JuniKhyat (UGC Care Group I Listed Journal)
ISSN: 2278-4632 Vol-8 Issue-02 Oct 2018

A Legacy-Controller-Based Solution for Building Resilient Assembly Lines with Minimal Overhead

Dilleep Kumar Bhadra
College of Engineering Bhubaneswar

***Abstract*— Because it is expensive and time-consuming to constantly purchase, program, test, and use new programmable logic controllers (PLCs), many factories continue to use possibly insecure legacy systems. The PLC error detection and correction method described in this letter makes use of shifted time redundancy, which is made possible by a complete crossbar connection network. Its goal is to successfully identify, isolate, and mitigate cybersecurity threats and malfunctions in PLC devices in a manufacturing environment where identical tasks are completed by numerous assembly lines operating in parallel. Each PLC's trustworthiness is rated using a trust scoring system, which also lowers required overhead.**

I. INTRODUCTION

OVER the last few decades, assembly line systemsand industrial processes have become more advanced, requiring more precise timing and control, and as a resulthave matured from human controlled processes to automatedsystems. With the advent of the micro-controller and embed-ded systems, programmable logic controllers (PLCs) presentedan absolute step change to the manufacturing sector, allowingprecise control to automate industrial processes and react tosystem fluctuations in real time. While the newer generationof PLCs presents significant advantages over legacy systems,companies are faced with a difficult tradeoff; purchase newdevices and undergo a complicated and expensive replacementprocess, or continue working with legacy devices, slowly pay-ing more over time as the legacy PLCs are less efficient, lessprecise, and more fault prone and unsecure. As PLCs age,they are more likely to experience random faults [1]. Morecrucially, they are opened to new attack vectors within thecybersecurity realm as a result of networking and modular design.

As with system-on-chip design, PLC design has become outsourced to third party design houses which produce intel- lectual property licensed by system integrators and make a customizeable chip for use in the PLC. The designed chip is produced by a fabrication facility and then deployed to the customer. At any other point in this supply chain malicious logic (usually known as hardware Trojans) can be embed-ded into the device. which could be so well hidden that even extensive testing fails to discover it.

Furthermore, as PLCs became more advanced, they were naturally networked together to facilitate advanced monitoringor control from remote locations. While these manufacturing and industrial networks are encouraged to be “air-gapped” (not connected in any way to the Internet as a whole) for security, not all of them are, and in some cases, such as the infamous Stuxnet malware [2], poor employee security practices can allow malware to “jump” the air-gap through the use of USB flash drives or other devices. In a factory or manufacturing setting this can cause massive economic loss or critical safety situations where human operators are endangered.

To handle errors in PLCs, one implementation [3] utilizesa single redundant sensor to provide stronger reliability to a geothermal system. Another group [4] utilizes three PLCs ina triple-modular-redundancy (TMR) configuration to control a heater in an industrial process. The work in [5] departs from these standard ideas and examines the control flow of the PLC, building a model and checking for control flow errors. The work in [6] takes a more nuanced look at the possible failure points within a PLC and suggests the areas to be improvedfor reliability enhancement. While these works are effectivein their own measure, they add overhead to the system or are too nuanced to be broadly applicable to factories.

This letter aims to successfully detect, isolate, and mitigate cybersecurity attacks and faults and in legacy PLCs. It focuseson a setting where multiple parallel assembly lines complete the same task, and proposes a solution which intelligentlyadds reliability to the setting via shifted time redundancy. A trust scoring system is also utilized to develop a novel “recy- cling” step, which prevents devices from being completely removed from the system if they only experienced a random transient fault. Individual PLCs not only can be abstracted from an assembly line but also are treated as nonpermutable blackboxes, allowing for both broad compatibility and thepotential for in-place piecemeal upgrades. The proposed PLC management solution has the following properties.

1) *Increased Reliability:* Jobs are executed on multiple redundant PLCs while meeting real-time constraints.
2) *Low Hardware Overhead:* The proposed shifted time redundancy enables multiple executions per output valuebut with fewer active PLCs.
3) *High Compatibility:* The proposed solution is suitable for legacy devices wherein it is impossible or impracticalto modify PLC code.

