2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #53086

Preliminary Testing of an Educational Assembly Cell Designed to Study Flexible Manufacturing Operations

Seyi Oluwadare, Clarkson University

Seyi Oluwadare is a doctoral student at Clarkson University in Potsdam, NY. He has a dual master's degree in mechanical engineering, with one focus on thermofluids and the other in computational modeling. His current research focuses on integrating Allen's Interval Algebra and temporal reasoning analytics to model sequence dependencies in both manufacturing operations and curriculum structures, with the broader goal is to advance intelligent systems that enhance efficiency, flexibility, and equity across engineering and educational domains.

Dr. Carl David Hoover, Clarkson University

Dr. Carl D. Hoover is Associate Professor and Director of Laboratories in the Department of Mechanical and Aerospace Engineering at Clarkson University where he teaches engineering courses and researches data science applications. Dr. Hoover's MAE lab courses Measurement & Instrumentation, Experimental Methods, and Test Engineering are at the heart of Clarkson's hands-on engineering education. Dr. Hoover oversees several engineering research facilities in the Center for Advanced Materials Processing (CAMP) building at Clarkson including our wind tunnel facility and several mechanical testing labs. Carl leads the Academic Test Equipment Asset Management & Safety (A-TEAMS) committee which he founded. Working with industry partners, Dr. Hoover co-developed Solinsky Challenge manufacturing-focused microcredential courses in Statistical Process Control, Basic Statistics for Manufacturing, Design for Manufacturing and Assembly, and Manufacturing Assembly Methods. Carl brings over 15 years of industry experience to the classroom with experience ranging from aerospace test engineering to executive leadership. Prior to academia, Carl served as CTO of Elder Research, where he now consults as a Senior Advisor. Carl has managed R&D, led a world-class team of scientists, and served on several DoD projects. Dr. Hoover previously taught NGA College's Program of Study in Data Science graduate courses through Mizzou. Classes he's taught include Introduction to Data Science, Data Science for Managers, Data Visualization, Statistical & Mathematical Foundations of Data Analytics, and Data Analytics from Applied Machine Learning. Dr. Hoover's research interests are in machine learning and testing applications in engineering, manufacturing and measurement systems. Current projects include temporal relational dependence models in STEM and manufacturing digital twins.

Preliminary Testing of an Educational Assembly Cell Designed to Study Flexible Manufacturing Operations

Seyi Oluwadare & Carl Hoover
Clarkson University, Potsdam, NY

Abstract

Manufacturing systems increasingly demand scheduling methods that are both efficient and resilient to variability in task times, rework, routing order, and resource availability. Most existing Manufacturing Execution Systems (MES) and related manufacturing teaching approaches are limited to static routings, offering little opportunity to allow routing sequence flexibility as learning and production evolve. This work-in-progress study involves preliminary testing of an educational assembly cell designed to allow students to investigate flexibility in manufacturing of an Arduino-based robot. This work is motivated particularly by the assumption that exploring task relationships using Allen temporal relations will allow operators to better understand allowable variation, thereby improving overall process efficiency. Preliminary work successfully configured the assembly cell, work instructions, alternate routings, process logging, and measurement techniques. A proposed Assembly Sequence Difference Index (ASDI) was introduced to measure overall deviation relative to a baseline routing. Initial testing designed to assess the cell and metrics yielded an experimental dataset comprising 10 pilot trials conducted by the researcher and a single undergraduate student trial. Initial results show promise that the education assembly cell and the ASDI metric are useful for learning insights about manufacturing flexibility. Results showed that assembly tasks can be successfully reordered. Several alternative routing trials achieved shorter cycle times than the baseline. Further, precedence deviation appears higher in robot disassembly than assembly. ASDI of disassembly is higher at the beginning of trials when the operator is new to the system than it is in assembly; however, as the trial progresses the ASDI between the two operations becomes similar. The preliminary station was limited to simple sequence changes (before, after, meets, etc). In addition to capturing trials for more student operators, ongoing work involves development of a wiring harness assembly cell and a quality testing station that allow study of more complicated temporal relations (overlaps, during, starts, etc).

1. Background

Traditional manufacturing systems rely on rigid, predefined assembly sequences that operators must follow step by step. However, this approach fails to capture the natural flexibility that exists in many assembly operations, where certain tasks can be performed in different orders without affecting the final product quality. Understanding and formalizing this assembly sequence flexibility is crucial for optimizing manufacturing workflows and improving process efficiency. With a better understanding of the relationships and constraints among tasks in the assembly or disassembly of manufacturing operations, tasks can be reordered to achieve optimal operation or to respond to unexpected delays, machine breakdowns, or maintenance in the assembly line. In flexible discrete manufacturing, some tasks or operations are blocked or executed in sequence due to physical, logical, temporal, state, and resource dependencies.

Among all constraints and dependencies, temporal relationships are most important for this study in determining sequence flexibility. To mathematically define the temporal relationship between the tasks, Allen interval algebra (AIA), a calculus for temporal reasoning, was applied. AIA was introduced by James Allen in 1989 [1] and defines 13 possible temporal relations between time intervals (as shown in Fig. 1) and provides a composition table that can be used as a basis for reasoning about temporal descriptions of events. AIA was initially conceptualized for artificial intelligence, but the calculus was later applied in planning and scheduling in manufacturing. The presentation of temporal keys underlies allowable sequence variation in assembly processes.

S/N	Relation Name	Symbol	Formal Definition (Endpoint Relations)	Illustration (X above Y)
1.	X precedes Y	p (<)	X+ < Y-	
2.	Y preceded-by X	pi (>)	Y+ < X-	
3.	X meets Y	m	X+ = Y-	
4.	Y met-by X	mi	Y+ = X-	
5.	X overlaps Y	o	X- < Y- < X+ < Y+	
6.	Y overlapped-by X	oi	Y- < X- < Y+ < X+	
7.	X starts Y	s	X- = Y- and X+ < Y+	
8.	Y started-by X	si	Y- = X- and Y+ < X+	
9.	X during Y	d	Y- < X- and X+ < Y+	
10.	Y contains X	c or di	X- < Y- and Y+ < X+	
11.	X finishes Y	f	X+ = Y+ and Y- < X-	
12.	Y finished-by X	fi	Y+ = X+ and X- < Y-	
13.	X equals Y	= (eq)	X- = Y- and X+ = Y+	

Figure 1: “Allen” temporal interval relations

Seow and Devanathan (1993) [2] built on Allen's work by demonstrating how temporal logic programming can automate assembly sequence planning. They implemented the Mechanical Assembly Sequence Satisfiability Checker (MASS-C) in Prolog, using temporal operators as logic predicates to synthesize and verify assembly sequences. Their propositional temporal logic framework introduced operators like Precede (P), Until (U), Always (□), and Eventually (◇) to express precedence constraints, providing both expressive power and formal inference mechanisms for verifying sequence properties and automatically generating feasible sequences.

