FEAT JOURNAL
FARM ENGINEERING AND AUTOMATION TECHNOLOGY JOURNAL
วารสารวิศวกรรมฟาร์มและเทคโนโลยีการควบคุมอัตโนมัติ

แนวทางการพัฒนาโปรแกรมระบบอัตโนมัติในการควบคุมแบบซีเควนซ์ สำหรับระบบอัตโนมัติราคาประหยัด

Programming Guidelines for Sequential Automation Tasks in Low-Cost Systems

ชัชชล เปรมชัยสวัสดิ์[1)*] ณรงค์ บุญเสนอ[1)] กฤช ตราชู[1)] ชาญชัย เหลาหา[1)] และ พิจิตร บัวระภา[1)]

Shutchon Premchaisawatt[1)*] Narong Boonsaner[1)] Krit Tashoo[1)] Chanchai Laoha[1)] and Pijit Buarapha[1)]

[1)] สาขาวิชาเทคโนโลยีอุตสาหการ คณะครุศาสตร์อุตสาหกรรม มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน วิทยาเขตขอนแก่น

[1)] Department of Industrial Technology, Faculty of Industrial Education,

Rajamangala University of Technology Isan Khon Kaen Campus

Received: 26 April 2023

Revised: 16 June 2024

Accepted: 17 June 2024

Available online: 26 June 2024

บทคัดย่อ

งานวิจัยนี้นำเสนอแนวทางการเขียนโปรแกรมสำหรับควบคุมระบบอัตโนมัติที่ออกแบบตามแนวคิด แผนภูมิสถานะเครื่องจักร ซึ่งจะช่วยในการพัฒนาซอฟต์แวร์อย่างมีประสิทธิภาพตามหลักการวงจรการพัฒนาซอฟต์แวร์ และลดปัญหาความไม่สอดคล้องกันในการเขียนโปรแกรม ปรับปรุงความเข้าใจโค้ดโดยไม่คำนึงถึงภาษาโปรแกรม การทดลองแบ่งออกเป็นสองกรณีพิจารณควบคู่กัน ในกรณีแรก ภาษา C++ ใช้ในการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ ในกรณีที่สอง ภาษาแลดเดอร์ใช้ในการเขียนโปรแกรมบน PLC ซึ่งจะพัฒนาซอฟต์แวร์ตามข้อกำหนดของการแข่งขันสำหรับชุดฝึกอบรมของระบบการเจาะอัตโนมัติ MPU-A ผลการทดสอบแสดงให้เห็นว่าการปฏิบัติตามแนวทางใหม่นี้ส่งผลให้คอนโทรลเลอร์ทั้งสองทำงานตามข้อกำหนดของ ระบบ โดยมีรูปแบบการเขียนโปรแกรมร่วมกัน อย่างไรก็ตาม เวลาตอบสนองอาจแตกต่างกันไปขึ้นอยู่กับฮาร์ดแวร์ของตัวควบคุม

คำสำคัญ: การเขียนโปรแกรมระบบอัตโนมัติ แผนภูมิสถานะเครื่องจักร ไมโครคอนโทรเลอร์ พีแอลซี

Abstract

This research presents programming guidelines for controlling automated systems designed based on the state machine diagram concept. This will help to develop software efficiently within the software development cycle and reduce the problem of inconsistency in programming, improving code understanding regardless of the programming language. The experiments are divided into two cases. The first part uses the C++ language to write programs on a microcontroller. The second part uses the ladder language to write programs on a PLC. This will develop software according to the requirements of the competition for the training set of the MPU-A automatic drilling system. The test results show that following this new approach results in both controllers working properly according to the system requirements, with a common programming style. However, response times may vary depending on the hardware of controllers.

Keywords: Automatic system programming: State machine diagram: Microcontroller: PLC

*ติดต่อ: E-mail: shutchon.pr@rmuti.ac.th,

1. **บทนำ**

ระบบอัตโนมัติเป็นเทคโนโลยีที่มีความสำคัญสูงในปัจจุบันเนื่องจากมีผลกระทบต่อการทำงานและชีวิตของมนุษย์สูง [1] ตัวอย่างเช่นในด้านการเพิ่มประสิทธิภาพการทำงาน ระบบอัตโนมัติช่วยลดเวลาที่ใช้ในการทำงานและเพิ่มประสิทธิภาพของการทำงานโดยที่ไม่ต้องใช้งานมนุษย์ในการดำเนินการทำให้สามารถทำงานได้เร็วขึ้นและลดความผิดพลาดได้ ในด้านการลดความเสี่ยงและความเป็นอันตราย ระบบอัตโนมัติสามารถทำงานในสภาวะที่อันตรายหรือไม่เหมาะสมสำหรับมนุษย์ เช่น ในการทำงานในสภาวะอันตรายในอุตสาหกรรมหรือการตรวจสอบสิ่งที่อยู่ในสภาพแวดล้อมที่อันตราย ในด้านความแม่นยำ ระบบอัตโนมัติมีความแม่นยำในการทำงานโดยไม่ได้รับผลกระทบจากความเหนื่อยล้าหรือความผิดพลาดของมนุษย์ ทั้งหมดนี้เป็นประโยชน์ของระบบอัตโนมัติในปัจจุบัน และมีแนวโน้มที่จะนำไปผสมกับเทคโนโลยีอื่นมากขึ้น [2] เช่น ปัญญาประดิษฐ์ ระบบคลาวด์ เป็นต้น

ระบบอัตโนมัติจึงถูกใช้ในงานอุตสาหกรรมอย่างกว้างขวาง แต่ในอดีตการเรียนรู้เรื่องการควบคุมระบบอัตโนมัติทำได้ยากเนื่องจากฮาร์ดแวร์มีราคาสูง ทำให้การเรียนการสอนเรื่องระบบอัตโนมัติอยู่ในวงที่จำกัด [3] ปัจจุบันฮาร์ดแวร์มีราคาที่ถูกลงและมีอุปกรณ์ทางเลือกมากขึ้น เช่นไมโครคอนโทรลเลอร์ทำให้มีการนำระบบอัตโนมัติมาใช้ในงานส่วนบุคคล เช่น การทำฟาร์ม [4][5] การอำนวยความสะดวกในบ้าน [6] เครื่องจักรอำนวยความสะดวก [7] เป็นต้น แต่ในการจะใช้งานระบบอัตโนมัติต้องมีกฎหรือสิ่งที่จะทำให้เครื่องจักรทำงานด้วยตนเอง ซึ่งก็คือการเขียนโปรแกรมควบคุมระบบอัตโนมัติ [8] หนึ่งในรูปแบบการควบคุมระบบอัตโนมัติที่นิยมคือการเขียนโปรแกรมการควบคุมแบบซีเควนซ์ [9] ในอุปกรณ์ระดับสูงมีเครื่องมือในการเขียนโปรแกรมให้

