Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025 ASEE
Paper ID #47788

WIP: An Advanced Automation Final Project Using a Finite-State Machine to Automate Motion Control

Mr. Bradley Lane Kicklighter P.E., University of Southern Indiana

Brad holds a BS in Electrical Engineering from Rose-Hulman Institute of Technology (1989) and an MS in Electrical and Computer Engineering from Purdue University (2001).

His past work experience includes eleven years at Delphi (formerly Delco Electronics) as an Advanced Project Engineer, eleven years at Whirlpool Corporation as a Lead Engineer/Solution Architect, and three years at Ivy Tech Community College as an Instructor/Program Chair Pre-Engineering. Since 2015, he has been employed at the University of Southern Indiana as a Clinical Associate Professor of Engineering Technology.

He holds three patents, has served as an IEEE section officer since 2004, and has been a Licensed Professional Engineer in the State of Indiana since 2005.

Brad is the current chair of the ASEE Instrumentation Division.

WIP: An Advanced Automation Final Project Using a Finite-State Machine to Automate Motion Control

Abstract

Programmable logic controllers (PLCs), finite-state machines (FSMs), human-machine interfaces (HMIs), and motion control are common topics in industrial automation courses. In industry, PLCs are commonly used to automate processes and machines. Using an FSM in a PLC program is a convenient way to implement the steps necessary to control a process or machine. HMIs provide interaction with a control system for a process or machine. Motion control, especially when using stepper motors, is widely used in industry to control the movement of a machine.

Advanced Automation is an elective course for junior and senior engineering and engineering technology students. This course is the second of two automation courses and was taught for the first time in spring 2024. For the final project in the Advanced Automation course, students must program an Allen-Bradley Micro850 PLC using ladder diagram (LD), structured text (ST), or a combination to control a two-axis motion control module which consists of stepper motors providing rotary motion. In addition to allowing students their choice of programming language, they are also able to choose the form of their FSM: based on Boolean values or integer values. Students must also program an Allen-Bradley PanelView 800 human-machine interface (HMI) to provide screens to control the system.

This work-in-progress paper will describe the Advanced Automation final project and the Two-Axis Motion Control Module that was designed and fabricated by the author.

Introduction

Programmable logic controllers (PLCs) along with human machine interfaces (HMIs) are commonly used to automate machines and processes in industry in general and manufacturing in particular. Many machines and processes require some form of motion control. Open-loop motion control in the form of stepper motors is quite common. If the control of a machine or process is in the form of a set of steps, then the use of finite-state machines (FSMs) is a convenient approach.

Here at the University of Southern Indiana, members of our Engineering Advisory Board have expressed interest in our students getting more experience with automation. Additionally, our students have asked for a follow-on course to ENGR 382 SCADA Systems Design which introduces the programming of PLCs (using ladder logic) and HMIs. As a result, ENGR 383 Advanced Automation was created and first taught in the spring of 2024. This new course is open to juniors and seniors who have taken ENGR 382 and is an elective for engineering and engineering technology students.

ENGR 383 teaches Allen-Bradley Micro800 ladder diagram (LD) programming ["ladder diagram" is also known as "ladder logic"], structured text (ST) programming, Allen-Bradley

PanelView 800 HMI programming, finite-state machines, industrial networking, motion control, program flow control, designing for failure, documentation, functional safety, Industrial Internet of Things, wiring standards, and control panel standards.

A PLC/HMI trainer and two-axis motion control motion module, both designed and fabricated by the author, are used in the course (see Figure 1).