Temporal reasoning extends to active control and decision-making in manufacturing systems. Parthasarathy and Kim (1989) [3] developed a hybrid time representation combining time-point and time-interval languages, asserting that while interval-based representations offer expressive power, point-based representations provide efficiency advantages. In discrete event systems, Seow & Devanathan (1995, 1996) [4], [5] proposed a temporal logic approach to discrete event control using invariance formulae, defining controllability and synthesizing optimal supervisors by incorporating physical constraints and feedback mechanisms.

Zhu and Qiao (2015) [6] demonstrated that sequence selection significantly influences geometric deviation and assembly precision. Using homogeneous transformation matrices and Monte Carlo simulation, they showed that feature deviations change systematically with sequence choice, proving that optimal sequence selection enhances precision. Huang and Lee (1989) [7] and Ayoub and Doty (1989) [8] distinguish between hard and soft constraints. Seow & Devanathan (1994) [9] define hard constraints as geometric and physical relationships, while Baldwin et al. (1991) [10] identify soft constraints as preferences based on non-geometric criteria such as ease of assembly or ergonomic considerations. This relationship is formalized through temporal logic integration: if H represents hard constraints defining possible sequences M_H, and S represents soft constraints defining preferred sequences M_S, then desired sequences are obtained through $H \wedge S$, corresponding to $M_H \cap M_S$.

De Fazio and Whitney (1987) [11] formulated a precedence-based liaison model that generates all valid assembly sequences by encoding geometric accessibility, fastening order, and stability constraints into compact state–transition graphs. Gottipolu and Ghosh (1997) [12] introduced the Assembly Sequence Graph (ASG), which derives feasibility directly from CAD-based contact and motion constraints and represents alternative assembly paths within a unified graph structure that supports qualitative and quantitative evaluation. Homem de Mello and Sanderson (1988) [13] proposed a formal method utilizing a relational assembly model that transforms assembly into a disassembly problem, identifying feasible decompositions through cut-sets of the assembly connection graph and representing valid sequences in an AND/OR graph.

Manufacturing involves temporal uncertainty about event timing and relationships. Siebra and Wac (2022) [14] address this by extending ontological temporal representations with uncertain moments defined through membership functions and uncertainty qualifiers (almostSurely,

strongSurely, weakSurely, almostSurelyNot), maintaining compatibility with standard Web Ontology Language (OWL) and reasoning engines. McDuffie et al. (1995) [15] demonstrate practical handling of temporal constraints in fuzzy rule-based expert systems for flexible manufacturing scheduling. Giustozzi et al. (2018) [16] developed an ontology with temporal relations among resources, processes, and locations for Industry 4.0, differentiating temporal operators according to moments, time intervals, and their combinations.

Haage et al. (2017) [17] present a method for teaching assembly through demonstration, capturing temporal dependencies by identifying key frames and generating executable programs using Sequential Function Charts (SFC), facilitating the transfer of assembly skills from human demonstrations to automated execution.

To investigate allowable variation in manufacturing assembly routings in an educational context, this study seeks to extend prior work using a framework grounded in temporal logic with a methodology that allows process flexibility. The study is designed to examine how different work routings influence operator performance in assembly tasks, using temporal logic as the formal basis for defining which sequence variations are permissible. A secondary motivation for this preliminary study is the observation that assembly instruction in manufacturing education is often delivered as rigid, step-by-step procedures that obscure the temporal constraint structures that actually govern valid sequence execution.

The work serves two parallel purposes: 1) to formally model sequence flexibility in manufacturing operations and 2) to provide a basis for teaching undergraduate students about allowable variations in manufacturing environments. This positions temporal interval analytics not merely as a theoretical method but as a practical approach for both operational analysis and engineering education

2. Methodology

Preliminary work involves assembling a small robot designed for undergraduate engineering education. The activities completed during this stage include preparing, validating, and testing a manufacturing assembly cell. In the subsequent phase, undergraduate engineering students will act as assembly operators and record operational data as they execute alternative assembly routings.

The assembly cell station was designed for an academic teaching laboratory for use in an undergraduate manufacturing engineering course. As shown in Figure 2, the setup included a mobile toolbox with integrated drawers, a perforated pegboard for organized tool mounting, and a whiteboard of specified dimensions supported by a frame. The workstation also featured a desktop PC with a monitor and a set of clearly labeled drawers, supporting efficient part kitting. The layout was configured to support both standing and seated working positions, ensuring modularity, mobility, and ergonomic comfort throughout the assembly process.

Figure 2: Manufacturing assembly cell for drawing robot

An Arduino-based drawing robot from an educational engineering kit was chosen as a suitable product for the assembly task. The Arduino engineering kit is relatively inexpensive US $290, well-organized, and ideal for practical academic applications. Additionally, the kits include three robot platforms: drawing, rover, and bicycle. Selecting a drawing robot kit had several benefits. The kit comes with comprehensive documentation and video tutorials, and it is approved for educational use. Figure 3 displays the contents of the kit.

Figure 3: Arduino Engineering Kit Rev 2 (AKX00022)

Arduino provides a sequence of predefined tasks that guide users through essential robot functions, including introductory use of the Arduino Toolbox, establishing communication with a computer, executing basic movements, tuning control, and performing drawing routines such as rendering the MATLAB logo or generating custom images. These step-by-step tasks provided as MATLAB Live Script code were particularly effective in familiarizing users with the robot's operation and control architecture.

The Arduino platform aligns with core engineering learning objectives covering MATLAB/Simulink programming, kinematics, dynamics, mechatronics, control, image processing, and AI/ML computer vision. The required tools, such as screwdrivers, Allen keys, and wrenches, are simple and readily accessible. The system utilizes an Arduino microcontroller, a platform widely familiar to students, and includes multiple circuit boards and wiring harnesses.

The human-machine interface (HMI) for the assembly cell was designed to vary work instructions and the manufacturing process through a desktop PC. This allowed varied routings, sequences, text instructions, and graphic/visual aids to guide manufacturing and capture assembly data. This required developing a method to record start and end times for each task and creating a database capable of storing multiple trials representing different operators and

alternative routings. Data was collected using a spreadsheet, with a simple VB macro (triggered by Ctrl+T) to log timestamps throughout the assembly process efficiently. The interface and sample work instructions are displayed on the assembly cell PC in Figure 2.

