Research Paper
Academic Platform Journal of Engineering and Smart Systems (APJESS) 10(2), 106-114, 2022
ISSN: 2822-2385
Received: 31-Jan-2022 Accepted: 16-Apr-2022
https://doi.org/10.21541/apjess.1065390

LabVIEW and Embedded System based New IoT Solution for Industrial applications

[1]Görkem SUNGUR, [*2]Barış BORU

[1] Bursa Technical University, Electrical and Electronics Engineering Department, Bursa, Turkey, gorkem.sungur1925@gmail.com,
[2] Sakarya University of Applied Sciences, Faculty of Technology Mechatronics Engineering, Sakarya, Turkey,

Abstract

The Internet of Things (IoT) is becoming increasingly popular around the world. Efficiency in terms of time and cost is ensured in all sectors, thanks to the internet of things technology. A novel IoT system design was created in this study to add Internet of Things technology to a new perspective. All clients can connect to the server using this IoT platform, regardless of their hardware or software capabilities, even if they are not capable of industrial communication. The platform brings flexibility and ease of use to IoT applications. Thanks to the platform's architecture, any device, whether inputs or outputs, can be easily added to an IoT network while it is running using the developed modules. The server software was created using the LabVIEW visual programming language and Raspberry Pi and MSP340F5529 embedded systems are used to develop hardware modules. To provide IoT data security, the TCP/IP protocol is used for data communication, and all data is encrypted using the AES (Advanced Encryption Standard) algorithm with a 128-bit key. A dynamic 1-N server-client IoT system has been implemented and tested with various analog, digital, and smart sensors, as well as smart devices. When the results are evaluated, it is found that industrial applications can be developed with the platform. Particularly in the case of transformation of industrial automation applications to IoT containing non-smart sensors and actuators, the developed platform can be preferred. In this case, the platform will bring opportunities to engineers in terms of cost and ease of development phase.

Keywords: IoT, AES, Plug-in Architecture, Labview, Raspberry Pi, Energia

1. INTRODUCTION

The Internet of Things (IoT) generally refers to the interconnection of different types of computers and devices to support various monitoring and control applications. A wide variety of innovative applications such as smart homes and smart cities are developed, as well as industrial applications, with the addition of digital communication features like sensors, actuators, subsystems, etc., which are expressed as objects. The concept of the Internet of Things, which emerged with the Industry 4.0 industrial revolution, which is also called the digital revolution, has started to spread around the world. Studies are carried out on remote control and monitoring applications in order to make automation lines autonomous. It is ensured that production is carried out more efficiently, quickly and without errors. This study focuses on Industry 4.0 and industrial applications of IoT technology. In various applications of industrial automation, monitoring and management systems, each of the sensors and actuators must have communication capability. Considering the transformation of existing systems, the renewal of all hardware is costly, and the transformation of the software requires time.
With the proposed method, it is aimed that many new clients, independent of hardware and software, can easily connect and adapt to the server without stopping the running server and making any changes to its software. For this purpose, an embedded system solution, a new communication method and general management software that will enable the IoT transformation of end objects has been developed. When studies in this field in are examined in the current literature, the ones listed below draw attention.

Chuan and Ruslan created a remote control and monitoring application to keep the materials in a medical warehouse at an appropriate temperature and humidity limits. In their study, they managed to keep the temperature and humidity data within the desired limits [1].

* Corresponding Author

Kaya et al. carried out a remote monitoring application to monitor the system in the automation lines of the cable production factory. Thus, they increased their productivity in production by monitoring the lines remotely [2].

Bolivar and Silva established a solar radiation monitoring system in their study. They measured the sun rays coming from the optical sensors by means of a microcontroller. They sent this value to the Raspberry Pi card and sent it to the server over the UDP communication protocol. The server saved the data both in the local database and in the cloud-based data storage system [3].

Jegan and Nimi performed measurements with improved performance on filtering techniques for processing the ECG signal and provided the opportunity to monitor these measurements in LabVIEW software over TCP/IP [4].

Hnidka and Rozehnal sent the sensor data they collected with NI DAQ cards to the GUI over Wi - fi. They monitored the strain gauge data from the LabVIEW software interface via TCP/IP protocol. They recorded the data collected in real time in TDMS format [5].

Shah and Mishra designed a customized IoT system that enables wireless sensing and monitoring for smart buildings In their study, they succeeded in collecting data such as light, humidity and temperature for building automation via RF [6].

Vujovic and Maksimovic, on the other hand, created a home automation system using the Raspberry Pi card in their study. They used each Raspberry Pi board as a sensor node within the scope of the Internet of Things. They collected data from different sensors and transferred them to the web environment. [7].

Swain et al. designed an autonomous vehicle that can reach areas that require security, without waking the enemy. They aimed to use this vehicle for operations such as bomb detection and destruction and threat diagnosis for security guards in border security zones or battlefields [8].

