agriculture

MDPI

Article

Design and Implementation of a Malfunction Detection System for Livestock Ventilation Devices in Smart Poultry Farms

Seung Jae Kim [1] and Meong Hun Lee [2,*]

[1] Department of Information and Communication Engineering, Sunchon National University, Suncheon 57922, Republic of Korea
[2] Department of Smart Agriculture Major, Sunchon National University, Suncheon 57922, Republic of Korea
* Correspondence: leemh777@scnu.ac.kr

Abstract: Smart livestock farming aims to improve the productivity of livestock through the provision of optimal housing, and it is developed using various sensors and actuators. Ventilation systems play a crucial role in smart livestock farming, including disease prevention and the processing of pollutants (ammonia and hydrogen sulfide) that are severely detrimental to livestock growth. Malfunctions in animal housing ventilation systems lead to mass mortality events. To address such issues, this study reports the design and implementation for a smart detection system for malfunctions in the ventilation devices installed in animal housing. This system is based on recurrent neural networks (RNNs) and implements the ontology method, considering sensor and controller data as the standard. A semantic sensor network ontology founded on a knowledge base was used to detect malfunctions, and stimulus-sensor-observation patterns were used to determine a sensor network within the smart barn. System activation and RNN model tests were used to test the malfunction detection system, and the error between actual data and predicted values was found to be 0.06889. These findings provide insight into the development of autonomous detection systems for device malfunctions and are essential for the development of smart livestock farming technologies.

Keywords: smart agriculture; Internet of Things; failure prediction; recurrent neural network; ontology

Citation: Kim, S.J.; Lee, M.H. Design and Implementation of a Malfunction Detection System for Livestock Ventilation Devices in Smart Poultry Farms. *Agriculture* **2022**, *12*, 2150. https://doi.org/10.3390/agriculture12122150

Academic Editors: Francesco Marinello and Claudia Arcidiacono

Received: 18 October 2022
Accepted: 7 December 2022
Published: 14 December 2022

1. Introduction

Smart livestock farming involves the use of technologies that monitor the internal and external environments of animal housing using sensors and actuators so as to optimize the housing environment [1,2]. In highly developed nations such as South Korea, the smart livestock farming industry is developing differentiated technologies centered around the integration of software and hardware platforms, intellectualization of data, and fusion with artificial intelligence, cloud computing, and the Internet of Things (IoT). These techniques can help develop an intelligent livestock farming industry with the use of information communication technology (ICT) devices. Research in this industry also requires the technological evolution of intelligent data fusion systems through data collection, analysis, and prediction [3]. The ultimate goal of smart livestock farming is to improve productivity by maintaining the optimal growth environment for livestock and by predicting factors such as gestation periods [4].

Barn ventilation is one of the environmental factors that can significantly impact livestock productivity [5]. The ventilation system plays a crucial role in processing pollutants—such as ammonia and hydrogen sulfide—that are generated within the barn and can be detrimental to the health of livestock [6]. As such, ventilation systems are related to disease prevention and the productivity rate of livestock. Malfunctions in these systems can lead to mass mortality events within the barn, thereby reducing productivity and adversely impacting livestock management [7,8]. Therefore, it is necessary to closely monitor the ICT equipment and related facilities, verify the soundness of the system, and implement

state-based predictive maintenance technologies. These measures can help prevent damage to livestock farms due to accidental faults or failures caused by aging [9].

To address the above-mentioned issues, in this study, we propose a smart system for detecting malfunctions in ventilation devices in poultry farms. We also discuss ontology-based malfunction detection using recurrent neural networks (RNNs), considering sensor and controller data collected throughout the operation of a smart barn as the standard. During the experiment, RNN training data were generated by accounting for major problem factors, including environmental factors—such as external temperature changes and airflow rate—and the use of old filters or aging equipment.

Chapter 2 presents a review of literature on malfunction detection devices, and Chapter 3 introduces our proposed system—including the architecture and composition of the platform—to detect device malfunctions in a smart poultry farm. Chapter 4 discusses various developments, techniques, and experimental procedures based on the aforementioned information, and the collected data are analyzed in the Results section.

2. Review of Literature

Researchers in various industries have investigated the detection of malfunctions in ICT devices. Studies typically identify various types of data abnormalities (with an emphasis on sensor networks technology) and present malfunction detection techniques for each type of data abnormality [10]. Malfunction detection techniques for sensor data can assess a single datum or multiple data within a given space. Most research on malfunction detection has been based on the moving average method with a focus on time series analysis. Techniques have also been developed to detect the accuracy of data classification based on the spatial understanding of sensor values [11]. In the Gaia project—a ubiquitous ontology-based computing system [12]—Bayesian networks (which use prior data to distinguish abnormal values) have been used to estimate inaccuracy and to apply significant corrections.

To date, many studies in other industries have used ontology-based methods, but there is a lack of such studies in the agricultural industry. Therefore, in this study, we aimed to develop a basic model of data analysis that would allow the application of artificial intelligence technology in agriculture. Noisy or ambiguous data obtained from malfunctioning sensors can be processed using sensor outlier models [13]. However, we concluded that the existing malfunction detection technologies primarily utilize individual data types (such as temperature and humidity values), which limits their applicability.

To overcome these limitations, we introduce an ontology-based semantic sensor network founded on a knowledge base and design a malfunction detection system for ventilation devices in a smart barn. Ontology-based semantic sensor networks can handle various attributes of sensors, such as the detection target, detection method, and metadata creation [14]. As such, although this network has been designed for versatility, it is limited by its complexity. The novel concept of stimulus-sensor-observation ontology design pattern has been developed previously, and major patterns related to sensors and batch systems have been introduced to mitigate this limitation of ontology-based semantic sensor networks. This makes it possible to define sensor networks within the context of a smart livestock barn.

The design of this malfunction detection system will lead to standardization of the service interface to respond to abnormal situations in the future of malfunction in the smart agricultural field, thereby increasing the diversity of services related to malfunction in smart livestock and enabling efficient communication of agricultural information through smart terminals.

3. Designing a Smart System for Detecting Malfunctions in Ventilation in Poultry Farms

This chapter presents a diagram of ventilation devices and systems in a smart poultry farm, and we discuss the structural design and detailed general design of a malfunction

detection system. The structural design includes the architecture of the system, software, and integration of internal systems, whereas the general design includes the interface, ontology, and database design.

3.1. Ventilation Devices in a Smart Poultry Farm

The ventilation system of a poultry farm can be described as follows. As shown in Figures 1 and 2, For axial ventilation, pressure is generated by a fan, installed on the outer wall, and a chimney. These components create a temperature difference inside and outside the ventilation shaft, thereby facilitating ventilation. The ventilation device uses serial communication based on the RS-232C communication standard [15] and is configured with several settings, including fan speed, temperature, and Wi-Fi-based remote control systems. The device controls the fans installed inside the inner chimney and walls of the poultry farm. The system processes poisonous gases generated by the livestock and their manure by controlling the coolers and heaters using data obtained from the temperature sensors installed within the barn.

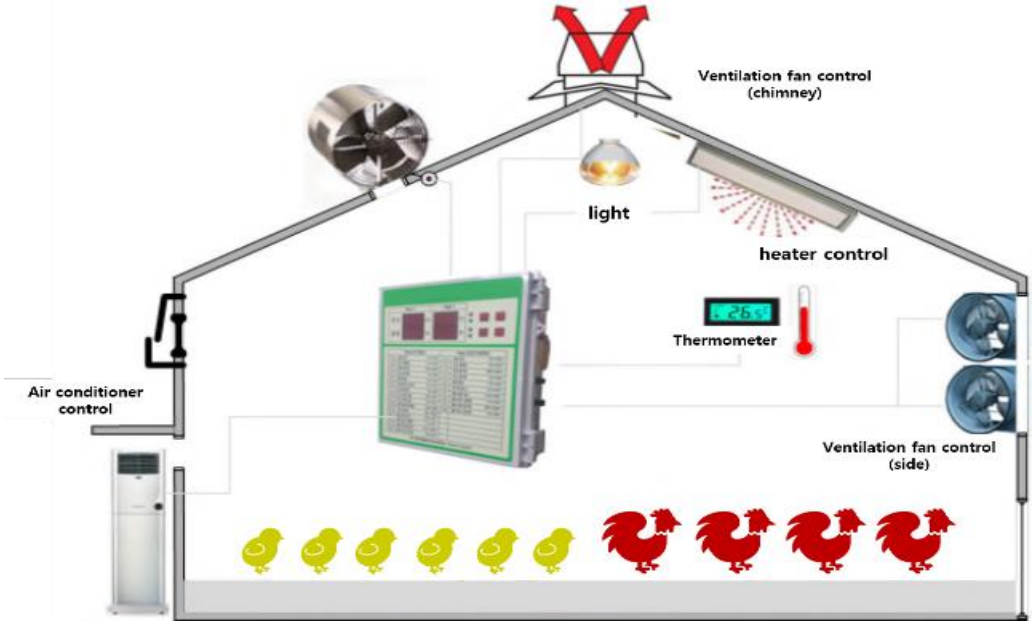

Figure 1. Sample ventilation devices installed in a smart poultry farm.