เป็นระเบียบเช่น ภาษา SFC แต่ในอุปกรณ์ระดับเริ่มต้นหรืออุปกรณ์เช่นไมโครคอนโทรลเลอร์ อุปกรณ์เหล่านี้การเขียนโปรแกรมในระบบอัตโนมัติยังไม่มีแนวทางที่แน่นอน [8][9] โดยจะพึ่งพิงกับประสบการณ์ของผู้เขียน ทำให้ตรวจสอบและแก้ไขโดยผู้อื่นได้ยากเวลาเปลี่ยนผู้ดูแล [1][8]

บทความวิจัยนี้นำเสนอรูปแบบการพัฒนาโปรแกรมตามแนวทาง SDLC และออกแบบการควบคุมตามหลักการแผนภูมิสถานะที่จะทำให้ดูแลและแก้ไขโปรแกรมแบบซีเควนซ์ในระบบอัตโนมัติสะดวกมากขึ้นในอุปกรณ์ราคาประหยัด โดยจะใช้ PLC ราคาประหยัด และไมโครคอนโทรลเลอร์ทดสอบกับอุปกรณ์ที่ใช้ในการเรียนรู้เรื่องการเขียนโปรแกรมระบบอัตโนมัติ ซึ่งทั้งรูปแบบการออกแบบและการเขียนโปรแกรมจะกล่าวถึงในหัวข้อถัดไป

2. ทฤษฎีที่เกี่ยวข้อง

การควบคุมแบบซีเควนซ์ (Sequential control) หมายถึงการใช้ตัวควบคุมการทำงานของเครื่องจักร ในที่นี้คือ PLC เพื่อควบคุมกระบวนการหรือเครื่องจักรตามลำดับขั้นตอนการทำงาน โดยอาศัยการโปรแกรม เพื่อควบคุมการทำงานของอุปกรณ์ต่างๆในกระบวนการนั้นๆ โดยกำหนดลำดับของการทำงานแต่ละอุปกรณ์จนครบกระบวนการทำงาน (Process) โดยหนึ่งในเครื่องมือที่ช่วยออกแบบลำดับขั้นตอนคือแผนภูมิสถานะเครื่องจักร (State machine diagram) ซึ่งจะช่วยบอกว่ามีกี่สถานะ มีลำดับสถานะอย่างไร เงื่อนไขการเปลี่ยนสถานะ แต่ละสถานะมีอุปกรณ์ใดทำงานได้บ้าง ซึ่งจะช่วยให้กระบวนการทำงานเป็นไปได้อย่างมีประสิทธิภาพและมีความน่าเชื่อถือในการทำงาน เมื่อออกแบบแผนภูมิสถานะเสร็จ การที่จะเขียนโปรแกรมลงพีเอลซีเพื่อสั่งการสามารถทำได้ด้วยการเขียนโปรแกรมด้วยภาษาซีเควนซ์เชียลฟังก์ชันชาร์ท (Sequential function chart) หรือเรียกอย่างย่อว่าเอสเอฟซี (SFC) ซึ่งใช้หลักการของกราฟเพนทริ (Pentri graph) [10] ซึ่งแสดงการจัดการกระบวนการและโครงสร้างของระบบต่างๆ ในการสร้างโปรแกรม ซึ่งมีความคล้ายกับแผนภูมิสถานะเครื่องจักรที่มนุษย์สามารถอ่านและทำความเข้าใจได้ง่าย จะทำให้การเขียนโปรแกรมและการแก้ไขโปรแกรมมีความสะดวกมากขึ้น แต่อย่างไรก็ตามอุปกรณ์ราคาประหยัดเช่น PLC ระดับล่าง หรือไมโครคอนโทรลเลอร์ [11] ไม่สามารถเขียนโปรแกรมในลักษณะนี้ได้เนื่องจากข้อจำกัดทางฮาร์ดแวร์ อย่างไรก็ตามการวางแผนในการพัฒนาโปรแกรมที่ดีสามารถช่วยเพิ่มประสิทธิภาพในการพัฒนาและดูแลรักษาโปรแกรมได้ [12]

ดังนั้น วิธีการที่จะนำเสนอ คือ การพัฒนาโปรแกรมเป็นขั้นตอนตามลักษณะของวงจรชีวิตการพัฒนาซอฟต์แวร์เพื่อนำมาประยุกต์ให้พีเอลซีและไมโครคอนโทรลเลอร์สามารถเขียนและแก้ไขโปรแกรมลักษณ์ซีเควนซ์ได้สะดวกมากขึ้น

2.1. วงจรชีวิตการพัฒนาซอฟต์แวร์

วงจรชีวิตการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) เรียกย่อๆว่า SDLC เป็นกระบวนการที่ใช้ในการพัฒนาและบริหารจัดการโปรแกรมหรือซอฟต์แวร์ต่างๆ โดย SDLC จะประกอบด้วยขั้นตอนต่างๆ ที่ต้องผ่านเพื่อให้เกิดซอฟต์แวร์ที่มีคุณภาพและสามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ ขั้นตอนหลักๆ ของ SDLC ประกอบด้วยการวางแผน การออกแบบ การพัฒนา

การทดสอบ การนำเสนอและการบำรุงรักษาซอฟต์แวร์ [10] ซึ่งมีอยู่หลายแบบจำลอง [12] เช่น จำลองแบบน้ำตก (Water fall) แบบจำลองแบบทรงวี (V-Shape) หรือแบบจำลองแบบอาจาย (Agile) เป็นต้น โดยในงานวิจัยใช้จะเป็นแบบจำลองแบบน้ำตก เพราะเข้าใจง่ายมีลักษณะไหลไปทางเดียวแบบขั้นน้ำตก ประกอบด้วยขั้นตอนดังต่อไปนี้

2.1.1. วิเคราะห์ความต้องการ (Requirement Analysis) - การวิเคราะห์และเข้าใจความต้องการของระบบ เพื่อกำหนดขอบเขตและคุณสมบัติของระบบที่จะพัฒนา

2.1.2. ออกแบบระบบ (System Design) – การออกแบบและวางแผนระบบที่ตอบสนองต่อความต้องการและคุณสมบัติของระบบที่จะต้องมี

