Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 1 February 2025

Translation of article 10.22146/jnteti.v14i1.7058

Automatic Liquid-Filling Machine Using Arduino and LabVIEW

Syafriyadi Nor[1], Zaiyan Ahyadi[1]

[1] Electrical Engineering Study Program, Department of Electrical Engineering, Politeknik Negeri Banjarmasin, Banjarmasin, Kalimantan Selatan 70123, Indonesia

[Submitted: 5 February 2024, Revised: 3 May 2024, Accepted: 24 February 2025]
Corresponding author: Zaiyan Ahyadi (email: z.ahyadi@poliban.ac.id)

ABSTRACT — The automatic liquid-filling machine plays a vital role in improving efficiency and productivity in modern manufacturing and packaging industries. However, challenges such as high costs, complexity, and limited technical knowledge often hinder its adoption. This research aimed to develop an educational system that is simple, affordable, and easy to implement, helping students grasp the fundamental principles and real-world applications of automatic liquid-filling machines. The system integrates LabVIEW for visual processing and an Arduino Nano microcontroller with the Modbus remote terminal unit (RTU) protocol to simulate industrial communication standards. LabVIEW controls the conveyor belt, filling, and capping processes using ladder logic while recording the number of filled bottles. The Arduino microcontroller manages conveyor belt operations and allows users to set volume and bottle count via a keypad. Serial communication between LabVIEW and Arduino through Modbus RTU provides hands-on experience in configuring industrial systems. Experimental tests under various operational scenarios confirmed the system's accuracy in filling bottles within a volume range of 250–1,000 ml at a speed of 10 ml/s, handling up to five bottles per cycle. The system demonstrated stable operation without disruptions. This research enhances instrumentation and control system education by offering an interactive, cost-efficient learning tool. The successful use of Modbus RTU underscores its reliability in supporting automatic liquid-filling machines while enriching students' understanding of industrial automation.

KEYWORDS — Instrumentation, Arduino, LabVIEW, Filling Machine, Capping, Modbus.

I. INTRODUCTION

Automatic liquid-filling machines are widely used in various industries, such as food, pharmaceuticals, chemicals, and cosmetics, to fill liquids of different volumes and quantities into containers. Precision in liquid-filling is essential to prevent financial losses, while automation enhances efficiency and quality in the industrial sector [1]. Manual liquid-filling requires significant labor and time, reducing overall efficiency and productivity [2]. Moreover, automation aims to improve product quality, increase production rates, reduce labor dependency, lower labor costs, minimize raw material wastage, and reduce unproductive time [3].

Automatic liquid-filling machines function by filling liquid into bottles, glasses, or cans. Additionally, these machines can package various liquid products, such as mineral water, oil, soft drinks, and wine, into bottled, glass, or canned packaging [4]. The machine automatically fills liquid into containers according to pre-set volume and quantity parameters. With Industry 4.0 advancements, these machines require minimal human intervention in control processes [5], effectively addressing workforce shortages [6].

The filling and capping processes in liquid product manufacturing often face challenges, such as control issues when errors or system failures occur [7]. Furthermore, implementing automatic liquid-filling machine systems requires complex hardware and software configurations to adjust volume measurements and bottle counts. By integrating software and hardware technologies such as LabVIEW and Arduino microcontrollers, the implementation provides insight into the liquid-filling process within packaging systems. Additionally, the required devices are more cost-effective. LabVIEW is a graphical programming language first introduced by National Instruments in 1986 [8]. LabVIEW programming does not involve textual code but uses dataflow and graphical representations through block diagrams. LabVIEW offers numerous features and advantages, including ease of use, an intuitive interface, and real-time data processing capabilities. Meanwhile, Arduino microcontrollers serve as an open-source platform capable of controlling various machines and systems [9]. Furthermore, Arduino microcontrollers are cost-effective.

Research on automatic liquid-filling machines integrating LabVIEW technology and Arduino microcontrollers remains limited. The implementation of programmable logic controllers (PLCs) as control processes has been explored in previous studies [10]–[14]. The design and implementation of automatic liquid-filling systems using Arduino microcontrollers and LabVIEW have also been investigated [15]–[18]. Other studies have developed systems for monitoring water turbidity, temperature, and pH using a LabVIEW and Arduino interface [19]. Besides Arduino, integration with LabVIEW can also be achieved using a programmable logic controller (PLC). One study developed a windscreen washer-filling machine system using PLCs and LabVIEW [20]. The communication interface between PLCs and LabVIEW employed open process control (OPC). System monitoring was conducted using supervisory control and data acquisition (SCADA) implemented through LabVIEW's data logging and supervisory control (DSC) module. However, this study did not address the input determination for volume measurements and bottle counts. The integration between LabVIEW and Arduino was established using the Modbus remote terminal unit (RTU) protocol [20]. The utilization of Modbus has become an industry-standard protocol, facilitating hardware and software integration and compatibility across various manufacturers, including Arduino and LabVIEW. Therefore, this study aimed to design an

industrial plant system for automatic liquid-filling machines with volume measurement and bottle count inputs through a keypad, utilizing LabVIEW and Arduino microcontrollers to simulate real industrial processes.