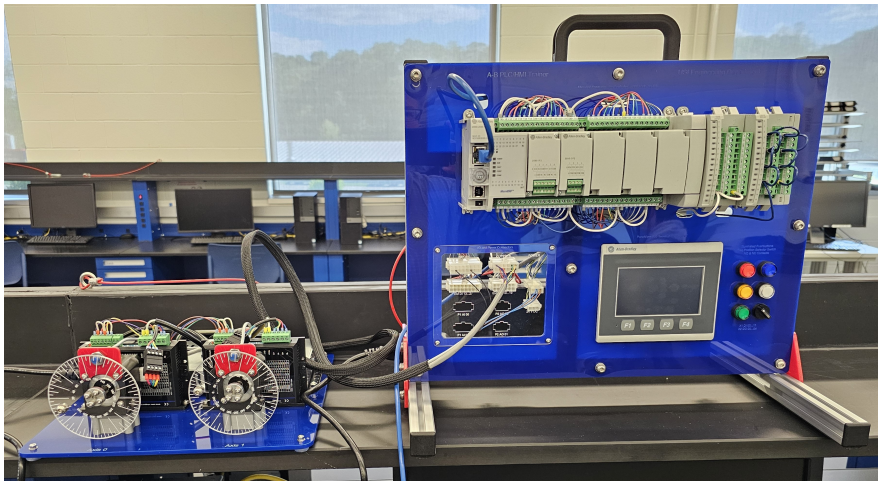

Figure 1: Allen-Bradley PLC/HMI Trainer with Two-Axis Motion Control Module.

The trainer is designed around an Allen-Bradley Micro850 PLC and Allen-Bradley PanelView 800 HMI. Both the PLC and HMI are programmed using the Connected Components Workbench software from Rockwell Automation. The motion control module consists of a power supply, two stepper motor drivers, two stepper motors, and two inductive proximity sensors (used for homing the axes). Acrylic disks with markings every five degrees are attached to the stepper motors. See Figure 2.

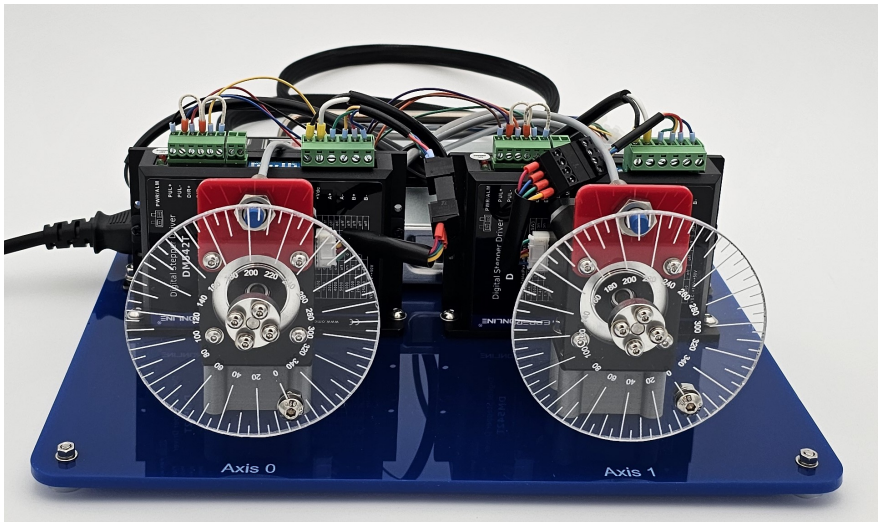

Figure 2: Two-Axis Motion Control Module.

Each student completes their own labs and projects. There are enough trainers and motion control modules so that each student has their own. The course size is capped at sixteen students.

Finite-State Machines

A finite-state machine (FSM) is a class of automata where it has a finite set of states, can only be in one state at a time, has an initial or start state, has a finite set of inputs, has a finite set of outputs, and transitions from one state to another based on the inputs [1].

Ladder Diagram Implementations

Two ladder diagram implementations are taught in the course based on a white paper by Anderson[2].

One ladder diagram implementation uses ladder diagram primitives (DirectContact, ReverseContact, and DirectCoil) which are Boolean values. See Figures 3 and 4.

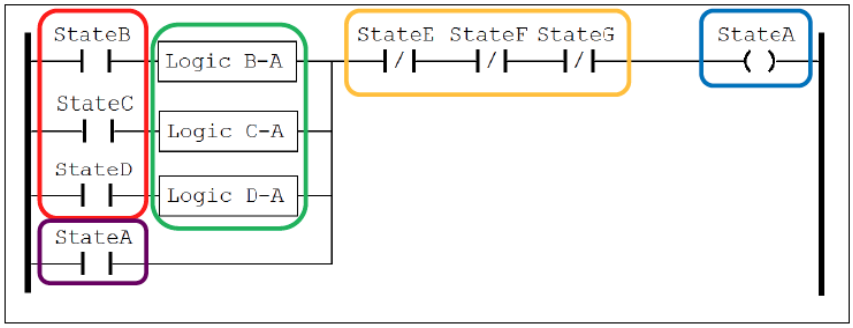

Figure 3: FSM implementation using LD primitives. Previous states are outlined in red, transition logic is outlined in green, seal in is outlined in purple, next states are outlined in yellow, and current state is outlined in blue. (Source: [2])

