REPORT ON MECHATRONICS SYSTEM INTEGRATION

5-PLC INTERFACING
GROUP 7

SECTION 2, SEMESTER 1, 24/25

TEAM MEMBERS

NAME	MATRIC NO.
MUHAMMAD ZAMIR FIKRI BIN MOHD ZAMRI	2212515
MUHD AKMAL HAKIM BIN SAIFUDDIN	2216093
NUR SHADATUL BALQISH BINTI SAHRUNIZAM	2212064
NORHEZRY HAKIMIE BIN NOOR FAHMY	2110061
NUR AMIRA NAZIRA BINTI MOHD NASIR	2110026

DATE OF EXPERIMENT: 6 NOVEMBER 2024
DATE OF SUBMISSION:12 NOVEMBER 2024

ABSTRACT

Programme Logic Controller (PLC) is a specialized industrial computer used for controlling and automating electromechanical processes in various applications. While microcontrollers serve as versatile, low-cost solutions for embedded systems applications.This experiment's purpose is to understand both software and hardware aspects of PLC interfacing with microcontrollers. Build a Start-Stop Control Circuit and develop it by using PLC. After that, compile, simulate and transfer the ladder diagram to the arduino board. After uploading it, the LED should be light on when the start button was pushed and light off when the stop button was pushed. This experiment combined the advantages of PLC and microcontroller and how both of them work together.

TABLE OF CONTENT

INTRODUCTION

This experiment explores the interfacing between a PLC and a microcontroller by using a Start-Stop control circuit. The Start-Stop control is one of the most fundamental control circuits used in most industrial applications. It allows the operator to initiate or stop a process based on simple input commands, typically using push buttons or switches. By interfacing a PLC with a microcontroller, we can demonstrate how these two systems can collaborate to control and monitor the start-stop sequence for an industrial process or motor. We can make an observation on both hardware and software. In hardware, we can observe the serial communication happen between the programmed and arduino and wired to perform control tasks. While in software, we focus on programming the microcontroller to handle control logic, process input signals, and respond to commands from the PLC while ensuring proper coordination and synchronization.

MATERIALS AND EQUIPMENT

- OpenPLC Editor Software
- Arduino board
- 2 push button switch
- Jumper wire
- LED
- Resistor
- Breadboard

EXPERIMENTAL SETUP

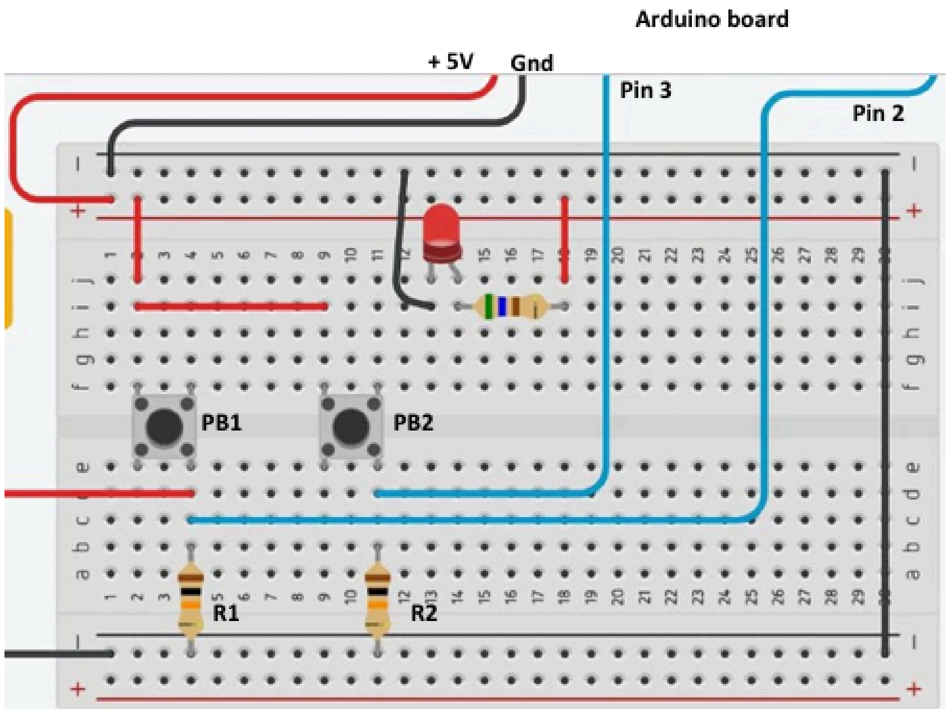


Fig. 6

METHODOLOGY

1. Create the ladder diagram shown in Fig. 5.
2. Specify all variables used in the ladder diagram.
3. Compile and simulate the ladder diagram in OpenPLC Editor.
4. Upload the ladder diagram to the Arduino board.
5. Ensure to select correct COM port number and all pin association between the OpenPLC variables and Arduino board.
6. Build the circuit as shown in Fig. 6.
7. Test the functionality.