2.1.3. การสร้างระบบ (Implementation) - การพัฒนาระบบโดยใช้เทคโนโลยีและเครื่องมือที่เหมาะสม รวมถึงการเขียนโปรแกรมและการสร้างส่วนติดต่อผู้ใช้ (Human machine Interface)

2.1.4. การทดสอบ (Testing) - การทดสอบระบบเพื่อตรวจสอบว่าระบบทำงานได้ตามความต้องการและคุณสมบัติที่กำหนดไว้ รวมถึงการทดสอบการทำงานร่วมกันของส่วนต่างๆของระบบ

2.1.5. การใช้งานในสภาพแวดล้อมจริง (Deployment) - การนำระบบที่พัฒนาไปใช้งานจริง รวมถึงการติดตั้ง การกำหนดค่า และการทดสอบระบบในสภาพแวดล้อมจริง

2.1.6. การบำรุงรักษา (Maintenance) - การดูแลและบำรุงรักษาระบบเพื่อให้ระบบทำงานได้อย่างมีประสิทธิภาพและประสิทธิผลตลอดเวลา รวมถึงการปรับปรุงและพัฒนาระบบเพื่อตอบสนองความต้องการเพิ่มเติม

2.2. แผนภูมิสถานะเครื่องจักร

แผนภูมิสถานะเครื่องจักร (State machine diagram) [14] เป็นส่วนหนึ่งของยูเอ็มแอล (UML) [13] ซึ่งเป็นเครื่องมือที่ใช้สำหรับออกแบบซอฟต์แวร์ เป็นแผนภาพที่บ่งบอกสถานะ (State) ทำงานต่างๆ ของเครื่องจักร จะใช้สัญลักษณ์สี่เหลี่ยมแทน State โดยมีชื่อของ State ระบุอยู่ และจะใช้เครื่องหมายลูกศรเพื่อแทนการเปลี่ยนแปลงสถานะ (Transition) โดยลากจาก State เริ่มต้นไปยัง State ที่ต้องการ บนลูกศรจะมีเงื่อนไขของ การเปลี่ยนแปลง Transition กำกับอยู่ด้วย ส่วนด้านในจะมีคำอธิบายสถานะ (State Notation) คือส่วนที่บอกว่าใน State นั้นมีอุปกรณ์ใดทำงานบ้าง แผนภูมิสถานะเป็นสิ่งที่ง่ายที่สุดที่จะอธิบายถึงการทำงานของระบบอย่างครอบคลุม

2.3. ภาษาแลดเดอร์

ภาษาแลดเดอร์ (Ladder) [15] เป็นภาษาโปรแกรมสำหรับการควบคุมอุตสาหกรรม ถูกออกแบบให้เป็นรูปแบบของสัญลักษณ์และภาพแสดงการทำงานเป็นแผนผังทางไฟฟ้า (Electrical Circuit Diagram) ซึ่งมีการเรียงลำดับการทำงานในลักษณะของเส้นสัญญาณไฟฟ้าตามลำดับการทำงานของตัวอุปกรณ์ต่างๆ เช่น การควบคุมรีเลย์ การเปิด/ปิดวงจร เพื่อบังคับให้อุปกรณ์ทำงาน ภาษาแลดเดอร์เป็นภาษาที่ได้รับความนิยมสูงที่สุดในการเขียนโปรแกรมควบคุมด้วย PLC เนื่องจาก PLC แทบทุกรุ่นในปัจจุบัน รองรับการเขียนโปรแกรมด้วยภาษาแลดเดอร์

2.4. ภาษา C++

ภาษา C และ ภาษา C++ เป็นภาษาที่ได้รับความนิยมในการเขียนควบคุมไมโครคอนโทรลเลอร์

โดยเฉพาะตระกูล Arduino เหมาะกับงานควบคุมอุปกรณ์ขนาดเล็ก เนื่องจากมีราคาไม่แพงและสามารถหาได้ง่าย ทำให้ Arduino ได้รับความนิยมอย่างสูง [4] ทำให้มีนักพัฒนาช่วยเขียนไลบรารี่ให้เลือกใช้งานจำนวนมาก แต่อย่างไรก็ตาม การเขียนโปรแกรมใน Arduino ไม่สามารถเขียนฟังก์ชันบางอย่างได้ เช่น STL library และ Exception เป็นต้น ทำให้การตรวจสอบโปรแกรมทำได้ยากขึ้น [16]

ในการเขียนโปรแกรมควบคุมด้วยภาษาแลดเดอร์และภาษา C++ ต่างมีความเหมาะสมในงานของตนเอง แต่เมื่อทำงานในแบบซีเควนซ์ ทั้งการเขียนโปรแกรมแลดเดอร์และ C++ ทำได้หลากหลายวิธี ทำให้ทำความเข้าใจและดูแลรักษายาก [17] ดังนั้นจึงควรจะหาแนวทางให้การพัฒนาโปรแกรมเป็นไปทางเดียวกัน

2.5. การแปลงแผนภูมิสถานะเครื่องจักร

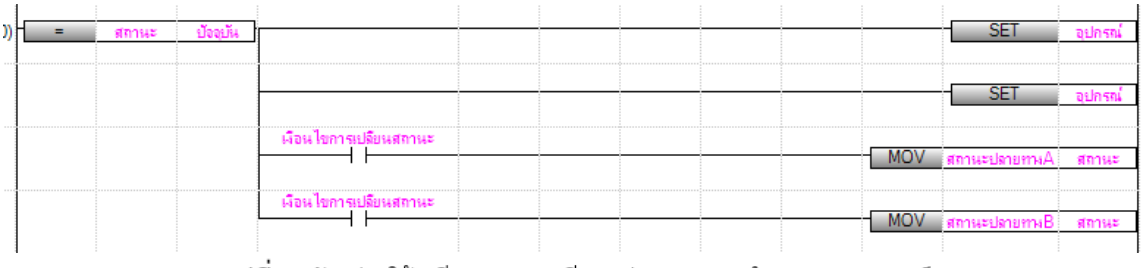
จากแผนภูมิสถานะ จะเห็นได้ว่ามีส่วนสำคัญอยู่สามส่วน คือชื่อของสถานะ การเปลี่ยนแปลงและคำอธิบายสถานะ แนวความคิดสำคัญคือเชื่อมโยงสถานะก่อนหน้าและเงื่อนไขการเปลี่ยนสถานะเข้าสู่สถานะปัจจุบัน ซึ่งทางผู้วิจัยขอนำเสนอการแปลงทั้ง 3 อย่างนี้ เพื่อให้สามารถเขียนอยู่ในรูปแบบของโค้ดที่สามารถพัฒนาตามหลักการของ SDLC ได้ โดยสิ่งที่ต้องทำก่อนการแปลงเป็นโค้ด คือ การเปลี่ยนชื่อ State เป็นตัวเลขและเก็บไว้ในตัวแปรแบบจำนวนเต็มเพื่อใช้ในการเป็นดัชนีของสถานะ และการสร้างตัวแปรเพื่อแทนจุดเชื่อมต่อของอุปกรณ์อินพุตเอาต์พุต หลังจากนั้นจะแบ่งออกเป็น 2 กรณี