Gómez et al. aimed to develop an architecture based on the Internet of Things (IoT), which allows the use of sensors with the capacity to collect information about environmental variables and also allows easy integration with other relevant sensors [9].

Prada et al. proposed the use of MQTT protocol, a lightweight protocol for communication with resource-constrained devices, and a tool for easy integration of new devices into the system, especially for those who use web standards such as JavaScript for creating user-interactive interfaces in educational applications [10].

Oksanen et al. worked on adapting the industrial automation protocol OPC UA (Open Platform Communications Unified Architecture) technology to IoT systems for remote monitoring of mobile agricultural machinery. They successfully completed the telemetry application by keeping the end-to-end delay time detected over the internet connection below 250 msec [11].

In their study, Shah and Bharadi collected biometric data with the Raspberry Pi card they used as an IoT device and encrypted the data they collected using RSA and AES-256 crypto algorithms. They sent the encrypted biometric data to the Azure cloud system. Thus, they have succeeded in keeping biometric data securely in the cloud system by providing information security [12].

In their study, Raja Singh et al. realized intelligent monitoring of power quality events using cloud-based embedded IoT and real-time logging of their data to the Firebase database on the server. With this study, it is aimed to detect the fault in industrial power applications remotely and to intervene in a short time [13].

Vakaloudis et al. aimed to organize the tasks and skills needed to quickly and seamlessly deploy an IoT platform in an industrial environment. While doing this, they talked about the difficulties and restrictions to be tackled. They have done their work using their experience in deploying small and medium-sized IoT solutions in a variety of fields (agriculture, health, manufacturing, water and energy) and industrial and community environments. They explained the applications that included real-life adaptations of the structure they established [14].

IoT system in 1-N structure with easy add-on was realized thanks to multi-channel programming and special add-remove protocol. Using the created add-drop protocol format and TCP/IP protocol, the integration of any hardware and software-independent client to server is easily achieved. In theory, an unlimited number of clients can connect to the server and send their data to the server asynchronously within a specified time. An innovative IoT system with a fast, high performance and dynamic structure was created by writing the server software in the LabVIEW environment.

The remaining parts of the article are planned as follows. In the second part, the materials and software also hardware technologies used are explained, in the third part, the findings of the study are presented. In the discussion and conclusion section, practical results, limitations, pros and cons of the developed system has been presented also a comparison table has been added for a relevant study in the existing literature.

2. MATERIALS AND METHOD

In this study, the server interface software was written using LabVIEW visual programming language. Virtual client software has also been developed in the LabVIEW environment. Raspberry Pi 2 and MSP430 cards were used as an example for embedded system hardware client cards. The client software of the Raspberry Pi board was implemented using the LabVIEW for Raspberry Pi library. The client software of the MSP430 board was written in C-based software in the Energia development environment. Sensors used as examples in the system are LM35 temperature sensor, MZ80 infrared sensor, 4-channel relay and different colored LEDs. MariaDB platform is used in the database management system where the data is recorded. TCP/IP and special add-remove protocols are used for

communication between server and client. The e-mail system is written with the SMTP mail protocol.

2.1. System Software

LabVIEW is a visual programming language, that is faster in terms of software development time compared to other text-based languages. Thanks to the dynamic structure it offers, reproducible multi-channel copy function structures can be created. These duplicate functions are executed by calling them from the generated template VI. After the Template VI software is written, it is configured as "Pre-allocated Reentrant VI". This setting allocates a separate data space to template VI each time it is called. In other words, the data that these sub-functions keep in their memories are recorded in separate areas.

In LabVIEW, copies of template VI are called parallelly and asynchronously. The position of the template VI to be copied is converted into a reference. Template of VI dynamic aspect summoning is shown Figure 1. While the template VI is called dynamically, the first data is sent over the VI SERVER. The template VI reference that is opened is closed, eliminating unnecessary memory occupancy of this reference.

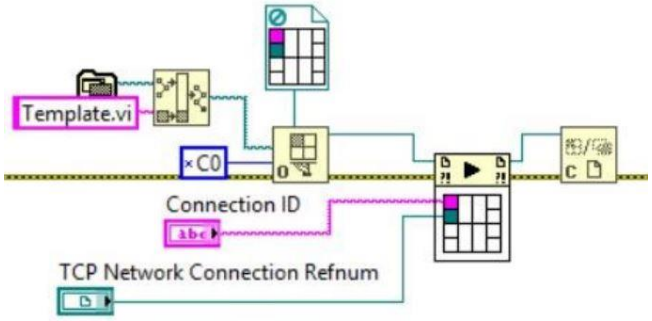

Figure 1. Template of VI dynamic aspect summoning

Template VI and the Producer/Consumer design template are used for the server. While user-interactive events are handled in the producer loop, the states of the events that occur in the consumer loop are processed. The cycle time of the event generating loop must be higher than the consuming loop. Queuing structures are used to create this design template and to provide inter-loop communication.

