

A Virtual Assistant Design and Application on Industrial Database*

Muhammed ÇINAKLI¹ 

Melike DEMİRDAĞ² 

Merve ARTA³ 

Ahmet Çağdaş SEÇKİN⁴ 

DOI:10.33461/uybisbbd.952310

Abstract

Article Info

Paper Type:
Research Paper

Received:
14.06.2021

Accepted:
02.09.2021

©2021 UYBISBBD
All rights reserved.



The development and advancement of technology allows people to access information in a variety of ways. Whether in commerce, marketing or communication, the amount and variety of data is increasing in every field. Collecting this data in different fields and drawing conclusions based on the data has become very important for the development of individuals, companies, industries, and countries, and especially collecting this data in databases has become popular. In this paper, an industrial database was created, a wireless sensor network was set up using ZigBee protocol for data collection, the collection of data in the database was done quickly and cheaply, and a natural language processing based Virtual Assistant was created to query the data. Unlike other traditional rule-based methods, Virtual Assistant has a non-predetermined structure that allows Virtual Assistant to answer different types of questions from the user. The communication between the created Sensor Network, database and Virtual Assistant is done using XMPP message protocol. Again, with the help of this protocol, a system is presented to communicate with the user. With the method presented in the study, the data in the relational database was made queryable regardless of location and platform. With the virtual help system provided, the distance between natural language and computer language is shortened and people can get information quickly and easily.

Keywords: virtual assistant, natural language processing, database, ZigBee, XMPP

Atıf/ to Cite (APA): Çınaklı, M., Demirdağ, M., Arta, M. ve Seçkin, A.Ç. (2021). A Virtual Assistant Design and Application on Industrial Database. International Journal of Management Information Systems and Computer Science, 5(2), 122-143

* This study was produced within the scope of the graduation project supported by TUBITAK (2209).

¹ Aydın Adnan Menderes University, Engineering Faculty, Computer Engineering Department, muhmmndenkl@gmail.com,

² Aydın Adnan Menderes University, Engineering Faculty, Computer Engineering Department, melikedemirdag@protonmail.com,

³ Aydın Adnan Menderes University, Engineering Faculty, Computer Engineering Department, merve.arta@gmail.com,

⁴ Doç. Dr. Aydın Adnan Menderes University, Engineering Faculty, Computer Engineering Department, seckin.ac@gmail.com

1. INTRODUCTION

The Wireless Sensor Networks (WSN) is a special type of network that hosts a large number of nodes with processing unit, wireless communication, sensors, actuators and power supplies (Akyildiz et al. 2002; Yick, Mukherjee, and Ghosal 2008). WSN devices have limited processing power, low data transmission and independent power supply. In WSNs, the general aim is to send the information received from many end nodes to a coordinator node. WSNs are used in military, environmental, health and home applications (Akyildiz et al. 2002). In the last two decades, the desire to connect everything to the Internet emerged due to the large addressing capacity achieved by the widespread use of IPv6 instead of IPv4 (Ziegler et al. 2015). Nowadays, the technologies used to connect physical objects to each other and to larger networks are referred to as the Internet of Things (IoT) (Atzori, Iera, and Morabito 2010; Gubbi et al. 2013). The main applications of IoT are home automation, smart cities, industrial monitoring systems, health systems, and security (Bandyopadhyay and Sen 2011). According to the reports prepared by CISCO, it is predicted that by 2022, 1 trillion sensors will be connected to the Internet using IoT (Anon n.d.).

If we roughly consider WSN technology and IoT technologies, we can conclude that both systems consist of sensors or WSN is the ancestor of IoT (Group et al. 2015; Mainetti, Patrono, and Vilei 2011). However, unlike IoT, WSN technologies are not directly connected to the Internet and generally the individual processing power of the WSN nodes are less than IoT devices. IoT systems operate with fewer devices and smaller areas than WSN systems. Usually, WSN applications collect sensor data, while most IoT systems only execute switching commands. In these applications, the common communication technology is Wi-Fi and the hardware platform is generally Raspberry-Pi or NodeMCU. Although these devices are very useful, they require many access points and infrastructure to spread over large areas such as industry, forest, state borders (Jeschke et al. 2017; Sadeghi, Wachsmann, and Waidner 2015). In these applications, the number of sensor nodes can reach hundreds or even thousands and spread over kilometers squares. Each wireless communication protocol has characteristic properties such as number of network devices, bandwidth, range, network topology, routing protocols, etc. (Baker 2005; Lee, Su, and Shen 2007). ZigBee protocol is the most suitable wireless networking technology when it comes to the study of various studies in the problem of networking with large area coverage and large number of nodes.

There are three types of devices involved in ZigBee network, namely coordinator, router and end device. Generally, we use star topology with Wi-Fi based IoT devices while in ZigBee star, tree and mesh network topologies are used. In mesh topology, each network device is connected to each other through router devices. This allows nodes to propagate over very large areas. This allows one node to access another through a hierarchical routing system that cannot be reached directly through its own transmission power. For security, authentication is required to join a ZigBee network. Authentication and association are managed by the coordinator node. The hibernation algorithm defined by ZigBee allows the end nodes to save energy. The end node does not communicate in hibernation mode. In this method, the communication module is switched to a power-saving mode with event-driven wake-up or periodic wake-up (Farahani 2011). Therefore, it can be used in wildfire monitoring (Yu, Wang, and Meng 2005), habitat monitoring (Mainwaring et al. 2002), animal tracking (Kiani 2018), and border security (Sun et al. 2011).

However, WSN protocols such as Bluetooth, ZigBee, MiWi, etc. are not designed to connect directly to the Internet. The main challenges in connecting each node to the Internet are the integration of the large IP header, global addressing scheme, limited bandwidth, node hibernation, and transport protocol (Rodrigues and Neves 2010). To overcome this problem, several solutions have been proposed to integrate ZigBee with the Internet (Khalil et al. 2014; Yazar and Dunkels 2009; Zhu et al. 2010). All of these methods can be outlined as a combination of a ZigBee interface, a message/data translator, an Internet interface, and a service provider. The basic process can be named as WSN gateway design. It is an intermediate device that can communicate with both the internet and the WSN.

IoT devices are divided into three as edge, fog and cloud in terms of calculation methods. Edge Computing is systems that perform computation with persistent data near the data source in the IoT system. In such systems, the amount of data that goes out of the region where the system is installed is made concise. In this way, both the external data is more secure, and the costs such as internet traffic and data storage can be reduced. In fog computing-based systems, the computational load occurs both locally and on remote devices. In Fog systems, after some internal data is extracted locally, processes that require higher processing power are performed on remote devices. In cloud computing IoT systems, the operations to be performed on all data collected locally are calculated on remote devices where the IoT system is installed.

The service providers used in IoT systems are usually a cloud based. CoAP(Constrained Application Protocol), MQTT(Message Queuing Telemetry Transport), AMQP (Advanced Message Queuing Protocol), XMPP (Extensible Messaging and Presence Protocol), etc. are application-level protocols to create service provider systems for end users (Ali et al. 2017; Mohamudally and Peermamode-Mohaboob 2018). CoAP is an application protocol that works using UDP on devices with limited resources (Shelby, Hartke, and Bormann 2014). MQTT is a lightweight protocol that supports TCP connections. MQTT works with multiple clients over a server (Banks and Gupta 2014). AMQP is used for message queuing of messaging applications. XMPP is a public protocol based on XML that enables instant messaging (Saint-Andre 2004). When using this protocol, each client has its own name and communicates with another client through the appropriate server that is authorized to forward. Such application layer applications offer advantages such as security, reliability, quality of service, flexibility, and standardization. In this case, especially due to the large number of users and nodes, problems such as traffic and packet size are not compatible with ZigBee(Ali et al. 2017; Thangavel et al. 2014; Vermesan and Friess 2014). In short, our main problem is that ZigBee is not designed to connect directly to the Internet and humans need to transmit commands and receive information through ZigBee nodes without being compromised by ZigBee over the Internet.

In the study, unlike the traditional methods, an attempt is made to create a more powerful gateway by setting up a database on the gateway and adding a Virtual Assistant. The concept of a Virtual Assistant (VA) is software that communicates with humans via text or speech, just like a real assistant. It understands sentences and gives a response corresponding to the user using Natural Language Processing (NLP). It is also known as chatbots and is used in business, education, commerce, health, and entertainment (Bansal and Khan 2018). Adalı, studied the structure and rules of language, which occupies an important place in the history of mankind, and presented the basic issues in the framework of natural language processing, applications of natural language processing and properties of languages (Adalı 2016). The main advantages of VA are its ease of use and the fact that no expertise is required. It can quickly handle daily or specific tasks.