DATA COLLECTION

To analyze the interfacing between a PLC (using OpenPLC software) and a microcontroller, evaluating data exchanges and the performance of control signals in real-time.

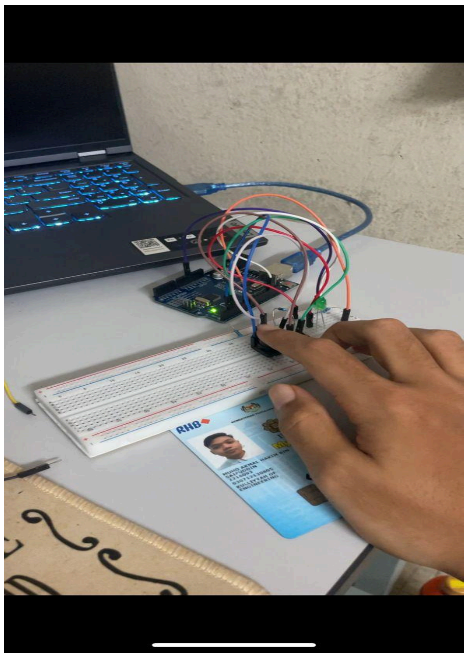
Instruments Used

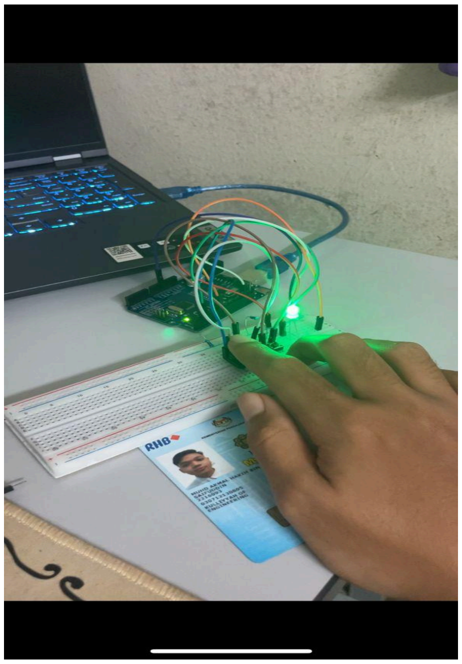
1) OpenPLC Editor
    a) Used to design and simulate PLC logic.
2) Microcontroller (e.g., Arduino or ESP32)
    a) Receives control signals from the PLC, triggering outputs such as LED indicators or motor movements.
3) Input/Output Modules
    a) Channels for the microcontroller to send and receive signals from external devices.
4) Power Supply Module
    a) Provides stable power to the microcontroller and connected components.

Data Acquisition Setup

1) Setup OpenPLC Editor to simulate a basic control system (e.g., a start-stop circuit or timer-based process).
2) Connect the microcontroller to the PLC via digital I/O pins, allowing the PLC program to control LED indicators on the microcontroller.
3) Vary control inputs to evaluate system responsiveness.

RESULT





DISCUSSION

1. Implications
   - Skill Development: The exercises provide foundational skills in PLC programming and interfacing with hardware, which are valuable for roles in automation, manufacturing, and system control.
   - Educational Accessibility: Using OpenPLC and Arduino for PLC simulation democratizes access to industrial automation concepts, as these tools are affordable and widely available.
   - Industrial Relevance: The principles learned here—such as ladder logic, timing, and start-stop mechanisms—are directly applicable in industrial automation settings, making this knowledge highly transferable to real-world scenarios.