In this study, a queue-based Asynchronous Messaging structure (AMC) was used. Thanks to the AMC library, the server communicates asynchronously with all template VIs.

Messages created in producer loops are queued to be sent to consumer loops. In order to give priority to the message in the queue structure, which works with the first-in, first-out logic, the message sending function is adjusted. The consumer loop runs event states to process the messages it receives from the queue. Interprocess communication diagram is shown in Figure 2. Queues use queue names to communicate among themselves.

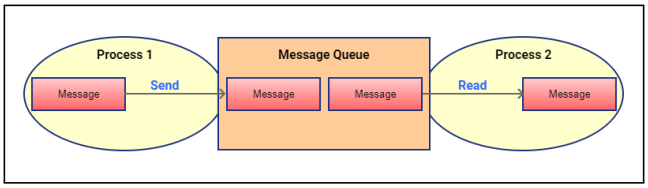

Figure 2. Interprocess communication diagram

LabVIEW for Raspberry Pi library is a tool that converts LabVIEW code to Python code both with and without an interface and transfers it to Raspberry Pi as an executable file. Thus, the software of the Raspberry Pi 2 embedded system board is written with the LabVIEW programming language without using the Python language.

The MSP430F5529 LP embedded system board was written in the Energia development environment. Energia is a development environment that can program Launchpad series such as MSP430 and 432, CC3100.
MariaDB is a relational database management system. A database management system is a system and software designed to create, use and modify databases and to meet all kinds of operating needs related to database systems. In this study, MariaDB database management system was used. Access to client data by HeidiSQL GUI is shown in Figure 3.

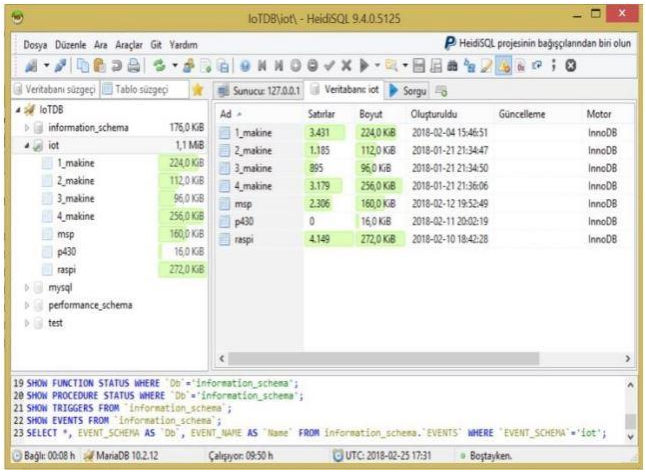

Figure 3. Access to client data by HeidiSQL GUI

Database Connectivity library is used in LabVIEW program. The database is accessed using this library. A separate table is automatically set up for each client connecting to the server. The name of this table consists of the machine name, and the columns consist of the data names. In addition to the data names, they are recorded at that date and time. When a client that was previously connected to the server reconnects to the system with the same name, it sends its data to the existing table. However, when it is connected with a new name, a different table is created and the new data is saved in this table.

Client data can be accessed from the server interface as well as from the HeidiSQL database client. HeidiSQL is a database client that is installed alongside MariaDB and supports database management system servers such as MySQL and SQL.

2.2. System Hardware

The Raspberry Pi 2 client is a credit card-sized single board minicomputer. It has a 900Mhz quad-core ARM Cortex-A7 CPU on it. Apart from the client, there are 4 USB ports, 1GB RAM, 40 GPIO pins, HDMI port, Ethernet port, audio output, camera connection interface, display interface, SD card slot and 3D graphics video core. Raspberry Pi client is connected to the server via Wi -fi through a plug-and-play USB Wi - fi adapter.

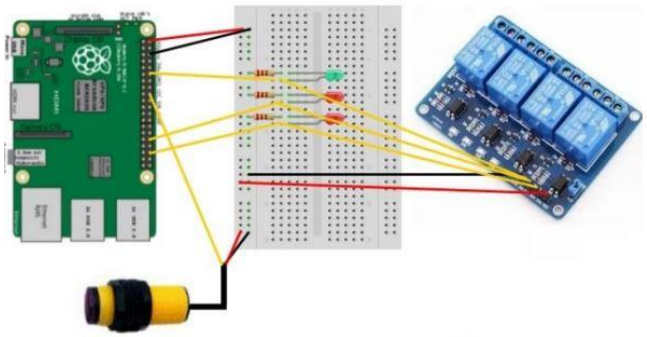

Figure 4. Sensor connections of Raspberry Pi device

Three digital outputs and one digital input are connected to the Raspberry Pi board. The digital outputs are 3 LEDs and 3 relays. The digital input is 1 MZ80 infrared sensor with 80 cm range. LED and relay outputs consist of the same pins. In addition, there is a random virtual data in the software of the RPI hardware to generate analog values between 0 and 10. Sensor connections of Raspberry Pi device is shown in Figure 4. The client card is prepared to send this data and digital input sensor data to the server every 300 milliseconds. Raspberry Pi client can be converted into an embedded system board that can be used in industry with additional cards.