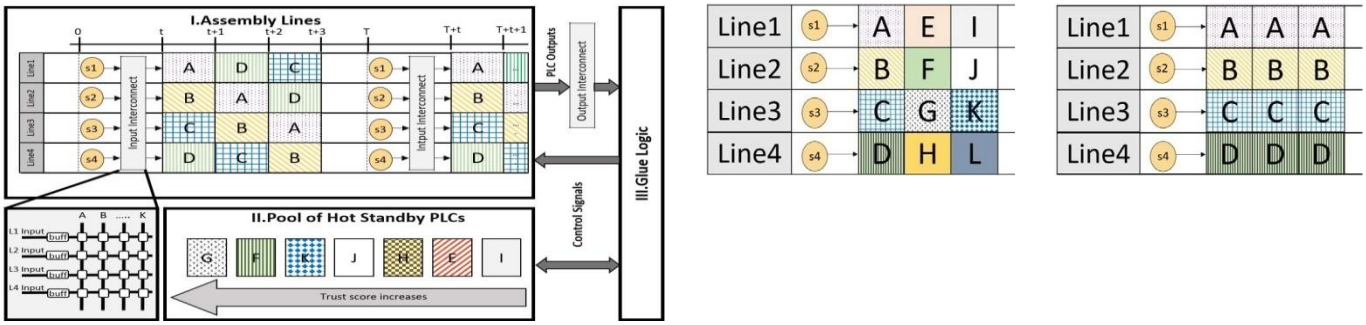

Fig. 1. Framework overview. *T* is the data sensing interval, *s*1–*s*4 representsensors in each assembly line, and *A*, *B*, *C*, and *D* are different PLCs.

The rest of this letter is presented as follows. Section II discusses the proposed design, Section III details testing and results, and finally Section IV concludes this letter.

II. Proposed Framework

This section presents the proposed framework which aims todetect, isolate, and mitigate cyber attacks and faults in assem- bly lines while satisfying the system's real-time constraints and minimizing the hardware overhead. An overview of the framework is first provided, followed by a detailed descriptionof its major parts.

A. Framework Overview

This letter is developed for a factory system with *N* assem- bly lines, each of which is equipped with a sensor, an actuator,and a PLC. The system performs the following process itera- tively: data sensing, where the input from each line's sensor isgathered and sent to a PLC; data processing, where a PLC exe-cutes the latest assembly line's job based on the latest sensed input; and activating actuators, where the PLC output is used toactivate the actuator. Outputs need to be ready before the new inputs are sensed. In other words, there is a real-time constraint*T* for the system which equals to the sensing interval.

The proposed framework detects, isolates, and mitigatesfaults and/or cyber attacks in assembly lines: 1) applying a shifted time redundancy algorithm to efficiently schedule redundant jobs on a series of PLCs; 2) using a nonbinary majority voting to detect faults; 3) isolating and mitigat- ing faults by selectively re-executing jobs; and 4) tabulating trust scores for PLC recycling and overhead reduction. Fig. 1 presents an overview of the proposed framework with four parallel assembly lines.

One advantage of the proposed framework is that it treats each PLC as a black-box and hence is compatible with many systems, especially those ones where the underlying PLC codecannot be modified. The framework incorporates a glue logic to produce the correct output for each assembly line's actuator,as shown in Fig. 1. The glue logic schedules jobs in the assem-bly lines following the shifted time redundancy scheduling algorithm, aiming to minimize hardware overhead. The logic also includes nonbinary voters to perform majority voting on(b)

Fig. 2. TMR uses 3*N* PLCs to control *N* assembly lines, while time redun- dancy uses *N* PLCs but can only handle transient faults. (a) Traditional TMR. (a) Time redundancy.

the PLC outputs and a set of counters to track the trust scoreof each PLC. A further function of the glue logic is to control accesses to hot-standby PLCs, sorted in descending order of their individual trust scores. Fig. 1 shows a hot-standby pool with 7 PLCs *(E K)* in which *G* is the most trusted. These PLCs can be used to replace untrustworthy PLCs running in the system.

It is important to deliver the inputs and outputs of the PLCs to the glue logic in a timely manner. To this end, the framework adopts the following two implementation policies. First, two full crossbar networks are utilized in the design, one on the input side and one on the output side. Fig. 1 shows an exampleof the input crossbar network. It is able to deliver the inputof each assembly line to any PLC device either active or inthe hot standby pool. Similarly, an output crossbar network connects PLC outputs to the majority voter of each assembly line. Second, the system adopts a two-level scheduling policy;a single master process (i.e., the glue logic) schedules PLCsto assembly lines, while the assembly line processes act as the slaves in the system and ensure the timely execution and data handling for their assigned PLCs. The master process utilizes the voting information returned by the assembly lines to adjustPLC trust scores and move PLCs into or out of the hot-standbypool.