2.5.1. กรณีภาษาแลดเดอร์เขียนบน PLC

ตัวอย่างจากรูปที่ 1 แยกออกเป็นสามส่วน คือ ส่วนเปรียบเทียบสถานะอยู่ด้านหน้าสุด (มุมบนซ้าย) เป็นส่วนที่บอกว่าสถานะนี้ทำงานหรือไม่ ส่วนถัดมา (มุมบนขวา) คือส่วนที่ต่อจากจากส่วนแรกมักจะตามด้วยคำสั่ง SET หรือ RST เป็นส่วนที่บอกว่าในสถานะนี้อุปกรณ์อะไรทำงานหรือไม่ทำงาน ส่วนสุดท้าย(มุมล่างขวา) คือเงื่อนไขการเปลี่ยนสถานะมักจะเป็นหน้าสัมผัสปกติเปิด (NO) หรือ ปกติปิด (NC) วางอยู่ก่อนคำสั่ง MOV ซึ่งเป็นตัวกำหนดว่าจะไปสถานะใดต่อไป โดยแต่ละสถานะสามารถมีเส้นทางเพื่อไปสถานะอื่นได้ไม่จำกัดจำนวน

2.5.2. กรณีภาษา C++

ในภาษา C++ ส่วนของการทำงานของแต่ละ State จะถูกครอบไว้ด้วยการเขียนโปรแกรมด้วยโครงสร้าง ‘Switch-Case’ ภายในฟังก์ชั่น (loop())



รูปที่ 1 ตัวอย่างโค้ดเทียมของการเขียนรูปแบบสถานะในภาษาแลดเดอร์

แต่ละ case เคสก็คือแต่ละ State ภายในนั้นจะมีส่วนควบคุมอุปกรณ์ และตามด้วยเงื่อนไขการเปลี่ยน State ครอบด้วย ‘If Statement’ ซึ่งเป็นเงื่อนไขการเปลี่ยน State และจะถูกยับยั้งแต่ละ State ด้วยคำสั่ง ‘break’ เพื่อเริ่มต้นวงวนใหม่และทำให้ State ปัจจุบันทำงาน ดังรูปที่ 2

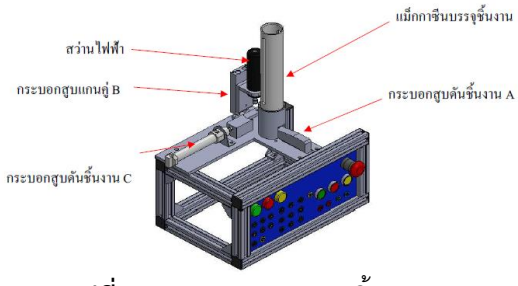
```
void loop() {
        switch (State) {
                case 0: //สถานะที่ 0
                // ส่วนอุปกรณ์ทำงาน
                 digitalWrite(Y_1, LOW);
                // ส่วนเงื่อนไขการเปลี่ยนสถานะ
                if (เงื่อนไขการเปลี่ยนสถานะ) {
                State = 1; // ไปสถานะที่ 1
                }
    break;}}
```

รูปที่ 2 ตัวอย่างโค้ดเทียมของการเขียนรูปแบบสถานะในภาษา C++

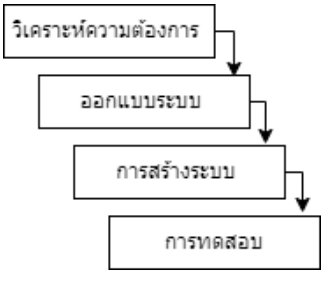
โดยการเปรียบเทียบโค้ดของทั้งสองภาษาจะถูกนำเสนอในผลการทดลอง

3. ขั้นตอนการทดลอง

การทดลองได้ทำในห้องปฏิบัติการรวมระบบอัตโนมัติและหุ่นยนต์ คณะครุศาสตร์อุตสาหกรรม มหาวิทยาลัยเทคโนโลยีราชมงคลอีสาน วิทยาเขตขอนแก่น โดยใช้ชุดการเรียนรู้เกี่ยวกับระบบอัตโนมัติในเรื่องการเขียนโปรแกรมแบบซีเควนซ์เชียล โดยใช้ชุดการเจาะชิ้นงาน MPU-A ของ บริษัท ออโตไดแด็กติก จำกัด เป็นระบบตัวอย่างที่นำมาใช้ในการประยุกต์ใช้หลักการเขียนโปรแกรม โดยมีลักษณะดังภาพที่ 3 ซึ่งระบบจะมีขั้นตอนใหญ่ๆ คือ จ่ายชิ้นงาน เจาะชิ้นงาน และส่งชิ้น จากนั้นจะประยุกต์เข้ากับการพัฒนาตามแนวทางของ SDLC แบบน้ำตกดังแสดงไว้ในภาพที่ 4 ซึ่งมีลักษณะดังนี้



รูปที่ 3 MPU-A ชุดการเจาะชิ้นงาน



รูปที่ 4 แนวทางในการพัฒนาโปรแกรม

1. ขั้นตอนวิเคราะห์ความต้องการ บ่งบอกถึงพฤติกรรมที่ระบบจะปฏิบัติซึ่งความต้องการจะได้มาจากการเก็บข้อมูลหรือเอกสารที่เกี่ยวข้อง

2. ขั้นตอนการออกแบบจะเป็นการนำผลลัพธ์ความต้องการมารวมกับข้อมูลกลไกลของฮาร์ดแวร์ โดยผลลัพธ์ที่ได้คือแผนภูมิสถานะ และการออกแบบการเดินสาย