The MSP430F5529 LP is an inexpensive and easy development kit for the MSP430F5529 USB microcontroller. MSP430F5529, a 16-bit MCU, has 128KB flash memory, 8KB RAM, 25MHz CPU speed, 12-bit A/D converter, 5 timers, onboard USB and many peripherals. The operating voltage of the device is between 1.8V-3.6V. There are 40 pins on the device.

Wi-fi add-on package is used to connect MSP430F5529 client wirelessly to a public network over the internet. Data is sent to the CC3100 add-on package via SPI. One LM35 temperature sensor and one LED are connected to this client. Sensor connections of MSP430F5529 LP device is shown in Figure 5.

2.3. System Communication Protocols

The protocol that provides data transmission between the server and the client is the TCP/IP protocol. This protocol includes both TCP and IP protocol. TCP/IP protocol consists of a 4-layer model. It consists of the application layer, transport layer, network layer and physical layer. The application layer contains applications that will process communication data. The transport layer determines how the data travels. The network layer is also known as the IP layer. Addresses are added to the packets and the data finds its destination. The physical layer determines the transmission path of the data.

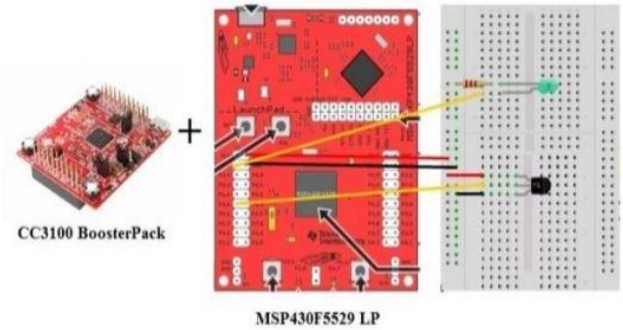

Figure 5. Sensor connections of MSP430F5529 LP device

TCP protocol splits the data coming from the upper layer into appropriate lengths. Gives a sequence number to the fragmented data. Allows reprocessing if parts are lost or sent damaged. It adds the header information to the packet to identify the tasks assigned to it. The IP layer, which is a lower layer, adds an IP header to the incoming packets and sends it to the receiver [15].

LabVIEW program, server-client software has been developed by using TCP/IP library in many software languages. In the server software, port 3584 is listened to at intervals of 100 milliseconds. The client software connects to the server with the connection function using the server's IP address and port 3584. TCP read and write functions also provide data exchange between server and client. In this study, while providing data communication, firstly the size of the data to be sent is calculated. The length data of a one-byte text is converted to text and added to the main data, and the total data is sent. The function that receives the data first reads the first byte of data and then determines the length of the data to be read. In the second reading, the main data is read and communication is provided. Server connection check, get initial text data is shown in Figure 6.

The server warns the users by e-mail system when an error occurs in any of the clients. This system is implemented through the SMTP (Simple Mail Transport Protocol) Simple Mail Transport protocol.

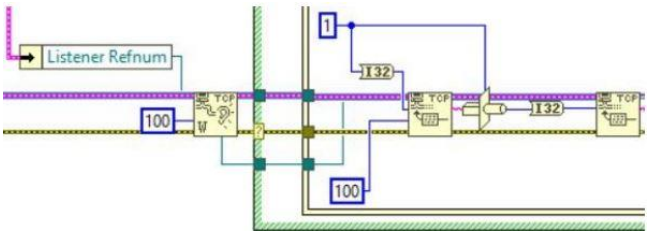

Figure 6. Server connection check, get initial text data

A special client add-remove protocol has been developed for new clients to connect to the server dynamically and asynchronously while the server is running and without making any changes to the server's software. Thanks to this protocol, any hardware created with client software suitable for the formats shown in the figure can be easily integrated

into the server. Thus, it is possible to add both hardware and software independent clients to the server at runtime.

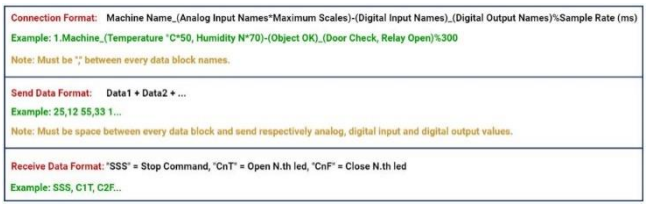

Figure 7. Application formats and examples for the client

The format that the client should send the first time it connects to the server is the connection format. In this format, the machine name, input-output names, scales, units and data collection time are entered. The data sending format is the format of the data that the client sends to the server during data collection. The data import format is the commands that go from the server to the client.