In VAs developed using a pattern matching approach, patterns or rules for the intended use are first created and then natural language sentences sent by the user are analyzed and an attempt is made to match these patterns. Based on the result of matching, the decision mechanism generates a response and this response is sent to the user as a reply. VAs using pattern matching select the response to be given to the user from the defined responses before replying to the user using the pattern matching algorithm. In rule-based systems, no new answers are created. All answers to be used are created by the developer in the form of patterns. Pattern matching algorithms can repeat as they proceed with human responses and responses ready for authenticity. For a VA to fully communicate with humans, thousands, perhaps millions, of rules must be written. Creating a complete VA is a very complicated process and introduces many errors. Instead of using a pattern matching algorithm, VAs that use machine learning use the structure of Natural Language Processing (NLP). NLP is a field of artificial intelligence that studies how language used in communication by computer systems is interpreted and controlled. Natural language processing is nowadays popularly used for topics such as social media, sentiment or document analysis (Başarslan and Kayaalp 2021; TOÇOĞLU 2020). However, natural language processing can be used for processes such as analysis and recognition as well as synthesis. This structure is not only about answering, but also about addressing the communication environment.

NLP systems do not require a predefined set of rules. However, it is a structure that requires an extensive training set. The NLP approach to VA is an in-depth analysis that examines and breaks down the language used in communication in detail. Most NLP approaches are based on Machine Learning (ML).

The main problem is that the rule-based VA, which are also used in this project, match the user's sentences with a certain rule pattern and give a corresponding predefined answer. So, these VAs do not go beyond the conversation patterns written in the system. The rule-based VAs has knowledge base, which contain a list of written responses that match the user's input. To deal with this problem, we extended the list of written answers indicating possible synonyms in VA knowledge base. To facilitate the interaction of the created virtual assistant with IOT devices, we also created VA gateway. Wireless Sensor Network (WSN) plays an important role in creating a Virtual Assistant Gateway.

There are various studies on virtual assistants. It is seen that academic studies on virtual assistant focus especially on data security, speed, and transmission quality problems. Toğay et al. in their study, they carried out several security and encryption processes related to the Internet of Things security system (Toğay et al. 2019). It is mentioned about the encryption system of short messages thanks to an improved chip This system provides benefits in terms of data security, but within the project we created, the security process is provided by communication protocols. Erdogan et al. Message Queuing Telemetry Transport Protocol (MQTT) used protocol (Erdoğan, Küçük, and Khan 2020). The designed gateway has been programmed with an algorithm that can detect the difference on the data. End-to-end delay times and statistical results of data messages with loads of different sizes at different levels of service quality (Quality of Service, QoS) are given. In this project, data speed and quality have been developed with the foreground, but in the newly created project, data speed and quality are provided by WSN systems. Taştan has implemented real-time data monitoring and control using an IOS / Android interface developer (Taştan, 2019). Thanks to the Android-based remote monitoring and control interface, it has been able to control the heating system using the temperature and humidity data of a smart home. In this project, data tracking and quality are at the forefront. However, in the newly created project, up-to-date and instant communication with the database can be established without the need for data tracking. Dener has made a location-based and sensor-based control and tracking system for users (Dener, 2019). A secure communication was provided between the sensor nodes and the home gateway with the help of the AES encryption algorithm. The study has provided data security, but the newly created project is also supported by these communication protocols. The user-friendliness and convenience of the solutions found were evaluated as the current shortcomings. While the benefits of the systems to the user cannot go beyond the security framework, this project is user-friendly and has been structured in a way to provide the user with the highest convenience and benefit. Recent studies on the Internet of Things and virtual assistants are compared in terms of Aim, IoT Support, IoT Level, VA sup Support, NLP Support, and Query Support presented in a summary in Table 1.

Reference	Aim	IOT Support	IOT Level	VA Support	NLP Support	Query Support
(Erdoğan et al. 2020)	An industrial application has been made for Iot. In the application, monitoring and management processes are carried out over the cloud with MQTT and IBM Watson.	yes	cloud	no	no	no
(Toğay et al. 2019)	A system has been developed that provides	yes	edge	no	no	no

	authentication and encrypted communication between the gateway and the broker.					
(Taştan, 2019)	A smart home application has been made for Iot, it is known that management and control operations can be performed over the cloud in the application.	yes	Fog	no	no	no
(Dener, 2019)	A smart home gateway application has been made for Iot, emergency messages from sensors are transmitted to the user via the gateway over the internet.	yes	edge	no	no	no
(Parthornratt et al. 2018)	Cloud-based home automation application	yes	cloud	yes	Rule based	no
(Roca et al. 2020)	Patient follow-up application in the field of health	no	cloud	yes	yes	no
(Al-Rasyid et al. 2020)	Cloud-based AquaCulture monitoring and management system	yes	cloud	yes	Rule based	no

Table 1: Recent Studies Comparison

In this paper, the project was implemented in a distributed system to achieve usability in many different functions. The advanced system in which we improve the tasks of a fully distributed system provides the simultaneous return of requests from different clients. It also performs the process of adding the data from the sensors to the database through WSN. It also aims to provide an alternative system that can be used with ZigBee for IoT systems. For the WSN which consists of ZigBee nodes, the interface device is connected to a messaging system such as VA. The system has been tested as a design and application of a virtual assistant on an industrial database. The data received from different types of sensor nodes were sent to the VA and the answers to the corresponding questions were provided to the user. In this way, unlike previous studies, the aim is to contribute to the literature with a cheaper, simpler and customizable application. In the paper, the system architecture, materials and methodology, results and discussion are presented and finalized with a conclusion section.

3. MATERIAL AND METHOD

The general structure of the system is shown in Figure-1 and consists of a WSN, a VA -WSN gateway, a SQL server, an XMPP server, and an end-user device. The WSN network consists of 4 network nodes. The gateway device is the device that acts as an interpreter between the WSN and the Internet. The SQL server is the device that stores the data of the WSN network. The XMPP server is a messaging system and provides messaging between the gateway and the end user. The end-user device is a computer, smartphone, or tablet running an XMPP messaging program. An XMPP server system in the system architecture performs all instant messaging. As shown in Figure 1, the gateway

device (s) defined in WSN (s) and user devices are connected to the XMPP server via the Internet connection. In the system design, devices which are coordinator or router from WSN devices can be operated as a gateway. With this feature of XMPP, it is also possible to use multiple gateways in the same WSN to provide blockchain-like collective data security that can be added in future studies. In this section, firstly XMPP and ZigBee technologies are explained and then gateway design is presented and the algorithm to be used is explained Also Virtual Assistant Method is generally based on Natural Language Processing. When a Natural Language Query is given to the system, the system uses the query and other information from the database such as database name and columns and makes a SQL Query from all this information given to it. First, the given Natural Language Query is taken from the user via the XMPP Client. The other parameters such as database path are taken and are loaded ready into the variables so that it would ease rendering of the system. Also, stop word lists are loaded. The Natural Language Query is translated to desired language. The input sentence is tokenized, meaning all words of the sentence are converted into a list format. The unwanted characters such as comma, quotes, etc. are removed from the input sentence. Along with removing unwanted characters, the stop words are filtered from the input sentence given by the user which leaves us with only important and useful words to help us construct the query. A list of keywords is also loaded which would signify/intend to mean any syntactic function of SQL in Natural Language, so it is easier for us to detect such keywords in the given input sentence. Once the filtered input sentence is received, it is compared with the keywords list that is loaded and such keywords are used to find out the kind of operation that needs to be performed. Then keywords such as table name and column names are also simultaneously found out. Since all these keywords are jotted and where to use them is assigned, a SQL query is created using this, which contains all the conditions. Generated SQL query is executed as a SQL script. The answer to the query is taken. Then the answer is translated or reformatted to desired language. Finally, the answer is sent to user via the XMPP Client. In the following subsections, the WSN architecture and configuration, VA -WSN Gateway Design, Database Design and the messaging system using XMPP are introduced respectively.