3. ขั้นตอนการสร้างระบบ ในที่นี้คือเขียนโปรแกรมตามแผนภูมิสถานะโดยยึดแนวทางตามที่นำเสนอในงานวิจัยนี้ และการเดินสายฮาร์ดแวร์ตามที่ออกแบบไว้โดยการทดสอบจะแบ่งออกเป็นส่วนของ PLC และส่วนของไมโครคอนโทรลเลอร์

4. ขั้นตอนการทดสอบ ทำการทดสอบระบบตามเงื่อนไขที่ออกแบบไว้เพื่อให้ได้ครบตามความต้องการและหลีกเลี่ยงความผิดพลาดที่อาจจะเกิดขึ้น จะสังเกตได้ว่าในงานวิจัยนี้การทดลองจะสิ้นสุดอยู่ขั้นตอนที่ 4 ของแบบจำลองแบบน้ำตก เนื่องจากเป็นชุดฝึกการเรียนรู้จึงไม่มีการติดตั้งใช้งาน และการบำรุงรักษา

4. ผลการทดลอง

การทดลองได้ติดตั้งและทดสอบกับ PLC Mitsubishi FX5U ส่วน ไมโครคอนโทรลเลอร์ใช้ Arduino Mega 2560 ที่มีขา DIO มากกว่า UNO และเนื่องจาก Arduino ไม่สามารถจ่ายไฟ 24V ได้โดยตรงจึงต้องใช้ตัวแปลงไฟกระแสตรงระหว่าง 24V และ 5V จากขั้นตอนการทดลองนำไปสู่ผลการทดลองได้ ดังนี้

4.1. ขั้นตอนวิเคราะห์ความต้องการ

ความต้องการในที่นี้ได้รับจากโจทย์ชุดฝึกเพื่อการแข่งขันระบบอัตโนมัติซึ่งมีขั้นตอนดังนี้

4.1.1. กดสวิตช์ S_1 สั่งเริ่มทำงาน เฉพาะเมื่อกดสวิตช์ S1 กระบอกสูบ A ผลักชิ้นงานจากแม็กกาซีนเข้าเจาะและผลักชิ้นงานค้างไว้จนจบขั้นตอนที่ 3

4.1.2. กระบอกสูบ B เลื่อนสว่านลงเจาะ Motor_1 เริ่มหมุนดอกสว่าน

4.1.3. กระบอกสูบ B เลื่อนสว่านขึ้น Motor หยุดหมุนเมื่อจบขั้นตอน

4.1.4. กระบอกสูบ A ถอยกลับ

4.1.5 กระบอกสูบ C ผลักชิ้นงานออกทางด้านขวา

4.1.6. กระบอกสูบ C ถอยกลับ

4.2. ขั้นตอนการออกแบบ

ตารางที่ 1
ข้อมูลการต่อฝั่งอินพุต

อินพุต			คำอธิบาย
MPU-A	PLC	MCU	
Reed_1	X1	2	กระบอกสูบ A เคลื่อนที่เข้าสุด
Reed_2	X2	3	กระบอกสูบ A เคลื่อนที่ออกสุด
Reed_3	X3	4	สว่านไฟฟ้า B เคลื่อนที่ขึ้นสุด
Reed_4	X4	5	สว่านไฟฟ้า B เคลื่อนที่ลงสุด
Reed_5	X5	6	กระบอกสูบ C เคลื่อนที่เข้าสุด
Reed_6	X6	7	กระบอกสูบ C เคลื่อนที่ออกสุด
S_1	X10	8	ปุ่ม S1
S_2	X11	9	ปุ่ม S2
S_3	X12	10	ปุ่ม S3

ออกแบบการต่อสายระหว่าง MPU-A กับ PLC และ MPU-A กับ ไมโครคอนโทรเลอร์ (MCU) มีการเชื่อมต่อ ดังตารางที่ 1 และ ตารางที่ 2

ตารางที่ 2
ข้อมูลการต่อฝั่งเอาท์พุต

เอาท์พุต			คำอธิบาย
MPU-A	PLC	MCU	
Y_1	Y1	22	ควบคุมการเคลื่อนที่กระบอกสูบ A
Y_2	Y2	24	ควบคุมสว่านไฟฟ้า B

Y_3	Y3	26	ควบคุมการเคลื่อนที่กระบอกสูบ C
L_1	Y4	28	ไฟสัญญาณ L1
L_2	Y5	30	ไฟสัญญาณ L2
L_3	Y6	32	ไฟสัญญาณ L3
Motor_1	Y7	34	ควบคุมการหมุนของสว่านไฟฟ้า

แต่ในเงื่อนไขความต้องการนี้อุปกรณ์ S_3 L_1 L_2 L_3 ไม่ได้ถูกใช้งาน จึงจะไม่ปรากฏอยู่ในการออกแบบโปรแกรม จากความต้องการและการต่ออุปกรณ์สามารถวิเคราะห์และออกแบบลักษณะการทำงานให้สอดคล้องกับความต้องการซึ่งจะเขียนแผนภูมิสถานะเครื่องจักรได้ ดังรูปที่ 5

State 0 สถานะเริ่มต้นเมื่อ PLC หรือไมโครคอนโทรลเลอร์เริ่มต้นทำงาน หรือ ถูกรีเซท ระบบจะกลับเข้าสู่สถานนะนี้ ซึ่งอุปกรณ์ทำงานแต่ละตัวจะถูกรีเซทให้กลับเป็นสถานะเริ่มต้น เพื่อเตรียมพร้อมที่จะทำงานต่อไป โดยสถานะที่เป็นค่าเริ่มต้นจะต่อท้ายด้วยเครื่องหมาย * หลังสถานะ ดังรูปที่ 6

State 1 สถานะที่ 1 จะเริ่มต้นทำงานเมื่ออยู่ที่ State0 และ Reed_1 มีสถานะเป็น ON ปุ่ม S_1 ถูกกด จากนั้นเมื่ออยู่ใน State 1 กระบอกสูบ A (Y_1) จะกางออก ส่งผลให้ชิ้นงานออกจากแม็กกาซีนเข้า สู่กระบวนการเจาะ ดังรูปที่ 7

State 2 สถานะที่ 2 จะเริ่มต้นทำงานเมื่ออยู่ที่ State1 และกระบอกสูบ A กางออกสุดจน Reed_2 มีสถานะเป็น ON จากนั้นเมื่ออยู่ใน State 2 กระบอกสูบ B (Y_2) จะเลื่อนลงและสว่าน Motor_1 ทำงาน ส่งผลให้ชิ้นงานได้รับการเจาะ ดังรูปที่ 8