In the data import format “SSS” is the command to stop the client. “CnT” and “CnF” commands n. These are commands that specify the active or passive digital outputs. In the data sending format, a space must be placed between the data. In addition, the data must be sent in order of analog input, digital input and digital output. A comma must be placed between each data collection in the link format. The “%” sign should be used before the data collection time expression. “*” sign is used between scale data and data names. Detailed application formats and examples for the client is shown in Figure 7.

The firmware loop communicates via the TCP/IP protocol with the server TCP/IP connection operations consumer loop to establish the initial connection, and the template VI user interface consumer loop to send and receive data. If the server TCP/IP connection operations are the consumer loop, the server user interface communicates with the consumer loop and the template VI user interface communicates with the consumer loop. Inter-loop communication communicates over the AMC structure. The general communication diagram of the IoT system is set up in this way. Messaging and communication diagram of the system is shown in Figure 8.

2.4. Server software

The server software is written in LabVIEW programming language. TCP/IP listener is created after general reset settings when the server software is started. The server receives the initial data with the connection request from the clients. The initial information and connection reference are sent to the template VI, which will constantly communicate with the client. The client sends its data to the matching template VI at specified time intervals. The digital output of the current client has been checked over Template VI. If the client stops, the template VI is also closed and the server and client are disconnected. The client can also be stopped via the server and template VI.

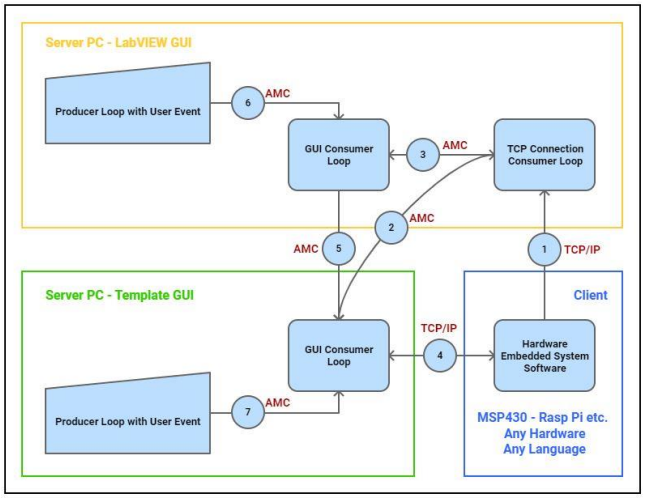

Figure 8. Messaging and communication diagram of the system

The server interface consists of two parts. The first part is where all clients are checked and general checks are made. The second section is the template VI interface section that opens specifically for each client. During data collection, data recording and instant data monitoring are performed from this interface, operations on historical data are carried out from the main server interface.

Clients connected to the server appear in the Connections selection bar. It is possible to operate on the client selected from this section. The client's template VI windows are hidden, shown, or terminated. Data from the database can be called to the table or output to excel.

More than one e-mail address can be registered in the system and e-mail can be sent. Warnings are presented in both written and visual form. Warning texts can be recorded. A competency-based password can be set for security purposes. A client can connect to the server only as long as the server allows it.

Template VI client windows open automatically as the client connects to the server. Data is monitored instantly from these windows.

2.5. Client software

A client's software can be written on any preferred platform. The client, whose software is written in accordance with the determined rules, can easily connect to the server. In this study, Energia software development platform for MSP430F5529 and LabVIEW for Raspberry Pi 2 for Raspberry Pi library is used. In client software, TCP/IP functions and client cards are generally connected to the internet. Clients connecting to the Internet via Wi-fi send a connection request to the server. The client that successfully connects to the server sends the startup information to the server. Sample client window opened with Template VI is shown in Figure 9.

This initial data includes information such as machine name, I/O names, scales, units and data collection times. Communication is achieved when the server creates a

template VI for the client. The client software sent its data to the template VI over the TCP/IP protocol. The client has processed any data from template VI, if any. Thus, the server both controlled the client and collected the client's data. The server stops the client software by sending the "SSS" command to the client.

Virtual clients and clients with operating systems such as Raspberry Pi have the interface. Thanks to these interfaces, users can see the data sent by the clients and the controlled digital output LEDs. Thus, there is no need to change the existing client software for simple operations.

Code block of the client software for MSP430 is shown in Figure 10. Server software interface is shown in Figure 11 and software flow chart of clients is shown in Figure 12.