2. Discrepancies between expected and observed outcomes

   <u>Expected vs. Observed Timing in LED Blinking</u>

   - Expected: The LED should blink at the specified intervals set in the ladder logic timer block.
   - Observed: If the LED blink rate was slower or faster than expected, this could indicate a timing mismatch or a processing delay on the Arduino.
   - Possible Causes: Delays may arise if the microcontroller has limited processing power or if other processes interfere with timing.
   - Solution: Adjust the timer values, check for code delays, or troubleshoot potential conflicts with other components on the Arduino board.

   <u>Start-Stop Button Response</u>

   - Expected: The Start button should activate the LED, and the Stop button should turn it off instantly.
   - Observed: If there was a delay, intermittent response, or the LED did not turn on/off as expected, it might indicate issues with button wiring, debounce, or pin assignment.
   - Possible Causes: Loose connections, insufficient pull-up/pull-down resistors, or physical button bounce can cause signal inconsistencies.
   - Solution: Verify the circuit connections and consider software debounce logic to stabilize button inputs.

   <u>COM Port Recognition and Program Uploading</u>

   - Expected: OpenPLC should recognize the Arduino on the correct COM port and upload the ladder program without error.
   - Observed: If there were upload errors, mismatches in the COM port, or issues in transferring the code, the program may not run as intended.

- Possible Causes: COM port misidentification or driver conflicts are common with USB-based connections.
- Solution: Ensure the correct COM port is selected and that drivers are updated for Arduino. Reconnect or restart if needed.

Unexpected Behavior in the Circuit

- Expected: The circuit should follow the programmed logic strictly, with no unexpected LED behavior.
- Observed: If the LED turned on/off unexpectedly or behaved erratically, it could indicate issues with code logic, hardware connections, or variable assignment in OpenPLC.
- Possible Causes: Misconfigurations in the ladder logic or incorrect mapping of variables to Arduino pins may lead to unpredictable behavior.
- Solution: Recheck the ladder logic diagram, ensure correct variable associations, and inspect connections.

3. Sources of error or limitations of the experiment

Hardware Limitations

- Arduino Processing and Timing Constraints: Arduinos are less powerful than industrial-grade PLCs, which may cause timing inconsistencies, especially in tasks requiring precise intervals. For example, an Arduino’s processor might struggle with exact millisecond timing, leading to slightly variable LED blinking intervals.
- Wiring and Connections: Breadboard and jumper wire connections can introduce noise or loose connections, leading to intermittent behavior in button response or LED operation. This setup differs from more robust, soldered connections used in industrial settings, potentially causing signal instability.

Software and Communication Errors

- COM Port Issues: Connection errors or port conflicts may occur, affecting the consistency of the Arduino’s communication with OpenPLC. This can delay or disrupt code uploading and prevent consistent operation of the Start-Stop control circuit.
- Button Debounce Issues: Mechanical buttons often exhibit a "bouncing" effect, where a single press registers multiple signals. Without software debouncing, this effect may cause erratic behavior, making the system seem less responsive or misinterpreting a single press as multiple inputs.

RECOMMENDATION

An extra LED or a tiny LCD display are examples of feedback indicators that may be added to this project to make it nicer. The Start-Stop circuit might be turned on or off, for example, by using an extra LED. As an alternative, you could use an LCD to show messages like "System Running" or "System Stopped." This extra information facilitates the visualization of circuit activity and can be helpful in promptly identifying and resolving problems. Given that feedback is crucial in actual control systems, providing this functionality also gives the project an authentic appearance.

The addition of a safety feature, such an emergency stop button, is another significant enhancement. For safety purposes, this could involve a push button that, when pressed, instantly stops the LED from flashing and simulates a shutdown. These kinds of safety measures are commonplace in industrial settings to safeguard the workers and the machinery. In addition to making the project more comprehensive and realistic, including an emergency stop mechanism provides an extremely important safety lesson.

Additionally, we also may improve the project by experimenting with the many timer types that are available in the OpenPLC program, such as off-delay and on-delay timers. An off-delay timer might keep the LED on for a particular amount of time after pushing the Stop button, but an on-delay timer would only cause the LED to flash a few seconds after pressing the Start button. Implementing these various timers illustrates how PLCs can manage more intricate timing needs in automated systems while helping in your understanding of timing controls.

Lastly, consider using many LEDs to create a sequential control system if you want to make the project more difficult. For example, you might program two LEDs to go on and off in a predetermined order when the Start button is pushed. This would demonstrate how multi-step processes may be handled by PLCs, which is helpful in scenarios such as manufacturing lines. Adding extra stages to the project prepares you for more complex automation positions by teaching you how to control multiple outputs in an ordered manner.