To pilot test the data acquisition, database, and interface, ten trials of the actual assembly sequence were carried out by a graduate researcher familiar with the assembly process, and an undergraduate student conducted one trial to verify that the data collection procedure could be replicated across operators. The trial lasted two weeks, starting with one trial per day and gradually increasing to two trials per day as the operator became more comfortable with the process.

Detailed work instructions were developed for the Arduino drawing robot assembly process, as shown in Table 1. Each instruction outlines a distinct assembly task that was ordered according to the robot's manufacturer's specifications. Based on the manufacturer's instructions and the standard procedure for assembly and disassembly, these instructions were organized in a conventional linear order. The task description, list of necessary parts (in Table 2), tools for the assembly process, diagrams illustrating the proper assembly state, and images of the final assembly state, as depicted in Figure 4, are all included in the work instructions. The process consists of 17 discrete steps using step numbers 0 to 17 as identifiers. Step 15 was skipped in the majority of trials because it adds little value to the experiment. Step 0 is the initial state at the very beginning, when the process was about to start, and it was later used as a reference point in the flow process. Also, step number 999 was used to indicate a pause or rework in the operation, as all step numbers have to be numbers; therefore, any task not in the main assembly process has to be given a numerical identifier to make data processing easier later.

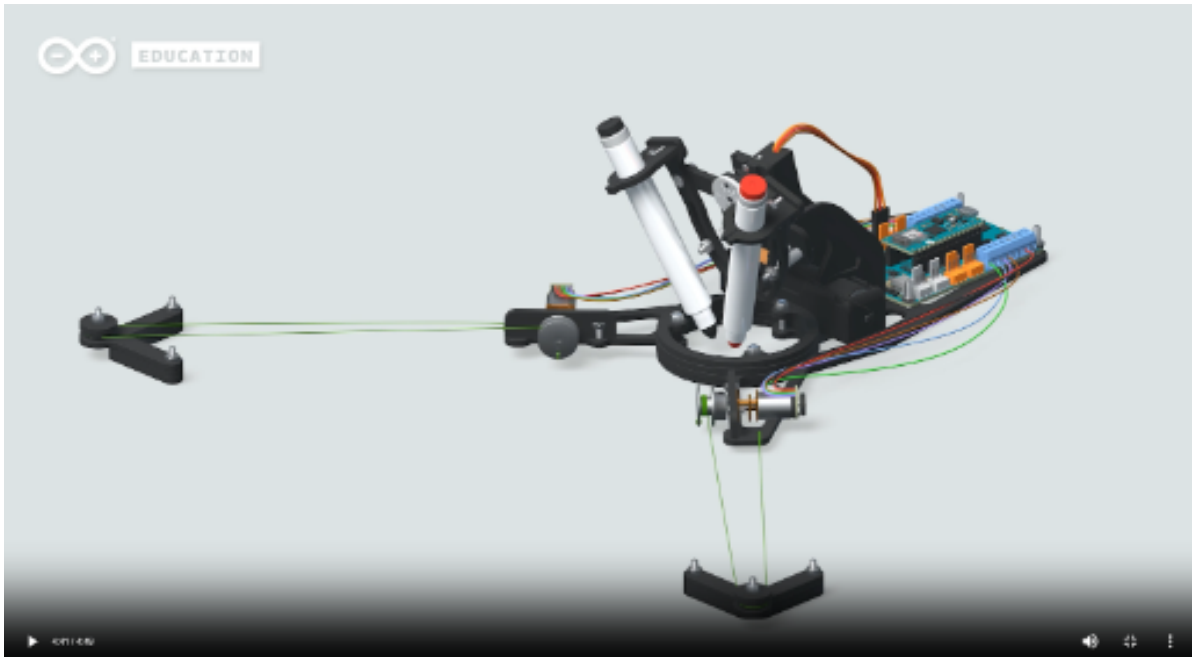

Figure 4 Final assembled state of the Arduino drawing robot

Table 1: Standard detailed work instructions of the Arduino drawing robot

Step No	Work Instruction
1	Join the motor mounting plate (M2) and two motor side plates (M3) using M3x 16mm bolts (H1)
2	Insert L6 mm distance spacer (H3) in the M3x16mm bolts (H1)
3	Place the gasket ring (M4) on the motor side plate and tighten with "M3 nutlocks" (H2)
4	Install the micro servo (G2) on the servo mounting bracket (M5) and tighen with M2x10mm bolt (H5) and nuts (H6)
5	Attach the round servo horn (M7) to the double arm servo horn (M6) using the servo screw (H13), and attach the couple to the micro servo bolt (H12)
6	Fix the mounting bracket arm (M8) to the servo mounting plate (M5) and tighen with M3x10mm bolts (H7) and nuts (H8)
7	Fix the two arm clip (M9) to the double arm servo horn (M6) and tighten with M3x10mm bolts (H7) and locknuts (H2)
8	Install L12 distance spacers (H4) between the arm clip (M9) and the servo mounting plate (M5) using M3X25mm bolts (H9) and locknuts (H2)
9	Fix the marker holder (M15) to the robotic arm clip (M9) using M3x10 bolt and nuts
10	Installing the complete set of 3rd servo on the base motor mounting plate (M2) using M3x10mm bolt (H7) and nut (H8)
11	Install geared motor bracket (M10) to the Micro Geared DC motor (G1) using M1.6x5mm (H10)
12	Install 1st and 2nd Micro Geared DC motor (G1) on the motor side plate (M3) using M3x10mm bolt /nut (H7 & H8)
13	Install the Nano Motor carrier (A2) on the base (M2) and wiring using M3x16mm (H1)
14	Fixing the pulley (M1) to the shaft of the two Micro Geared DC motor (G1) using a Grubscrew (H14) and an allen key (T7)
15	Coiling of the fishing rope (M16) on the pulley (M1)
16	Couple V-shape pulley holder (M11) and L-shaped pullley holder (M12) to form the board pulley using M3x16 mm bolt (H11) and nut (H8), Also add pulley spacer (M13) and pulley washer (M14) together
17	Insert the battery (H16)