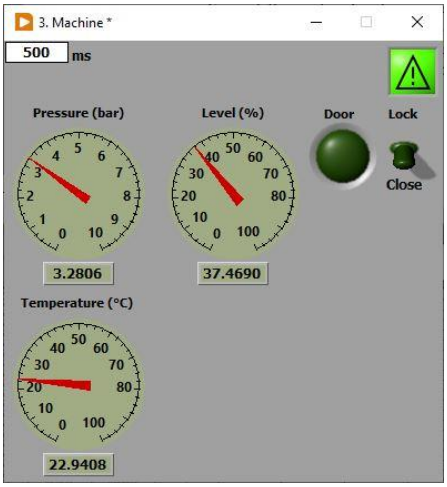

Figure 9. Sample client window opened with Template VI

```
{
 TempVal = analogRead(TempPin);
 TempVal /= 12.42;
 Serial.println(TempVal);
 String TempNew = String (TempVal);
 Serial.print("Temp=");
 Serial.print(TempVal);
 char TempLen = char (TempNew.length());
 String TotalTemp = TempLen + TempNew;
 client.print(TotalTemp);
 delay(500);
}
```

Figure 10. Code block of the client software for MSP430

2.6. Data Encryption

The communication between the server and the clients has been established by encrypting with AES algorithms with a 128-bit key for the security of IoT data. Therefore, the sender encrypted the data with this key. The receiver, on the other hand, decrypted the data with the same key and processed it. Even if the protocol is known, if the key is not known, it cannot be interrupted and the communication is rendered undecipherable. This process has become an IoT security solution suitable for today's crypto technology [16].

