4].

Adaptive negotiation between sensor nodes and gateways is crucial for dynamic environments, as it enables efficient energy management and quality of service (QoS) resolution. Ortega (2021) and Udoh (2020) both propose negotiation models that allow nodes to dynamically adapt their strategies, with Ortega focusing on energy negotiation [5] and Udoh on QoS negotiation [6]. Callebaut (2020) and Banerjee (2020) further enhance energy efficiency through proactive adjustment strategies [7] and temperature-adaptive sleep scheduling [8], respectively. García (2020) and Mazumdar (2021) address the challenges of dynamic environments in specific applications, such as precision agriculture and harsh environments, by proposing the use of remote sensing drones as mobile gateways [9] and a hierarchical data dissemination strategy [10], respectively. Bouarourou (2021) introduces a bio-inspired adaptive model for sensor selection [11], while Siva (2022) presents a negotiation-based approach for robot navigation in unstructured terrains [12]. These studies collectively highlight the importance of adaptive negotiation in addressing the challenges of dynamic environments in IoT and sensor networks.

A negotiation method that takes into account client preferences to ensure safe and efficient data transmis-

The manuscript was received on November 16, 2023, revised on July 2, 2024, and published online on July 26, 2024. `Emitor` is a Journal of Electrical Engineering at Universitas Muhammadiyah Surakarta with ISSN (Print) 1411 – 8890 and ISSN (Online) 2541 – 4518, holding Sinta 3 accreditation. It is accessible at https://journals2.ums.ac.id/index.php/emitor/index.

sion on Wireless Sensor Networks (WSN) [13]. The article [14] provides a solution with gateway discovery in LoRa systems, making it more efficient. A range of studies have highlighted the importance of connected and coordinated sensors in improving data collection efficiency and accuracy. Khattar (2023) and Wala (2020) both emphasize the role of sensors in smart agriculture, with Khattar focusing on soil moisture data transmission [15] and Wala discussing energy-efficient data collection in smart cities [16]. Dehury (2020) and Hou (2021) propose frameworks for coherent coordination of data migration and computation [17], and for optimizing web service-based data collection systems [18], respectively. Alejandrino (2021) and Chehri (2020) both address the challenges of data transmission in precision farming, with Alejandrino proposing a protocol-independent data acquisition system [19] and Chehri focusing on the deployment of IoT devices [20]. Lastly, Sletcha (2020) and Li (2020) discuss the real-time dataflow architecture for oil and gas rigs [21] and the use of edge computing-enabled wireless sensor networks in smart agriculture [22], respectively. These studies collectively underscore the critical role of connected and coordinated sensors in enhancing data collection efficiency and accuracy.

This research designs a coordinated auto-pairing protocol in a wireless sensor network that facilitates sensor nodes connecting with a gateway/controller. The design is divided into two parts: first, how the sensor nodes connect with the gateway; second, how data transmission is coordinated between the gateway and the nodes. The experiments in this study use a series of sensor nodes consisting of nRF24L01 modules, Arduino Nano, and a temperature sensor. The gateway assembly is composed of an nRF24L01 module and an Arduino Uno. The gateway is used as a hub for sensors located nearby.

II. Research Methods

i. Hardware Design

The sensor node circuit used in this study consists of the nRF24L01 module, Arduino Nano, and DS18B20 temperature sensor. All components are connected to the Arduino Nano microcontroller. The nRF24L01 module functions for wireless communication between sensor nodes and the controller. Meanwhile, the controller circuit consists of the nRF24L01 module and Arduino Uno. Figure 1 shows the schematic diagram for the controller module.

Figure 2 displays the schematic diagram of the sensor module design.

The hardware implementation for the controller and sensor node is shown in Figure 3.