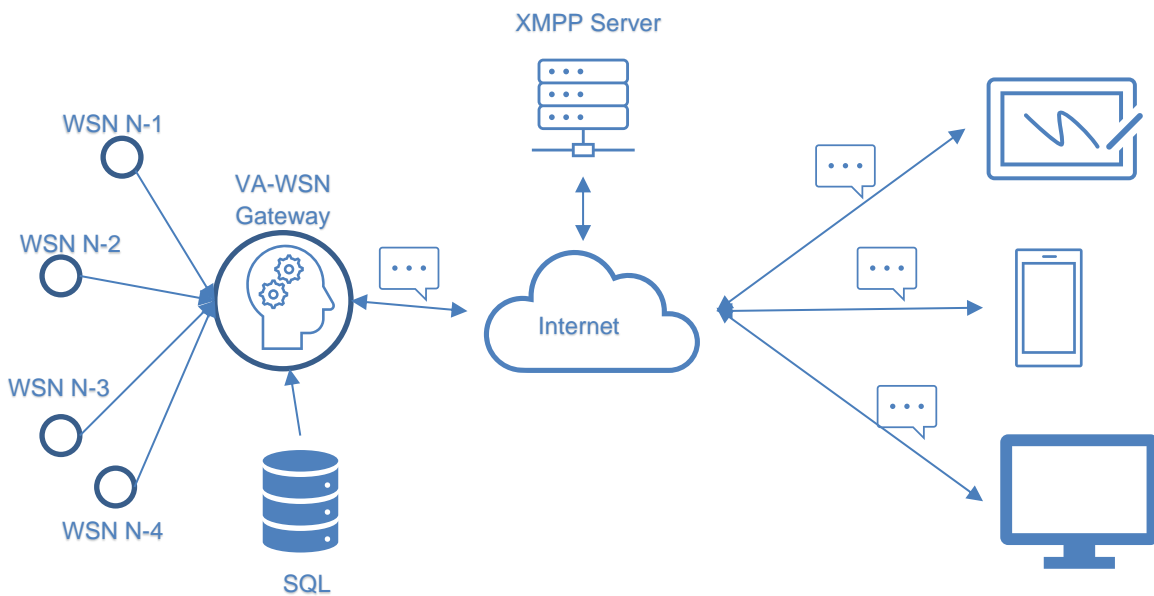


Figure 1: General System Design

3.1. WSN Configuration

While configuring the WSN, the hardware connections of the sensors with the Xbee modules are made, and the software of the sensors is loaded and made ready. To configure the WSN, it is necessary to make sure that the Xbee modules communicate with each other, so a USB cable is connected to the computer using the Xbee adapter. Then the XCTU program is downloaded and

installed. The XCTU program is used to enter the Xbee modules into the interface and make the necessary configuration settings. When making the configuration settings, the Xbee modules to be configured are assigned the specified roles. The assigned roles are as follows: Coordinator, Router, and Edge devices. The coordinator's functions are as follows: It selects a channel and PAN ID (both 64-bit and 16-bit) to start the network, can allow routers and edge devices to join the network, can also help in routing data, and must be powered by the network. The characteristics of the router are as follows; It must join a ZigBee PAN before sending, receiving, or forwarding data, can allow routers and terminal devices to join the network after joining, and can help in routing data after joining must be powered from the network. The characteristics of the end device are as follows; must join a ZigBee PAN before sending or receiving data, devices cannot be allowed to join the network and must always send and receive RF data through its parent, data cannot be routed, can go into power saving mode to save power and run-on battery. When starting configuration settings, Xbee S2 module and adapter are connected. Clicking the Next button will start the device search and if the device is found, clicking the Add Selected Devices button will add the device. This is done for all Xbee modules. The settings of the added Xbee modules are set as Coordinator and the roles Router for each sensor and End Device for the end devices, and after the settings are made, the changes made to the Xbee modules are saved by clicking the Write button. The changes made are as follows:

1. Any PAN ID is entered and the entered PAN ID must be the same for Coordinator, Router, and End Devices.
2. For the coordinator, these parameters are set: DH = 0, DL=FFFF, NI = COORD.
3. For the router these parameters are set JV = ENABLED [1], CE = DISAPLED [0], DH =0, DL = 0, NI= ROUTER.
4. For end devices, only NI = END_DEVICE is determined in the configuration for router devices. In addition, the SH value of the connecting router device is written to the DH parameter, the router SL value is written to the DL parameter.

There are 3 types of ZigBee topology; Star topology: such networks have a coordinator and end devices that communicate directly with each other, and if the coordinator fails, the entire network fails. Mesh topology: It includes coordinators, routers, and end devices with connections as shown in the Figure 2. Tree topology: this is a combination of star and network topology. The ZigBee topology used in this study is the mesh topology.

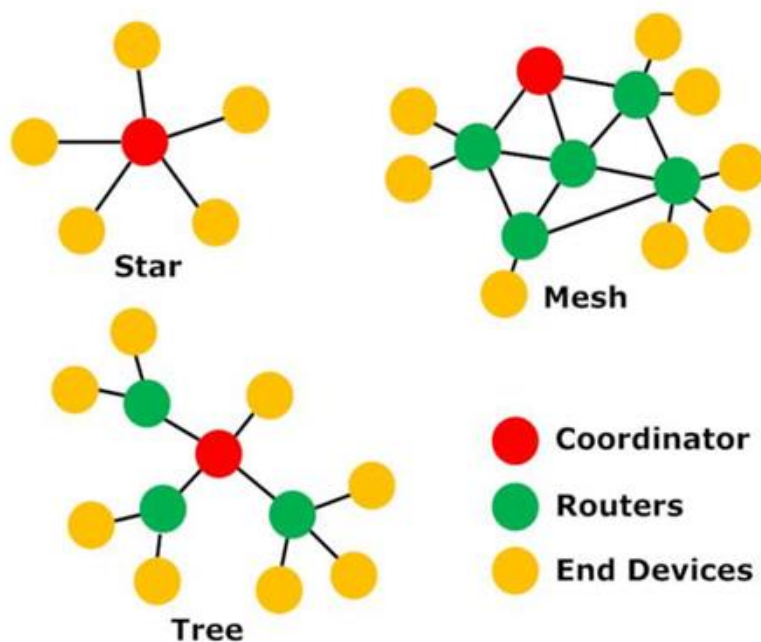


Figure 2: ZigBee topologies

After the configuration, The Coordinator Xbee module is connected to the Raspberry Pi using a USB cable, and the system is made operational.

3.2. Virtual Assistant Gateway Design

The Gateway device performs the translation of the command, query, warning and information messages between the Internet and the WSN. The message flow in the gateway is shown in Figure 3. Messages received over the Internet reach the gateway via XMPP. Messages from the client are accepted as two types of command or query message. Each command and query sent by the client is recorded in the database.

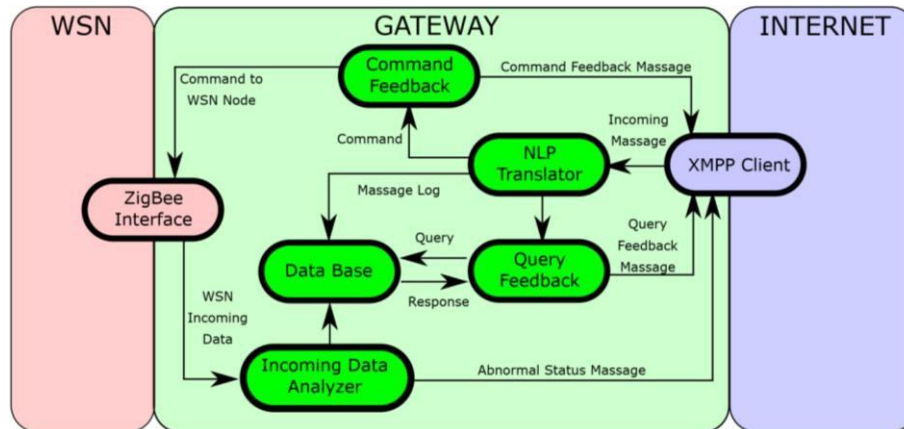


Figure 3: Gateway Design

3.3. Database Design

The main purpose of the prepared project starts with the user entering a query into the system using natural language. At this stage, the targeted situation in the database has been created with the aim of enabling all kinds of users, mostly distant or inexperienced people to access information easily. The data collection process of this project takes shape according to the work area and demands of the user. Before coming to the database design stage, firstly determining the needs of the user related to the project is valuable for the first step and is a step that will provide convenience for other steps. What data will be kept in the database, which data will be stored, the tables are prepared according to the user's demands and needs, however, the content in these tables generally complement the first step of the project's operation. In the database creation phase of the project, MySQL was used as a large database system that can meet many needs. The database created after the SQL query is obtained from natural language expressions is mapped to the table name in the database that is in the user's query and related to the information to be learned. Thus, the first relationship between natural language and database is established.