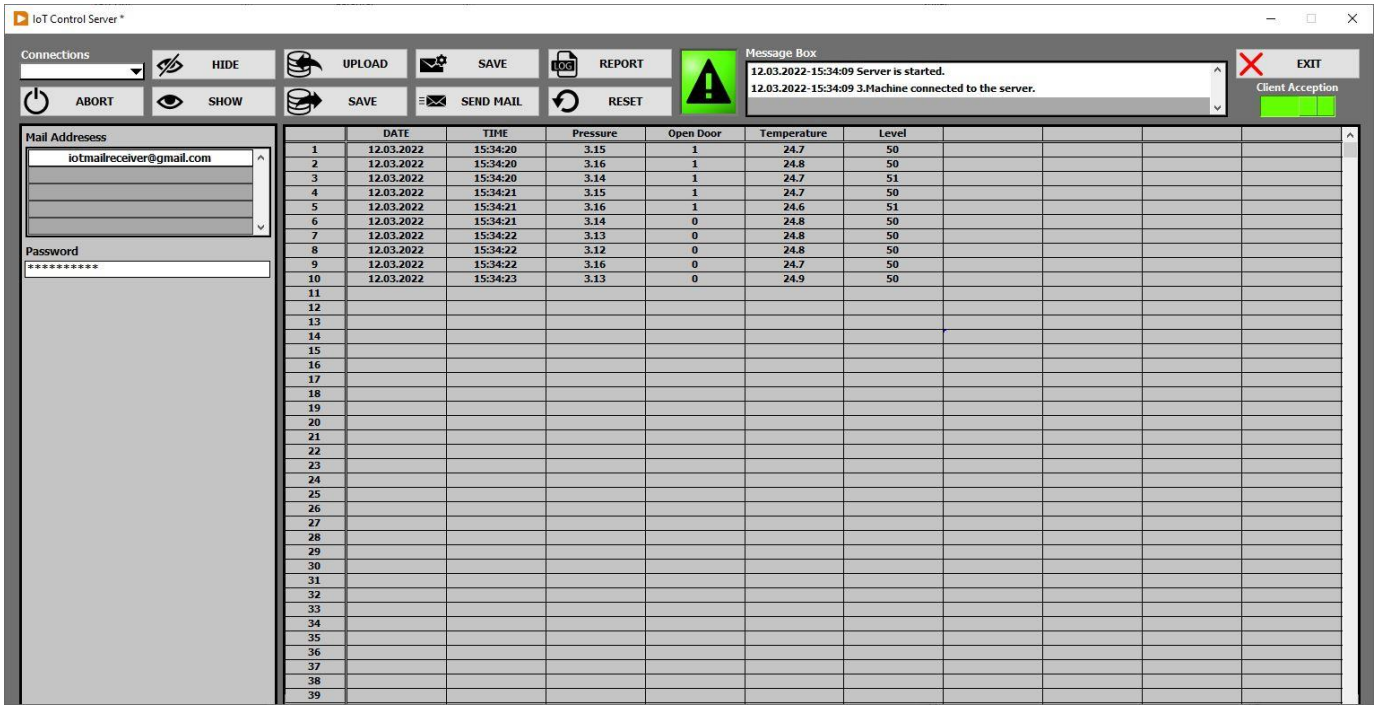

Figure 11. Server software interface

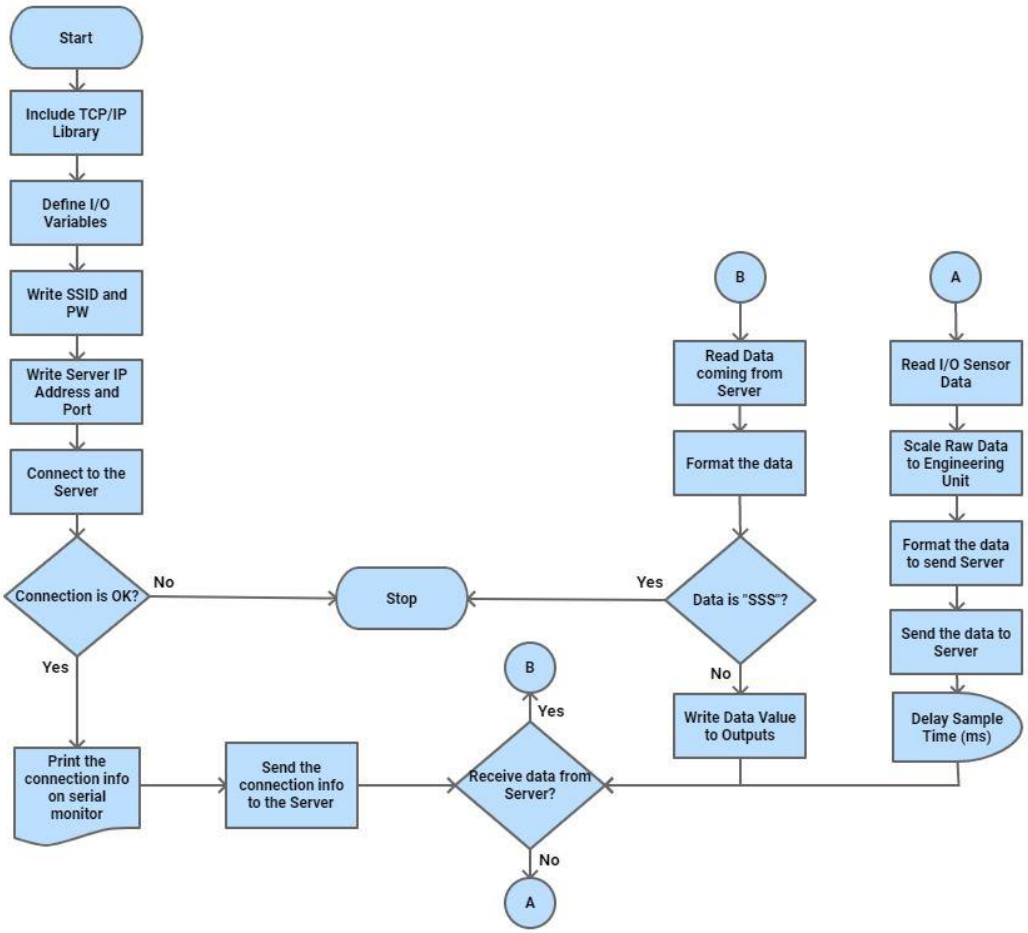

Figure 12. Software flow chart of clients

3. FINDINGS

In this study, a total of six clients, two real and four virtual, were used. By connecting different sensors to all clients, data collection processes were carried out successfully at different data collection times. Since the server receives the data asynchronously with multi-channel software, data collection was carried out full-time independent of the number of clients.

The 1st machine client in 300 milliseconds, the 2nd machine client in 200 msec, the 3rd machine client in 500 msec, the Raspberry Pi client in 300 msec and the MSP430 client in 400 msec. The server managed to collect data at intervals of at least 30 milliseconds. The client and server work simultaneously.

It has been determined that the ideal data collection time is at least 100 milliseconds in terms of ease of tracking the data coming to the computer. Data collection and data control processes do not affect each other. Data control was also provided at the time of data collection. Six clients were connected to the system sequentially and their data was collected and recorded in the database.

4. DISCUSSION AND CONCLUSION

In the implemented system, there is no theoretical limit on the number of clients to be added to the server. However, since each client used calls a template VI, it creates a processing load on the computer the server is running on. The hardware capacity of the server computer is a factor that will limit the number of clients. Six clients were run smoothly on a server computer with Intel Core-i7-4510U, dual core, quad channel, 2.00 GHz processor speed, 4 MB cache, which was used during the experimental study.

In this study, it was determined that the minimum data collection rate was 30 milliseconds. This period can be further reduced by increasing the bandwidth of the internet infrastructure used or by increasing the hardware capacity of the computer used.

Table 1. Comparison of the current thesis study with the study of Kaya et al. [2]

Parameters	Design and implementation of flexible automation systems data tracking system [2]	LabVIEW and Embedded System based New IoT Solution for Industrial applications
Sample Rate	Unclear	Min. 30 milliseconds depending on system hardware and internet speed
Cost	More cost	Less cost
Communication Protocol	TCP/IP	TCP/IP and Plug-in architecture application protocol
Scalable Structure	Not have	Easy Plug-in without stopping
Software Architecture	Static	AMC – Dynamic
Client Availability	Only PLC	Hardware independent
Maximum Client Number	8	Theoretically unlimited
Server Software Language	LabVIEW	LabVIEW
Client Software Language	PLC, LabVIEW	Software independent
Data Record Option	Text, Excel	Text, Excel, MySQL (MariaDB)
User Warning System	E-mail, LED	SMTP protocol, E-mail, LED

As given Table 1, the advantages of the current thesis study over other studies can be listed as follows:

- Instead of putting a costly computer per machine, the cost of working is reduced by using inexpensive industrial-based embedded system cards that can be connected to the monitor, called mini-computers.
- Thanks to the AMC software architecture built on the dynamic structure used in the study and the plug-in application protocol created, the cost is reduced as it becomes easy to integrate new machines to the system by connecting client cards to the server without changing the server software.
- As a hardware and software-independent client structure is created, the dependency on a single type of data collection device is eliminated, and the integration of different systems into the server increases the variety for needs.