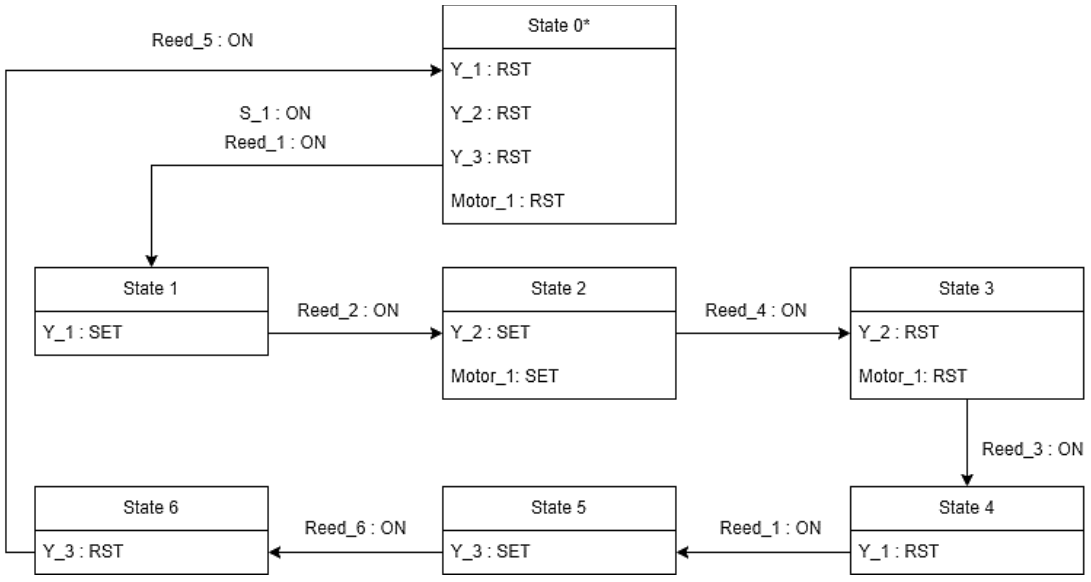
State 3 สถานะที่ 3 จะเริ่มต้นทำงานเมื่ออยู่ที่ State2 และกระบอกสูบ B เลื่อนลงสุดจน Reed_4 มีสถานะเป็น ON เมื่อเข้าสู่ State 3 กระบอกสูบ B จะเลื่อนขึ้นและสว่าน Motor_1 จะหยุดหมุน ส่งผลให้หยุดการเจาะชิ้นงาน ดังรูปที่ 9

State 4 สถานะที่ 4 จะเริ่มต้นทำงานเมื่ออยู่ที่ State3 และกระบอกสูบ B เลื่อนขึ้นสุดจน Reed_3 มีสถานะเป็น ON เมื่อเข้าสู่ State 4 กระบอกสูบ A ที่กางอยู่จะเคลื่อนกลับ ส่งผลให้ขากระบอกสูบ A หุบกลับ ดังรูปที่ 10

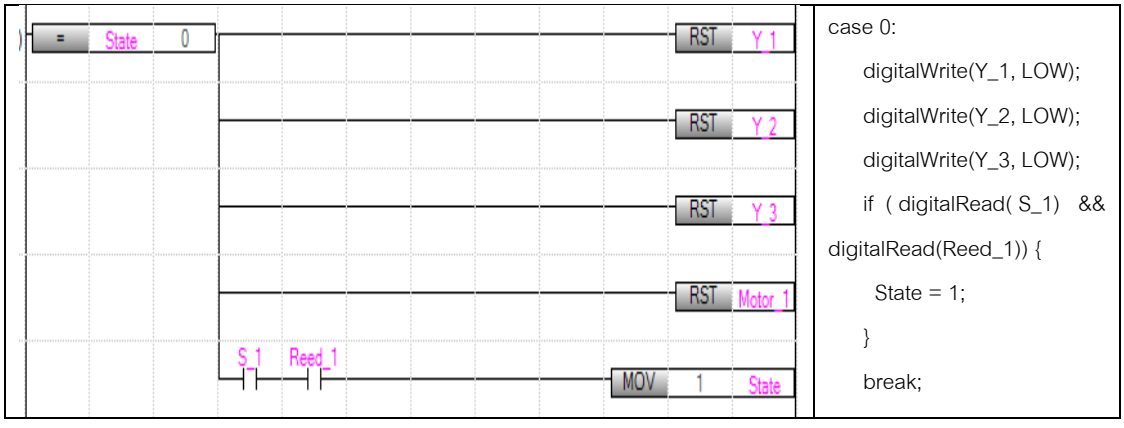
State 5 สถานะที่ 5 จะเริ่มต้นทำงามเมื่ออยู่ที่ State 4 และกระบอกสูบ A จะเคลื่อนกลับสุดจน Reed_1 ทำงาน เมื่อเข้าสู่ State 5 กระบอกสูบ C (Y_3) จะเลื่อนออกสุด ส่งผลให้ชิ้นงานถูกส่งออกจากกระบวนการเจาะ ดังรูปที่ 11

State 6 สถานะที่ 6 จะเริ่มต้นทำงานเมื่ออยู่ใน State 5 และ กระบอกสูบ C เลื่อนออกสุด Reed_6 เมื่อเข้าสู่ State 6 กระบอกสูบ C จะเลื่อนเข้าสุด ส่งผลให้กระบอกสูบ C เก็บ รอจนกว่า Reed_5 มีสถานะ ON จะกลับเข้าสู่ State 0 ดังรูปที่ 12

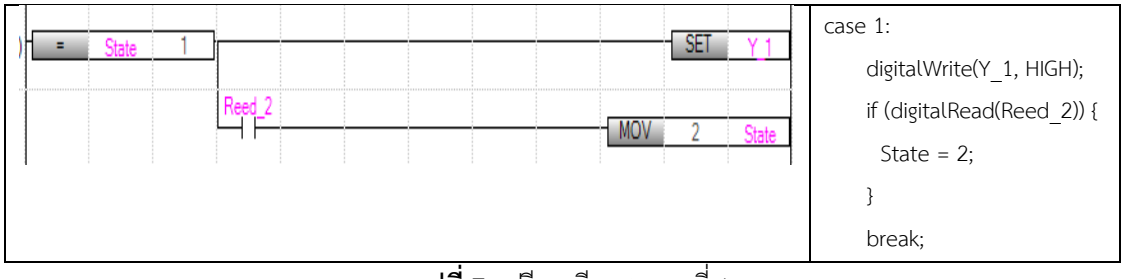
สถานะที่ไม่ได้อยู่ในการออกแบบ แต่ถูกเพิ่มเข้ามาเพื่อใช้ในการตรวจสอบและแก้ไขระหว่างพัฒนา คือ การกดปุ่ม S_2 ในการกลับเข้าไปที่ State 0 ในการเขียนภาษาแลดเดอร์ การเปรียบเทียบถูกใช้สองครั้ง เพื่อระบุสถานะที่สามารถทำการหยุดในที่นี้คือ State 1 ถึง State 6 สามารถกลับมาที่ State 0 ในขณะที่ภาษาซีเขียนเงื่อนไขนี้อยู่ข้างนอกการเปรียบเทียบแบบ switch-case จะทำให้รูปแบบโค้ดกระชับกว่า ดังรูปที่ 12