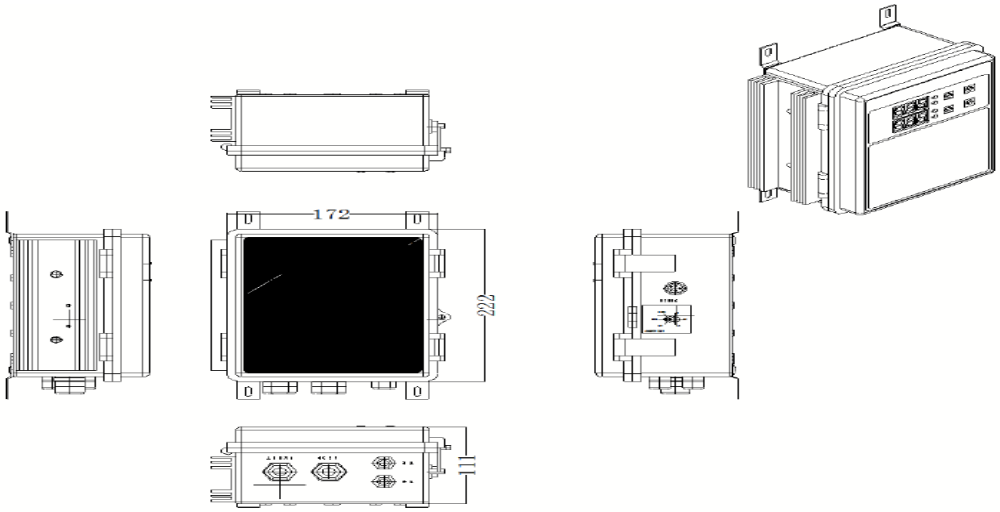

Figure 2. The composition of a ventilation device.

3.2. Structural Design of the System

As shown in Figure 3, The oneM2M-interoperable open API [16] is used as a public interface that links the cloud to the internal data of ventilation devices and sensor data, both of which are obtained locally from the smart poultry farm. Actual data—extracted from the internal interface of the malfunction detection system through the cloud—are obtained using the public interface

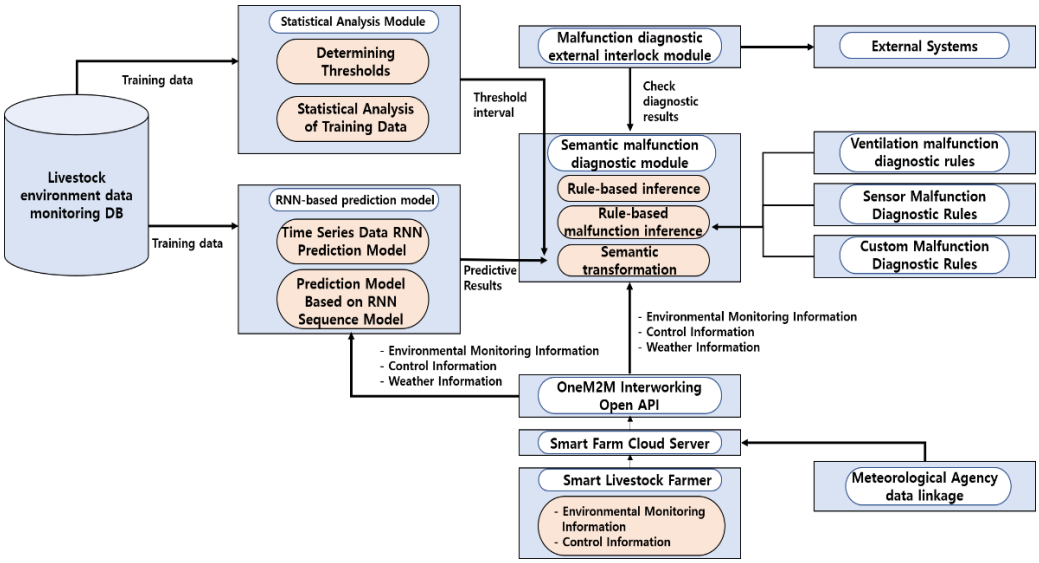

Figure 3. System architecture of a malfunction detection system for ventilation in poultry farms and the functions of various modules.

The environmental data are monitored and logged in a database, which provides training data for RNN-based prediction models. Discrepancies between the actual results and the prediction results of the RNN-based prediction model are determined through statistical analysis by modules, which compute the statistical parameters needed to distinguish malfunctions from normal data. The RNN-based prediction model is trained using time series data and sequential modeling. The system is configured to predict sensor values using the trained model, and the sensor and controller values are presented as input. The semantic module for the malfunction detection system is composed of the smart farm, time, space, event, and malfunction detection ontology. The malfunction detection ontology defines concepts such as threshold values and prediction information and helps identify malfunctions based on statistical analysis. Finally, the external interoperability module provides an interface for relaying the results to users (such as farmers) through a notification interface provided by the cloud [17,18].

3.2.1. Software Architecture

The software as shown in Figure 4, composed of the following layers: an execution environment, a malfunction detection engine, an interface, and a user interface. The execution environment layer runs on a Java runtime environment (JRE) and consists of JRE code (that facilitates development in the Java language), the Hadoop platform (for the distributed processing of big data), Spark (for big data processing and real-time data streaming), and HBase (that facilitates non-stop data-saving for massive volumes of distributed data) [19–24]. The malfunction detection engine layer is composed of an ontology-based predictor (for deducing malfunctions using the RNN-based prediction model and information collected from smart livestock farms [25]), an RNN-based prediction model (for training and prediction using input data [26]), and a statistical computation module (that detects malfunctions using the basic statistics of the training data and the predicted values provided by each model). Finally, the interface and user interface layers are

composed of a public API, an internal API for malfunction detection, a GUI for monitoring the sensors installed in the livestock farm, and another GUI for expressing requests and responses for RNN-based malfunction detection tests.

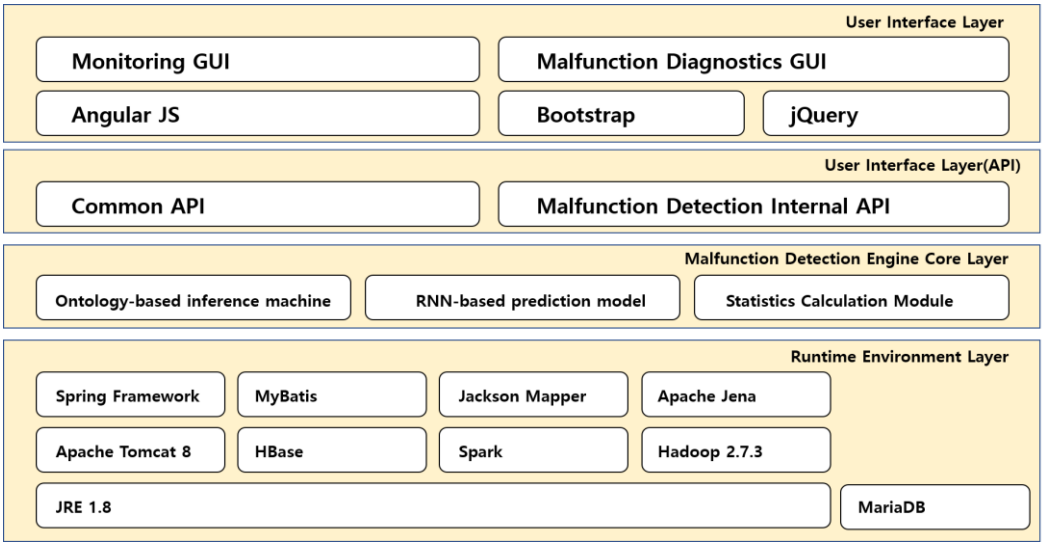

Figure 4. Software architecture of the malfunction detection system for ventilation devices in smart poultry farms.