The applications that can be made in addition to this study can be summarized as follows:

- With the mobile application, the data can be tracked on the mobile phone.
- By opening the server to the outside world with port forwarding, data can be collected from every client connected to the internet.
- The data storage size problem can be eliminated by uploading the data to the cloud system.

Author contributions: Concept, Literature Search, Writing - G. S. , B. B. Software and Hardware Development - G. S.
Conflict of Interest: This study was produced from the MSc thesis entitled "Suitable for Industrial use LabVIEW and Embedded System based a new IoT Solution" by Görkem SUNGUR, which was accepted in 2018.
Financial Disclosure: The authors declared that this study has received no financial support.

REFERENCES

[1] OW Chuan, SH Ruslan , “Medical warehouse monitoring and control system using LabVIEW ”, International Conference on Electrical , Electronics and Optimization Techniques (ICEEOT), pp . 2396-2401, 2016.

[2] B. Kaya, A. Altintas, U. Kök, “Design and Implementation of a Flexible Automation System Data Tracking System, Süleyman Demirel University Journal of Technological Sciences (UTBD), pp . 30-38, 2015.

[3] LEP Bolivar , AG Alexandre da Silva , “Solar radiation monitoring using electronic embedded system Raspberry Pi database connection MySQL , Ubidots and TCS-230 sensor”, CHILEAN Conference on Electrical , Electronics Engineering , Information and

Communication Technologies (CHILECON), pp . 473-479, 2015.

[4] R. Jegan , WS Nimi, “Low cost and improved performance measures on filtering techniques for ECG signal processing and TCP/IP based monitoring using LabVIEW ” 4th International Conference on Advanced Computing and Communication Systems (ICACCS), pp . 1- 7, 2017.

[5] J. Hnidka , D. Rozehnal , “Strain gauge measurement system with Wi - Fi data transfer in LabVIEW ”, International Conference on Military Technologies (ICMT), pp . 520-525, 2017.

[6] J. Shah , B. Mishra , “ Customized IoT enabled wireless sensing and monitoring platform for mart buildings ”, Procedia Technology , Vol.23, pp . 256-263, 2016.

[7] V. Vujovic , M. Maksimovic , “Raspberry Pi as a sensor web node for home automation ”, Computers and Electrical Engineering , Vol.44, pp . 153-171, 2015.

[8] BK Swain , S. Dash , SS Gouda , “ Raspberry Pi based integrated autonomous vehicle using LabVIEW ”, 3rd International Conference on Sensing , Signal Processing and Security”, pp . 69-73, 2017.

[9] JE Gómez , FR Marcillo , FL Triana , VT Gallo , BW Oviedo , VL Hernández , “ IoT for environmental variables in urban areas ”, Procedia computer Science , Vol . 109, p . 67-74, 2017.

[10] MA Prada , P. Reguera , S. Alonso , A. Morán , JJ Fuertes , M. Domínguez , “Communication with resource-constrained devices through MQTT _ control education ”, IFAC- PapersOnLine , Vol.49, pp . 150-155, 2016.

[11] T. Oksanen, R. Linkolehto , I. Seilonen , “Adapting an industrial automation protocol to remote monitoring of mobile agricultural machinery : a combine harvester with IoT ”, IFAC -PapersOnLine , Vol.49, pp . 127-131, 2016.

[12] D. Shah, V. Harad , “IoT Based biometrics Implementation on Raspberry Pi”, Procedia computer Science , Vol.79, pp . 328-336, 2016.

[13] R. Raja Singh, Yash S.M., Shubham S.C., Indragandhi V., Vijayakumar V., Saravanan P., Subramaniyaswamy V., “IoT embedded cloud-based intelligent power quality monitoring system for industrial drive application”, Future Generation Computer Systems, Vol.112, pp. 884-898, 2020.

[14] A. Vakaloudis and C. O’Leary, "A framework for rapid integration of IoT Systems with industrial environments," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pp. 601-605, 2019.

[15] Network Security and Network Protocols, Department of Electrical and Electronic Technologies, Ministry of National Education Publications, 2011

[16] Z. Rahman, X. Yi, I. Khalil, M. Sumi, “Chaos and Logistic Map based Key Generation Technique for AES - driven IoT Security”, Computer Science - Cryptography and Security, H.4.1, 2022.