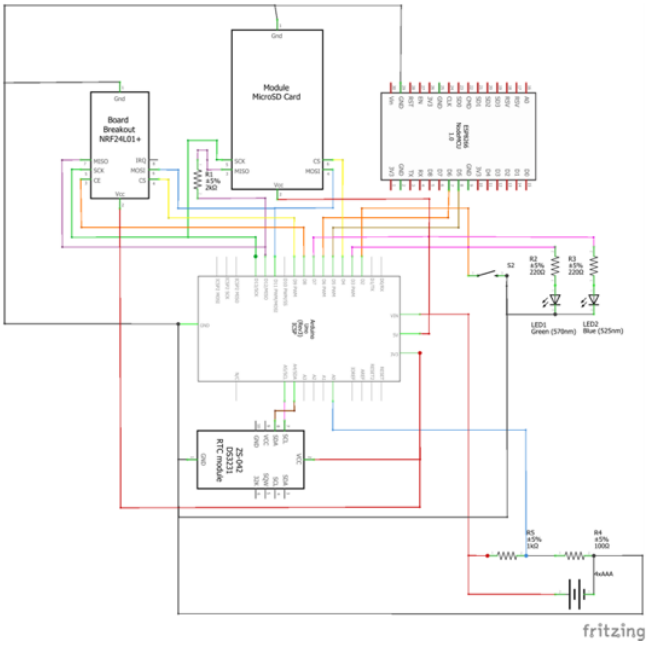

Figure 1: Schematic diagram of controller/gateway circuit

ii. Design of Coordinated Auto-Pairing Protocol

The coordinated auto-pairing protocol is designed for wireless sensor networks using the gateway-to-sensor handshake method. The gateway coordinates the establishment of connections and the transmission of data from the sensor node to the gateway. The terms gateway and controller are used interchangeably and will be frequently mentioned throughout this article. This protocol is designed with two phases: the first phase is the connection phase, and the second phase is the data transmission phase.

The connection phase is the initial stage of building a wireless sensor network. This phase begins with the gateway becoming active to discover sensor nodes that will join the network. Figure 4 shows the state diagram of the gateway/controller in the connection and data transmission phase.

The gateway begins by checking its list of nodes to determine the number and index position of the next node. Each node must have a unique identity, called nodeID, within the gateway's node list. The gateway generates tmpID as a candidate nodeID, as shown in Figure 4 at the initial state. Candidate nodes compete for the tmpID in a BEACON broadcasted by the gateway. The gateway then waits for an acknowledgment (ACK) from the node that successfully receives the candidate nodeID. The gateway temporarily stores the candidate nodeID after receiving an ACK from the successfully accepted node.

The connection phase up to this point can also be

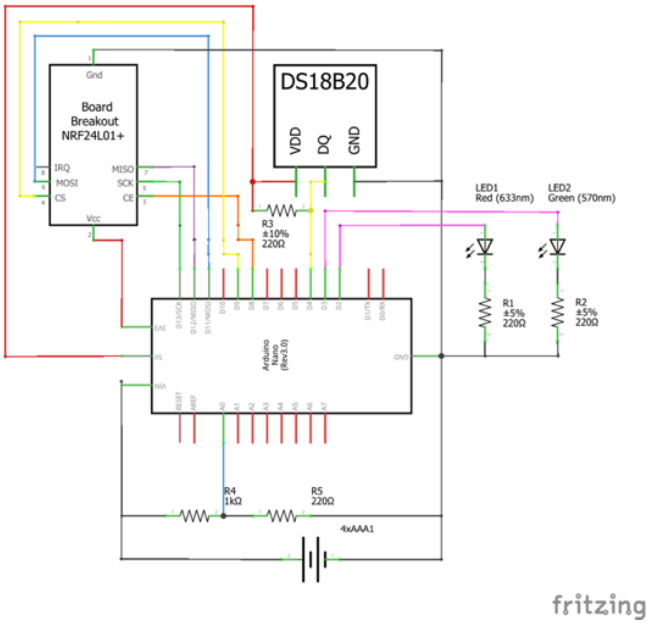

Figure 2: Schematic diagram of sensor node circuit

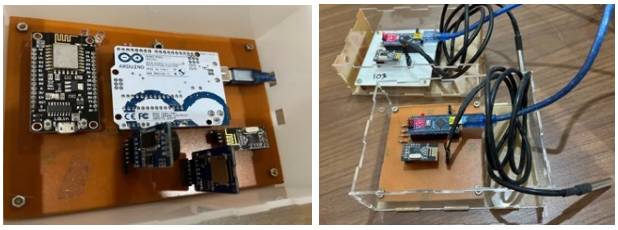

Figure 3: Hardware implementation (left) Gateway/Controller (right) Sensor node

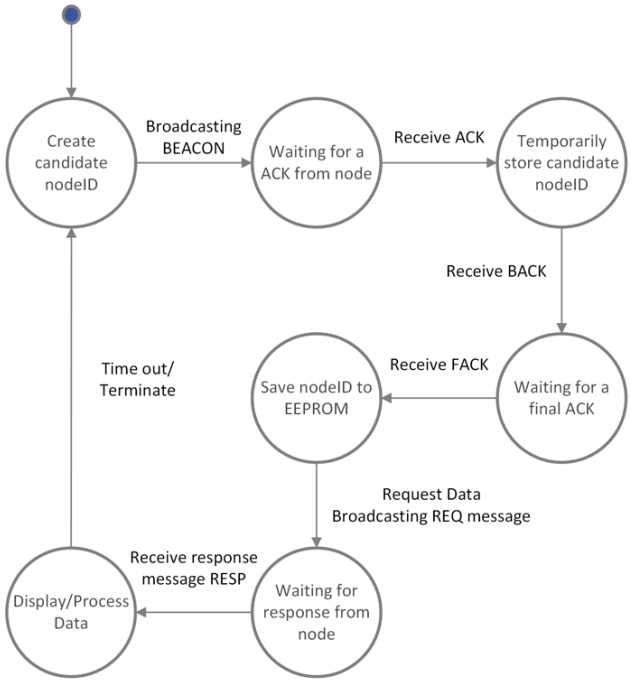

Figure 4: State diagram of gateway/controller in the connection and data transmission phase