Table 2: List of Parts in the Arduino Drawing Kit

Part Number	Part Name	QTY
	Nuts and Bolts	
H-1	M3x 16mm bolt	3
H0	M3 locknut	5
H1	L6 mm distance spacers	3
H2	L12 mm distance spacers	2
H3	M2x10mm bolt	2
H4	M2 nut	2
H5	M3x 10mm bolt	8
H6	M3 nut	14
H7	M3x25mm bolt	2
H8	M1.6x5mm bolt	4
H9	M3x16mm	6
H10	Micro servo bolt	1
H11	Micro servo screw	1
H12	Grub screw	2
H13	LI-ION Battery	1
H14	Battery holder	1
	DC and Servo Motors	
G1	Micro Geared DC motor W/Encoder	2
G2	Standard Micro Servo	1
	Arduino Board and Carrier	
A1	Nano 33 IOT	1
A2	Nano Motor carrier	1
	Mechanical parts	
M-1	Pulley	2
M0	Motor base mounting bracket	1

M1	Motor side plate	2
M2	Gasket ring	1
M3	Servo mounting bracket for Micro servo	1
M4	Double Arm Servo Horn	1
M5	Round Servo Horn	1
M6	Mounting bracket arm	2
M7	arm clip	2
M8	Micro geared bracket	2
M9	L-shaped pulley holder	8
M10	V-shape pulley holder	2
M11	whiteboard pulley spacer	2
M12	small round washer	2

	Additional Components	
C1	Whiteboard drawing pens	2
C2	5m thread	2

	Tools needed for Assembly.	
T-2	Philips screwdriver	1
T-1	2.5mm Flat head screwdriver	1
T0	Scissors	1
T1	Tape	1
T2	Pliers	1
T3	Whiteboard	1
T4	Allen key	

3. Data Analysis

A spreadsheet served as the database, where data were entered and stored. Eleven unique data points were collected for each task: EventID, Trial, StepNum, StartTime, EndTime, RelStart, RelEnd, Duration, T_i(s), Operation, and OperatorID. In each operation, Trial 1 served as the

baseline; the baseline sequence corresponded to the manufacturer's prescribed assembly order and was executed by the graduate researcher during the first trial. This baseline was not assumed to be optimal; rather, it served as a consistent reference sequence against which temporal and precedence deviations could be measured. The baseline trial likely reflects both manufacturer recommended ordering and initial learning effects. Accordingly, it is treated as a structural reference, not as an efficiency benchmark. So, the assembly process was assembled in the same order to form a baseline that was used for sequence metric computation later in the study.

There are a total of 11 pilot trials in the preliminary part of the study; aside from trial 1, any other trials were conducted out of sequence. Table 3 shows an example of the database information. The order of operations in subsequent trials was randomized naturally as the operator explored alternative feasible sequences within known physical and logical constraints. This exploratory approach was intentionally chosen to mimic natural sequence flexibility rather than enforce artificial randomization that could violate assembly feasibility.

Table 3: Sample Data Collection Information of the Assembly Process

EVENT ID	TRIAL	STEP NUM	STARTTIME	ENDTIME	REL START	REL END	DUR	T_i(s)	OPERATION	OPERATOR ID
35	2	0	5:16:00 PM	5:16:00 PM	0:00:00	0:00:00	0:00:00	0	Assembly	957392
36	2	4	5:16:00 PM	5:22:00 PM	0:00:00	0:06:00	0:06:00	360	Assembly	957392
37	2	5	5:22:00 PM	5:29:00 PM	0:06:00	0:13:00	0:07:00	420	Assembly	957392
38	2	6	5:29:00 PM	5:32:00 PM	0:13:00	0:16:00	0:03:00	180	Assembly	957392
39	2	1	5:32:00 PM	5:35:00 PM	0:16:00	0:19:00	0:03:00	180	Assembly	957392
40	2	2	5:35:00 PM	5:36:00 PM	0:19:00	0:20:00	0:01:00	60	Assembly	957392
41	2	3	5:36:00 PM	5:43:00 PM	0:20:00	0:27:00	0:07:00	420	Assembly	957392
42	2	7	5:43:00 PM	5:51:00 PM	0:27:00	0:35:00	0:08:00	480	Assembly	957392
43	2	8	5:51:00 PM	5:55:00 PM	0:35:00	0:39:00	0:04:00	240	Assembly	957392
44	2	11	5:55:00 PM	6:07:00 PM	0:39:00	0:51:00	0:12:00	720	Assembly	957392
45	2	12	6:07:00 PM	6:09:00 PM	0:51:00	0:53:00	0:02:00	120	Assembly	957392
46	2	10	6:09:00 PM	6:16:00 PM	0:53:00	1:00:00	0:07:00	420	Assembly	957392
47	2	9	6:16:00 PM	6:18:00 PM	1:00:00	1:02:00	0:02:00	120	Assembly	957392
48	2	14	6:18:00 PM	6:41:00 PM	1:02:00	1:25:00	0:23:00	1380	Assembly	957392
49	2	13	6:41:00 PM	6:59:00 PM	1:25:00	1:43:00	0:18:00	1080	Assembly	957392

In Table 3, the EventID uniquely identifies each recorded action within the sequence data, while the Trial indicates the specific experimental or operational run in which the event occurred. StepNum represents the task's position in the workflow, and StartTime and EndTime record when the step begins and ends. RelStart and RelEnd are the start and end times, expressed relative to the beginning of the trial, to enable easy comparison across steps. Duration captures the total time spent performing the step in the format hh:mm:ss, and T_i(s) represents the conversion of the Duration format to seconds. The Operation field describes the specific task or activity performed, and OperatorID identifies the worker or agent who executed the step, enabling human-performance analysis and sequence optimization.

The knowledge acquired during the initial setup of the assembly was to follow the Arduino cloud course instruction step by step to familiarize the researchers or volunteers with the process. There were some observations during the process, such as making mistakes in coiling the robot pulley. Instead of counterclockwise with both pulleys in the same direction, it was done clockwise; at times, both pulleys coiled in opposite directions, which posed a challenge during robot operation. Some faults occur during the process as well, such as the servo motor failing or the equipment requiring the robot to maintain an accurate position. Communication between the robot and the PC requires either a USB cable or WiFi. It required a long USB cable to navigate all areas of the board, so the researcher attempted a WiFi connection to set up communication, but for an unknown reason, the connection timed out intermittently. A USB cable was used as a means of tethered communication in the experiment. In addition to the challenges of the experiment, some nuts were over- or undertightened, making the bearing surface too tight or too loose. Fishing line is a challenge because it is so small and easily tangles in the spool.