A novel aspect of this research is the integration of LabVIEW and Arduino via the Modbus RTU protocol. The Arduino microcontroller functions as the physical controller operating the plant within LabVIEW. In similar studies, PLCs have generally been used as hardware controllers. This combination addresses accessibility limitations and the availability of industry-relevant equipment. This research is essential in assisting students in control system instrumentation laboratory exercises and providing insights into the processes within automatic liquid-filling machine systems.

II. SYSTEM DESIGN

In this study, a simulation plant for an automatic liquid-filling machine system was designed using the LabVIEW 2017 application. Figure 1 presents the block diagram of the automatic liquid-filling system, which consists of an Arduino microcontroller, push buttons, a keypad, an LCD, LEDs, and LabVIEW control. The Arduino is utilized to process input data from the keypad and push buttons, while the LCDs the menu interface and input data from the keypad. LEDs function as indicators for commands from the push buttons. The LabVIEW control receives control signals and executes the liquid-filling process.

This system employs the Arduino Nano ATMega 328 microcontroller modules. The Arduino Nano features 22 digital pins for input/output ports, 8 analog pins, and several serial communication protocols, such as universal asynchronous receiver-transmitter (UART), serial peripheral interface (SPI), and inter-integrated circuit (I2C) [21]. The Arduino Nano is equipped with two push buttons for start and stop functions, along with two LEDs as indicators. A green LED indicates that the conveyor belt is operating, while a red LED signifies that the conveyor belt has stopped. To input the fill volume and the number of bottles, a 3 × 4 keypad is used alongside a 16 × 2 LCD with an I2C module. To interface the push button and keypad inputs with LabVIEW, the Arduino's UART serial protocol is connected to the LabVIEW serial protocol using the National Instruments Virtual Instrument Software Architecture (NI-VISA). NI-VISA is an application programming interface (API) that provides serial communication programming interfaces for National Instruments (NI) applications [22].

In LabVIEW control, the system design consists of a conveyor belt, a proximity sensor for detecting the presence of bottles as the conveyor belt moves, a liquid-filling machine (FILLING), and a bottle-capping machine (CAPPING). These machines are developed using the datalogging and supervisory control (DSC) module, an add-on software module for LabVIEW that facilitates the development of human-machine interfaces (HMI).

Although LabVIEW is a graphical programming environment based on block diagrams, this study implements the control process for the automatic liquid-filling system simulation using ladder logic. Ladder logic was selected because it is commonly used in the development of PLC software and has been widely implemented in industrial control applications. In this context, ladder logic is used to construct sequential logic for the conveyor belt process, including filling, capping, and bottle counting. The control algorithm can be programmed and structured based on ladder logic, while

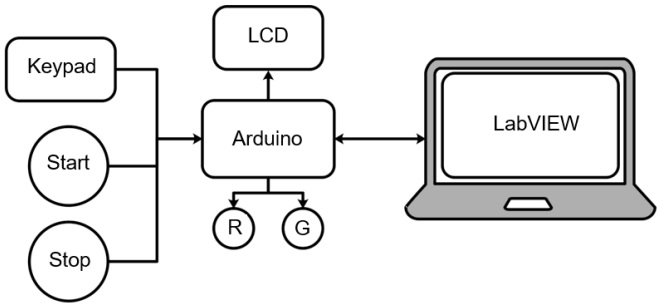


Figure 1. System block diagram of automatic liquid-filling machine.

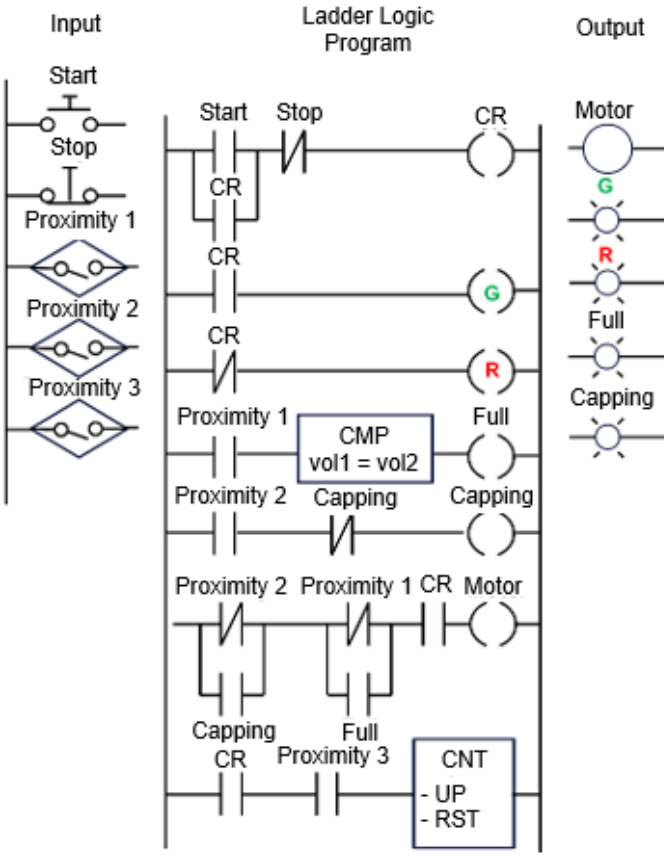


Figure 2. Ladder logic on automatic liquid-filling machine system.

counter and timer functions are adjusted according to the programming rules in LabVIEW.