referred to as the competition phase, where active candidate nodes compete to obtain the candidate nodeID from the broadcast message sent by the gateway and strive to be accepted by the gateway to continue the registration process. The gateway then responds to the node's ACK with a back acknowledgment (BACK), which is further confirmed by the corresponding node as a final acknowledgment (FACK). The FACK is received by the gateway, concluding the node registration process by assigning the candidate nodeID as the nodeID of the sensor node. This nodeID is then permanently stored in the gateway's node list. This stage ends the connection phase between the gateway and one of the nodes in the wireless sensor network.

The connection phase between the controller/gateway and several nodes in forming the wireless sensor network is called coordinated auto-pairing. It is called auto-pairing because the formation of unique identities for each node is done automatically without any hardcoded intervention in each node. The method used involves a handshake (conversation) interaction between the controller and the nodes.

The data transmission phase is where the controller coordinates the sending of data from each registered node to the controller. As shown in Figure 4, there are two states: the controller waiting for a response from the node, and the display or processing of data. The controller requests data from a node by sending a request message (REQ) containing the nodeID of the target node, and then waits for a response from that node. Once the controller receives a response message (RESP) from the node, the data is ready to be processed. In this experiment, the data is displayed on the serial monitor. The controller will request data from all registered nodes, and upon completion, the controller will repeat the connection phase to obtain other candidate nodes that will join this wireless sensor network.

Nodes have several states that synergize with the states of the controller as previously explained. Figure 5 shows the state diagram of a node in the connection and data transmission phases.

When a node is first activated, it will create a self-identity used to communicate with the controller during the connection phase. This self-identity of the node is called localID, which is randomly generated within the range of 1 to 255. The node will receive a broadcast message (called BEACON) sent by the controller. As previously explained, this BEACON contains a candidate nodeID. The node will respond by sending an ACK containing the localID and the candidate nodeID. If the node receives a BACK confirmation from the controller, it successfully wins the competition and sends a final confirmation (FACK) to the controller, which concludes the connection phase with the status that the