The core of the methodology involves analyzing temporal dependencies between assembly tasks. Among the 13 temporal relationships available from Allen interval algebra, only three temporal keys—before, after, and equal—were used in this analysis for preliminary purposes to analyze the precedence dependence.. For instance, A < B means Task A was completed before Task B began; conversely, B > A means Task B started after Task A was completed. A=B means Tasks A and B start and end at the same time, they both have an equal timespan. As noted in the analysis, < means before, > means after, and finally = means equal in temporal matrix intervals.

To formulate a sequence metric for assembly and to assess how much it differs from the baseline sequence, the ASDI metric was developed. ASDI is the Assembly Sequence Difference Index; the formula for the metric is as follows:

$$ASDI = TD + PD + PerfD \quad (1)$$

Where TD is the temporal deviation, PD is the performance deviation, $PerfD$ is the performance deviation. ASDI uses three different parameters as seen in Eqn (1). Temporal deviation, also called timing difference, is one of the parameters on which ASDI depends.

Temporal deviation measures how far the timing (start/finish times or duration) of each task in your new sequence differs from the baseline.

$$TD = \frac{1}{N}\sum_{i=1}^{N}\frac{|T_i^{new} - T_i^{base}|}{T_i^{base}} \quad (2)$$

Where T_i^{new} is the actual time of the task i, T_i^{base} is the time of the task i in the baseline, and N is the total number of tasks.

The precedence deviation, also known as order differencing, comes in the second term of equation (1). It measures the degree to which the order of operations changed, such as when part B was assembled before part A rather than after.

$$PD = \frac{M}{N_p} \quad (3)$$

Where M is the number of mismatched precedence pairs (pairs where order differs), and $N_p = \frac{N(N-1)}{2}$ is the total number of possible pairs of tasks.

To compute the precedence deviation, an Allen adjacent matrix table was computed for each trial of operations per operator ID. The table formulates the temporal dependence of each operation on other operations in the trial. The same was done for the baseline trial; in the baseline, the sequence flow was in order from SEQ1…SEQ17, while there was a deviation or out-of-order in the subsequent trials. The baseline matrix is symmetrical, and the diagonal is smooth throughout the matrix. Table 4 is the baseline relation of the assembly process. There is an equal "=" operation in Table 4 that shows that operation has the same temporal relation, e.g., SEQ1 = STEP1, because it is the same event that takes place at that interval of time. Trial 2 has a different order of sequence from baseline, as shown in Table 5; therefore, the diagonal is nonlinear.

Table 4: Trial 1 Temporal Relation of Baseline Assembly of Arduino Drawing Robot

	STEPS															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**16**	**17**
SEQ1	=	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ2	<	=	>	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ3	<	<	=	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ4	<	<	<	=	>	>	>	>	>	>	>	>	>	>	>	>
SEQ5	<	<	<	<	=	>	>	>	>	>	>	>	>	>	>	>
SEQ6	<	<	<	<	<	=	>	>	>	>	>	>	>	>	>	>
SEQ7	<	<	<	<	<	<	=	>	>	>	>	>	>	>	>	>
SEQ8	<	<	<	<	<	<	<	=	>	>	>	>	>	>	>	>
SEQ9	<	<	<	<	<	<	<	<	=	>	>	>	>	>	>	>
SEQ10	<	<	<	<	<	<	<	<	<	=	>	>	>	>	>	>
SEQ11	<	<	<	<	<	<	<	<	<	<	=	>	>	>	>	>
SEQ12	<	<	<	<	<	<	<	<	<	<	<	=	>	>	>	>
SEQ13	<	<	<	<	<	<	<	<	<	<	<	<	=	>	>	>
SEQ14	<	<	<	<	<	<	<	<	<	<	<	<	<	=	>	>
SEQ16	<	<	<	<	<	<	<	<	<	<	<	<	<	<	=	>
SEQ17	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	=

Table 5: Trial 2 Temporal Relation of Arduino Drawing Robot

	STEPS															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**16**	**17**
SEQ4	<	<	<	=	>	>	>	>	>	>	>	>	>	>	>	>
SEQ5	<	<	<	<	=	>	>	>	>	>	>	>	>	>	>	>
SEQ6	<	<	<	<	<	=	>	>	>	>	>	>	>	>	>	>
SEQ1	=	>	>	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ2	<	=	>	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ3	<	<	=	>	>	>	>	>	>	>	>	>	>	>	>	>
SEQ7	<	<	<	<	<	<	=	>	>	>	>	>	>	>	>	>
SEQ8	<	<	<	<	<	<	<	=	>	>	>	>	>	>	>	>
SEQ11	<	<	<	<	<	<	<	<	<	<	=	>	>	>	>	>
SEQ12	<	<	<	<	<	<	<	<	<	<	<	=	>	>	>	>
SEQ10	<	<	<	<	<	<	<	<	<	=	>	>	>	>	>	>
SEQ9	<	<	<	<	<	<	<	<	=	>	>	>	>	>	>	>
SEQ14	<	<	<	<	<	<	<	<	<	<	<	<	<	=	>	>
SEQ13	<	<	<	<	<	<	<	<	<	<	<	<	=	>	>	>

After the Allen table is formed for the baseline, the next step is to create the comparison table that compares the subsequent trial to the baseline. The same approach was done for the baseline as well, comparing baseline to baseline yields 0 throughout the matrix, since there is no mismatch in the trial, but for the second trial, where a mismatch exists, 1 is used to represent a mismatch, i.e., 0 = same Allen relation as baseline, 1 = deviation from baseline. Summing all the ones in the matrix gives the total mismatch or precedence deviation that is present in the trial.

Table 6: Baseline – Comparison Table

	STEPS															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**16**	**17**
SEQ1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The baseline comparison table contains only zeros because it compares the Allen relation matrix of the baseline sequence against itself. Since no task order or temporal relationships differ, all precedence relations remain identical, resulting in a matrix with no mismatches. In contrast to the baseline (in Table 6), the comparison table for Trial 2 (in Table 7) shows several “1” entries, indicating deviations from the baseline. Each “1” reflects a pair of tasks whose temporal or precedence relationship changed because the operator performed steps out of the prescribed sequence. Tasks that were executed earlier or later than in the baseline produced altered Allen relations (<, >, =), leading to mismatches across multiple rows and columns. These deviations confirm that Trial 2 was performed out of sequence and illustrate the natural flexibility present in the assembly process. The comparison matrices provide a clear quantitative indication of how operator behavior differs from the engineered baseline and where temporal reordering occurs.