3.2.2. Integration Architecture for Internal Systems

The malfunction detection module as shown in Figure 5 collects real-time data from the public and internal APIs—including the device control information and measurement data—through the HTTP RESTful API [27]. The HTTP RESTful API stores the collected data for a certain period of time and then performs diagnostic tests based on a scheduler. The scheduled jobs perform RNN-based predictions using the collected information, and execute semantic translation procedures to add the original data and predicted results to the ontology [28]. Once the data have been translated into the triple format, they are added to the semantic storage. Pre-registered rules are used to judge the validity of malfunction detections, and the ontology is tasked with sending cloud notifications for future malfunction events. Such events can be saved and managed in RDBMS/HBase [29]. Queries related to the malfunction detection history are directly sent to RDBMS and HBase through the API, which allows users to inspect previous malfunction events.

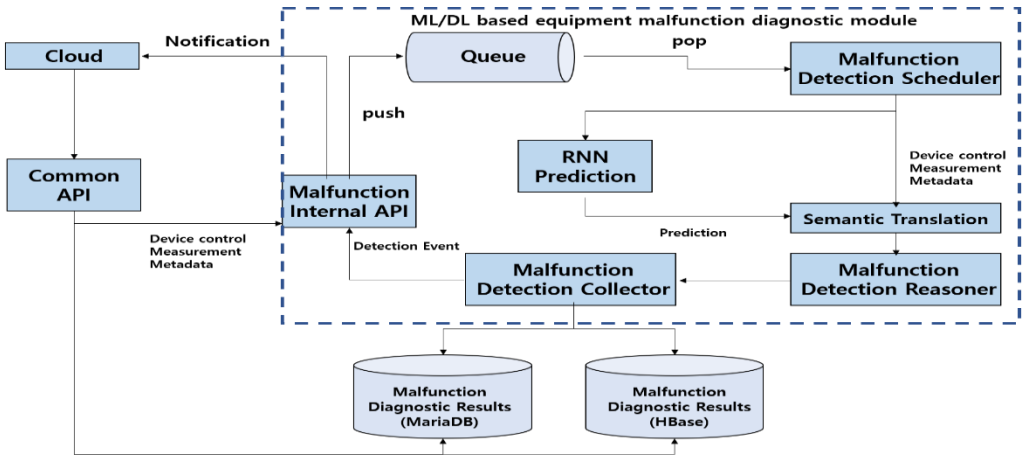

Figure 5. Integration architecture of a malfunction detection system for ventilation in poultry farms.

3.3. Detailed Design of the System

This section discusses the components of a malfunction detection system in detail, including the design of the interface, ontology, and database.

3.3.1. Interface Design

The interface is an open API based on the HTTP RESTFul API. Inquiries and notifications regarding malfunction detection results use the JSON format for each identification code. Each interface as shown in Table 1 configured to add malfunction events based on pre-defined rules.

Table 1. System rules for the addition of malfunction events.

Standard	Content
Rule 1	When the device type (e.g., temperature sensor) is identical to another device installed in the same zone, and the sensor values differ from each other, a malfunction event is added.
Rule 2	When a heater in another device is close to a temperature sensor, a malfunction event is added.
Rule 3	When temperature and humidity sensors are installed in the same zone, and the temperature is $\leq$5 °C degrees and the humidity is $\leq$10 °C, a malfunction event is added.
Rule 4	When the device has been initialized but the measurements of sensors installed in the same zone are 0, a malfunction event is added.
Rule 5	When an RNN-based predicted value exists for an installed equipment, and the absolute predicted value exceeds the absolute value of the threshold for the equipment, a malfunction event is added.

When the malfunction-retrieve command is identified, as shown in Figure 6, the system extracts specific inspection requests from the history of malfunction diagnosis results and relays the relevant results as malfunction-response messages. The results associated with malfunction detection are communicated along with the malfunction-notification message.

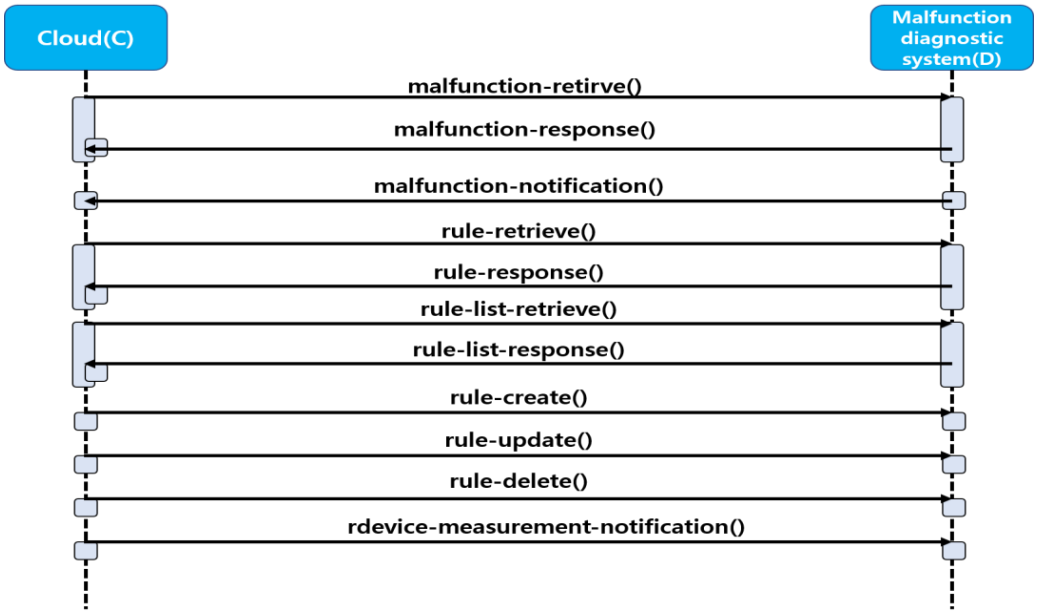

Figure 6. Example of a system interface for users to diagnose device malfunctions through the cloud.

The inspection interface displays as shown in Figure 7 the rules for malfunction detection in individual devices in the form of URLs and messages. When the user inserts a rule-retrieve message in the system, it returns the values that define malfunction detection thresholds for individual devices.

Malfunction diagnostic rule inquiry	**C**	**D**	**GET** **http://etrl/malfunction/rule/farms/(livestock_owner_ID)/(livestoct_unqno)** **200 OK** **{** **"rule" :** **{** **"livestock_owner_ID" : "1",** **"livestock_unqno" : "101",** **"rule_id" : "100000",** **"rule_name" : "Multi-sensor condition rule",** **"rule_id" : "threshold",** **"device_class_cd" : "temp_sensor",** **"min_value" : "5",** **"max_value" : "45"** **"operator1" : ">",** **"operator2" : "<=",** **"operator3" : ">=",** **"value1" : "30",** **"value2" : "70",** **"value3" : "300",** **"power_consum" : "100",** **"user_rule" : " ",** **}** **}**

Figure 7. Items of the inspection interface that define the rules for malfunction detection in individual devices.

3.3.2. Ontology Design

To detect malfunctions in ventilation devices in a smart barn, the system ontology must be defined to allow the execution of diagnoses founded on a knowledge base. In a barn, key environmental factors are measured by sensors, and the data are compiled through sensor networks. Therefore, we defined the ontology of our system based on the semantic sensor network ontology [30].

The semantic sensor network ontology was designed for versatility in various aspects, including the detection target, detection method, metadata, batch system with sensors, and other attributes. Due to its complexity, the semantic sensor network ontology (in its original form) is inefficient and inappropriate for real-world applications. Therefore, in the context of a smart barn, the sensor network was defined based on the stimulus–sensor–observation pattern [31] and other patterns related to batch systems with sensors.

The W3C ontology was used to depict time [32]. To depict ventilation-related devices installed throughout the smart barn, the concept of space was defined with reference to publicly available spatial information ontologies. As shown in Figure 8 multiple ontologies were integrated to facilitate malfunction detection in a smart barn environment.

The different ontologies are described as follows:

1. The event ontology defines major events indicating the outcomes of malfunction detection. Information related to device configuration and actual measurement values are interpreted through rule-based inferences, based on rules provided by the user. Following semantic translation, the results are saved in the form of RDF Triples [33].
2. The detection ontology indicates the malfunction threshold for each device. The time series of sensor data are used to train an RNN model, and the results are used to predict sensor values. Various concepts are defined to indicate the predicted values, which are needed to identify device malfunctions. When certain thresholds are exceeded, a device malfunction is indicated based on the discrepancy between predicted values and actual sensor measurements.
3. The sensor network ontology consists of concepts that represent various devices installed throughout the smart barn and their measurements. The measurements obtained from these devices (such as sensors and actuators) are utilized as basic patterns with reference to the semantic sensor network ontology.