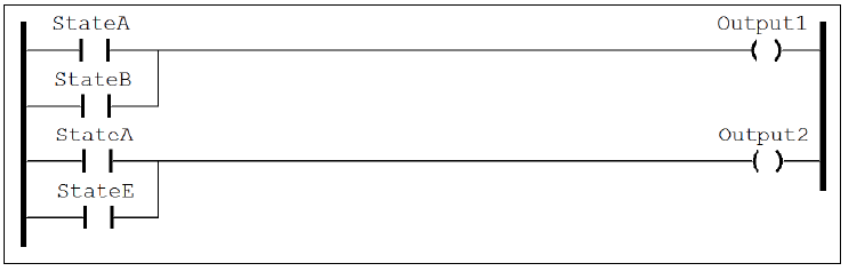

Figure 4: Decoding of states into outputs. (Source: [2])

The other ladder diagram implementation assigns an integer value to each state and uses Equal (=) and Move function blocks. See Figures 5 and 6.

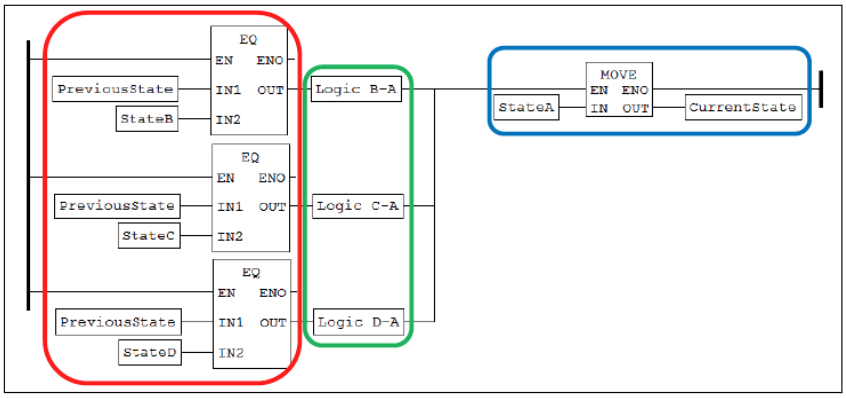

Figure 5: FSM implementation using LD using Move and Equal. Previous states are outlined in red, transition logic is outlined in green, and current state is outline in blue. (Source: [2])

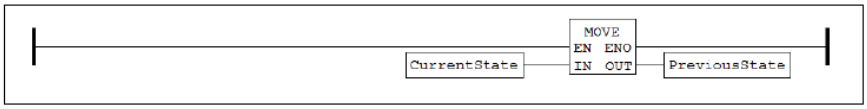

Figure 6: Once the current state is updated, then the previous state is set to the current state. This rung must be placed after the state rungs. (Source: [2])

Cohenour describes a version of the Boolean finite-state machine implementation in ladder logic for a final project using a fluid process rig [3].

Structured Text Implementations

Two structured text implementations are taught in the course.

One structured text implementation is similar to the ladder diagram primitives implementation where states are Boolean values and the If-Then-Else statement is used. The following is a generic FSM example implemented in ST using If-Then-Else statements:

```
// Input Conditioning
R_TRIG_1(Input1_In);
Input1 := R_TRIG_1.Q;
R_TRIG_2(Input2_In);
Input2 := R_TRIG_2.Q;

// States
...
// State1
IF State0 AND Input1 THEN
State1 := TRUE;
ELSIF State2 THEN
State1 := FALSE;
```

```
END_IF;

// State0
IF __SYSVA_FIRST_SCAN OR (State4 AND Input2) THEN
State0 := TRUE;
ELSIF State1 THEN
State0 := FALSE;
END_IF;

// Output Decoding
IF State0 THEN
Output1 := TRUE;
ELSE
Output1 := FALSE;
END_IF;

IF State1 THEN
Output2 := TRUE;
ELSE
Output2 := FALSE;
END_IF;
```