Table 7: Trial 2 vs Baseline – Comparison Table

	STEPS															
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**16**	**17**
	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SEQ4	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
SEQ5	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
SEQ6	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
SEQ1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
SEQ2	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
SEQ3	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
SEQ7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SEQ11	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
SEQ12	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
SEQ10	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
SEQ9	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
SEQ14	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
SEQ13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

The number of tasks per trial differs because some trials did not include all 16 baseline tasks. Finally, there is the performance deviation, also known as the difference in output efficiency. This evaluates the new sequence's deviation in terms of outcome metrics like cost, time, energy, or resource usage.

$$PerfD = \frac{Metric_{new} - Metric_{base}}{Metric_{base}} \tag{4}$$

Where $Metric_{new}$ is the metric at the new sequence, and $Metric_{base}$ is the baseline metric. A generic name, "metric," was used in this part as a placeholder because the metric can depend on the factor being considered, such as assembly total cost, energy consumption, or cycle time, depending on the application. In this case, the cycle time was used as the metric for calculating performance deviation. The cycle time is the total time required to complete all tasks in the trial. Therefore, eqn (4) becomes

$$PerfD = \frac{Cycletime_{new} - Cycletime_{base}}{Cycletime_{base}} \tag{5}$$

Substituting Eqns. (2), (3), and (5) into the equation. (1) becomes

$$ASDI = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{|T_i^{new} - T_i^{base}|}{T_i^{base}}\right) + \left(\frac{M}{N_p}\right) + \left(\frac{Cycletime_{new} - Cycletime_{base}}{Cycletime_{base}}\right) \quad (6)$$

Since the overall system output is the cycle time for preliminary testing, a weakness of this metric is that Eqn (2) and Eqn (3) are correlated. Both terms are retained to preserve conceptual separation between task-level timing and overall system output, which can be output or cost in other cases. For simplicity of the preliminary analysis, the ASDI metric assumed equal weight of each term; however, future analysis will incorporate weights to optimize the effects of temporal deviation, precedence deviation, and performance terms.

4. Preliminary Results

The analysis of the drawing robot assembly process revealed significant temporal flexibility that was not apparent in the manufacturer's work instructions. Of the 16 tasks in the assembly, 10 distinct dependency chains have been identified and could be flexibly reordered within constraints. The adjacency matrix representation clearly shows clusters of independent tasks, such as motor assembly subsystems (left and right motors can be assembled in parallel), electronics installation (power, board, and wiring have minimal interdependencies), servo installation, and bracket and gasket installation.

The graph in Fig. 5 shows the relationship between the Temporal Deviation (TD) and Performance Deviation (perfD) across the ten trials. Both metrics exhibit a similar trend, showing a strong correlation between the two metrics since performance is based on cycle time. For the assembly case, trials 2 and 6 exhibit negative deviations, indicating that the trial took a longer time to complete than the baseline trial. In contrast, trials 9–11 show the peak performance in time, despite significant task reordering. The positive deviation in other processes in Fig. 5 indicates that the tasks were completed in less time than the recommended baseline. The graph demonstrates that out-of-sequence execution does not compromise productivity but instead shows acceptable flexibility in the assembly system. For disassembly (red line), deviations are generally lower than in assembly. Trial 6 shows a dip in both metrics. From Trial 8 onward, performance rises, reflecting significant reordering and timing variation alongside improved performance.

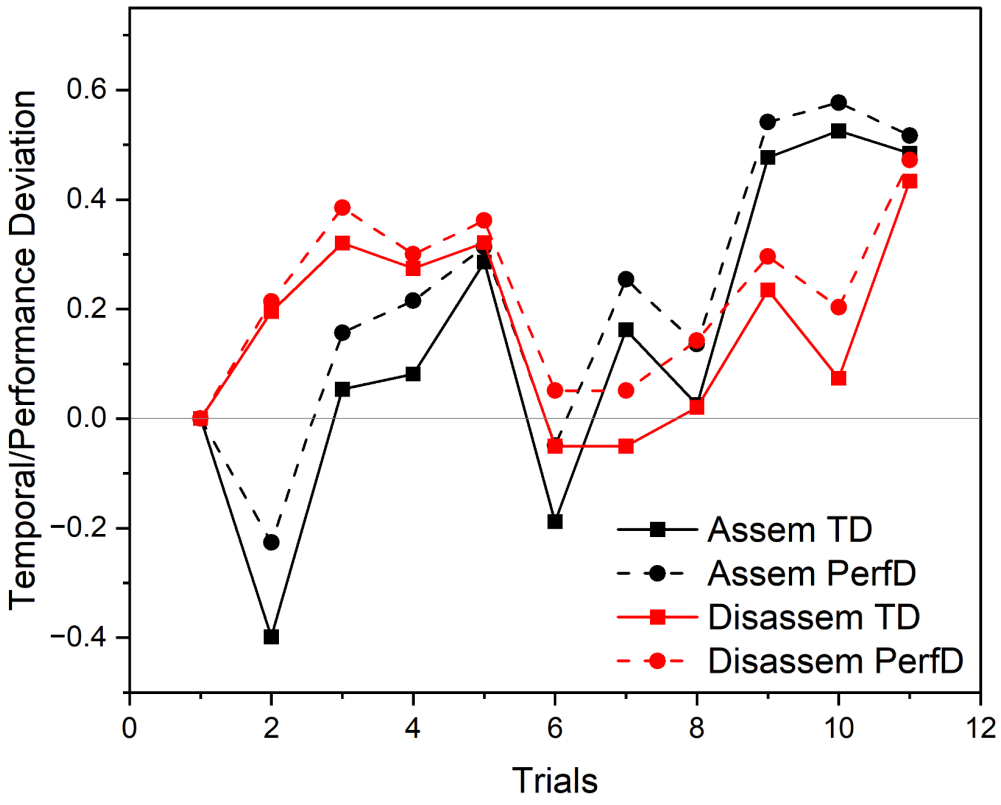

Fig. 5 Temporal and Performance Deviation of the Assembly and Disassembly

Performance improvements across trials reflect both operator learning and sequence adaptation. Because the same operator conducted most trials, learning effects cannot be fully separated from sequence efficiency. Rather than isolating these factors, this study treats learning as an inherent part of flexible manufacturing. As experience increases, operators not only work faster but also identify and apply alternative valid sequences. Thus, learning and sequence flexibility are considered coupled phenomena, consistent with real-world manufacturing, where adaptation in timing and ordering occurs simultaneously.