4. The GeoSpatial ontology utilizes the concept of space and structural relationships to define the space (such as an animal barn) where the devices (such as sensors and actuators) are installed.
5. The OWL-2 DL ontology defines temporal concepts and is used to describe the temporal characteristics of resources [34]. It also provides the lexicon for describing information related to duration, the relationships between the temporal locations—including date and time—of each moment, the intervals between moments, and their temporal order.

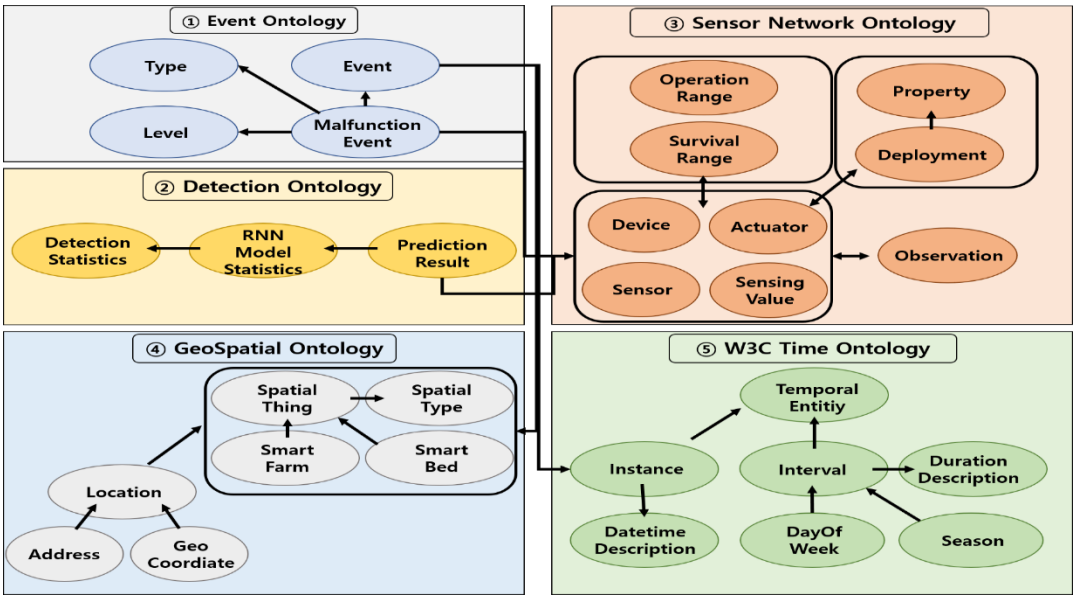

Figure 8. Relationships of various ontologies implemented in the malfunction detection system.

3.3.3. Database Design

As shown in Table 2, the results of malfunction detection are presented in the form of a table that contains information related to the detection results, the livestock farm where the malfunction event occurred, the device number, the time of detection, and the contents of the notification sent to the user.

As shown in Table 3, the rules for malfunction detection are listed in a separate table. This is mainly based on a comparative analysis of data obtained from different devices. The values are inspected and analyzed according to the detection ontology, following which the results of the malfunction detection are reported. Discrepancies between the predicted and actual values are analyzed through comparisons between sensors installed in different zones within the barn.

Table 2. Database showing the results of malfunction detection and the corresponding responses, including the contents of the notification sent to the user.

Malfunction Detection Results						
SYSTEM	**Common Framework Based on ML/DL**	**SUB SYSTEM**	**Malfunction Detection**			
TABLE ID	Malfunction_ detection	TABLE NAME	Malfunction Detection Results			
NO	COLUMN ID	COLUMN NAME	DATA TYPE	NULL	KEY	REMARK
1	Detection_id	Diagnostic Result Identifier	VARCHAR(20)	NOTNULL	PK	md_yymmddhhmmssSSS
2	Livestock_owner_ID	Livestock Farm ID	VARCHAR(30)	NOTNULL	PK	1

Table 2. *Cont.*

Malfunction Detection Results						
SYSTEM	**Common Framework Based on ML/DL**	**SUB SYSTEM**	**Malfunction Detection**			
3	Livestock_unqno	Livestock Farm Identifier	VARCHAR(6)	NOTNULL	PK	101
4	Barn_unqno	Place Identifier	VARCHAR(10)	NOTNULL	PK	1001
5	Instal_device_unqno	Equipment unique Number	VARCHAR(6)	NOTNULL	PK	1001-1
6	Instal_device_nm	Installation Equipment Name	VARCHAR(50)	NOTNULL		Internal temperature
7	Device_class_cd	Equipment classification code	VARCHAR(6)	NOTNULL		Temp_sensor
8	Detection_time	Detection time	TIMESTAMP	NOTNULL		2000-00-00-00:00:00
9	Mesur_value	Mesur value	DECIMAL(10,2)	NOTNULL		50
10	Detection_type	Detection type	VARCHAR(64)	NOTNULL		Over threshold
11	expt	exception	VARCHAR(1024)	NOTNULL		Temperature sensor threshold exceeded.

Table 3. Database showing the rules for malfunction detection.

Malfunction Detection Rules						
SYSTEM	**Common Framework Based on ML/DL**	**SUB SYSTEM**	**Malfunction Detection**			
TABLE ID	Malfunction_ detection	TABLE NAME	Malfunction Detection Results			
NO	COLUMN ID	COLUMN NAME	DATA TYPE	NULL	KEY	REMARK
1	Livestock_owner_ID	Livestock Farm ID	VARCHAR(30)	NOTNULL	PK	1
2	Livestock_unqno	Livestock Farm Identifier	VARCHAR(6)	NOTNULL	PK	1
3	Rule_id	Rule Identifier	VARCHAR(32)	NOTNULL	PK	The_rule_001
4	Rule_name	Rule name	VARCHAR(64)	NOTNULL		Sensor threshold rule
5	Rule_type	Rule type	VARCHAR(16)	NOTNULL		Threshold, multisensory, actuator, User_defined
6	Device_class_cd	Equipment classification code	VARCHAR(6)	NOTNULL		Temp_sensor
7	Min_value	Min value	DECIMAL(10,2)			30.5
8	Max_value	Max value	DECIMAL(10,2)			50.5
9	Device_calss_cd2	Equipment classification code	VARCHAR(6)			A_sensor
10	Device_calss_cd3	Equipment classification code	VARCHAR(6)			B_sensor
11	Operator1	Equipment 1 Comparison Operator	VARCHAR(3)			>, <, >=, <=, ==, !=
12	Operator2	Equipment 2 Comparison Operator	VARCHAR(3)			>, <, >=, <=, ==, !=

Table 3. *Cont.*

Malfunction Detection Rules				
SYSTEM	**Common Framework Based on ML/DL**	**SUB SYSTEM**	**Malfunction Detection**	
13	Operator3	Equipment 3 Comparison Operator	VARCHAR(3)	>, <, >=, <=, ==, !=
14	Value1	Equipment 1 Comparison Value	DECIMAL(10,2)	50.5
15	Value2	Equipment 2 Comparison Value	DECIMAL(10,2)	50.5
16	Value3	Equipment 3 Comparison Value	DECIMAL(10,2)	50.5
17	Power_consum	Power consumption	DECIMAL(10,2)	Power Consumption Comparison
18	Power_consumption	Actuator Power consumption	DECIMAL(10,2)	390.0 (Wh)
19	User_rule	Custom Rule Sentences	VARCHAR(2048)	

4. Implementation and Outcomes of a Malfunction Prediction System for Smart Livestock Farming

Section 4.1 discusses IoT sensors and their control, the implementation of an RNN-based malfunction prediction model for field devices, and coordination between servers across different hubs. Section 4.2 discusses the results of on-site experiments in which we tested the RNN model and device interoperability by applying the proposed system in practice. To apply the ontology method for diagnosing device malfunctions using this system, it is necessary to identify the critical point of each sensor in a smart farm. Using this method, if the sensor indicates an abnormal value or shows a large deviance from the predicted value, this may indicate a device malfunction.

4.1. Implementation of an RNN-Based Prediction Model for Malfunction Detection

Before implementing the RNN model, we first identified various factors that affected the operation of ventilation devices in the poultry farm selected for the study. We analyzed the data collected over a period of 1 year and identified the following major factors: changes in external temperature due to seasonal change, excessive control of the airflow rate, and the use of old filters. The data were cleaned and organized while accounting for these factors, and this database was used to train the RNN. Using these training data, a many-to-one model was created, followed by training with a time series of actual sensor values classified into sequences. The differences between prediction values—calculated at the time of training—and actual measurements followed a normal distribution with a set of confidence intervals. The thresholds were determined by semantic translation. Ultimately, this system predicted malfunctions using sensor data and was implemented to resolve the discrepancy between predicted and actual values. The discrepancies underwent semantic translation and were used to determine whether a device was malfunctioning. This determination was based on pre-defined rules and comparisons with the thresholds calculated by the model.