Figure 2 illustrates the ladder logic used for system control in the LabVIEW block diagram. The operation of this plant is as follows: when the start button is pressed, the control relay (CR) activates and turns on the green (G) lamp as an indicator that the conveyor belt is running to move bottles into and out of the filling station. When proximity sensor 1 detects a bottle, the conveyor belt stops to allow liquid-filling. The fill volume is determined by the keypad input, which is transmitted by the Arduino via serial communication.

Once the desired quantity of liquid has been dispensed, the motor resumes conveyor belt operation. When proximity sensor 2 detects a bottle, the conveyor belt stops again, initiating the bottle-capping process. Once the capping process is complete, the conveyor belt reactivates to move the bottle forward. When proximity sensor 3 detects a bottle, the bottle is subjected to a counting process.

The total counter resets to zero upon reaching a predefined setpoint. The process stops when the stop button is pressed; the

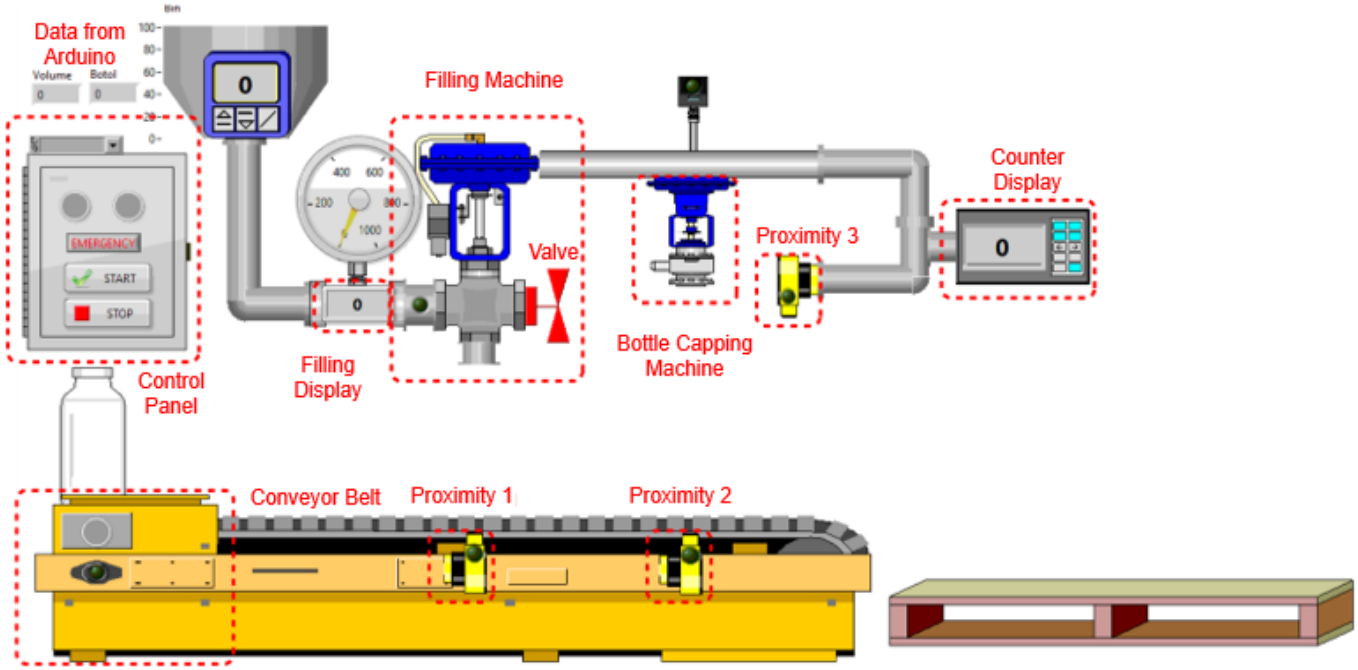


Figure 3. Automatic liquid-filling machine system components on LabVIEW Front Panel.

G lamp turns off, and the red (R) lamp illuminates. All components of the automatic liquid-filling machine system on the LabVIEW Front Panel are displayed in Figure 3.

The control panel is used to control components within the LabVIEW front panel. The LED indicators on the panel illuminate simultaneously with the LEDs connected to the Arduino. The conveyor belt is responsible for transporting bottles to the filling station until the capping process is complete. The proximity sensors detect bottle presence as the conveyor belt moves. The filling display provides information regarding the fluid flow rate into the bottle through a pipeline or channel

Data from the Arduino is transmitted to LabVIEW using serial communication. The filling machine fills liquid into bottles when proximity sensor 1 detects a bottle, while the capping machine caps bottles when proximity sensor 2 detects a bottle. The counter display provides information on the number of bottles detected by proximity sensor 3. A valve serves as an indicator for the open or closed status during the filling process.

The next stage involves programming in the Arduino IDE to acquire data from push button and keypad readings and subsequently display this data on the LCD. Additionally, a selection menu is developed on the LCD, consisting of VIEW and SET menus.