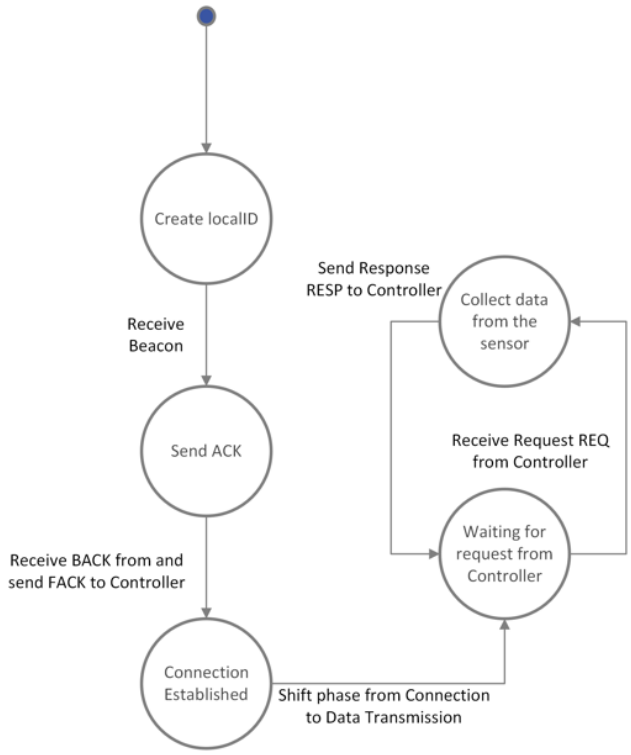

Figure 5: State diagram of a node in the connection and data transmission phase

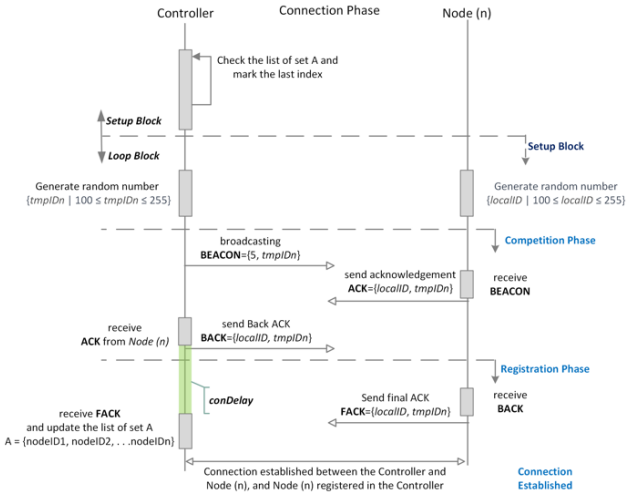

Figure 6: Interaction diagram of gateway (controller) and node in the connection phase

node is now connected to the controller/gateway.

The node then enters the data transmission phase by waiting for a request message (REQ) from the controller. The REQ message received by the node is responded to by retrieving data from the existing sensor module, which in this experiment is the DS18B20 temperature sensor. The node will send the data with a RESP message to the controller, where the RESP message contains the nodeID and temperature data. The node will continuously alternate between waiting for requests and reading the temperature sensor module as long as the node is active/operational.

iii. Implementation of the Coordinated Auto-Pairing Protocol (Connection Phase)

The hardware implementation of the auto-pairing concept is shown in Figures 1, 2, and 3. Meanwhile, the software implementation can be seen in the form of an interaction diagram between the controller and the node, as depicted in Figure 6.

The software implementation uses Arduino IDE, where there are two blocks: the setup block and the loop block. The code within the setup block will be executed once when the system is activated. Meanwhile, the loop block will continuously repeat all the code within it.

The controller has a function to check the `nodeID` list and mark the end index of the available location for the next node. This function is placed in the `setup` block because it is only needed once when the controller system is activated. The function that handles auto-pairing on the controller is named `cPairing`, which is placed in the `loop` block because it will run repeatedly as long as the controller system is active. Meanwhile, the auto-pairing function on the node side is named `nPairing`, which is placed in the `setup` block. As shown in Figure 6, after the controller enters the `loop` block area, it randomly creates a candidate identity for the node named `tmpID`, defined as $\{\texttt{tmpID}_n \mid 100 \leq \texttt{tmpID}_n \leq 255\}$. On the node side, it also begins by randomly creating a local identity with a definition similar to the controller, which is $\{\texttt{localID} \mid 100 \leq \texttt{localID} \leq 255\}$. Next, the controller creates an array variable named `BEACON` = $\{5, \texttt{tmpID}_n\}$, where 5 is the code indicating that this message is a beacon. `BEACON` is broadcasted for all candidate nodes to compete for. After receiving the `BEACON`, each node that receives it will respond by creating an array variable `ACK` = $\{\texttt{localID}, \texttt{tmpID}_n\}$ and sending it back to the controller. The `ACK` message received by the controller is one of the `ACK`s sent by several candidate nodes competing to connect with the controller. The controller will respond to the node's `ACK` message by sending a `BACK` = $\{\texttt{localID}, \texttt{tmpID}_n\}$ message, indicating that the node has successfully won the connection beacon.

The node that receives BACK from the controller will change the connection status connStatus from false to true and send a final confirmation in the form of an array variable FACK = {localID, nodeID}. Thus, the node is connected to the controller and registered with the node identity, which is nodeID, as defined in

Equation (1).

$$\begin{cases} \text{nodeID} = 0 \quad \text{and} \quad \text{connStatus} = \text{false} \\ \text{nodeID} = \text{tmpID}_n \quad \text{if} \quad \text{connStatus} = \text{true} \end{cases} \tag{1}$$

The FACK will be received by the controller, prompting it to check for a match between the received FACK and the BACK sent previously. The result of this comparison will update the node list stored in the EEPROM of the Arduino. The node list is defined in Equation (2),

$$A = \{\} \text{ and } \text{nodeID}_n \in A \text{ if } \text{FACK}_n \equiv \text{BACK}_n \tag{2}$$

where A is the set of nodes (node list), and nodeID is the node's identity.

iv. Data Transmission Implementation

The data transmission phase is fully controlled by the controller. The controller sequentially calls each node stored in its list, starting from the smallest index 0 (zero) to the largest. The controller sends a request to all nodes via broadcast in the form of an array variable REQ = {nodeID, 0}, where 0 is the request code. The node will respond with the message RESP = {nodeID, data}. After completing the entire node list, the controller will repeat the connection phase and continue this process throughout the controller's lifetime. The illustration of the data transmission phase is shown in Figure 7.