When the database was created, two databases were created. One of the databases was created for industrial data. In creating the industrial database, a standard industrial environment was considered. Moreover, the Industrial Database is shown in Figure 5. The other for data from the sensor network. A gas sensor table, a humidity sensor table, a motion sensor table, a temperature sensor table, a door sensor table, and a distance sensor table were created for the 6 sensors needed for the sensor network. In the industrial environment, environmental variables such as temperature, humidity, gas level and security were determined. Since these sensors are considered necessary in our project, the basic level is used to determine which situations are needed and the sensor network is built in this way. We have created a node table to which our sensors are connected. This table contains the information of all the sensors in conjunction with other tables. Also, the states of the environment variables store their information in tables related to the node table. The command state and the information it contains are created to be associated with the user table along with the command table. The database created for the sensors is shown in Figure 6. Also, the command tables are shown in Tables 2 and 3.

The commands to the sensors have a fixed structure and can only be sent by users with administrator privileges. The commands designed for the application are in the structure shown in Figure 4. The structure of a basic command sent by the client contains the target node ID, command and command parameter. After receiving the command and parse operation, a feedback message is generated and sent to the client according to the meaning of the command. The feedback message is used to understand that the received message has been parsed correctly and is adapted to the natural language as follows: “You sent the CMD command and PAR parameter to the node named NID”. If "yes" or "#" is sent after this message, the message is confirmed to be correct and subsequent operations are performed. Other details of the sensor command structure are shown in Table 2 and Table 3.

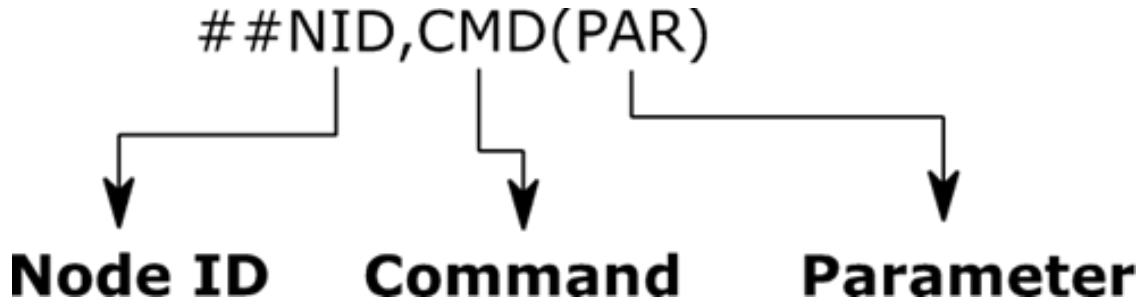


Figure 4: Sensor Command Structure

Node ID	Node Explanation
DL01	Door Node
TH01	Temperature and Humidity Node
SG01	Smoke and Gas Node
MD01	Motion Detector Node

Table 2: Node Structure

Command	Parameter Type	Explanation
SL	Integer	Set smoke level
DS	Boolean	Change door lock status. To open parameter equals 1 and to close parameter equals 0
SP	integer	Set period as minutes for all periodical values.

Table 3: Command Type

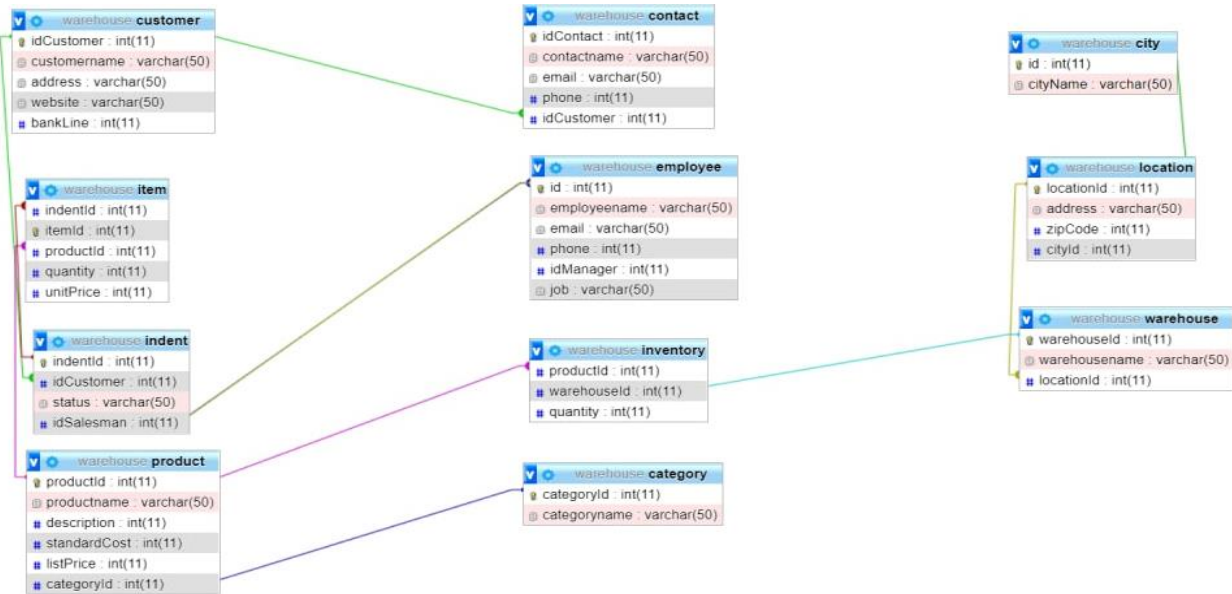


Figure 5: Database for Industrial

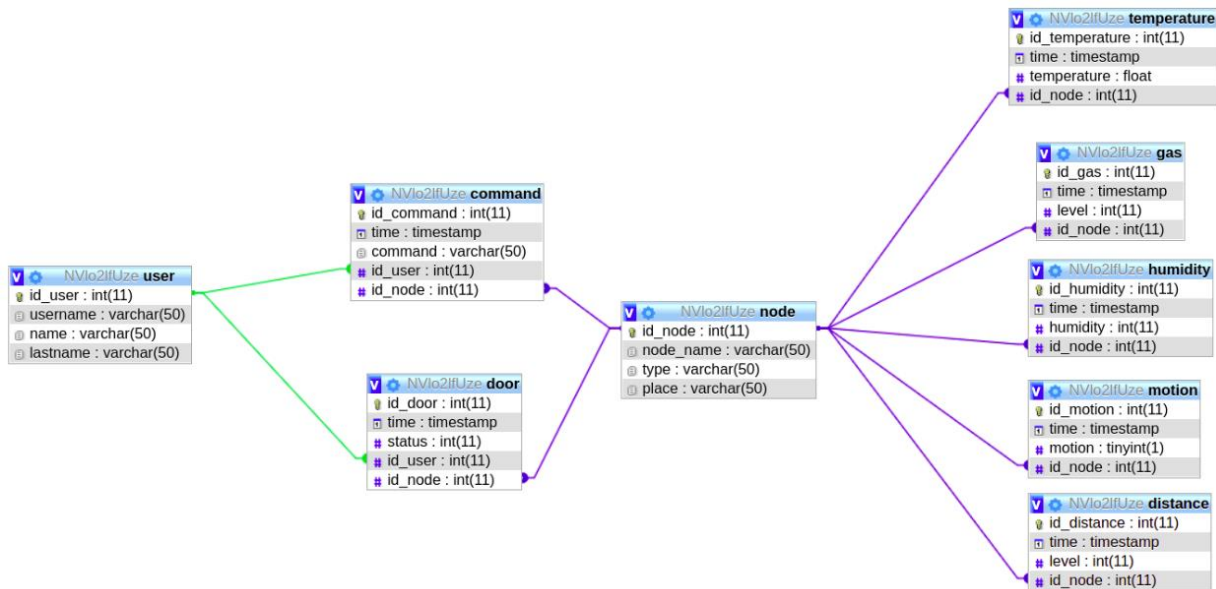
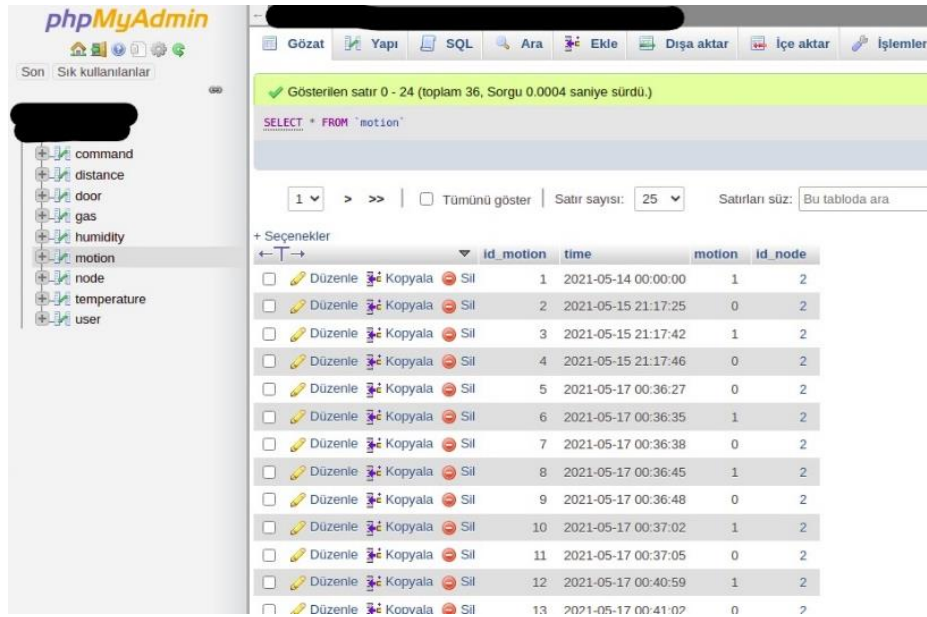