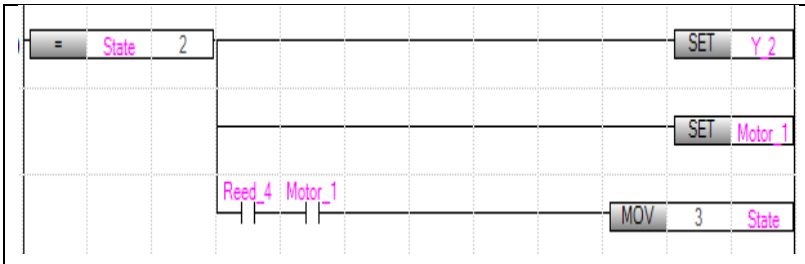
รูปที่ 5 แสดงแผนภูมิสถานะของ MPU-A



รูปที่ 6 เปรียบเทียบสถานะที่ 0

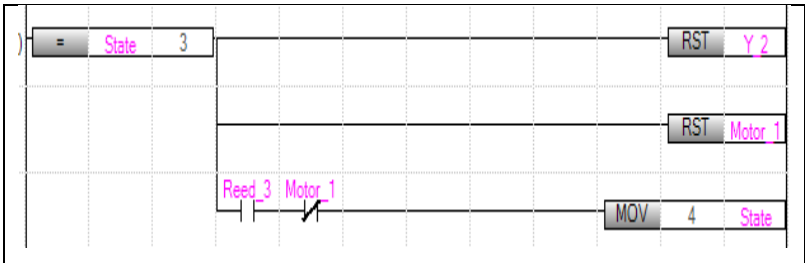


รูปที่ 7 เปรียบเทียบสถานะที่ 1



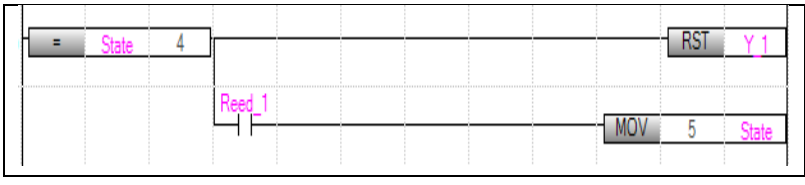

```
case 2:
    digitalWrite(Y_2, HIGH);
    digitalWrite( Motor_1, HIGH);
    if( digitalRead( Reed_4) && digitalRead(Motor_1)){
      State = 3;
    }
    break;
```

รูปที่ 8 เปรียบเทียบสถานะที่ 2



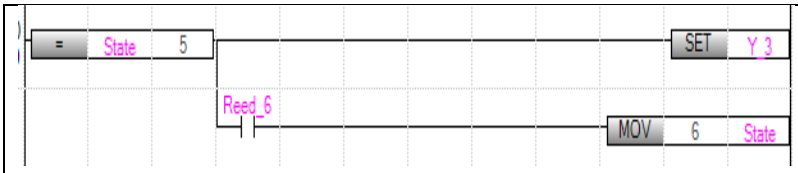

```
case 3:
    digitalWrite(Y_2, LOW);
    digitalWrite( Motor_1, LOW);
    if(digitalRead(Reed_3) && not digitalRead(Motor_1)){
      State =4;
    }
    break;
```

รูปที่ 9 เปรียบเทียบสถานะที่ 3



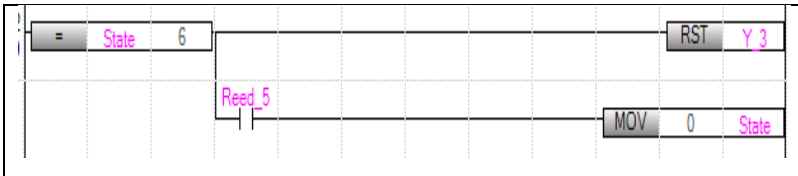

```
case 4:
    digitalWrite(Y_1, LOW);
    if(digitalRead(Reed_1));
    break;
```

รูปที่ 10 เปรียบเทียบสถานะที่ 4




```
case 5:
    digitalWrite(Y_3, HIGH);
    if(digitalRead(Reed_6)){
      State = 6;
    }
    break;
```

รูปที่ 11 เปรียบเทียบสถานะที่ 5




```
case 6:
    digitalWrite(Y_3, LOW);
    if(digitalRead(Reed_4)){
      State = 0;
    }
    break;
```

รูปที่ 12 เปรียบเทียบสถานะที่ 6

4.3 ผลของโปรแกรมที่ออกแบบ

การทดสอบระบบการควบคุม MPU-A โดยใช้ PLC และไมโครคอนโทรลเลอร์ พบว่าทั้งสองระบบสามารถทำงานได้ตามความต้องการ แต่ความเร็วในการตอบสนองใกล้เคียงกัน แต่ไมโครคอนโทรลเลอร์จะมีกลไกลจากวงจรแปลงแรงดันไฟฟ้าหรือรีเลย์เพื่อใช้กับ MPU-A เลยทำให้ตอบสนองช้ากว่าพีเอลซี ดังนั้นการเลือกใช้ตัวควบคุมในจะขึ้นอยู่กับความต้องการในเรื่องของความเร็วในการตอบสนองและงบประมาณที่มี