APPENDICES

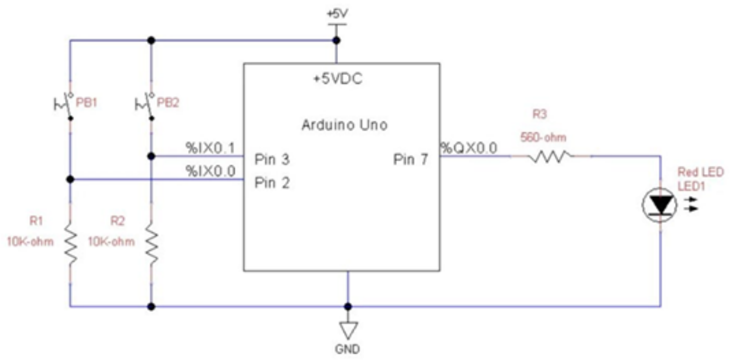
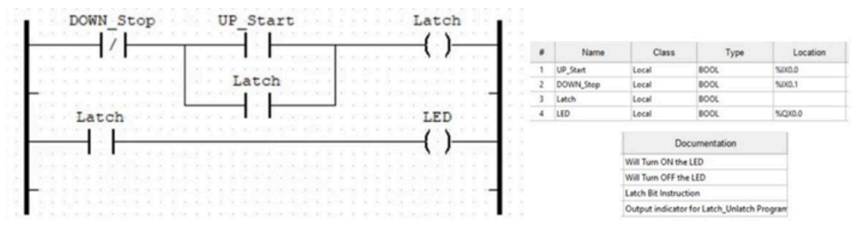


Fig. 4: Start-Stop Control Circuit



#	Name	Class	Type	Location
1	UP_Start	Local	BOOL	%IX0.0
2	DOWN_Stop	Local	BOOL	%IX0.1
3	Latch	Local	BOOL	
4	LED	Local	BOOL	%QX0.0

Documentation
Will Turn ON the LED
Will Turn OFF the LED
Latch Bit Instruction
Output indicator for Latch_Unlatch Program

Fig. 5: Ladder Diagram for the Start-Stop Control Circuit

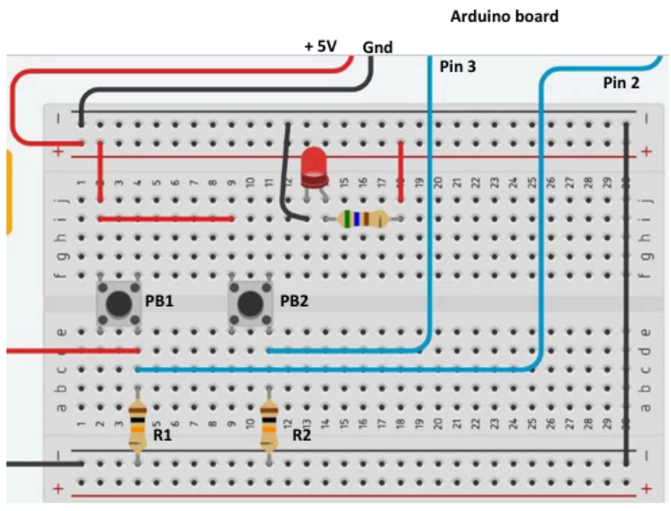


Fig. 6

STUDENT DECLARATION
<u>Certificate of Originality and Authenticity</u>

This is to certify that we are responsible for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us.**

Name : Muhammad Zamir Fikri Bin Mohd Zamri	**Read**	/
Matric No. : 2212515	**Understand**	/
[illegible] **Signatures :**	**Agree**	/

Name : Muhd Akmal Hakim Bin Saifuddin	**Read**	/
Matric No. : 2216093	**Understand**	/
Signatures : *akmal*	**Agree**	/

Name : Nur Shadatul Balqish Binti Sahrunizam	**Read**	/
Matric No. : 2212064	**Understand**	/
Signatures : shadatul	**Agree**	/

Name :NORHEZRY HAKIMIE BIN NOOR FAHMY	**Read**	/
Matric No. :2110061	**Understand**	/
Signatures :hezry	**Agree**	/

Name : Nur Amira Nazira Binti Mohd Nasir	**Read**	/
Matric No. :2110026	**Understand**	/
Signatures :amira	**Agree**	/