Figure 6: Database for Sensor

In these operations, the Python Programming language was used, and an XMPP client was created to transmit abnormal system messages. Incoming data was received over the serial port using Serial Library in Python. Mysql library was used to save the retrieved data in the database. The data from the sensors is filtered by the XMPP client. Since the incoming data is in a certain format, the data coming from the nodes can be easily filtered, easily distinguishing which node and which sensor in the node it is coming from, and can be stored in the database according to the specified conditions. Figures 7 and 8 show the retrieval and storage of data in the database.

```
Run: main
main.py
INSERT INTO temperature (time,temperature,id_node) VALUES ('2021-06-20 23:09:36', '26.70', '1');
INSERT INTO humidity (time,humidity,id_node) VALUES ('2021-06-20 23:09:36', '77.00', '1');
INSERT INTO temperature (time,temperature,id_node) VALUES ('2021-06-20 23:09:54', '25.80', '1');
INSERT INTO humidity (time,humidity,id_node) VALUES ('2021-06-20 23:09:54', '80.00', '1');
INSERT INTO temperature (time,temperature,id_node) VALUES ('2021-06-20 23:10:04', '25.80', '1');
INSERT INTO humidity (time,humidity,id_node) VALUES ('2021-06-20 23:10:04', '80.00', '1');
INSERT INTO temperature (time,temperature,id_node) VALUES ('2021-06-20 23:10:14', '25.80', '1');
```

Figure 7: Data Retrieval



	id_motion	time	motion	id_node
<input type="checkbox"/>	1	2021-05-14 00:00:00	1	2
<input type="checkbox"/>	2	2021-05-15 21:17:25	0	2
<input type="checkbox"/>	3	2021-05-15 21:17:42	1	2
<input type="checkbox"/>	4	2021-05-15 21:17:46	0	2
<input type="checkbox"/>	5	2021-05-17 00:36:27	0	2
<input type="checkbox"/>	6	2021-05-17 00:36:35	1	2
<input type="checkbox"/>	7	2021-05-17 00:36:38	0	2
<input type="checkbox"/>	8	2021-05-17 00:36:45	1	2
<input type="checkbox"/>	9	2021-05-17 00:36:48	0	2
<input type="checkbox"/>	10	2021-05-17 00:37:02	1	2
<input type="checkbox"/>	11	2021-05-17 00:37:05	0	2
<input type="checkbox"/>	12	2021-05-17 00:40:59	1	2
<input type="checkbox"/>	13	2021-05-17 00:41:02	0	2

Figure 8: Data Storage

3.4. Natural Language to SQL Query Translation Process

Firstly, a natural language query is received from the user via Virtual Assistant. The received query is translated into English with Translate API. In this translation phase, typos in the query entered by the user are automatically corrected because the translation application is used. After the query completes the translation phase, it begins to be parsed to be converted to SQL form. In2sql library is used for parsing (Couderc and Ferrero, 2015). In this parsing phase firstly, all words of the query sentence are converted into a list format. Then unwanted items removed from the list using the stop word list that has loaded. The mentioned unwanted items are commas, quotation marks, etc. The items that mean SQL command in the given natural language sentence are determined by using the keywords list. These determined words will be used in SQL query creating. These keywords are used to find words that indicate the table name or column name in the NL clause. When converting these queries to SQL, database and column names must match those in the query sentence (Bhadgale, Gavas, and Goyal, 2013). It is important to choose appropriate column names during database preparation. Possible table or column names to be entered in the NL query should be specified in the system to ensure matching. SQL query is generated after specifying command word and table name or column name. After these steps, the SQL query created is queried on the database associated with the system. The query result from the database is sent back to the user via Virtual Assistant. For security reasons, in this method, keywords such as UPDATE and INSERT, are isolated on the gateway. In this way, client messages can only be used to get information about WSN, in no way new

data entry, modification or deletion is possible. This detail ensures system security. The overall system design is shown in figure 9.

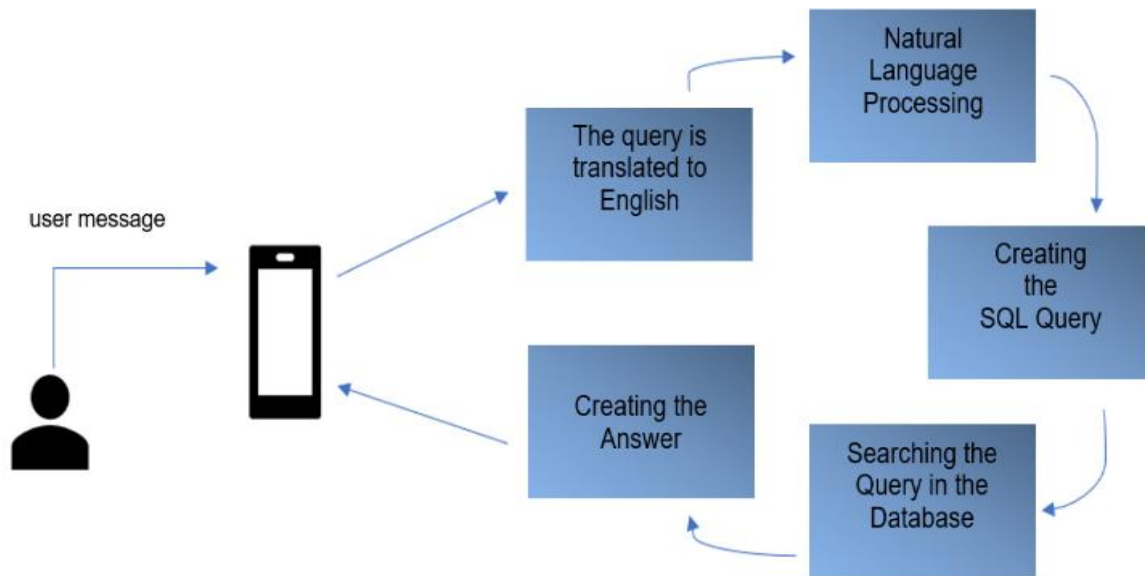


Figure 9: Overview of System

3.5. Messaging Protocol Implementation

The XMPP protocol entered our developing technology in 1998 under the name Jabber. It is an XML-based structure that supports voice-video transmission with multiple multi-chats while supporting messaging state (Saint-Andre 2004; Saint-Andre et al. 2009). Supports library software for server-client software in many different programming languages. Supported programming languages are neither C++, Java, Python etc. examples can be given. The easy accessibility provided by XMPP during the project development phase is extremely efficient in the initial development phase in terms of control and regulation thanks to both timesaving and simplicity in many chained situations to achieve the goals. Another reason why XMPP is preferred is that it has been developed for more than 20 years and it has proven itself to users in terms of reliability. The architectural structure of the XMPP network is another step that is familiar to the user, simplifying the work, and the architecture is very similar to the e-mail communication network we use today, and the user can easily operate the XMPP server. One of the technological problems while operating the server is that the communication network is secure. It is an indisputable fact that the security situation is extremely valuable during the information exchange situation, where we communicate with the server at every stage. Demonstrates XMPP users that it can benefit security through authentication layer and transport layer security protocols. Communication is one of the milestones of information exchange at every stage of the project. For the management part of messaging to send the correct answer to the user, the usability benefit of XMPP is that the server provides support for its own users to manage the network, for content and messaging, and to remotely control and monitor the system status.

The XMPP communication steps require that different servers connect and know each other, and that many clients can communicate with each other even though they are on different servers. Clients have addresses in the form "username@servername.com". The idea of using an off-the-shelf server in the initial editing and development process of the project supports development in other branches of the project and increases usability in the targeted steps in a simplified form for the initial development phase. It allows transfer of XML data blocks from XMPP. When you want to start a session with the XMPP server, a TCP connection is opened. After approval of server a data stream is sent to the client. When the client sends something to the XMPP server, the client XML file is processed to the server side. When the data crosses the network connection and reaches the client, it

is written to the client stream file. In the client stream file, the client requests the contact list for target "username@servername.com". The server responds with a contact list. This processing structure is used to create a client for use in the project that works with the natural language processing application. A web-based client or mobile messaging application using any XMPP protocol successfully and securely transmits user-requested queries to the program.