Structure of the RNN Prediction Model

Table 4 indicates the data used to implement the RNN prediction model in the malfunction detection system for ventilation devices in smart poultry farms. Each database lists the source of the training and prediction models, sensor data from livestock farms, and data used for the prediction tests. The training model was composed of constants and

global variables to which normalization and inverse normalization functions were applied. The training and test datasets were uploaded separately.

Table 4. File structure of the prediction model for malfunction detection.

Directory	File	Outline
malfunction_rnn	malfunction_rnn_prediction_model.py	Training model source
	malfunction_rnn_predicttion.py	Prediction model source
malfunction_rnn/model	Checkpoint Malfunction_predict.pd.data-00000-of-00001 Malfunction_predict.pd.index Malfunction_predict.pd.meta	Saved training model
malfunction_rnn/temp	PF_01_Train.csv PF_02_Train.csv PF_03_Train.csv PF_04_Train.csv	Temperature data for each training poultry farm
malfunction_rnn/test	PF_01_Pridict.csv PF_02_Pridict.csv PF_03_Pridict.csv PF_04_Pridict.csv	Data for prediction tests

As shown in Table 5, the main code of the training model loaded data from the directory of each farm, defined the RNN Cell/Multi-RNN Cell, and configured the RNN network. A fully connected layer code was used to verify the training and test datasets.

Table 5. Main code of the training model.

```
print('size of training: ' + str(len(tainX)))
print('size of test: ' + str(len(inputX)))

X = tf.placeholder(tf.float 32, [None, seq_length, data_dim])
Y = tf.placeholder(tf.float32, [None, 1])

cell = tf.contrib.rnn.GRUCell(
num_units = hidden_dim, activation = tf.tanh)
cells = tf.contrib.rnn.MultiRNNCell([cell] * NUMBER_OF_RNN_CELL_LAYERS);

outputs, _states = tf.nn.dynamic_rnn(cell, X, dtype = tf.float 32)

Y_pred = tf.contrib.layers.fully_connected(
outputs[:, -1], output_dim, activation_fn=None)
```

As shown in Table 6, a cost function was added to execute the train node, conduct a globally-established number of training sessions, and save the results. AdamOptimizer was used to minimize the cost.

Table 6. Code of the cost minimization process for the training model.

```
loss = tf.reduce_sum(tf.square(Y_pred - Y))
tf.summary.scalar("cost", loss)
summary = tf.summary.merge_all()
optimizer = tf.tarin.AclamOptimizer(learning_rate)
targets = tf.placeholder(tf.float32, [None, 1])
rmse = tf.sqrt(tf.reduce_mean(tf.square(targets - predictions)))

with tf.Session() as sess:
init = tf.global_variables_initializer()
sess.run(init)

# Create summary writer
writer = tf.summary.FileWriter(TB_SUMMARY_DIR)
writer.add_graph(sess.graph)
global_step = 0
```

As shown in Table 7, to obtain the training results, the test data—which had been created using data from livestock farms—were used to return test plots for the trained model. The prediction model returned predicted values that were calculated based on actual sensor sequences, where the input file path was provided through a command line argument.

Table 7. Code for displaying test plots for the training model.

```
test_predict = sess.run(Y.pred, feed_dict = {X: firstTestX})
rmse_val = sess.run(rmse, feed_dict = {
targets: firstTestY, predictions: test_predict})
print("firstTestX RMSE: {}".format(rmse_val))

correst_prediction = test_predict - firstTestY
accuracy = tf.reduce_mean(correct_prediction)

plt.figure(figsize =(20, 6))
plt.plot(RevMinMaxScaler(firstTestY), 'b-', label = 'Sensing')
plt.plot(RevMinMaxScaler(test_predict), 'r-', label = 'Prediction')
plt.xlabel("Time Period")
plt.Ylabel("Temperature")
plt.legend( lod = 'best')

plt.show()
})
```

The structure of the prediction model source as shown in Table 8, identical to that of the training model source. During the prediction phase, the prediction model calculated the discrepancy between actual measurements and predicted values to determine whether a device malfunction had been accurately identified.

Table 8. Code for displaying the test plots for the prediction model.

```
prdict = RevMinMaxScaler(sess.run(Y_pred, feed_dict = {X: inputX}))
print("Predict: " + str(predict))

variance = abs(predict - RevMinMaxScaler(sensingValueY))
print("Variance: " + str(variance))
```

4.2. System Activation Test

This section discusses the procedures for configuring the ontology-based malfunction detection system for ventilation devices in poultry farms.

To facilitate the diagnosis of device malfunctions, a test environment needs to be configured to verify the prediction model. Throughout the experiment, we tested the functioning of linkages between the API, diagnostic units, and various sensors. As shown in Figure 9 and Table 9, the equipment used in the experiment included a Zigbee sensor node, a Zigbee gateway equipment, and a poultry farm ventilation system.

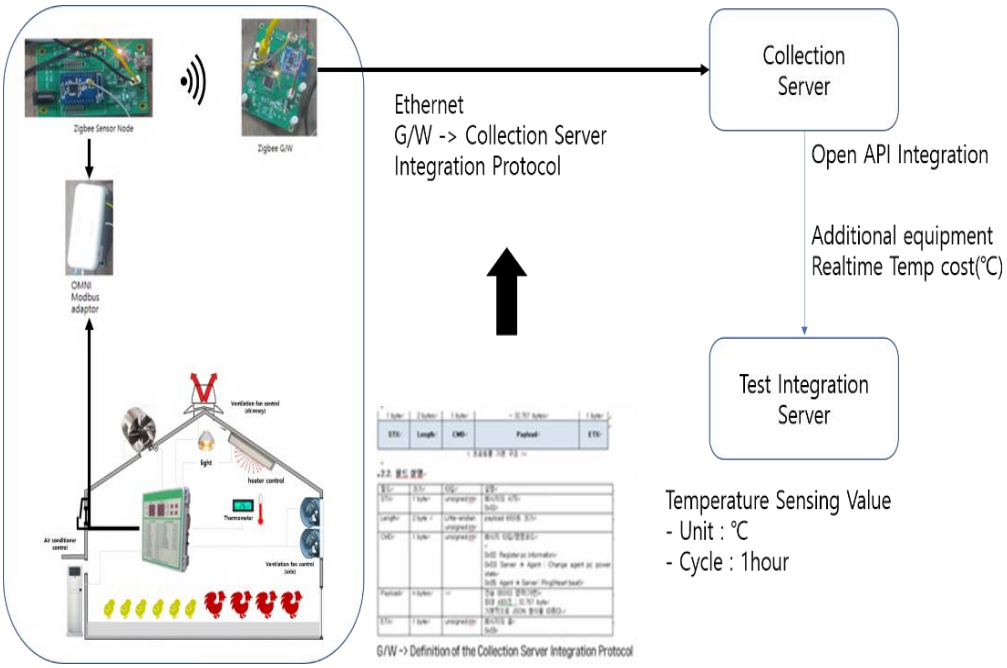

Figure 9. Procedures for configuring a test environment.

Table 9. Key functions of the test equipment used in the experiment.

Equipment	Description
Zigbee sensor node	Supports Zigbee communication capabilities Provides protocol support for communication with Zigbee gateways Provides RS-485-based communication assistance Temperature measurement range: −40 °C to 80 °C
Zigbee gateway	802.15.4 IEEE Zigbee provides RS-485-based communication capabilities Transmission of temperature measurement information using RS-485
Poultry farm ventilation system	Remote ventilation control system Provides functions to set fan speed and temperature Supports RS-485 serial, Zigbee, Wi-Fi communications

As shown in Figure 10, the communication protocol for data collection is displayed as a message block.

Figure 10. Message block structure of the communication protocol.

Table 10 describes the contents of each field in the communication protocol, including the size, type, and operational contents of each field.

Table 10. Communication protocol details.