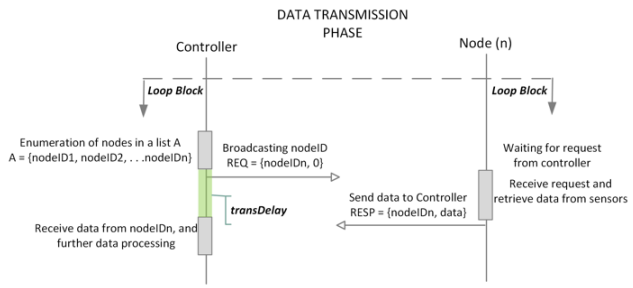

Figure 7: Interaction diagram of gateway/controller and node in the data transmission phase

The timing operation of this protocol can be seen in Figure 8, which illustrates the controller and 2 nodes. The controller continuously repeats both phases, the connection phase and the data transmission phase, while the nodes only enter the connection phase once and then repeatedly enter the data transmission phase as requested by the controller.

III. Results and Discussion

This research focuses on the performance of the coordinated auto-pairing protocol design in terms of the accuracy of both the connection phase and the data transmission phase. The delay parameter on the controller is observed to assess the impact on the accuracy of connection and data transmission.

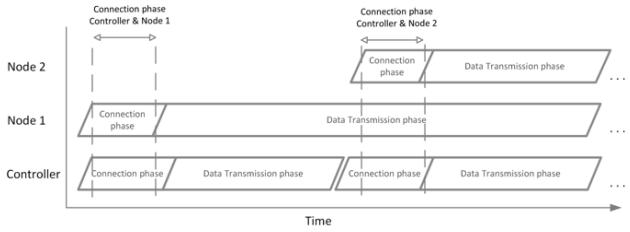

Figure 8: Time diagram of coordinated auto-pairing system

In the connection phase, there is a delay time called conDelay, which is the delay time between the controller receiving/reading ACK and FACK messages from the node, as shown in Figure 6 with green shading. The experimental results show that this delay has a significant impact on the accuracy of the connection phase. Table 1 shows the recorded interaction between the controller and 2 nodes based on time without any delay immediately after the controller receives the ACK and FACK messages. The impact of the absence of this delay is evident when the controller broadcasts the beacon message, as all nodes seem to receive the beacon well. This is shown in the timing of 14:18:14.340–341. When the controller sends BACK at 14:18:14.406, both nodes are not ready to receive it. This continues to repeat at 14:18:16.429–462. A false positive occurs at 14:18:18.521–865, where the controller successfully registers nodeID = 142, but it is not confirmed by either node 1 or node 2, so neither node 1 nor node 2 feels they have been registered with the controller.

Different results are observed in Table 2, which shows the interaction between the controller and two nodes with a conDelay set to about 10ms immediately after the controller receives the ACK and FACK messages from the nodes. The interaction between the controller and the two nodes runs smoothly. As seen in the timing from 13:32:51.090-53.093, the controller sends a beacon, and at 13:32:55.142-274, the beacon is successfully received by node 1, followed by the auto-pairing process from 13:32:55.142-630, where node 1 confirms connection with nodeID = 113. Node 2 also successfully completes the auto-pairing process, starting from 13:32:57.403-785. The interaction between the controller and each of the nodes takes 488ms and 382ms, respectively, so the average time required to complete the auto-pairing process per node is 435ms.

The accuracy of the data transmission phase is more influenced by the delay time in which the controller receives the response from the node. This delay time refers to the duration allocated by the controller