The SET menu function is used to input the fill volume into bottles within a range of 0–1,000 ml. The fill volume is entered by pressing numerical keys on the keypad "0" – "9." Since the keypad function returns values in char data type, conversion to an integer is required. The conversion is performed using (1):

$$x = x \times 10 + (key - '0'). \quad (1)$$

where x is an integer data type, *10* is a constant that is multiplied by x to add 1 each time a key is pressed, and the key represents a numerical value on the keypad "0" – "9," adjusted by subtracting the character "0" to obtain a decimal number from the American Standard Code for Information Interchange (ASCII) code. The character numbers "0" – "9" are converted into decimal numbers "48"–"57."

Once the fill volume is set, pressing the "*" symbol on the keypad saves the value into a variable. The next step involves inputting the number of bottles to be filled. Once this input is confirmed, pressing the "*" symbol again saves both the fill volume and the bottle count. The display then returns to the VIEW menu, where the fill volume and bottle count variables are displayed on the 16 × 2 LCD. When the START (1) or STOP (0) push button is pressed, the stored data is transmitted to LabVIEW using the Modbus RTU protocol.

This process enables LabVIEW to receive and process the fill volume and bottle count data from the system, offering enhanced flexibility and control in real-time monitoring and parameter adjustment. With Modbus RTU integration, the collected information can be efficiently accessed and utilized in LabVIEW applications. To communicate using the Modbus protocol on Arduino, a compatible library is required. In this study, the ModbusRTUSlave library was used [23].

Figure 4 presents the Modbus RTU message structure, which consists of the Slave ID (identification), function code, data, and cyclic redundancy check (CRC). If the Slave ID and CRC sections of a Modbus protocol are omitted, the resulting data unit is known as a protocol data unit (PDU).

The Slave ID is a part of the Modbus protocol that specifies the address of the Modbus slave device communicating with the Modbus master device. Each Modbus slave device is identified by a unique address known as the Slave ID. Additionally, the Slave ID is used by the Modbus master device to determine the target device for requests or instructions.

The function code is a numeric value identifying the type of operation requested by the Modbus master or performed by the Modbus slave. For instance, function code 03 reads a holding register, while function code 06 writes to a holding register. Function codes instruct the Modbus device on the action that should be performed. Each function code has a specific meaning and function in the context of Modbus operations.

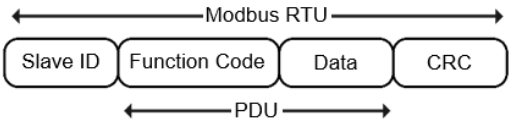


Figure 4. RTU Modbus message.

TABLE I
COMMON FUNCTION CODE IN MODBUS PROTOCOL

Function Code	Description	Function
01	Read Coils	Reads the values of multiple coils.
02	Read Discrete Inputs	Read the values of multiple discrete inputs.
03	Read Holding Registers	Reads the values of multiple holding registers.
04	Read Input Registers	Reads the values of multiple input register.
05	Write Single Coil	Writes a value to a single coil.
06	Write Single Register	Writes a value to a single register.
15	Write Multiple Coils	Writes values to multiple coils.
16	Write Multiple Registers	Write values to multiple registers.

Table I presents a comprehensive list of the common function codes within the Modbus protocol.

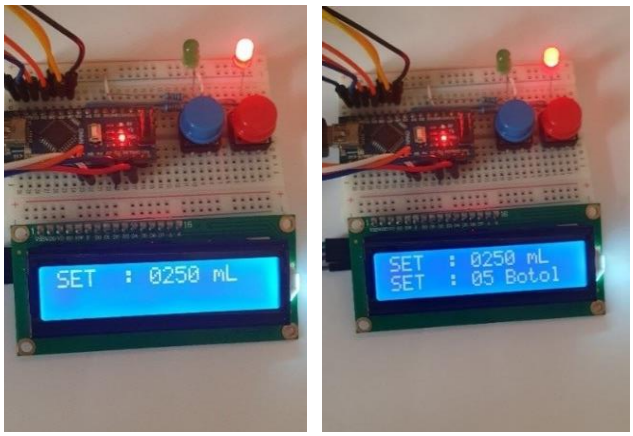
The data section in Modbus refers to information transmitted or received through the Modbus protocol. These data may include values from registers or other information transmitted between Modbus master and slave devices. The data stores the values required to execute the operation according to the function code transmitted. For example, if the function code is 03 (holding register), the data may contain information about the address of the register to be read and the number of registers to be fetched.

CRC is an error-checking method used to ensure data integrity in Modbus communication. This process involves calculating the CRC value from the data packet and including this CRC value within the packet.

Since Arduino functions as a slave, it responds to messages requested by LabVIEW, which operates as the master. The coil reading data, volume measurement, and bottle count are stored in variables. The CR variable is a Boolean data type representing two conditions: a value of 0, which signifies a STOP command, and a value of 1, which indicates a START command. To access and read this data, function code 01 is applied. Meanwhile, the volume and bottle count variables are of integer data type. The volume variable ranges from 0 to 1,000 ml, while the bottle count variable ranges from 0 to 10. Although these ranges can be flexibly set as needed, in this study, both variables are constrained within the predefined range.