To better understand the constraints of this design, we state our assumptions below.

1) While faults may appear in any device, they manifest in different ways. Two devices will never have the same faulty output for the same input.
2) Once activated, faults/attacks present themselves at the output of the device and can be either transient, specific to a single input or sequence of inputs, or permanent.
3) There exist enough PLCs to utilize as hot spares thatcan be brought on and offline on demand.
4) The logic is assumed to be secure and fault-free as it is designed in-house on a highly reliable device.

B. Shifted Time Redundancy

A conventional TMR system requires at least three redun- dant versions of a job to successfully detect and recover from one fault. For a system with N assembly lines, traditional TMRrequires $3N$ PLCs in total. Time redundancy, on the other hand,requires N PLCs to control N assembly lines, but is only able to tolerate transient faults. Fig. 2 shows a concrete example comparing these two schemes.

The shifted time redundancy outperforms time redundancy by providing hardware redundancy to each assembly line. It looks at the scheduling problem from a different point of

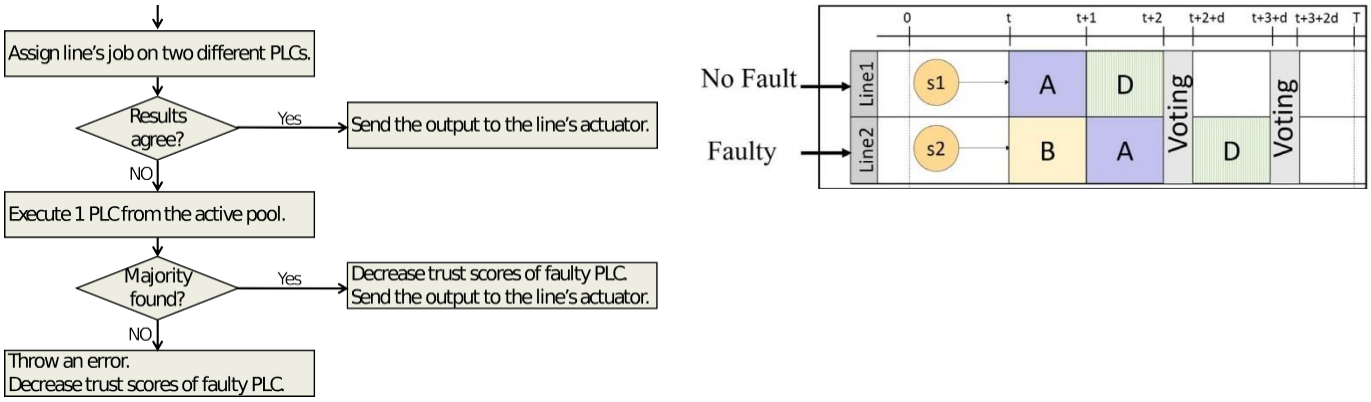

Fig. 3. Standard execution process flowchart for jobs on PLCs.

view. Since all the PLCs in the system perform the same func-tion and there is no dependency among jobs, an assembly line can be controlled with any PLC. Hence, the problem of scheduling jobs to PLCs can be converted to the problem of assigning different PLCs to different assembly lines. Fig. 1 presents a concrete example showing how PLCs are shifted across different assembly lines. In the first execution slot, the four PLCs A–D are, respectively, assigned to lines 1–4. In the second slot, A is shifted to line 2, B to line 3, C to line 4, and$D$ to line 1. Finally, in the third slot lines 1–4, respectively, uses PLC C, D, A and B. Only N PLCs are needed to con-trol N assembly lines, while within each sensing interval the redundant jobs on each assembly line are executed by differ- ent PLCs, thus achieving hardware redundancy and allowing the detection of both random errors and cyber attacks.

Assume a system with N assembly lines and M PLCs ($M > N$). T is the system sensing interval and t is execution latency of a job. Within each sensing interval there are Sexecution slots $slot_1, slot_2, \ldots, slot_S$ and $S = T/t$. To reach shifted time redundancy, the PLC to line assignment in each sensing interval T should satisfy the following constraint.

1) At $slot_k$ ($k \geq 1$), P_i ($1 \leq i \leq N$) executes the job of line $(i + k - 1)\%N$.
2) Let $(L_i, P_j)_{slot_k}$ be 1 if and only if PLC P_j is assignedto line L_i at $slot_k$. The overall assignment should satisfy

$$\forall i, j \quad \sum_{k=1}^{S} (L_i, P_j)_{slot_k} = 1.$$

Fig. 4. Example of different fault scenarios.

the nonbinary voters[1] provided by the glue logic. The system enters one of the following scenarios based on whether the results of the first two copies agree or not.