Field	Size	Type	Description
STX	1 byte	unsigned int	Start of message 0 × 02
Length	2 bytes	Little-endian unsigned int	payload byte size
CMD	1 byte	unsigned int	Message Type/Command Code 0 × 02: Register pc information 0 × 03: Server → Agent: Change agent pc power state 0 × 05: Agent → Server: Ping (Heartbeat)
Payload	n bytes	-	Transmission Data Area (Variable) Maximum size: 32,767 bytes Follows the JSON format.
ETX	1 byte	unsigned int	End of message 0 × 03

Next, the protocol for gateway registration is defined as shown in Table 11. The defined protocol registers network configuration information—such as the gateway MAC, ventilation device ID, and channel—and configures the gateway to transmit at the time of first connection to the server or on reconnection after connection termination. The communication is transmitted from the gateway to the server.

Table 11. Communication protocol for gateway registration.

Field	Size	Type	Description
STX	1	0 × 02	Message Start Indicator
Length	2	0 × 01	Payload data byte size
CMD	1	0 × 64	Massage type (0 × 64)
Payload	variable	JSON	KEY: mac: MAC address panId: PAN ID channel: channel ex) {“mac”: “E0-43-DB-0B-1F-20”, “panId”: 1, “channel”: 11}
ETX	1	0 × 03	Message End Indicator

The protocol for sensor node registration is defined as shown in Table 12. The defined protocol is set to register the MAC address and model name for the temperature sensor node.

The communication protocol for periodically transmitting temperature measurements (as recorded by the temperature sensor) to the server is shown in Table 13. These data are transferred from the sensor node to the server.

For the experiment, the testing environment for activation tests were configured as shown in Table 14.

Table 12. Communication protocol for sensor node registration.

Field	Size	Type	Description
STX	1	0 × 02	Message Start Indicator
Length	2	0 × 01	Payload data byte size
CMD	1	0 × 65	Massage type (0 × 65)
Payload	variable	JSON	KEY: mac: MAC address(optional) address: Equipment Identifier model: model name ex) {“mac”: “E0-43-DB-0B-1F-20”, “address”:1, “model”: “XXXX-111A”}
ETX	1	0 × 03	Message End Indicator

Table 13. Communication protocol for transmitting temperature measurements.

Field	Size	Type	Description
STX	1	0 × 02	Message Start Indicator
Length	2	0 × 01	Payload data byte size
CMD	1	0 × 04	Massage type (0 × 04)
Payload	variable	JSON	KEY: mac Sensor Node MAC value: Sensing Value unit: unit type: Data type(“byte”, “int”, “float”, “double”, “string”) ex) {“mac”: “E0-43-DB-0B-1F-20”, “value”: “35”, “unit”: “°C”, “type”:“int”}
ETX	1	0 × 03	Message End Indicator

Table 14. Testing environment for the activation test of our proposed malfunction detection system for ventilation devices.

Name of Device	Outline	Installation Software
DL API Test Server	DL Server	Hadoop 2.6.0 Hive 2.1.0 or higher Maria DB JDK 1.7
Client Device	Windows 10	POSTMAN Chrome Browser
Malfunction detection devices for ventilation	Gateway and Sensor Nodes	Gateway F/W Linkage of ventilation malfunction detection measurement devices and gateway communication F/W

Because the activation test required browsers capable of running AJAX, compatible browsers—POSTMAN and Chrome—were selected for use. Servers used Hadoop versions 2.6.0 or higher and Hive versions 2.x or higher. The client and server were configured in accordance with this, with the firewall being opened at the HTTP 80 port. Finally, the malfunction detection systems for ventilation devices were configured with an appropriate

network composition, IP and firewall settings, and port forwarding, so that they could be connected to the server.

API Test for Interoperability in Malfunction Detection

The interoperability tests as shown in Figure 11, included verifying the results after using a browser in the POSTMAN REST Client sphere to test whether a malfunction occurred on the device in accordance with predefined rules. The interoperability API test executed the following procedures: inspection of the results of device malfunction; creation of thresholds, actuator rules, multi-sensor rules, and user-defined rules; inspection of rules; inspection of lists of rules; alteration of threshold rules, actuator rules, multi-sensor rules, and user-defined rules; deletion of user-defined rules; decision on whether to collect measurements from devices; and an interoperability test for malfunction detection in ventilation devices.

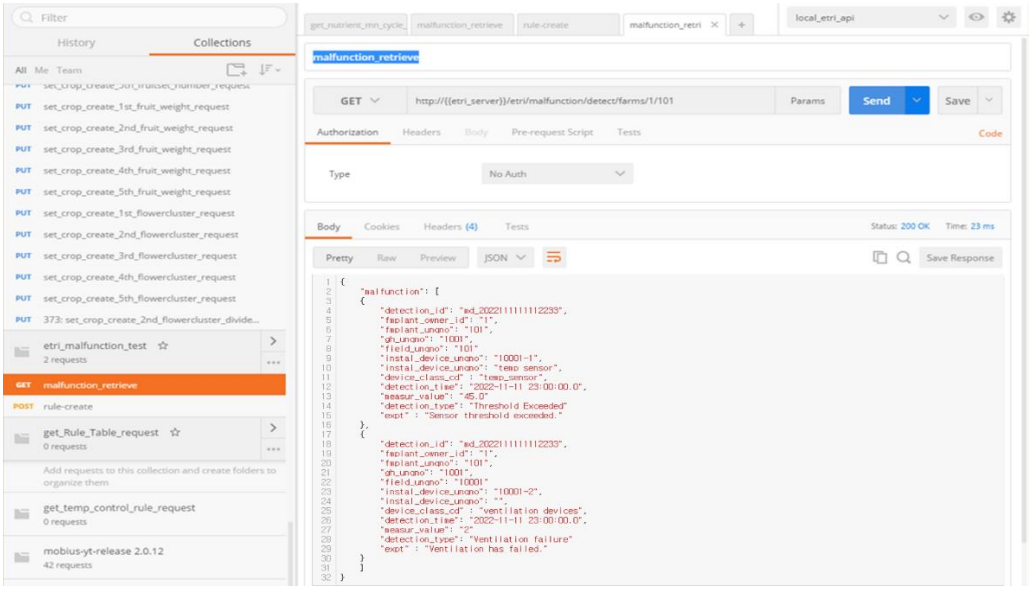

Figure 11. API test for interoperability in malfunction detection in ventilation devices.

Figure 11 shows the process of setting input data and checking HTTP response in conjunction with an actual server. One shows the API test of whether the temperature sensor installed in the ventilation system may malfunction, and the other shows whether the ventilation system itself may malfunction.

4.3. Training and Validation of the RNN Model

This section discusses the process of modifying data for training the RNN model. Approximately 20% of the original data were used as test data to verify the model. The biggest advantage of RNN as shown in Figure 12, that it can create various structures in a flexible manner owing to its network architecture, which can take inputs of any length.

For data analysis, we used test bed data measured at 1-h intervals for 24 h every Saturday. These data were classified into various categories such as internal environment, external environment, usage of ventilation devices, and airflow rate of ventilation fans. The training dataset was created by modifying the temperature data because temperature was the primary factor affecting the ventilation devices within the barn. The temperature within the barn was heavily impacted by period (1–365), time (0–23), external temperature, quantity of solar radiation, rainfall, and internal temperature [35–38]. The training dataset as shown in Figure 13, included the following items: A (period), B (time), C (external temperature), D (quantity of solar radiation), E (rainfall), and F (internal temperature).

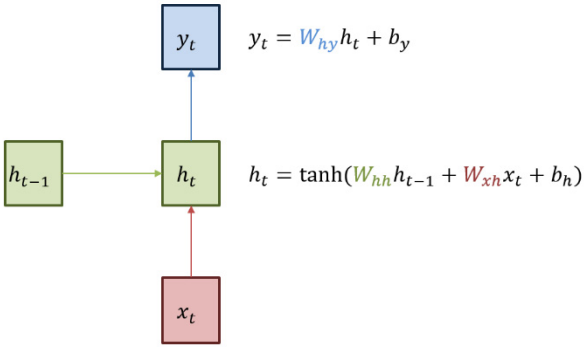

Figure 12. Basic architecture of the recurrent neural network. The green boxes indicate hidden states (h), the red box is the input (x), and the blue box is the output (y). The current hidden state (h_t) is updated by using the previous hidden state (h_{t-1}). The current output (y_t) is updated using h_t, as shown by the formula.