To store data from the volume and bottle variables, an array data structure is utilized as a register. With this implementation, the data contained in the array can be accessed and read using function code 03.

Subsequently, LabVIEW sends a request to the slave (Arduino) to access and read the data stored in the register. To read coil data, the read coil block is used with an initial address of 0 (00001), while to read volume measurement and bottle count data, the Read Holding Registers block is employed with



(a) (b)

Figure 5. Process of inputting values, (a) volume measurement, (b) number of bottles.

an initial address of 0 (40001 for volume measurement, 40002 for bottle count).

At the initial stage of programming, supporting libraries and variables for storing volume measurement and bottle count values are declared. When the user presses the '*' symbol on the keypad, the selection switches to the SET option to configure the volume measurement and bottle count values. Subsequently, the selection returns to VIEW mode. From the VIEW mode, the user can press the START or STOP button to send data to LabVIEW.

The Arduino microcontroller programming flow begins with the declaration of supporting libraries and variables used to store volume measurement and bottle count values. Initially, the system is in VIEW mode as the default mode. In this mode, the user can monitor the previously configured volume measurement and bottle count values.

If the user presses the '*' symbol on the keypad, the system switches to SET mode. In this mode, the user can input new values for liquid volume measurement (in ml) and the desired number of bottles. Once these values are saved, the system automatically returns to VIEW mode.

From VIEW mode, the user has the option to start or stop system operation. Pressing the START button on the push button will send a coil value of 1 (ON) to the LabVIEW software via the Modbus RTU communication protocol to initiate the liquid-filling process. Conversely, pressing the STOP button will send a coil value of 0 (OFF) to halt the ongoing process. This flow ensures that the user has full control over the system configuration and operation with a simple and structured interface.

III. DESIGN AND SIMULATION RESULTS

The simulation of the automatic liquid-filling machine has been designed in LabVIEW, utilizing hardware components such as the Arduino microcontroller, push button, LED, 16 × 2 LCD, and a 3 × 4 keypad. The first step was assembling the aforementioned components and hardware. Then, the program was uploaded to the Arduino ATMega 328P microcontroller. After uploading the program, the initial display showed the VIEW menu, which presented the volume measurement and bottle count values. When the "*" symbol was pressed on the keypad, the menu switched to the SET menu. In the SET menu, the user could input the volume measurement within the range of 0 – 1,000 ml. After this, the "*" symbol was pressed to save the volume and then display the bottle count or total number of

Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 1 February 2025

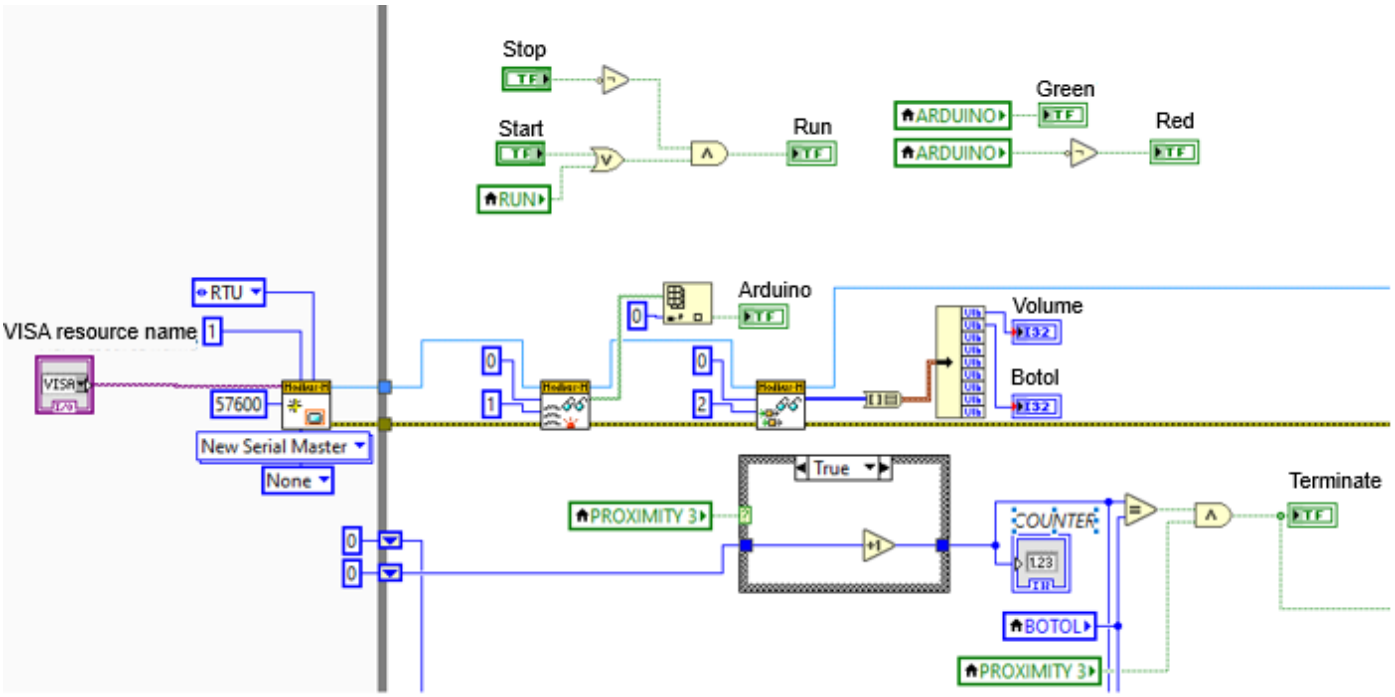
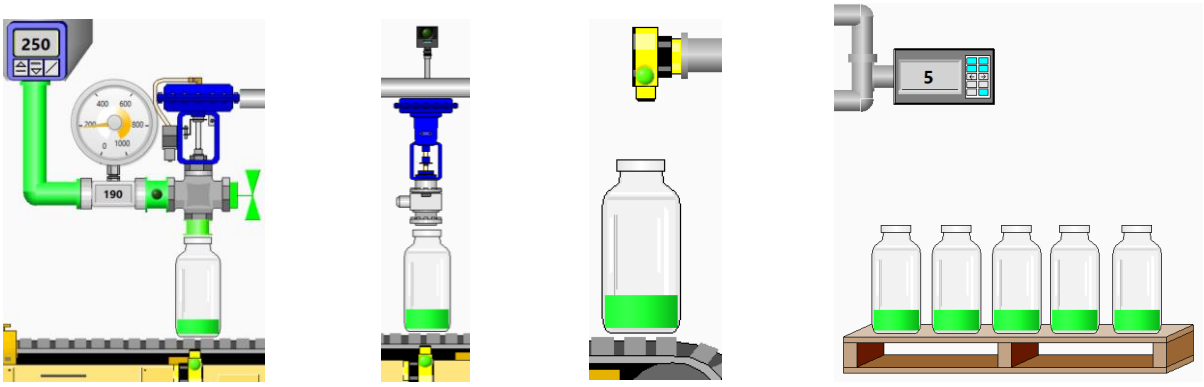