No Fault Scenario: If the voter shows no mismatch betweenthe two redundant copies, the agreed value is sent to the cor-responding actuator as the correct output and there is no needfor assigning a third PLC to the assembly line. Fig. 4 showsan example of this scenario in line 1. The voting process exe-cuted at time t 2 shows agreement between the results of $A$and D. Upon finishing the voting, there is no need to assignanother PLC to line 1 to execute the third iteration of the job.*Faulty Scenario:* If the results of two redundant executions do not agree, the glue logic schedules another redundant jobcopy in the third execution slot following shifted time redun-dancy. Fig. 4 shows an example of this scenario in line 2. Thevoting process shows a disagreement between the results of $B$and A, and PLC D is scheduled in the third slot to mitigate thedetected fault. Once the fault is located, the trust score ofthe faulty PLC is decremented. If the trust score drops belowthe trust score of the most trustworthy hot-standby PLC, it isreplaced by the latter. Note that the framework provides theability to mitigate more than one fault via assigning multiplePLCs in the third time slot, but the choice is left to the system supervisor.

In summary, the proposed fault tolerance framework com- bines the advantages of both TMR and time redundancy. On one hand, it provides the same fault tolerant capability as TMRas it is able to detect and mitigate N faults, one per assembly line. On the other hand, same as time redundancy, the proposedwork only imposes minimum hardware overhead, i.e., N PLCsare shared by N assembly lines.

D. Recycling Hot Standby Pool

As PLCs become labeled untrustworthy, they will neces-

sarily be moved to the hot standby pool and be replaced by more

trustworthy PLCs. The hot standby pool is kept sortedto ensure that the system can quickly find and utilize the most

This assignment rotates a PLC across N assembly lines one by one. In other words, for up to S ($S \leq N$) consecutive timingslots, a PLC will not be assigned to the same assembly line twice. Furthermore, the assignments in the first two slots are purely given by the constraint, while at $slot_3$ and later PLCs are assigned to lines only if needed. If more than one PLC is available, the most trustworthy PLCs will be used.

C. Fault Detection, Isolation, and Mitigation

The procedure of detecting, isolating, and mitigating cyberattacks and faults is represented in the flowchart of Fig. 3.

Once the first two copies of a job are finished at the end ofthe second execution slot, the two results are compared usingtrustworthy PLCs.

As we make the optimistic assumption that faults causedby cybersecurity threats may not be permanent, PLCs in the hot standby pool are periodically tested to determine whether their faults were temporary. These PLCs are tested by schedul-ing them into empty execution slots on an assembly line with no faults. If a PLC agrees with the agreed value, its trustscore is increased. If the PLC produces a different output, itis marked as permanently fault and removed from the system. For each failed device, the exact inputs that caused it to fail

[1]A nonbinary voter [1] with a predefined margin of error σ counts twoinput values in1 in2 as agreeing if $|in1 - in2| \leq \sigma$. The value of σ can be adjusted to loosen or restrict the error margin. A σ of 0 results in the functionality of a binary voter.

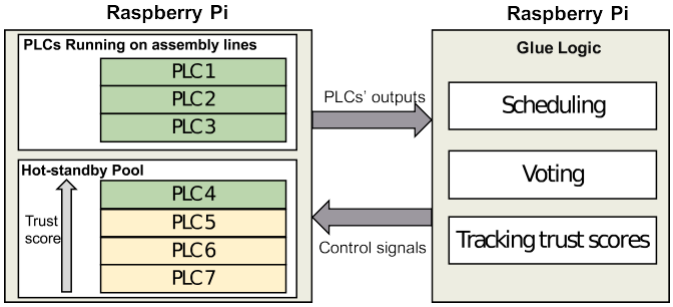

Fig. 5. Hardware setup overview.

are recorded. This provides the ability to potentially deter- mine specific triggers for each device (if the fault is the result of a triggered Trojan) and adapt with this knowledge, test-ing whether the device continues to fail on this input and necessitating its removal from the system.

III. Implementation and Results

The proposed fault tolerance framework was tested in both hardware and software environments. The hardware environ- ment (utilizing OpenPLC [7]) proved the feasibility and timingof the system, while the software environment tested different fault probabilities.