	A	B	C	D	E	F
1	300	0	4.55	0	0	13.5
2	300	1	4.52	0	0	13.2
3	300	2	4.05	0	0	13.6
4	300	3	3.8	0	0	13.8
5	300	4	3.5	0	0	14.12
6	300	5	3.82	0	0	13.8
7	300	6	3.91	0	0	13.91
8	300	7	4.1	0	0	13.8
9	300	8	4.5	0	35.8	14.3
10	300	9	5.2	0	36.2	15.06
11	300	10	7	0	35.2	15.62
12	300	11	7.2	0	34.1	16.21
13	300	12	7.62	0	92.5	16.52
14	300	13	7.8	0	94.6	17.2
15	300	14	7.51	0	150.1	19.5
16	300	15	7.6	0	175.5	21.32
17	300	16	7.51	0	180.2	23.56
18	300	17	7.2	0	200.5	20.52
19	300	18	6.15	0	222.3	19.56
20	300	19	6.02	0	235.6	18.55
21	300	20	5.51	0	255.2	17.42
22	300	21	5.12	0	261.5	15.2
23	300	22	4.81	0	270	14.21
24	300	23	4.72	0	285	13.92

Figure 13. Configured training dataset for detecting malfunctions in ventilation devices using the recurrent neural network model.

As shown in Table 15, the model parameters were defined to facilitate the training of the RNN model.

The number of training sessions were 500, 1,000, 5,000, 10,000, 50,000 and 100,000. The results indicated that there were no significant differences beyond $\geq$ 50,000 training sessions. We used the root mean squared error (RMSE) to verify the results of the training. As shown in Table 16, the RSME values of the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models were derived from the RNN cells.

Table 15. Parameters defined to facilitate the training of the recurrent neural network model.

Parameter Name	Value
Training Data Size	3590
Test Data Size	1006
RNN Cell	LSTM
Layer Stack Size	5
Data Dim	6
Hidden Dim	10
Sequence Len	5
Iteration	500
Learning rate	0.01
Optimizer	AdamOptimizer

As shown in Figure 14, the results of the training were used to deduce cost values.

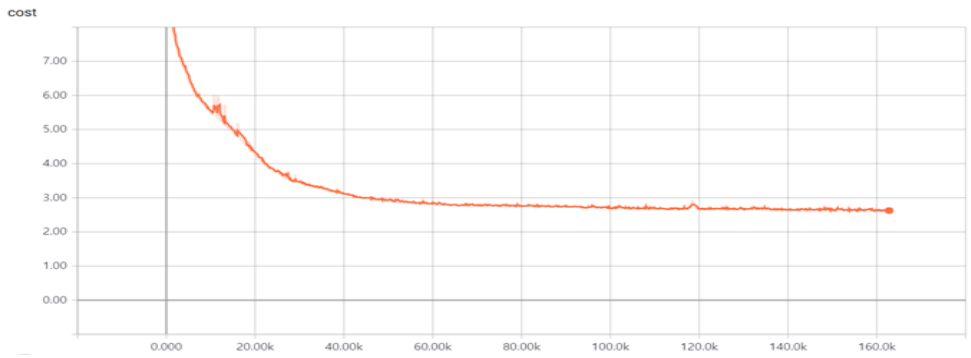

Figure 14. Cost analysis based on the training results of the recurrent neural network model.

Table 16. Root mean squared error (RMSE) values of the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models.

RNN Cell	RSME
LSTM	0.0689
GRU	0.0643

The results showed that the RSME values of LSTM were 0.004 higher than those of GRU. Therefore, the RNN cells were found to be suitable for LSTM. Table 17 summarizes the RSME values according to the sequence length.

Table 17. Root mean squared error (RMSE) values based on sequence length.

Sequence Length	RSME
3	0.0697
5	0.0682
7	0.0725
10	0.0716

There were no significant differences in RSME between different sequence lengths. However, a sequence length of 5 provided the optimal RSME, and this was used for the

experiment. As shown in Table 18, we also evaluated the RMSE for different values of the hidden dimension.

Table 18. Root mean squared error (RMSE) values based on the value of the hidden dimension.

Hidden Dimension	RSME
3	0.0832
10	0.0692

A hidden dimension value of 10 provided the least RMSE and was used for the experiment. In addition, we evaluated the RMSE associated with different numbers of hidden layers (that is, the number of RNN cells stacked in the multi-RNN layer).

The RMSE was lowest when the number of hidden layers was five (Table 19). Therefore, five hidden layers were used for analysis. Finally, we evaluated the differences between different numbers of training sessions (500, 1,000, 5,000, 10,000, 50,000 and 100,000) (Table 20).

Table 19. Root mean squared error (RMSE) values based on the number of hidden layers.

Hidden Layer	RSME
5	0.0692
10	0.0713

Table 20. Root mean squared error (RMSE) values based on the number of training sessions.

Number of Training Iterations	RSME
500	0.0755
1000	0.0734
5000	0.0725
10,000	0.0691
50,000	0.0689
100,000	0.0958

The RSME value was lowest when the model was trained >50,000 times, but it increased as the number of training sessions increased to 100,000 times (overfitting). As shown in Figure 15, the prediction results using the RNN model indicated an RMSE of 0.0689.

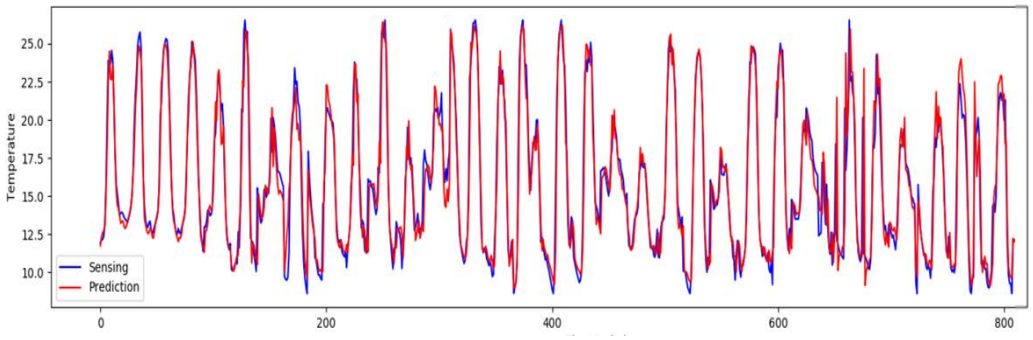

Figure 15. Prediction results using the recurrent neural network model. Errors between actual sensor data (blue lines) and predicted values (red lines) are shown.

As shown in Figure 15, the overall test outcomes revealed a small error between actual sensor data and predicted values.

Errors in temperature data are the key cause of malfunctions in ventilation devices. By predicting temperature data using RNN-based prediction models, we can predict device malfunctions and provide preemptive notifications to farmers. This can help prevent malfunctions in ventilation devices and contribute to the prevention of mass mortality events among livestock.

5. Conclusions

In enclosed barns (such as those used in poultry farms), dust accumulation around ventilation devices and the release of poisonous gases (such as ammonia and hydrogen sulfide) can lead to problems such as device malfunctions. The current study proposes the design and implementation of a new technology to detect malfunctions in ventilation devices in livestock farming, which is one of the necessities of a smart poultry farm. Here, we used a semantic sensor network ontology founded on a knowledge base to detect device malfunctions in a barn. We also implemented several key concepts (e.g., stimulus-sensor-observation patterns) to define a sensor network within the smart barn environment. We introduce an interface module to the malfunction detection system, which is designed to calculate the discrepancy between actual measurements and predicted values through RNN-based data prediction. This module also sends notifications to the farm if the established rules indicate the possibility of a malfunction event. In addition, a rule database was implemented to analyze the degree of difference between actual measurements and predicted values by comparing the data from multiple sensors installed in different regions of a poultry barn. The actual error rate in the field was measured by conducting on-site experiments. Following experimentation, the error was determined to be 0.06889, indicating a small error between actual sensor data and predicted values. Malfunctions in ventilation devices can cause critical damage to livestock production, as even a single malfunction can be fatal for poultry. This necessitates research on the immediate detection or prediction of device malfunctions. Our findings contribute to the development of autonomous detection systems for malfunctions in ventilation devices—a core technology in smart livestock farming—and provide evidentiary basis for the development of smart livestock farming technologies. The study presented in this paper analyzed training data related to temperature, which is a major factor in ventilation device malfunctions, then made predictions with this data. However, additional data such as vibration and dust amount measurements, are factors that may cause malfunctioning of the device. In future studies, these factors are to be examined carefully in the training data for livestock ventilation system so that more precise malfunction prediction can be performed. In addition, it plans to contribute to measures to cope with malfunctions, such as submitting a standard agenda for rapid response to the failure of these smart livestock ventilation devices.