Figure 6. Modbus block diagram in LabVIEW.



(a) (b) (c) (d)

Figure 7. Process simulation results on the LabVIEW front panel, (a) filling, (b) capping, (c) counting, (d) number of bottles.

bottles. Next, the '*' symbol was pressed again to save the input data, and the display returned to the VIEW menu.

The process of entering volume measurement and bottle count values using the keypad is illustrated in Figure 5. Three variables are transmitted via Arduino serial communication: the CR variable, which holds the state 0 or 1 from the push button; the volume variable 0 – 1,000 ml; and the bottle count variable 0 – 10 bottles. However, in this study, the number of bottles is limited to five.

Data reception by LabVIEW was conducted via Modbus RTU and Virtual Instrument Software Architecture (VISA) in LabVIEW. The automatic liquid-filling machine program employed a block diagram in LabVIEW. Each block was interconnected, forming a data flow consistent with ladder logic. Figure 6 only displays a partial screenshot of the entire block diagram.

Modbus Create Master Instance is a function or block in LabVIEW used to instantiate a Modbus master. This instantiation can be either a serial or TCP Modbus master. The parameters used include the type of serial connection (such as RTU or ASCII), the Modbus slave device address, the VISA resource name for serial communication, the data transfer rate (baud rate), and the error-checking method (parity) to ensure data integrity.

Read Coils is a block used to read the status of multiple coils (relays or switches) in the Modbus slave device. In this study, this block was used to read the value of the CR variable. The Modbus master can obtain information about the relay or switch status on the Modbus slave device by configuring parameters such as the starting address of the coil to be read, the number of coils to be read, and the values representing the on/off status of the coils that had been read.

The Index Array function was used to read values based on array indices. Read Holding Registers were utilized to read values from holding registers in the Modbus slave device. Holding registers serve as memory locations that store data accessible and modifiable by the Modbus master. The parameters include the starting address of the holding register, the number of holding registers to be read, and the register values. The output from this block is typically an array or a set of values, where each array element contains a value from one holding register. The Array to Cluster function converted a 1D array into a group of elements of the same type as the array elements, while the Unbundle function was used to separate the

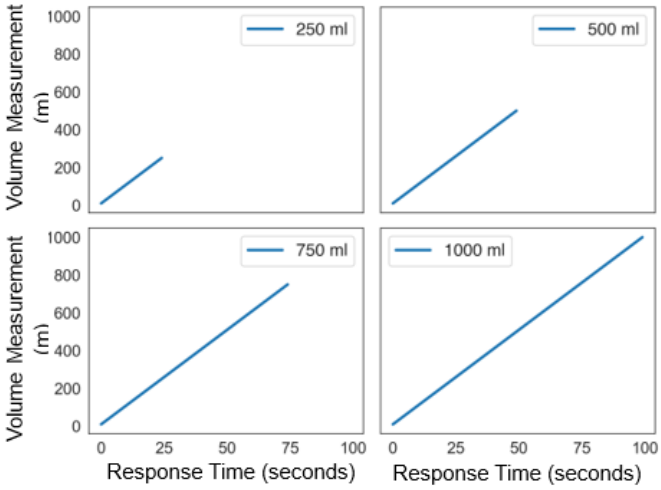


Figure 8. Response time to volume measurement.

cluster into individual elements. These elements (such as volume and bottle count) served as set points. Logical functions such as AND, OR, and NOT are represented in the control logic shown in Figure 2 in the block diagram.