Table 1: Conversation between the controller and 2 nodes without time delay

Controller	Node 1	Node 2	Description
Time --> Status	Time --> Status	Time--> Status	
14:18:14.340 -> Send BEACON with tmpID: 238	14:18:14.341 -> Receive BEACON with tmpID: 238	14:18:14.340 -> Receive BEACON with tmpID: 238	
14:18:14.373 -> Receive ACK with tmpID: 238	14:18:14.445 -> Send ACK[1]: 238	14:18:14.445 -> Send ACK[1]: 238	
14:18:14.406 -> Send BACK to node: 55	14:18:15.202 -> Establishing Connection!!!	14:18:15.202 -> Establishing Connection!!!	**Controller cinfirming receiveBEACON but,**
14:18:16.429 -> Send BEACON with tmpID: 209	14:18:15.953 -> Establishing Connection!!!	14:18:15.992 -> Establishing Connection!!!	Node1 dan 2 failed to get BEACON
14:18:16.462 -> Receive ACK with tmpID: 209	14:18:16.461 -> Receive BEACON with tmpID: 209	14:18:16.462 -> Receive BEACON with tmpID: 209	
14:18:16.538 -> Send BACK to node: 90	14:18:16.538 -> Send ACK[1]: 209	14:18:16.538 -> Send ACK[1]: 209	Controller cinfirming receiveBEACON but,
14:18:18.521 -> Send BEACON with tmpID: 142	14:18:17.308 -> Establishing Connection!!!	14:18:17.309 -> Establishing Connection!!!	Node1 dan 2 failed to get BEACON
14:18:18.767 -> Receive ACK with tmpID: 142	14:18:18.067 -> Establishing Connection!!!	14:18:18.067 -> Establishing Connection!!!	
14:18:18.800 -> Send BACK to node: 84	14:18:18.564 -> Receive BEACON with tmpID: 142	14:18:18.564 -> Receive BEACON with tmpID: 142	Controller cinfirming receiveBEACON but,
14:18:18.800 -> Receive FACK from nodeID and save: 142	14:18:18.640 -> Send ACK[1]: 142	14:18:18.641 -> Send ACK[1]: 142	Node 1 dan 2 failed to get BEACON
14:18:18.865 -> 0: 142	14:18:19.395 -> Establishing Connection!!!	14:18:19.396 -> Establishing Connection!!!	Controller do false positive in confirming a connectio
14:18:18.865 -> Send request REQ to: 142	14:18:20.162 -> Establishing Connection!!!	14:18:20.163 -> Establishing Connection!!!	then connection is failed to establish
14:18:20.836 -> Send BEACON with tmpID: 251	14:18:20.926 -> Establishing Connection!!!	14:18:20.926 -> Establishing Connection!!!	

Table 2: Conversation between the controller and 2 nodes with time delay 10ms

Controller	Node 1	Node 2	Description
Time --> Status	Time --> Status	Time --> Status	
13:32:51.090 -> Send BEACON with tmpID: 134	13:32:54.529 -> Establishing Connection!!!		
13:32:53.093 -> Send BEACON with tmpID: 222	13:32:55.274 -> Establishing Connection!!!		
13:32:55.142 -> Send BEACON with tmpID: 113	13:32:55.274 -> Receive BEACON with tmpID: 113		Node 1 successfully get the beacon
13:32:55.346 -> Receive ACK with tmpID: 113	13:32:55.379 -> Send ACK[1]: 113		
13:32:55.379 -> Send BACK to node: 236	13:32:55.379 -> Receive BACK with nodeID: 113		Node 1 ready for register
13:32:55.379 -> Receive FACK from nodeID and save: 113	13:32:55.415 -> Send FACK to Contnroller.	13:32:55.596 -> Establishing Connection!!!	
13:32:55.444 -> 0: 113	13:32:55.630 -> Connection Established with nodeID: 113	13:32:56.412 -> Establishing Connection!!!	Node 1 successfully registered
13:32:55.444 -> Sending request to: 113		13:32:57.132 -> Establishing Connection!!!	Connection established with Node1
13:32:57.403 -> Send BEACON with tmpID: 222		13:32:57.442 -> Receive BEACON with tmpID: 222	Node 2 successfully get the beacon
13:32:57.475 -> Receive ACK with tmpID: 222		13:32:57.542 -> Send ACK[1]: 222	
13:32:57.509 -> Send BACK to node: 103		13:32:57.542 -> Receive BACK with nodeID: 222	Node 2 ready for register
13:32:57.542 -> Receive FACK from nodeID and save: 222		13:32:57.575 -> Send FACK to Contnroller.	
13:32:57.575 -> 0: 113		13:32:57.785 -> Connection Established with nodeID: 222	Node 2 successfully registered
13:32:57.607 -> 1: 222			Connection established with Node2

Table 3: Recording of 3 forms of data transmission patterns from the controller side

Stable Pattern of Data Transmission	Unstable Pattern of Data Transmission #1	Unstable Pattern of Data Transmission #2
Time --> Status	Time --> Status	Time --> Status
16:09:35.892 -> 0: 205	16:09:41.703 -> 0: 205	16:10:25.002 -> 0: 205
16:09:35.892 -> 1: 251	16:09:41.703 -> 1: 251	16:10:25.002 -> 1: 251
16:09:35.924 -> 2: 184	16:09:41.736 -> 2: 184	16:10:25.002 -> 2: 184
16:09:35.924 -> Send request REQ to: 205	16:09:41.736 -> Send request REQ to: 205	16:10:25.046 -> Send request REQ to: 205
16:09:36.231 -> Receive response RESP from: 205.00	16:09:42.306 -> Receive response RESP from: 205.00	16:10:25.942 -> Receiving response from: 205.00
16:09:36.264 -> nodeID 205.00	16:09:42.338 -> nodeID 205.00	16:10:25.974 -> nodeID 205.00
16:09:36.264 -> Temperature: 25.06	16:09:42.338 -> Temperature: 25.12	16:10:25.974 -> Temperature: 25.12
16:09:36.304 -> Send request REQ to: 251	16:09:42.380 -> Send request REQ to: 251	16:10:25.974 -> Send request REQ to: 251
16:09:38.226 -> Receive response RESP from: 251.00	16:09:44.317 -> Send request REQ to: 184	16:10:26.274 -> Receiving response from: 251.00
16:09:38.258 -> nodeID 251.00	16:09:45.262 -> Receive response RESP from: 184.00	16:10:26.306 -> nodeID 251.00
16:09:38.291 -> Temperature: 24.62	16:09:45.296 -> nodeID 184.00	16:10:26.306 -> Temperature: 24.69
16:09:38.291 -> Send request REQ to: 184	16:09:45.296 -> Temperature: 25.12	16:10:26.306 -> Send request REQ to: 184
16:09:39.717 -> Receive response RESP from: 184.00		16:10:29.292 -> Send BEACON with tmpID: 138
16:09:39.750 -> nodeID 184.00		
16:09:39.750 -> Temperature: 25.06		