The other structured text implementation assigns an integer value to each state and uses the Case statement along with If-Then-Else statement. The following is a generic FSM example implemented in ST using a case statement:

```
// Initialization
IF __SYSVA_FIRST_SCAN THEN
NextState := State0;
CurrentState := State0;
END_IF;

// Input Conditioning
R_TRIG_1(Input1_In);
Input1 := R_TRIG_1.Q;
R_TRIG_2(Input2_In);
Input2 := R_TRIG_2.Q;

// States
CASE CurrentState OF
0:  // State0
Output1 := FALSE;
Output2 := FALSE;
IF Input1 THEN
NextState := State1;
END_IF;
```

```
1:  // State1
Output1 := TRUE;
Output2 := FALSE;
IF Input1 THEN
NextState := State2;
ELSIF Input2 THEN
NextState := State3;
END_IF;
...
END_CASE;

// Update CurrentState
CurrentState := NextState
```

Motion Control

When some part of a machine must move in a controlled manner, we apply a sub-field of automation called motion control[4]. The motion may be linear, rotary, or a combination. Motion control may be closed-loop (a feedback sensor indicates the position of the system) or open-loop (there is no feedback to verify actual position).

Examples of motion control applications include:

- Conveyors for material handling.
- Pick and place.
- Computer Numerical Control (CNC) machining.
- 3D printing.
- Laser engraving.
- Water jet cutting.
- Plasma cutting.
- Industrial robots.

Closed-Loop Motion Control

In a closed-loop motion control system, there is a feedback sensor that indicates the position of the system. In many cases an encoder is used as the feedback sensor. The encoder may be absolute or incremental. An incremental encoder requires the system to be "homed" to establish the home or origin of the axis.

Servo motors are commonly used in closed-loop systems. The encoder may be part of the servo motor or may be a separate device. In most cases, a servo drive is required to drive a servo motor. The PLC or controller in the system is connected to and commands the servo

drive which then drives the servo motor to the desired position and at the desired speed. Typically, the encoder in the system is connected to the servo drive.

Open-Loop Motion Control

In an open-loop motion control system, there is no feedback to verify actual position. Position is assumed based commands given to the axis motor. A system like this must be "homed" to establish a home position or origin. This requires each axis to have one or more limit switches, photoelectric sensors, or proximity sensors to sense when the system is at the home position.

The most common motor type used in open-loop motion control systems is the stepper motor. Usually, a stepper motor driver is used to drive the stepper motor phases directly and the PLC or microcontroller provides step pulses and a direction signal to the stepper motor driver.

For this course, open-loop motion control using stepper motors was selected since the system cost was much lower that using servo motors.

Examples in the literature where a PLC is used to control a stepper motor in labs and projects include Lee (elevator)[5][6], Tepe et al. (conveyor belt)[7], Fotouhi and Eydgahi (utility cart)[8], and Sokoloff (low level control of stepper motor)[9]. In all of these examples, ladder logic is used to program the PLCs.

The motion control instructions in the Micro800 series PLCs are based on PLCOpen Motion Control[10]. There are a set of function blocks for performing motion control such as controlling power to (enabling) an axis, resetting axis errors, performing an absolute position move, performing a relative position move, performing a velocity move, homing an axis, and halting axis motion.

Methods

ENGR 383 is a project-based course where students are assessed using exercises, quizzes, labs, and projects. The final project (final assessment) for the course has the following objectives:

1. Design and create a two-axis motion control system.
2. Use a finite-state machine to implement a motion sequence.
3. Design and program the human machine interface (HMI) to implement a user interface for the system.

The following is the problem statement for the project:

> Create a system that controls a two-axis motion control module. The motion control module (MCM) must execute a motion sequence. All system interaction must be done using one or more HMI screens.

Students are allowed to use ladder diagram (LD), structured text (ST), or a combination. In addition, students may choose to use a Boolean or integer based finite-state machine approach and use either absolute or relative positioning of the motion control axes. These choices allow students some freedom in their implementation and make the project less prescriptive.

The system must implement the following functions:

- Enable motion control axes.
- Reset motion control errors.
- Home each motion control axis.
- Execute a seven-step motion control sequence in the form of angular rotations (see Table 1).

Table 1: Motion sequence with absolute positions for each step in fractions of a revolution. Step 0 is the stopped state (waiting for the start button to be pressed). The sequence is 1 to 2 to 3 to 4 to 5 to 6 to 1 ...