The Case Structure was used to control program execution flow based on specific conditions. In this context, a subdiagram was utilized to operate the bottles on the conveyor belt.

Figure 7 illustrates the test results on the LabVIEW front panel with a volume measurement of 250 ml. When the START button is pressed, the Arduino sends a command signal (1) to activate the green LED, followed by initiating the conveyor belt. The bottles are arranged and positioned on the conveyor belt, which moves towards the filling machine. When proximity sensor 1 detects the presence of a bottle, the conveyor belt stops. The bottle-filling machine then operates by dispensing liquid into the bottle automatically at a rate of 10 ml/s. The valve opens, allowing the liquid to be poured into the bottle through a nozzle connected to the liquid tank. The volume of the dispensed liquid corresponds to the predetermined measurement. To determine the response time required to fill a volume of 250 ml, (2) is used:

$$time = \frac{volume}{flow\ rate} = \frac{250\ ml}{10\ ml/s} = 25\ s. \quad (2)$$

After the bottle had been filled according to the specified volume, the conveyor belt resumed movement toward the capping machine. When proximity sensor 2 detected the presence of a bottle, the capping machine operated by placing a cap on the bottleneck and pressing it down automatically until it was securely sealed. Subsequently, the conveyor belt continued moving toward the bottle-counting process.

Bottles passing through a proximity sensor 3 were counted. This sensor was programmed to count the number of bottles that have been filled and sealed. The system halted once the desired number of bottles was reached, as defined by the data previously set and transmitted by the Arduino.

Figure 8 illustrates the time required for the simulation to fill bottles with four different volume measurements. The testing was conducted with input volume measurements of 250 ml, 500 ml, 750 ml, and 1,000 ml, with a total of five bottles. The set points on the LabVIEW front panel accurately correspond to the predefined volume measurements and bottle counts entered via the keypad. The results displayed on the LabVIEW front panel do not specify the type of liquid used during the bottle-filling process. The developed control system visualizes the input data determination process for volume measurements and bottle counts via the keypad, as well as the control process for the conveyor belt, filling, capping, and counting on the LabVIEW front panel.

IV. CONCLUSION

This study has demonstrated that the integration of LabVIEW and microcontrollers can effectively simulate an automatic liquid-filling machine system. The system provides precise control over volume measurement and bottle count, ensuring that each container is filled according to the specified inputs of 250 ml, 500 ml, 750 ml, and 1,000 ml. The use of LabVIEW as a plant design tool resembling industrial processes, along with seamless integration with the Arduino microcontroller, has been successfully implemented. Communication between LabVIEW and Arduino in this study was conducted using the Modbus RTU protocol, which ensures stable and reliable data exchange, guaranteeing accurate and efficient communication. This highlights the potential of this system as a supporting tool for control system instrumentation practices. Further development and system optimization could lead to broader adoption in various industrial plant applications.

CONFLICTS OF INTEREST

The authors declare that there is no conflict of interest in the research and preparation of this paper.

AUTHORS' CONTRIBUTION

Conceptualization, Syafriyadi Nor and Zaiyan Ahyadi; methodology, Syafriyadi Nor; software, Syafriyadi Nor; validation, Syafriyadi Nor and Zaiyan Ahyadi; formal analysis, Syafriyadi Nor; investigation, Syafriyadi Nor and Zaiyan Ahyadi; resources, Syafriyadi Nor and Zaiyan Ahyadi; data curation, Syafriyadi Nor and Zaiyan Ahyadi; writing-original drafting, Syafriyadi Nor; writing-reviewing and editing, Zaiyan Ahyadi; visualization, Syafriyadi Nor and Zaiyan Ahyadi.

ACKNOWLEDGMENT

The authors would like to express their gratitude to the Microprocessor Laboratory of Politeknik Negeri Banjarmasin for providing the necessary facilities and support for this research.

REFERENCES

[1] S.K. Das *et al.,* "Design, development and FEA analysis of multi-sized bottle filling system," in *2023 Int. Conf. Integr. Comput. Intell. Syst. (ICICIS)*, 2023, pp. 1–5, doi: 10.1109/ICICIS56802.2023.10430283.

[2] R.L.W. Koggalage, A.G.M.I.S. Wijesinghe, H.P.S.S. Caldera, and R.R. Samarawickrama, "Design and implementation of an automated multi-purpose filling and capping machine," in *2021 From Innov. To Impact (FITI)*, 2021, pp. 1–5, doi: 10.1109/FITI54902.2021.9833035.

[3] M.M. Khan *et al.*, "Simulation of PLC ladder logic programming for an automated glass bottle molding and refilling plant," in *4th Smart Cities Symp. (SCS 2021)*, 2021, pp. 114–119, doi: 10.1049/icp.2022.0324.

[4] M.F. Rahaman, S. Bari, and D. Veale, "Flow investigation of the product fill valve of filling machine for packaging liquid products," *J. Food Eng.*, vol. 85, no. 2, pp. 252–258, Mar. 2008, doi: 10.1016/j.jfoodeng.2007.07.020.