To In order to use the XMPP protocol, the server must first be installed or create an account from the installed XMPP servers. Account creating is done from the previously installed XMPP server providers or the installed server. For example, XMPP accounts are created from the specified providers as "user@xabber.com" and the created XMPP clients are logged in with these accounts. In this article, the XMPP client was created using the PyCharm compiler with the PYTHON programming language. For Python Programming language, SleekXMPP library is used to create the client. The Turkish sentence received from the user is translated into English using translate-api. The translated sentences are translated into query language using NL2SQL library. The translated queries are sent to the database using the pymysql library and the response is received in this way. The following steps were followed while creating the XMPP client;

3.5.1. Creating the EchoBot Class:

There are three main types of entities in XMPP: servers, components, and clients. A client connection is used because our echo bot will respond to a few people for now, rather than remembering thousands of users. A client connection is the type of connection used in standard IM prompts like Pidgin or Psi. SleekXMPP ships with a ClientXMPP class that you can extend to add echoing of messages. Since ClientXMPP requires the jid and password parameters, the EchoBot class can also accept them, as shown in Figure 10.

```
class EchoBot(sleekxmpp.ClientXMPP):  
    def __init__(self, jid, password):  
        super(EchoBot, self).__init__(jid, password)
```

Figure 10: EchoBot class

3.5.2. Processing Session Start:

The XMPP feature requires clients to publish their presence and retrieve their list (friends list) when they connect to the XMPP server and create a session. Until these two tasks are completed, some servers cannot deliver or send messages or status notifications to the client. So, it is necessary to ensure that the list is received and an initial online status is sent after the session is started. This is done by registering an event handler for the session_start event, as in Figure 11.

```
def __init__(self, jid, password):  
    super(EchoBot, self).__init__(jid, password)  
    self.add_event_handler('session_start', self.start)
```

Figure 11: Event Handler

The self.start method must be executed when the session_start event is triggered, the self.start handler must also be defined, as in Figure 12.

```
def start(self, event):  
  
    self.send_presence()  
  
    self.get_roster()
```

Figure 12: Session Start Event

The event handler accepts a single parameter, usually the stanza that triggers the event. In this case, the event is simply an empty dictionary, since there is no associated data. The first task of sending the first entity is done using send_presence. Calling send_presence without any arguments sends the simplest stanza allowed in XMPP: <presence />. The second task is accomplished with get_roster, which sends a IQ continent requesting a list to the server and then waits for the response. The list data is stored by an internal handler in self.roster and, in the case of a clientXMPP instance, in self.client_roster. (The difference between self.roster and self.client_roster is that self.roster supports storing list information for multiple JIDs, which is beneficial for components, while self.client_roster stores list data only for the client's JID.) It is possible to time out. While waiting for the server to respond, the network may be too slow or the server may not respond. In this case an IQTimeout is generated. Similarly, an IQError exception can be raised if the request contains incorrect data or requests the list for the wrong user. In both cases, the get_roster() call can be used. In the roster_retrieval process, the XMPP stanzas are as shown in Figure 13.

```
<iq type="get">  
  <query xmlns="jabber:iq:roster" />  
</iq>  
  
<iq type="result" to="echobot@example.com"  
from="example.com">  
  <query xmlns="jabber:iq:roster">  
    <item jid="friend@example.com" subscription="both" />  
  </query>  
</iq>
```

Figure 13: XMPP Stanzas Result

3.5.3. Reply to Messages:

After an EchoBot instance has processed session_start, it is now ready to receive and reply to messages. A handler is registered for the message event that occurs when a message is received, as shown in Figure 14. For message stanzas, reply() accepts the body parameter, which is then used as the value of the message's element. Setting the appropriate JID is also handled by reply(). Another way to send the reply message is to use send_message, a convenient method for creating and sending a message based on the values passed to it. These methods are used to provide a quick response directly to the user under the conditions we specify for the incoming messages. An example can be seen in Figure 15. Also, the result can be seen in Figure 16. A connection with the XMPP server must be established and the client must be instructed to start executing as well as processing messages.

```
def __init__(self, jid, password):
    super(EchoBot, self).__init__(jid, password)

    self.add_event_handler('session_start', self.start)
    self.add_event_handler('message', self.message)
```

Figure 14: Message Handler Event

```
def message(self, msg):
    if msg['type'] in ('normal', 'chat'):
        self.send_message(mto=msg['from'],
                           mbody='Thanks for sending:\n%s' % msg['body'])
```

Figure 15: Quick Response Example

```
<message to="echobot@example.com"
from="someuser@example.net" type="chat">
  <body>selam!</body>
</message>

<message to="someuser@example.net" type="chat">
  <body>Sanada selam!:</body>
</message>
```

Figure 16: Quick Response Result

3.5.4. Connecting to the Server and Processing:

It needs to connect to the XMPP server and instruct the client to start executing as well as processing messages. After these steps have been performed, some configuration can also be done. For example, if wanted the bot supports service discovery and pings, it can be configured as shown in the Figure 17. After all the necessary processing is done, it is ready to connect to the server and start sending messages. If the dnspython package is installed, the `sleekxmpp.clientxmpp.ClientXMPP()` method performs a DNS query to find the appropriate server to connect to with a given JID. If `dnspython` does not exist, `sleekxmpp` will attempt to connect to the hostname used by the UID unless an address tuple is specified `clientxmpp.ClientXMPP()`. To begin responding to messages, `sleekxmpp` calls `sleekxmpp.basexmpp.BaseXMPP.process()`, which starts the event handling, send queue, and XML reader threads. It also calls `sleekxmpp.plugins.base.base_plugin.post_init()` method for all registered plugins, passing `block=True` to `sleekxmpp.basexmpp.BaseXMPP.process()` and running the main rendering loop on the main execution thread. The call to `sleekxmpp.basexmpp.BaseXMPP.process()` will not return until the connection to `SleekXMPP` is disconnected. If you need to run the client in the background for another program, `block=False` will create the rendering loop on a separate thread. Also, the SSL version can be specified as shown in Figure 18. After the necessary client creation processes have

been performed, the general working state of the system is shown as in Figure 19.

```
if __name__ == '__main__':  
    xmpp = EchoBot('asistan1@xabber.org', '12345')  
    xmpp.connect()  
    xmpp.process(block=True)  
    xmpp.register_plugin('xep_0030')  
    xmpp.register_plugin('xep_0199')
```

Figure 17: XMPP Client Configuration

```
import ssl  
xmpp.ssl_version = ssl.PROTOCOL_SSLv3
```