A. Hardware Implementation

The hardware-based implementation was completed with two Raspberry Pi 3s as shown in Fig. 5. One Raspberry Pi 3 was utilized as the "glue logic" and controller module. This device ran a python script that implemented the different partsof the algorithm, including scheduling, voting, and tracking ofPLC trust scores. A second Raspberry Pi 3 device ran sim- ulated PLCs on an OpenPLC server. Both Raspberry Pi 3s contained a quad core ARM A53 running at 1.2 GHz.

Testing in hardware was conducted by injecting faults viaa python script that accessed the general purpose I/O pins on the Raspberry Pi and inverted them. Two permanent fault sce-narios were examined. The base case was a single fault in the active pool. To pass this test, the system must detect a mis- match between the values returned by two active PLCs (fault detection), schedule a third PLC to break the tie (fault mitiga- tion), and lower the score of the offending PLC and replaceit with a PLC from the hot standby pool (fault isolation). Inall tested cases, the system successfully captured this type of faults within the very first iteration of the program withoutany error.

A far more difficult case was one fault in the active pooland one fault in the top rated hot standby PLC. To pass this test, the faulty PLC replacement must be executed twice. First,the faulty PLC in the active pool must be replaced. Since a fault was into the replacement PLC as well, the system must detect this fault at the next iteration and replace the PLC again.Again, this test was passed without any error, proving that the system was capable of executing each aspect of a fault tolerantassembly line controller.

TABLE I

TYPES OF FAULTS INJECTED AND SYSTEM ACCURACY

Fault Type	Fault Rate	% of Successfu Decisions	Overhead
Transient	2.0%	99.9%	$1.02N$
Transient	10.0%	97.4%	$1.18N$
Permanent	100%	99.9%	$1.003N$

B. Software Simulation

To test more advanced fault scenarios, a simulator was cre- ated. Within the simulator, PLCs can be adjusted to have a specific error model, assembly lines can have different input values to assign to each PLC, and the overhead of the system can be evaluated.

To determine how the system reacted to permanent and tran-sient faults, simulator testing was conducted with different fault profiles as shown in Table I. The investigated situa-tion was 5 assembly lines with 5 standby PLCs running for 5000 iterations. Testing consisted of random transient faults and permanent faults. Results demonstrated that the proposed framework is able to mitigate faults with very high accuracy. With a 10% fault rate, which is excessive for random faults within a system, 97.4% of decisions were correct. When low- ering the fault rate, the system further improves to the pointof a 99.9% success rate. The execution overhead is as low as $1.02N$, with N denoting the number of assembly lines.

IV. CONCLUSION

In this letter, a system for identifying, severing, and lessening defects in blackbox legacy PLCs in an industrial environment was proposed. Utilizing a full crossbar, a trust scoring system, and a pool of hot standbys, we give an adaptive and low overhead dependable solution to the threat presented by legacy PLC controllers. The outcomes of the experiments demonstrated that the suggested architecture, which treated devices like a blackbox and used fewer redundant devices than TMR, could mitigate up to 99.9% of injected defects.

REFERENCES

[1] I. Koren and C. M. Krishna, *Fault Tolerant Systems*. San Francisco, CA, USA: Elsevier, 2007.

[2] N. Falliere. *Exploring Stuxnet's PLC Infection Process*. Accessed: Sep. 2010. [Online]. Available: https://www.symantec.com/connect/ blogs/exploring-stuxnet-s-plc-infection-process

[3] G. Gabor, D. Zmaranda, C. Gyorodi, and S. Dale, "Redundancy method used in PLC related applications," in *Proc. 3rd Int. Workshop Soft Comput. Appl.*, Jul. 2009, pp. 119–126.

[4] V. Tipsuwanporn, A. Sangrayub, T. Suesut, A. Numsomran, and S. Gulphanich, "Development of PLC fiber-optic network for redundant system," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2002, pp. 303–306.

[5] N. Rajabpour and Y. Sedaghat, "A hybrid-based error detection techniquefor PLC-based industrial control systems," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2015, pp. 1–7.

[6] L. Huai and C. Cheng, "Reliability design of PLC-based control system,"in *Proc. 9th Int. Conf. Nat. Comput. (ICNC)*, 2013, pp. 1671–1675.

[7] T. R. Alves, M. Buratto, F. M. de Souza, and T. V. Rodrigues, "OpenPLC: An open source alternative to automation," in *Proc. IEEE Glob. Humanitarian Technol. Conf. (GHTC)*, San Jose, CA, USA, 2014, pp. 585–589.