[5] K.S. Kiangala and Z. Wang, "An Industry 4.0 approach to develop auto parameter configuration of a bottling process in a small to medium scale industry using PLC and SCADA," in *2nd Int. Conf. Sustain. Mater. Process. Manuf. (SMPM 2019)*, 2019, pp. 725–730, doi: 10.1016/j.promfg.2019.06.015.

[6] G. Selvaraj, R. Karthikeyan, S. Brindha, and K. Kumar S, "Low cost assorted sized bottles automated liquid filling system using SCADA," in *2023 Intell. Comput. Control Eng. Bus. Syst. (ICCEBS)*, 2023, pp. 1–4, doi: 10.1109/ICCEBS58601.2023.10448914.

[7] A. Mahrez *et al.*, "Design a PLC-based automated and controlled liquid filling-capping system," in *2022 Int. Eng. Conf. Elect. Energy Artif. Intell. (EICEEAI)*, 2022, pp. 1–5, doi: 10.1109/EICEEAI56378.2022.10050478.

[8] C. Elliott, V. Vijayakumar, W. Zink, and R. Hansen, "National instruments LabVIEW: A programming environment for laboratory automation and measurement," *J. Lab. Autom.*, vol. 12, no. 1, pp. 17–24, Feb. 2007, doi: 10.1016/j.jala.2006.07.012.

[9] Arduino. "What is Arduino? | Arduino." Access date: 20-Feb-2023. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[10] M.L. Ahmed, S. Kundu, and M. Rafiquzzaman, "Automatic bottle filling system using PLC based controller," *J. Adv. Mech.*, vol. 4, no. 1, pp. 17–24, Mar. 2019.

[11] R. Sureshkumar *et al.*, "IoT based bottle filling system using PLC," in *2023 Int. Conf. Energy Mater. Commun. Eng. (ICEMCE)*, 2023, pp. 1–5, doi: 10.1109/ICEMCE57940.2023.10433948.

[12] A. Kumar M and H.P. Kumar, "Automatic bottle filling system using PLC," *Int. J. Trend Sci. Res. Dev. (IJTSRD)*, vol. 2, no. 1, pp. 361–364, Nov./Dec. 2017, doi: 10.31142/ijtsrd5953.

[13] G.A. Laksmana, P. Santoso, and F. Pasila, "Aplikasi untuk memonitor PLC pada mesin filling dan capping," *J. Tek. Elekt.*, vol. 10, no. 2, pp. 48–53, Sep. 2017, doi: 10.9744/jte.10.2.48-53.

[14] D. Patil, "Automatic bottle filling, capping and labelling system using PLC based controller," *Ilkogretim*, vol. 20, no. 1, pp. 5750–5761, 2021, doi: 10.17051/ilkonline.2021.01.604.

[15] O.I. Abdullah, W.T. Abbood, and H.K. Hussein, "Development of automated liquid filling system based on the interactive design approach," *FME Trans.*, vol. 48, no. 4, pp. 938–945, Aug. 2020, doi: 10.5937/fme2004938A.

[16] M. Aria *et al.*, "Virtual simulation system with various examples and analysis tools for programmable logic controller training," in *3rd Int. Conf. Inform. Eng. Sci. Technol. (INCITEST 2020)*, 2020, pp. 1–7, doi: 10.1088/1757-899X/879/1/012108.

[17] A. El Hammoumi *et al.*, "Real-time virtual instrumentation of Arduino and LabVIEW based PV panel characteristics," in *Int. Conf. Renew. Energies Energy Effic. (REEE'2017)*, 2018, pp. 1–11, doi: 10.1088/1755-1315/161/1/012019.

[18] R.M. Shrenika *et al.*, "Non-contact water level monitoring system implemented using LabVIEW and Arduino," in *2017 Int. Conf. Recent Adv. Electron. Commun. Technol. (ICRAECT)*, 2017, pp. 306–309, doi: 10.1109/ICRAECT.2017.51.

[19] Y.K. Taru and A. Karwankar, "Water monitoring system using Arduino with LabVIEW," in *2017 Int. Conf. Comput. Methodol. Commun. (ICCMC)*, 2017, pp. 416–419, doi: 10.1109/ICCMC.2017.8282722.

[20] M. Gharte, "Automation of soap windscreen washer filling machine with PLC and LabVIEW," in *Int. Conf. Autom. Control Dyn. Optim. Tech. (ICACDOT)*, 2016, pp. 469–472, doi: 10.1109/ICACDOT.2016.7877630.

[21] Arduino. "Arduino ® Nano Arduino ® Nano Features." Access date: 6-Feb-2024. [Online]. Available: https://docs.arduino.cc/hardware/nano/?_gl=1*pj02od*_up*MQ..*_ga*MTA0NDA1MDQ1Ni4xNzQyMzUzMDcy*_ga_NEXN8H46L5*MTc0MjM1MzA2OS4xLjAuMTc0MjM1MzA2OS4wLjAuMTg0Mjc5MDA0Mg..#tech-specs

[22] *LabVIEW*, National Instruments, Austin, TX, USA, 1998.

[23] C.M. Bulliner. "ModbusRTUSlave." Access date: 6-Feb-2024. [Online]. Available: https://github.com/CMB27/ModbusRTUSlave