Figure 18: SSL Version Define

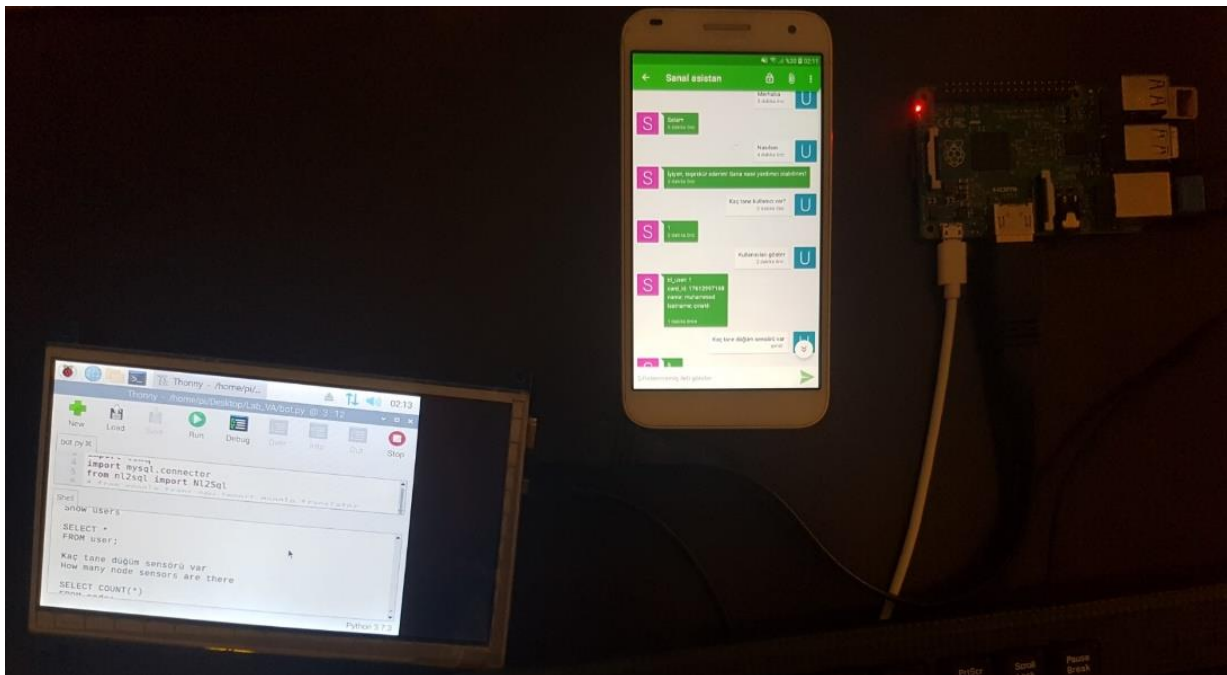


Figure 19: Real System Example

4. CONCLUSION

This study has reduced the communication between everyday language and computer language to a simple level. The user's expressions were directly translated into SQL queries, and natural language access was provided to the database. Thus, as stated in the project aim, users do not need the necessary SQL knowledge to get the data they want. Users can access the searched data with a single sentence as shown Figure 20, 22 and 23. In Figure 20, the list price and standard price of the product whose product name is pant was requested and the result was successfully obtained. In addition, the total list prices of the products were also requested, and the answer was successfully received. As can be seen in Fig. 21, the SQL query is shown again with the virtual assistant dialog example. First, the Turkish sentence is translated into English thanks to the translate api, then the English sentence is translated into SQL. The general information of the product named in Figure 22

and 23 has been requested and the answer has been received successfully. The SQL form of the query in the dialog is shown in Figure 24. Therefore, users can easily access any information registered in the database. When the user interacts with different platforms within the scope of the project, the user can use it, especially on the web or mobile. The virtual assistant can answer most questions quickly and in the specified area. However, the virtual assistant may not be able to answer some desired questions. For instance, if the sentence sent by the user to the virtual assistant is not in a form suitable for the Turkish language structure, it will be translated into English incorrectly. Therefore, the SQL query may not be built or maybe built incorrectly and an incorrect return will be provided, or an incorrect or incomplete translation of some sentences of the translate API will also cause an incorrect SQL query to be built. Some new goals have been set to solve this problem and make the virtual assistant more useful. The first of the targeted working phases of the project is the Turkish translation of the stage of examining the query sentence and parsing it. In the second stage, with the creation of a local server established by the project owners instead of security and communication protocols. In addition, with the virtual assistant, it is aimed to ask questions and get answers via voice. Also, there are a few performance and accuracy improvements/changes that are needed to be addressed within the system. Finally, against possible incomplete or incorrect query situations in the query, like a "Did you mean this ...?" query sentence is targeted to suggest a prediction. The virtual assistant, which will be realized in the direction with the goals, will be able to help users in most industrial areas. Furthermore, the virtual assistant will be able to answer every question asked by the user in the designated area and will provide maximum benefit in the area in which it is used. This article aims to realize the communication between a user and any industrial database in a simple and accurate way. Besides, the goals were achieved in line with the methodology we explained. The main goals we have achieved are as follows; The virtual assistant can access all the information in the database and show them to the user. The user can communicate with the virtual assistant from a mobile phone or computer. In addition, thanks to the created WSN, a fast, low-energy, and stable system exists together. In this study, the user can access the desired information simultaneously from the platform he/she wants, whenever he/she wants. Moreover, it is very convenient and economical to be integrated into different systems in today's industry. Finally, this study enables different systems to work together, enabling people to talk to machines and making people's lives easier with the help of machines.



Figure 20: Dialog with Virtual Assistant

```
Ürünlerin toplam liste fiyatı nedir  
What is the total list price of the products  
what is the total listPrice of the products  
  
SELECT SUM(product.listPrice)  
FROM product;
```

Figure 21. The SQL Query of Dialog with Virtual Assistant

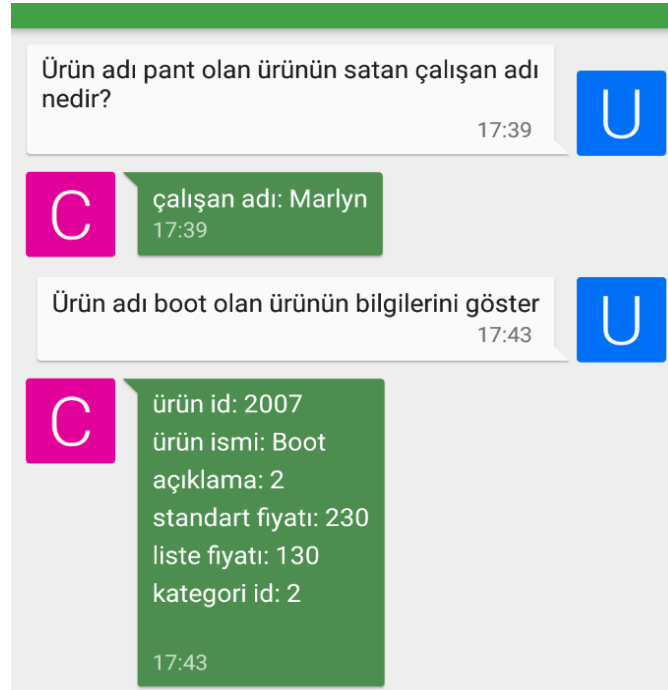


Figure 22: Dialog with Virtual Assistant

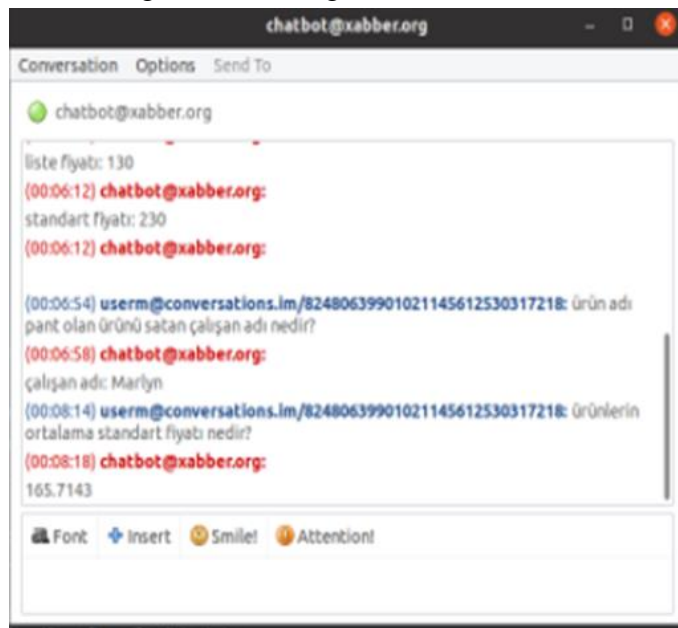


Figure 23: Dialog with Virtual Assistant on the Desktop Application

```
Ürün adı pant olan ürünü satan çalışan adı nedir?  
what is the employeefirstname that sells the product whose productname is pant?  
  
SELECT employee.employeefirstname  
FROM product  
INNER JOIN item  
ON product.productId = item.productId  
INNER JOIN indent  
ON item.indentId = indent.indentId  
INNER JOIN employee  
ON indent.idSalesman = employee.id  
WHERE product.productname = 'pant';
```

Figure 24: The SQL Query of Dialog with Virtual Assistant

5. ACKNOWLEDGMENT

Support was received from The Scientific and Technological Research Council of Turkey (TÜBİTAK) for this study within the scope of TÜBİTAK-2209 graduation project.