to wait for and receive data, as illustrated in Figure 7, which is called transDelay. This is the duration after the controller sends a REQ={nodeID, 0} message to a node until it receives a RESP message containing data from that node. Table 3 shows examples of three types of data transmission patterns from the controller's perspective. A stable pattern is when all nodes are successfully coordinated in data transmission correctly and in the correct order. An unstable pattern is when not all nodes are successfully coordinated in data transmission correctly and in the wrong order. In this experiment, two transDelay values, tD1 and tD2, were used, each being 2000ms and 5000ms. Each scenario was run for 180,000 ms (3 minutes), and the transmission accuracy results are shown in Table 4.

A longer transDelay will affect the accuracy of data transmission between the controller and the nodes. This can be explained by the fact that the controller allocates more time for the nodes to stably receive the REQ message from the controller and send the RESP within sufficient time.

Table 4: Data transmission accuracy based on variations in transDelay by the controller

transDelay	Time Frame (ms)	Accuracy (%)
tD1	2000	48
tD2	5000	97

IV. Conclusion

The coordinated auto-pairing design in a wireless sensor network using the gateway/controller to sensor node handshake method has been successfully developed. The establishment of connections between the controller and nodes in coordinated auto-pairing is influenced by the controller's delay time (conDelay) in

receiving ACK and FACK messages. Without this delay, the handshake process does not function properly, connections are not established, and there are false positive connections between the controller and nodes. A delay time of around 10 ms is sufficient to stabilize the connection establishment between the controller and nodes. The accuracy of data transmission is influenced by the delay time (transDelay) allocated by the controller to send the REQ request message to the node until the controller receives the RESP message containing data from the node. The best delay time from the experiment is tD2 = 5000 ms with an accuracy of 97%.

References

[1] P. Xu, "A brief overview of wireless sensor networks for internet of things," in *Proceedings - 2022 2nd International Conference on Networking, Communications and Information Technology, NetCIT 2022*, 2022. [Online]. Available: https://doi.org/10.1109/NetCIT57419.2022.00021

[2] S. A. Celtek, M. Durgun, and H. Soy, "Internet of things based smart home system design through wireless sensor / actuator networks," in *2nd International Conference on Advanced Information and Communication Technologies, AICT 2017 - Proceedings*, 2017. [Online]. Available: https://doi.org/10.1109/AIACT.2017.8020054

[3] B. S. Adiga, M. A. Rajan, R. Shastry, V. L. Shivraj, and P. Balamuralidhar, "Lightweight ibe scheme for wireless sensor nodes," in *2013 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2013*, 2013. [Online]. Available: https://doi.org/10.1109/ANTS.2013.6802866

[4] W. Meng, L. Xie, and W. Xiao, "Tdoa sensor pairing in multi-hop sensor networks," in *IPSN'12 - Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, 2012. [Online]. Available: https://doi.org/10.1145/2185677.2185692

[5] A. P. Ortega, G. Merrett, S. Ramchurn, and L. Tran-Thanh, "Partner selection in self-organised wireless sensor networks for opportunistic energy negotiation: A multi-armed bandit based approach," *Ad hoc networks*, 2021. [Online]. Available: https://doi.org/10.1016/j.adhoc.2020.102354

[6] I. Udoh and G. Kotonya, "A reinforcement learning qos negotiation model for iot middleware," in *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security*. SCITEPRESS - Science and Technology Publications, 2020. [Online]. Available: https://doi.org/10.5220/0009350102050212

[7] G. Callebaut, G. Ottoy, and L. Perre, "Optimizing transmission of iot nodes in dynamic environments," *Coins*, 2020. [Online]. Available: https://doi.org/10.1109/COINS49042.2020.9191674