Author Contributions: Conceptualization, S.J.K.; methodology, M.H.L.; software design and implementation S.J.K.; verification and testing S.J.K.; resources S.J.K. and M.H.L.; paper writing and editing S.J.K.; paper review and feedback, M.H.L.; project supervisor, M.H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) and Korea Smart Farm R&D Foundation (KosFarm) through the Smart Farm Innovation Technology Development Program funded the by Ministry of Agriculture, Food and Rural Affairs (MAFRA) and Ministry of Science and ICT (MSIT), Rural Development Administration (RDA) (421021-03).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Ku, H.; Lee, R.; Park, Y. Development of a U-IT Based Monitoring System for the Feeding and Environmental Management of Livestock Animal Production. 2014. Available online: https://scienceon.kisti.re.kr/srch/selectPORSrchReport.do?cn=TRKO201500011294 (accessed on 19 November 2022).
2. Jeong, W.; Ahn, Y.; Lee, R.; Koo, H. A Study of Ubiquitous-based Stall Management System. *J. Anim. Environ. Sci.* **2014**, *20*, 57–62. [CrossRef]
3. Kim, S.; Lee, J.; Choi, B. Design and implementation of malfunction detection of smart farm systems. In Proceedings of the Electronic Information and Communication Academic Conference (CEIC), Yeosu, Korea, 27–28 October 2017; pp. 175–178.
4. Lee, J.; Kim, S.; Lee, S.; Choi, H.; Jeong, J. A study on the necessity and construction plan of IoT-based open platform for the spread of smart agriculture. *J. Korea Multimed. Soc.* **2014**, *17*, 1313–1324. [CrossRef]
5. Jung, J.; Moon, B.; Nah, D.; Kim, J.; Kim, H. Analysis optimum thermal efficiency of the heat recovery ventilators according to the height of the duct. *J. Korean Soc. Agric. Mach.* **2015**, *20*, 147–148.
6. Wu, Z.; Heiseberg, P.; Stoustrup, J. Modeling and control of livestock ventilation systems and indoor environments. In Proceedings of the 26th Air Infiltration and Ventilation Center (AIVC) Conference: Ventilation in Relation to the Energy Performance of Buildings, Brussels, Belgium, 21–23 September 2005; pp. 335–340.
7. Jeong, S. Monthly Korean Chicken. *Korea Chick. Counc.* **2009**, *15*, 82–85.
8. Seo, S. 130,000 Chickens Died in the Heat, and the Ventilation System Was Broken. Available online: http://www.newsway.co.kr/news/view?ud=2014071019485131917(2022.07) (accessed on 20 November 2022).
9. Jeffery, S.R.; Alonso, G.; Franklin, M.J.; Hong, W.; Widom, J. Declarative support for sensor data cleaning. In Proceedings of the 4th International Conference on Pervasive Computing, Dublin, Ireland, 7–10 May 2006; Volume 3968, pp. 83–100.
10. Sharma, A.B.; Golubchik, L.; Govindan, R. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sens. Netw.* **2010**, *6*, 1–39. [CrossRef]
11. Mourad, M.; Bertrand-Krajewski, J.L. A method for automatic validation of long time series of data in urban hydrology. *Water Sci. Technol.* **2002**, *45*, 263–270. [CrossRef]
12. Ranganathan, A.; Al-Muhtadi, J.; Campbell, R.H. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Comput.* **2004**, *3*, 62–70. [CrossRef]
13. Elnahrawy, E.; Nath, B. Cleaning and querying noisy sensors. In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, USA, 19 September 2003; pp. 78–87.
14. Park, J.; Hong, J.; Kim, W. Study on interworking of intelligent IoT semantic information using IoT-lite ontology. *J. Inf. Technol.* **2017**, *16*, 111–127.
15. Hong, I. A study on functional test of damage detection sheet using RS-485 embedded board. *J. Knowl. Inf. Technol. Syst.* **2018**, *13*, 211–220.
16. Myung, S.; Kim, S. The design of Open IoT Platform based on one M2M Standard Protocol. *J. Korea Inst. Inf. Commun. Eng.* **2017**, *21*, 1943–1949.
17. Tomicic, I.; Grd, P.; Bernik, A. Smart internet of things modular micro grow room architecture. *Interdiscip. Descr. Complex Syst. INDECS* **2022**, *20*, 469–482. [CrossRef]
18. Vigneswari, T.; Vijaya, N. Smart livestock management using cloud IoT. *Cloud IoT Syst. Smart Agric. Eng.* **2022**, *1*, 55–74.
19. Lambert, J.; Monahan, R.; Casey, K. Accidental choices—How JVM choice and associated build tools affect interpreter performance. *Computers* **2022**, *11*, 96. [CrossRef]
20. Bawankule, K.L.; Dewang, R.K.; Singh, A.K. A classification framework for straggler mitigation and management in a heterogeneous Hadoop cluster: A state-of-art survey. *J. King Saud Univ.—Comput. Inf. Sci.* **2022**, *9*, 7621–7644. [CrossRef]
21. Dean, J.; Ghemawat, S. Mapreduce: Simplified data processing on large clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
22. Chen, C.T.; Hung, L.J.; Hsieh, S.Y.; Buyya, R.; Zomaya, A.Y. Heterogeneous job allocation scheduler for Hadoop MapReduce using dynamic grouping integrated neighboring search. *IEEE Trans. Cloud Comput.* **2017**, *8*, 193–206. [CrossRef]
23. Michael, A.; Reynold, S.X.; Cheng, L.; Huai, Y.; Liu, D.; Bradley, J.K.; Meng, X.; Kaftan, T.; Franklin, M.J.; Ghodsi, A.; et al. Spark SQL: Relational data processing in Spark. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Australia, 31 May–4 June 2015; pp. 1383–1394.
24. Vora, M.N. Hadoop-HBase for large-scale data. In Proceedings of the 2011 International Conference on Computer Science and Network Technology, Harbin, China, 24–26 December 2011; pp. 601–605.
25. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, *284*, 34–43. [CrossRef]
26. Lee, J.; Lee, H. Comparison of deep learning models using protein sequence data. *KIPS Trans. Softw. Data Eng.* **2022**, *11*, 245–254.
27. Ehsan, A.; Abuhaliqa, M.E.; Catal, C.; Mishra, D. RESTful API testing methodologies: Rationale, challenges, and solution directions. *Appl. Sci.* **2022**, *12*, 4369. [CrossRef]
28. Lee, W.; Kim, S.; Yu, Y.; Koo, B. Development of graph based deep learning methods for enhancing the semantic integrity of spaces in BIM models. *Korean J. Constr. Eng. Manag.* **2022**, *23*, 45–55.
29. Barznji, K.A. Big data processing frameworks for handling huge data efficiencies and challenges: A survey. *Int. J. Data Sci. Anal.* **2022**, *2*, 1–9.
30. Park, D. A study on semantic technology in the internet of things. *J. Inst. Electron. Inf. Eng.* **2015**, *42*, 25–32.

31. Janowicz, K.; Compton, M. The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In Proceedings of the 3rd International Workshop on Semantic Sensor Networks, Shanghai, China, 7 November 2010; Volume 668, pp. 1–15.
32. Kwon, S.; Lee, J.; Kim, S.; Lee, S.; Shin, Y.; Doh, Y.; Heo, T. Design of big semantic system for factory energy management in IoE environments. *J. Korean Soc. Inf. Process.* **2022**, *29*, 37–39.
33. Lee, M.; Lee, E.; Rho, J. Cataloging trends after LRM and its acceptance in KORMARC bibliographic format. *Korean Biblia Soc. Libr. Inf. Sci.* **2022**, *33*, 25–45.
34. Lembo, D.; Stantarelli, V.; Savo, D.F.; Giacomo, G.D. Graphol: A graphical language for ontology modeling equivalent to OWL 2. *Future Internet* **2022**, *14*, 78. [CrossRef]
35. Gwon, G. The influence and characteristics of fine dust on livestock in Korea. *Mag. Korean Soc. Agric. Eng.* **2020**, *62*, 15–23.
36. Kim, J. In the Hot Summer, the Temperature and Humidity of the Barn Must Be Lowered. Enrichment Oil Communication. 2016. Available online: http://www.amnews.co.kr/news/articleView.html?idxno=18865(2022.07) (accessed on 20 November 2022).
37. Abc Ei-Hack, M.E.; Alagawany, M. Managerial and nutritional trends to mitigate heat stress risks in poultry farms. *Sustain. Agric. Environ. Egypt Part II* **2018**, *77*, 325–338.
38. Ahaotu, E.O.; De los Ríos, P.; Ibe, L.C.; Singh, R.R. Climate change in poultry production system—A review. *Int. J. Sustain. Dev. Afr.* **2019**, *10*, 362–370.