The precedence deviation graph in Fig. 6 shows how much each trial's task order differs from the baseline sequence. A value of 0 represents perfect adherence to the baseline precedence, while higher values indicate a greater number of out-of-sequence tasks. As expected, Trial 1 records a deviation of zero since it is the baseline reference. Trial 3 exhibits a sharp increase, reflecting substantial out-of-sequence execution. Subsequent trials maintain moderate to high deviation levels, with values generally between 0.4 and 0.85 in assembly, and 1.0 and 1.2 in disassembly. This indicates that the operator completely assembled the robot differently relative to the baseline. The upward trend in the later trials (i.e., Trials 7–11) suggests consistent use of the alternative task sequence and confirms that operators rarely follow the baseline ordering.

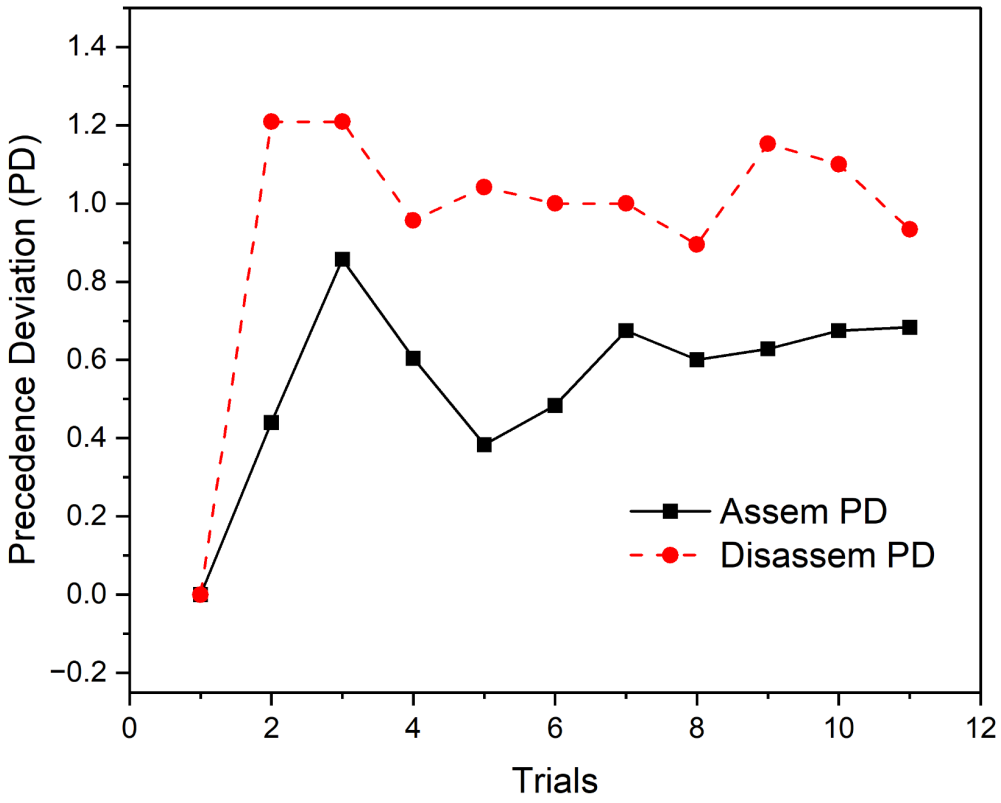

Fig. 6. Precedence Deviation of the Assembly and Disassembly Process

The ASDI of the assembly process shown in Fig. 7 exhibits a positive deviation and high reordering in all the trials except for trial 2, which shows a slightly negative deviation. The ASDI value in Fig. 7 does not indicate “good” or “bad” performance in isolation. Rather, higher ASDI values indicate greater deviation from the baseline structure. In this study, ASDI is used to characterize the magnitude of sequence variation, not to rank sequences as optimal or suboptimal. The important factor to consider is the operator's experience or learning curve, which helps yield higher ASDI as the operator repeats the task several times. The disassembly process consistently exhibits higher ASDI values than the assembly process, reflecting greater allowable sequence flexibility. This result reinforces the conclusion that disassembly processes benefit more strongly from flexible, non-linear sequencing models than assembly processes.