[8] P. Banerjee, S. Mandal, D. De, and B. Maiti, "Rl-sleep: Temperature adaptive sleep scheduling using reinforcement learning for sustainable connectivity in wireless sensor networks," *Sustainable Computing: Informatics and Systems*, 2020. [Online]. Available: https://doi.org/10.1016/j.suscom.2020.100380

[9] L. García, L. Parra, J. M. Jiménez, J. Lloret, P. V. Mauri, and P. Lorenz, "Dronaway: A proposal on the use of remote sensing drones as mobile gateway for wsn in precision agriculture," *Applied Sciences*, 2020. [Online]. Available: https://doi.org/10.3390/APP10196668

[10] N. Mazumdar, A. Nag, and S. Nandi, "Hdds: Hierarchical data dissemination strategy for energy optimization in dynamic wireless sensor network under harsh environments," *Ad hoc networks*, 2021. [Online]. Available: https://doi.org/10.1016/j.adhoc.2020.102348

[11] S. Bouarourou, A. Boulaalam, and E. H. Nfaoui, "A bio-inspired adaptive model for search and selection in the internet of things environment," *PeerJ Computer Science*, vol. 7, p. e762, Dec 2021. [Online]. Available: https://doi.org/10.7717/peerj-cs.762

[12] S. Siva, M. B. Wigness, J. Rogers, L. Quang, and H. Zhang, "Nauts: Negotiation for adaptation to unstructured terrain surfaces," *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2022. [Online]. Available: https://doi.org/10.1109/IROS47612.2022.9982207

[13] Z. Wang, H. Chen, and W. Wu, "Client-aware negotiation for secure and efficient data transmission," *Energies (Basel)*, vol. 13, no. 21, Nov 2020. [Online]. Available: https://doi.org/10.3390/en13215777

[14] A. R. 'Susanto, A. 'Bhawiyuga, and K. 'Amron, "Implementasi sistem gateway discovery pada wireless sensor network (wsn) berbasis modul komunikasi lora," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 2, pp. 2138–2145, Feb 2019.

[15] S. Khattar and T. Verma, "Enhancement of the performance and accuracy of soil moisture data transmission in iot," *IOP Conference Series: Earth and Environmental Science*, vol. 1110, no. 1, p. 012001, feb 2023. [Online]. Available: https://doi.org/10.1088/1755-1315/1110/1/012001

[16] T. Wala, N. Chand, and A. K. Sharma, *Energy Efficient Data Collection in Smart Cities Using IoT*. Springer International Publishing, 2020, pp. 632–654. [Online]. Available: https://doi.org/10.1007/978-3-030-40305-8_30

[17] C. K. Dehury, S. N. Srirama, and T. R. Chhetri, "Ccodamic: A framework for coherent coordination of data migration and computation platforms," *Future Generation Computer Systems*, vol. 109, pp. 1–16, aug 2020. [Online]. Available: https://doi.org/10.1016/j.future.2020.03.029

[18] C. Hou, Q. Zhao, and T. Basar, "Optimization of web service-based data-collection system with smart sensor nodes for balance between network traffic and sensing accuracy," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 2022–2034, oct 2021. [Online]. Available: https://doi.org/10.1109/tase.2020.3030835

[19] J. D. Alejandrino, R. S. C. II, V. J. D. Almero, M. G. Palconit, R. R. P. Vicerra, A. Bandala, E. Sybingco, and E. P. Dadios, "Protocol-independent data acquisition for precision farming," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 25, no. 4, pp. 397–403, jul 2021. [Online]. Available: https://doi.org/10.20965/jaciii.2021.p0397

[20] A. Chehri, H. Chaibi, R. Saadane, N. Hakem, and M. Wahbi, "A framework of optimizing the deployment of iot for precision agriculture industry," *Procedia Computer Science*, vol. 176, pp. 2414–2422, 2020. [Online]. Available: https://doi.org/10.1016/j.procs.2020.09.312

[21] B. Sletcha, C. Vivas, F. K. Saleh, A. Ghalambor, and S. Salehi, "Digital oilfield: Review of real-time data-flow

architecture for upstream oil and gas rigs," in *Day 2 Thu, February 20, 2020*. SPE, feb 2020. [Online]. Available: https://doi.org/10.2118/199298-ms

[22] X. Li, L. Zhu, X. Chu, and H. Fu, "Edge computing-enabled wireless sensor networks for multiple data collection tasks in smart agriculture," *Journal of Sensors*, vol. 2020, pp. 1–9, feb 2020. [Online]. Available: https://doi.org/10.1155/2020/4398061