REFERENCES

- Adalı, Eşref. 2016. “Doğal Dil İşleme.” *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi* 5(2).
- Akyildiz, Ian, Su WY, Y. Sankarasubramaniam, and E. Cayirci. 2002. “Wireless Sensor Networks: A Survey.” *Computer Networks* 38:393–422. doi: 10.1016/S1389-1286(01)00302-4.
- Ali, Anum, Ghalib A. Shah, Muhammad Omer Farooq, and Usman Ghani. 2017. “Technologies and Challenges in Developing Machine-to-Machine Applications: A Survey.” *Journal of Network and Computer Applications* 83:124–39. doi: 10.1016/j.jnca.2017.02.002.
- Al-Rasyid, M. Udin Harun, Sritrusta Sukaridhoto, Muhammad Iskandar Dzulqornain, and Ahmad Rifai. 2020. “Integration of IoT and Chatbot for Aquaculture with Natural Language Processing.” *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18(2):640–48.
- Anon. n.d. “Cisco Visual Networking Index: Forecast and Trends, 2017–2022.” *Cisco*. Retrieved August 21, 2021 (<https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>).
- Atzori, Luigi, Antonio Iera, and Giacomo Morabito. 2010. “The Internet of Things: A Survey.” *Computer Networks* 54(15):2787–2805. doi: 10.1016/j.comnet.2010.05.010.
- Baker, Nick. 2005. “ZigBee and Bluetooth Strengths and Weaknesses for Industrial Applications.” *Computing & Control Engineering Journal* 16(2):20–25.
- Bandyopadhyay, Debasis, and Jaydip Sen. 2011. “Internet of Things: Applications and Challenges in Technology and Standardization.” *Wireless Personal Communications* 58(1):49–69. doi: 10.1007/s11277-011-0288-5.

- Banks, Andrew, and Rahul Gupta. 2014. "MQTT Version 3.1. 1." *OASIS Standard* 29:89.
- Bansal, Himanshu, and Rizwan Khan. 2018. "A Review Paper on Human Computer Interaction." *International Journals of Advanced Research in Computer Science and Software Engineering* 8:53–56.
- Başarslan, Muhammet Sinan, and Fatih Kayaalp. 2021. "Sentiment Analysis on Social Media Reviews Datasets with Deep Learning Approach." *Sakarya University Journal of Computer and Information Sciences* 4(1):35–49.
- Bhadgale, Anil M., Sanhita R. Gavas, and P. R. Goyal. 2013. "Natural Language to SQL Conversion System." *International Journal of Computer Science Engineering and Information Technology Research* 3(2):161–66.
- Couderc, Benoît, and Jérémy Ferrero. 2015. "Fr2sql: Interrogation de Bases de Données En Français." in *22ème Traitement Automatique des Langues Naturelles*.
- Dener, Murat. 2019. "A New Home Gateway Design and A Sensor-Based Smart Home Application Including Privacy Protection." *Bilişim Teknolojileri Dergisi* 12(1):23–32.
- Erdoğan, Hamdi, Kerem Küçük, and Sajjad Ahmad Khan. 2020. "Endüstriyel IoT Bulut Uygulamaları İçin Düşük Maliyetli Modbus/MQTT Ağ Geçidi Tasarımı ve Gerçekleştirilmesi." *Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi* 7(1):170–83.
- Farahani, Shahin. 2011. *ZigBee Wireless Networks and Transceivers*. Newnes.
- Group, Somayya Madakam\$IT Applications, National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai, India+R Ramaswamy\$IT Applications Group, National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai, India+Siddharth Tripathi\$IT Applications Group, National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai, and India. 2015. "Internet of Things (IoT): A Literature Review." *Journal of Computer and Communications* 03(05):164. doi: 10.4236/jcc.2015.35021.
- Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems* 29(7):1645–60. doi: 10.1016/j.future.2013.01.010.
- Jeschke, Sabina, Christian Brecher, Tobias Meisen, Denis Özdemir, and Tim Eschert. 2017. "Industrial Internet of Things and Cyber Manufacturing Systems." Pp. 3–19 in *Industrial Internet of Things: Cybermanufacturing Systems, Springer Series in Wireless Technology*, edited by S. Jeschke, C. Brecher, H. Song, and D. B. Rawat. Cham: Springer International Publishing.
- Khalil, Nacer, Mohamed Riduan Abid, Driss Benhaddou, and Michael Gerndt. 2014. "Wireless Sensors Networks for Internet of Things." Pp. 1–6 in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*.
- Kiani, Farzad. 2018. "Animal Behavior Management by Energy-Efficient Wireless Sensor Networks." *Computers and Electronics in Agriculture* 151:478–84. doi: 10.1016/j.compag.2018.06.046.

- Lee, Jin-Shyan, Yu-Wei Su, and Chung-Chou Shen. 2007. "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi." Pp. 46–51 in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*.
- Mainetti, Luca, Luigi Patrono, and Antonio Vilei. 2011. "Evolution of Wireless Sensor Networks towards the Internet of Things: A Survey." Pp. 1–6 in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*.
- Mainwaring, Alan, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. 2002. "Wireless Sensor Networks for Habitat Monitoring." Pp. 88–97 in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, WSNA '02*. New York, NY, USA: Association for Computing Machinery.
- Mohamudally, Nawaz, and Mahejabeen Peermamode-Mohaboob. 2018. "Building An Anomaly Detection Engine (ADE) For IoT Smart Applications." *Procedia Computer Science* 134:10–17. doi: 10.1016/j.procs.2018.07.138.
- Parthornratt, Tussanai, Dollachart Kitsawat, Pasd Putthapipat, and Prapap Koronjaruwat. 2018. "A Smart Home Automation Via Facebook Chatbot and Raspberry Pi." Pp. 52–56 in *2018 2nd International Conference on Engineering Innovation (ICEI)*.
- Roca, Surya, Jorge Sancho, José García, and Álvaro Alesanco. 2020. "Microservice Chatbot Architecture for Chronic Patient Support." *Journal of Biomedical Informatics* 102:103305. doi: 10.1016/j.jbi.2019.103305.
- Rodrigues, Joel J. P. C., and Paulo A. C. S. Neves. 2010. "A Survey on IP-Based Wireless Sensor Network Solutions." *International Journal of Communication Systems* 23(8):963–81. doi: 10.1002/dac.1099.
- Sadeghi, Ahmad-Reza, Christian Wachsmann, and Michael Waidner. 2015. "Security and Privacy Challenges in Industrial Internet of Things." Pp. 1–6 in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*.
- Saint-Andre, Peter. 2004. "Rfc 3920: Extensible Messaging and Presence Protocol (Xmpp): Core." *Internet Engineering Task Force*.
- Saint-Andre, Peter, Kevin Smith, Remko Tronçon, and Remko Troncon. 2009. *XMPP: The Definitive Guide*. O'Reilly Media, Inc.
- Shelby, Zach, Klaus Hartke, and Carsten Bormann. 2014. *The Constrained Application Protocol (CoAP). Request for Comments*. RFC 7252. Internet Engineering Task Force. doi: 10.17487/RFC7252.
- Sun, Zhi, Pu Wang, Mehmet C. Vuran, Mznah A. Al-Rodhaan, Abdullah M. Al-Dhelaan, and Ian F. Akyildiz. 2011. "BorderSense: Border Patrol through Advanced Wireless Sensor Networks." *Ad Hoc Networks* 9(3):468–77. doi: 10.1016/j.adhoc.2010.09.008.
- Taştan, Mehmet. 2019. "Akıllı Ev Uygulamaları İçin Yeni Nesil IoT Denetleyici İle Gerçek Zamanlı Uzaktan İzleme ve Kontrol Uygulaması." *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi* 23(2):481–87.
- Thangavel, Dinesh, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan, and Colin Keng-Yan Tan. 2014. "Performance Evaluation of MQTT and CoAP via a Common Middleware." Pp. 1–6 in

2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP). IEEE.

- TOÇOĞLU, Mansur Alp. 2020. “Sentiment Analysis for Software Engineering Domain in Turkish.” *Sakarya University Journal of Computer and Information Sciences* 3(3):296–308.
- Toğay, Cengiz, Gökhan Mutlu, Durmuş KURTULUŞ, and Faik ÖZGÜR. 2019. “Secure Gateway for the Internet of Things.” *Avrupa Bilim ve Teknoloji Dergisi* (16):414–26.
- Vermesan, Ovidiu, and Peter Friess. 2014. *Internet of Things-from Research and Innovation to Market Deployment*. Vol. 29. River publishers Aalborg.
- Yazar, Dogan, and Adam Dunkels. 2009. “Efficient Application Integration in IP-Based Sensor Networks.” Pp. 43–48 in *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, BuildSys '09*. New York, NY, USA: Association for Computing Machinery.
- Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. 2008. “Wireless Sensor Network Survey.” *Computer Networks* 52(12):2292–2330.
- Yu, Liyang, Neng Wang, and Xiaoqiao Meng. 2005. “Real-Time Forest Fire Detection with Wireless Sensor Networks.” Pp. 1214–17 in *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005*. Vol. 2.
- Zhu, Qian, Ruicong Wang, Qi Chen, Yan Liu, and Weijun Qin. 2010. “IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things.” Pp. 347–52 in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*.
- Ziegler, Sébastien, Peter Kirstein, Latif Ladid, Antonio Skarmeta, and Antonio Jara. 2015. “The Case for Ipv6 as an Enabler of the Internet of Things.” *IEEE Internet of Things*.