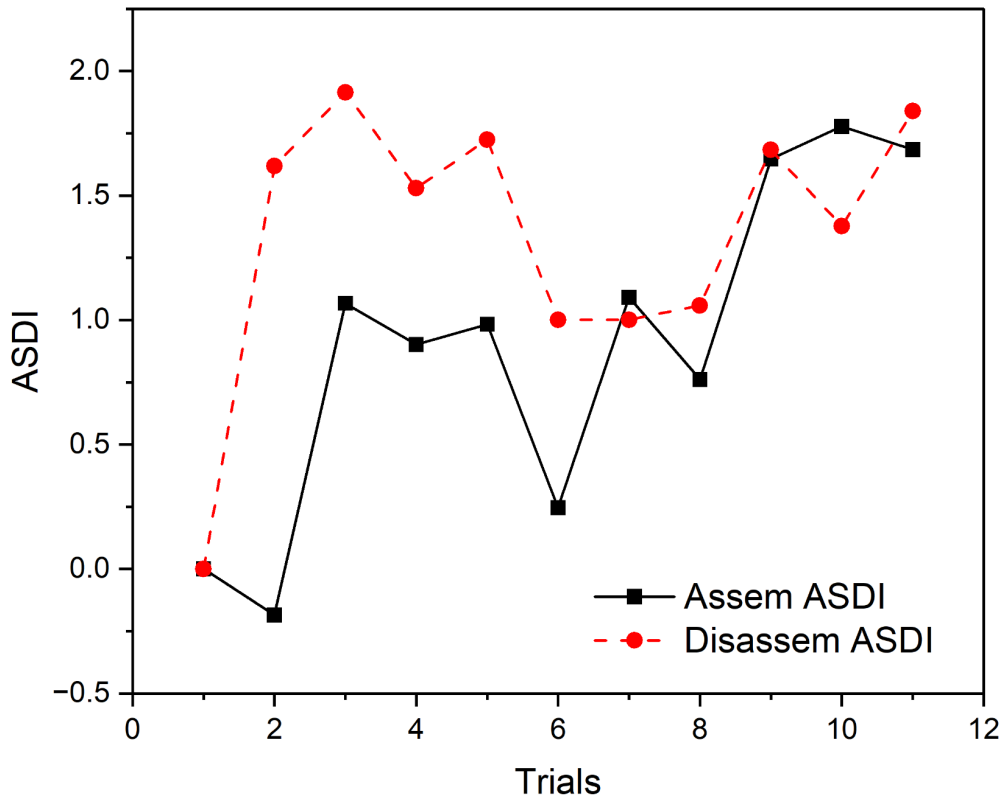

Fig. 7. ASDI of the Assembly and Disassembly Process

6. Conclusion

There are several ways to rearrange or carry out tasks without sacrificing assembly process efficiency, in contrast to traditional assembly, which is fixed and rigid. To determine how tasks are completed with alternate routings, the method presented in this study establishes the temporal relationships between events or operations for education applications. All study outcomes were compared using the Arduino-prescribed sequence as the baseline metric. It was discovered that because operator experience had an impact, the cycle time decreased as the assembly process was carried out iteratively. As the trial went on and the tasks were repeated, more expertise was gained. Unlike a repeated, rigid trial that is conducted in the same manner every time, the iterated trial was conducted in a different order each time.

By representing assembly processes using temporal logic rather than fixed linear instructions, students learn to reason about constraints, dependencies, and allowable variability in manufacturing systems rather than prescribed steps. The temporal logic approach strengthens systems thinking skills that generalize to scheduling, project management, and other digital

environments. Students are exposed to formal reasoning about time, precedence, and flexibility. The approach enables students to evaluate trade-offs, adapt to disruptions, and design resilient workflows. As such, the contribution of this work is not limited to performance outcomes but also to how manufacturing processes are taught, understood, and generalized across domains.

7. Future Work

This study is limited by a small sample size and the use of a single operator, both of which restrict statistical generalizability. These limitations are acknowledged explicitly, as the primary contribution of this work lies in the preliminary setup of the system for expressing Allen algebra in manufacturing and the development of manufacturing educational training rather than empirical optimization. Future work will involve multiple operators, randomized and counterbalanced sequences, and expanded use of Allen temporal relations to capture parallelizable tasks.

To allow exploration of processes with different temporal relations, additional workstations are being developed for wiring harness fabrication and quality testing. Strain relief processes being developed for the wiring harness station for example include gluing tasks that allow concurrent processing. A quality testing station will further allow refinement of the performance metric to address assembly problems, defects, rework, and/or functional failures such as the spool-winding issue. Future work will also integrate into a manufacturing execution system such as SAP. A 2-year trial academic license of Odoo has been secured to formalize work centers, work orders, and routings within a production planning software suite.

References

[1] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Commun ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983, doi: 10.1145/182.358434.

[2] K. T. Seow and R. Devanathan, “Temporal logic programming for assembly sequence planning,” *Artificial Intelligence in Engineering*, vol. 8, no. 4, pp. 253–263, Jan. 1993, doi: 10.1016/0954-1810(93)90008-4.

[3] S. Parthasarathy and S. H. Kim, “Temporal reasoning for manufacturing: A hybrid representation and complexity management,” *Robot Comput Integr Manuf*, vol. 6, no. 1, pp. 67–81, Jan. 1989, doi: 10.1016/0736-5845(89)90085-9.

[4] Kiam Tian Seow and R. Devanathan, “A temporal logic approach to discrete event control,” in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, IEEE, pp. 1435–1440. doi: 10.1109/ROBOT.1995.525479.

[5] K. T. Seow and R. Devanathan, “A temporal logic approach to discrete event control for the safety canonical class,” *Syst Control Lett*, vol. 28, no. 4, pp. 205–217, Aug. 1996, doi: 10.1016/0167-6911(96)00032-1.

[6] Z. Zhu and L. Qiao, “Analysis and Control of Assembly Precision in Different Assembly Sequences,” *Procedia CIRP*, vol. 27, pp. 117–123, 2015, doi: 10.1016/j.procir.2015.04.053.

[7] Y. F. Huang and C. S. G. Lee, “Precedence knowledge in feature mating operation assembly planning,” in *1989 IEEE International Conference on Robotics and Automation*, IEEE Computer Society, 1989, pp. 216–217.

[8] R. G. Ayoub and K. L. Doty, “A representation for discrete assembly sequences in task planning,” in *[1989] Proceedings of the Thirteenth Annual International Computer Software & Applications Conference*, IEEE, 1989, pp. 746–753.

[9] Kiam Tian Seow and R. Devanathan, “A temporal framework for assembly sequence representation and analysis,” *IEEE Transactions on Robotics and Automation*, vol. 10, no. 2, pp. 220–229, Apr. 1994, doi: 10.1109/70.282546.

[10] D. F. Baldwin, T. E. Abell, M.-C. M. Lui, T. L. De Fazio, and D. E. Whitney, “An integrated computer aid for generating and evaluating assembly sequences for mechanical products,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 78–94, 1991, doi: 10.1109/70.68072.

[11] T. De Fazio and D. Whitney, “Simplified generation of all mechanical assembly sequences,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 640–658, Dec. 1987, doi: 10.1109/JRA.1987.1087132.

[12] R. B. Gottipolu and K. Ghosh, “Representation and selection of assembly sequences in computer-aided assembly process planning,” *Int J Prod Res*, vol. 35, no. 12, pp. 3447–3466, Dec. 1997, doi: 10.1080/002075497194183.

[13] L. S. Homem de Mello and A. C. Sanderson, “Automatic generation of mechanical assembly sequences,” 1988.

[14] C. A. Siebra and K. Wac, “Engineering uncertain time for its practical integration in ontologies,” *Knowl Based Syst*, vol. 251, p. 109152, Sep. 2022, doi: 10.1016/j.knosys.2022.109152.

[15] E. L. McDuffie, M. Schneider, F. B. Buoni, E. Shnaider, M. Schneider, and L. A. Martin-Vega, “Using scheduling in flexible manufacturing systems,” *Eng Appl Artif Intell*, vol. 8, no. 6, pp. 681–688, Dec. 1995, doi: 10.1016/0952-1976(95)00052-6.

[16] F. Giustozzi, J. Saunier, and C. Zanni-Merk, “Context Modeling for Industry 4.0: an Ontology-Based Proposal,” *Procedia Comput Sci*, vol. 126, pp. 675–684, 2018, doi: 10.1016/j.procs.2018.08.001.

[17] M. Haage *et al.*, “Teaching Assembly by Demonstration Using Advanced Human Robot Interaction and a Knowledge Integration Framework,” *Procedia Manuf*, vol. 11, pp. 164–173, 2017, doi: 10.1016/j.promfg.2017.07.221.