Step	**Axis0**	**Axis1**
0	-	-
1	0.5	0.5
2	0.0	-
3	-	0.0
4	0.25	-0.25
5	-0.25	0.25
6	0.0	0.0

Results and Discussion

The final project was the first time the students were able to create an FSM or interact with the motion control module. This was due to late availability of the hardware. Even with that being the case, the majority of the students did quite well on the project. Some students required more help than others with the FSM, the motion control instructions, or both.

Overall, the author was happy with the results of the final project and the students enjoyed getting the motion control module to perform.

Students provided the following feedback on the course:

“Practical labs, learned a lot about HMI/PLC programming.”

“This course gave a better understanding of Industrial Controls Engineering, which reinforces the material from the previous course. The experience programming a PLC is valueable for those interested in working in the manufacturing area.”

"The labs and projects were very helpful. They allowed for a better understanding of more complex scenarios involving PLC programming."

"Some of the labs take quite a bit of time, some (Resistor Calculator) are big enough to be projects and some projects (hopper volume) are small enough to be just labs."

"I would like to learn how to build a PLC system as well as program it. I understand that the trainers are limited and expensive, but possibly having a lab to spec out a PLC and components for a specific automation scenario would be nice to implement."

Conclusions and Recommendations

ENGR 383 and its final project meet the need for exposing students to more automation topics and to explore some automation topics deeper than the prior course, ENGR 382. This helps satisfy the desire of our Engineering Advisory Board members for our students to get more automation experience.

The final project assesses major course topics including ladder diagram programming, structured text programming, HMI programming, finite-state machines, motion control, and documentation (in the form program comments).

Two of the students who graduated after taking the course remarked about how they were applying many of things they learned in ENGR 383 to their new jobs and were grateful for the course.

What could be done better? Since the hardware is now available, the students will get exposed to it earlier in the course. A lab devoted to creating an FSM should be added and a lab that introduces the motion control module should be added.

References

[1] "Finite-state machine," *Wikipedia*, Nov. 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Finite-state_machine&oldid=1187480774

[2] E. Anderson, "Sequential Function Chart to PLC Ladder Logic Translation." [Online]. Available: https://www.dmcinfo.com/latest-thinking/blog/id/10076/sequential-function-chart-to-plc-ladder-logic-translation

[3] C. Cohenour, "Teaching Finite State Machines (FSMs) as Part of a Programmable Logic Control (PLC) Course," in *2017 ASEE Annual Conference & Exposition*, Jun. 2017. [Online]. Available: https://peer.asee.org/teaching-finite-state-machines-fsms-as-part-of-a-programmable-logic-control-plc-course

[4] "Motion control," *Wikipedia*, Oct. 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Motion_control&oldid=1178610879

[5] S. Lee, "Integration Of Motion Control Teaching Components Into The Programmable Logic Controller Course," in *2009 Annual Conference & Exposition*, Jun. 2009, pp. 14.776.1–14.776.10. [Online]. Available: https://peer.asee.org/integration-of-motion-control-teaching-components-into-the-programmable-logic-controller-course

[6] ——, "Development Of A Four Story Elevator System For Teaching Motion Control Concept With Programmable Logic Controller," in *2010 Annual Conference & Exposition*, Jun. 2010, pp. 15.400.1–15.400.10. [Online]. Available: https://peer.asee.org/development-of-a-four-story-elevator-system-for-teaching-motion-control-concept-with-programmabl

[7] C. Tepe, A. S. Aslan, and İ. Eminoğlu, "Conveyor belt experiment setup for programmable logic controller education," *International Journal of Electrical Engineering & Education*, vol. 60, no. 3, pp. 258–272, Jul. 2023. [Online]. Available: https://doi.org/10.1177/0020720920958134

[8] K. Fotouhi and A. Eydgahi, "Using A Plc Trainer To Control A Utility Cart," in *2001 Annual Conference*, Jun. 2001, pp. 6.1100.1–6.1100.6. [Online]. Available: https://peer.asee.org/using-a-plc-trainer-to-control-a-utility-cart

[9] L. Sokoloff, "Plc Stepper Motor Controller," in *1998 Annual Conference*, Jun. 1998, pp. 3.447.1–3.447.12. [Online]. Available: https://peer.asee.org/plc-stepper-motor-controller

[10] V. Eldijk, "Motion Control," Apr. 2018. [Online]. Available: https://plcopen.org/technical-activities/motion-control