5. สรุป

งานวิจัยนี้นำเสนอรูปแบบการเขียนโปรแกรมสำหรับควบคุมระบบอัตโนมัติในแนวทางของแผนภูมิสถานะ เพื่อที่จะสามารถพัฒนาโปรแกรมในแนวทาง SDLC ซึ่งจะช่วยให้มีประสิทธิภาพในการพัฒนาโปรแกรม นอกจากนี้ยังทำให้สามารถอ่านโค้ดและเห็นภาพรวมของการทำงานได้ โดยในการทดสอบ จะทำการพัฒนาโปรแกรมควบคุมสำหรับชุดฝึกระบบอัตโนมัติในกระบวนการเจาะ MPU-A โดยใช้ตัวควบคุมราคาประหยัดแยกออกเป็น PLC ที่เขียนด้วยภาษาแลดเดอร์ และ ไมโครคอนโทรลเลอร์เขียนควบคุมด้วยภาษาซีพลัสพลัสซึ่งทั้งสองภาษาสามารถเขียนได้จากการแปลงแผนภูมิสถานะ โค้ดแยกเป็นสามส่วน คือ ส่วนสถานะ ส่วนการเปลี่ยนสถานะ และส่วนการทำงาน ทำให้สามารถพัฒนาและแก้ไขได้ง่ายขึ้นเนื่องจากแต่ละสถานะถูกแบ่งแยกออกจากกันและเป็นอิสระในการแก้ไข

ผลการทดสอบทั้งสองตัวควบคุมสามารถทำได้ตามความต้องการของระบบ แตกต่างกันตรงการ

ตอบสนองที่พีเอลซีตอบสนองได้รวดเร็วกว่าไมโครคอนโทรลเลอร์ที่ต้องผ่านวงจรช่วยทำงานก่อน เนื่องจากงานวิจัยนี้นำเสนอทางด้านการออกแบบโปรแกรม จึงยังไม่ได้ทดสอบด้านอื่น เช่น เวลาในการตอบสนองของเครื่องจักร การเดินสายเครื่อง การทนต่อสิ่งรบกวน หรือทางด้านความคุ้มค่าของอุปกรณ์ ซึ่งต้องทำการวิจัยเพิ่มเติมและทดสอบในโจทย์ปัญหาอื่นต่อไป

โดยสรุปรูปแบบการเขียนโปรแกรมที่นำเสนอสามารถช่วยอำนวยความสะดวกในการพัฒนาได้โดยไม่ขึ้นอยู่กับเทคโนโลยีของเจ้าของผลิตภัณฑ์ ภาษาที่ใช้เขียนโปรแกรม แต่ขึ้นอยู่กับการออกแบบแผนภูมิสถานะเครื่องจักร ทำให้มีความยืดหยุ่นในการพัฒนามากขึ้น และเห็นภาพรวมของระบบได้ง่ายขึ้น แต่ตัวอย่างในการทดลองยังเป็นเพียงการควบคุม แบบซีเควนเชียลง่ายๆ เท่านั้น หากระบบมีความซับซ้อนมากขึ้น สัญลักษณ์หรือการทำงานบางอย่างยังไม่ถูกกล่าวถึง ทางผู้วิจัยจะพัฒนาเพิ่มเติมในงานวิจัยภายหน้า

6. เอกสารอ้างอิง

[1] Fronchetti F, Ritschel N, Holmes R, Li L, Soto M, Jetley R, et al. Language impact on productivity for industrial end users: A case study from Programmable Logic Controllers. Journal of Computer Languages. 2022 Apr 1;69:101087.

[2] Chakraborti T. From Robotic Process Automation to Intelligent Process Automation: – Emerging Trends– . InBusiness Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18; 2020, Proceedings 18 2020; 215-228. Springer International Publishing.

[3] Beer P, Mulder RH. The effects of technological developments on work and their implications for continuous vocational education and training: A systematic review. Frontiers in Psychology 2020; May 8;11:918.

[4] Thong-un N, Wongsaroj W. Productivity enhancement using low- cost smart wireless programmable logic controllers: A case study of an oyster mushroom farm. Computers and Electronics in Agriculture. 2022 Apr 1;195:106798.

[5] Dwinugroho TB, Hapsari YT. Greenhouse automation: Smart watering system for plants in greenhouse using programmable logic control (PLC). In Journal of Physics: Conference Series 2021 Mar 1 (Vol. 1823, No. 1, p. 012014). IOP Publishing.

[6] Cheng YH, Chao PJ, Liang HY, Kuo CN. Smart Home Environment Management Using Programmable Logic Controller. Engineering Letters. 2020 Dec 1; 28(4).

[7] Livinsa ZM, Valantina GM, Premi MG, Sheeba GM. A modern automatic cooking machine using arduino mega and IOT. InJournal of Physics: Conference Series 2021 Mar 1 (Vol. 1770, No. 1, p. 012027). IOP Publishing.

[8] Tiegelkamp M, John KH. IEC 61131-3: Programming industrial automation systems. Berlin/Heidelberg, Germany: Springer; 2010.

[9] Erickson KT. Programmable logic controllers. IEEE potentials. 1996 Feb; 15(1): 14-7.

[10] Bornot S, Huuck R, Lakhnech Y, Lukoschus B. An abstract model for sequential function charts. Discrete event systems: analysis and control. 2000: 255-64.

[11] Sanver U, Yavuz E, Eyupoglu C, Uzun T. Design and implementation of a programmable logic controller using PIC18F4580. In2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) 2018 Jan 29; 231-235. IEEE.

[12] Balaji S, Murugaiyan MS. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. International Journal of Information Technology and Business Management. 2012 Jun 29; 2(1): 26-30.

[13] Vogel-Heuser B, Obermeier M, Braun S, Sommer K, Jobst F, Schweizer K. Evaluation of a UML-based versus an IEC 61131-3-based software engineering approach for teaching PLC programming. IEEE Transactions on Education. 2012 Nov 29; 56(3): 329-35

[14] Wagner F, Schmuki R, Wagner T, Wolstenholme P. Modeling software with finite state machines: a practical approach. CRC Press; 2006 May 15.

[15] DeGuglielmo NP, Basnet SM, Dow DE. Introduce Ladder Logic and Programmable Logic Controller (PLC). In2020 Annual Conference Northeast Section (ASEE-NE) 2020 Oct 16; 1-5. IEEE.

[16] Sanver U, Yavuz E, Eyupoglu C, Uzun T. Design and implementation of a programmable logic controller using PIC18F4580. In 2018 IEEE Conference of Russian Young Researchers in

Electrical and Electronic Engineering (EIConRus) 2018 Jan 29; 231-235. IEEE.

[17] Hudedmani MG, Umayal RM, Kabberalli SK, Hittalamani R. Programmable logic controller (PLC) in automation. Advanced Journal of Graduate Research. 2017 May 24; 2(1): 37-45.