A Programmable Open Architecture Testbed for CPS Education

Sumana Ghosh[1], Arnab Mondal[2], Philipp H. Kindt[1], Prateek Sharma[2], Yash Agarwal[2], Soumyajit Dey[2], Alok Kanti Deb[2], and Samarjit Chakraborty[3]

[1]Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany. E-Mail: sumana.ghosh@tum.de, philipp.kindt@tum.de
[2]Indian Institute of Technology (IIT) Kharagpur, Department of Computer Science and Engg., Kharagpur, West Bengal - 721302, India. E-Mail: amondal23@iitkgp.ac.in, mailtopraty@yahoo.com, yashaga98@gmail.com, soumya@cse.iitkgp.ac.in, alokkanti@ee.iitkgp.ac.in
[3]University of North Carolina at Chapel Hill, Department of Computer Science, 201 S. Columbia St., Chapel Hill, NC 27599-3175, USA. E-Mail: samarjit@cs.unc.edu

The topic of Cyber-Physical Systems (CPS) is very interdisciplinary and covers a wide variety of applications. A large class of CPS involves the combination of control theory and embedded systems. Many CPS curricula aim to teach different aspects of how designing control algorithms and embedded systems impact each other. Building an experimental platform to illustrate this interdependency is however non-trivial. A purely theoretical exposition on this topic without any real experiments is also not effective. To address this, in this paper, we propose a novel programmable testbed that models complex, networked CPS with minimal design effort. As the heart of this testbed, we propose a software tool that provides an easy-to-use interface for specifying various CPS configurations and implementing them in hardware.

***Index Terms*—Cyber-Physical Systems, CPS, Testbed, MCN, Control, Education, Wireless Network**

I. Introduction

In Cyber-Physical Systems (CPS), the behavior of the entire system is defined by the complex interactions between control algorithms, the hardware platform, and the communication network connecting the different devices. In particular, CPS in which controller and plant are connected through a multi-hop wireless network are deployed frequently, since they offer a higher degree of flexibility, lower maintenance and installation cost, and a better adaptability than conventional CPS [2], [4]. While a lot of research has been done on wired as well as wireless CPS, there are no platforms available that facilitate education of distributed CPS systems in general and wireless CPS in particular.

CPS Education: Understanding and developing CPS requires interdisciplinary knowledge from Computer Science, Mathematics, Electrical Engineering, Mechanical Engineering and other domains. Different design choices affect different layers of abstraction, and the complex interactions among multiple layers are often difficult to understand. Providing a good understanding of such cross-layer effects is of fundamental importance in CPS education and training [6], especially in the light of safety-critical CPS.

However, establishing a deeper understanding of cross-layer interactions that occur in real-world CPS cannot be achieved without performing experiments on feature-rich testbeds [7], which expose a future practitioner to the non-ideal characteristics of CPS implementation platforms and their effects on performance and safety. However, many of the available testbeds consist of simplistic setups, which typically comprise only a simple plant and a single microcontroller. Here, the behavior of the testbed does not differ significantly from what is predicted by the theory, because non-idealities occurring in real-world setups are not present. As a result, they are only of limited use for conveying the necessary insights into practical CPS.

Testbed Development: A networked CPS consists of multiple nodes with different roles (e.g., plant, controller, intermediate node, router, etc.), which all require their customly developed firmware. Moreover, these nodes have to be interconnected through a wireless network, which requires further design effort. As a result, developing such a setup from scratch remains a very complex and time-consuming task. Despite suitable text book material on embedded and cyber physical systems [5], in many courses and institutions, training on practical CPS is eluded. This leads to a gap between the growing importance of CPS on the one hand and appropriate training on the other hand.

This Paper: In this paper, we address this problem and present a novel programmable testbed for specifying, implementing, and analyzing various networked CPS configurations in a user-friendly way. The testbed provides a unique opportunity to design, run, and validate wireless control algorithms for different networked CPS configurations through hands-on experiments. The proposed testbed is complex enough to exhibit hardware effects like timing delays, packet drops, etc., which are also present in real-world setups. As a result, students and CPS practitioners can validate the robustness of their control algorithms against these non-idealities. Since our main goal is exhibiting this behavior for educational purposes, we have additionally included the option to insert programmable, artificial faults.

Proposed Testbed: At the heart of our testbed, there is a software tool that supports a graphical user interface (GUI)-based easy-to-use front-end for specifying a given CPS configuration, and then implementing it on real hardware. Based on a CPS specification, which is given in terms of network parameters, control parameters and plant dynamics, our tool automatically generates and deploys the firmware of the individual nodes in the network. It thereby configures the nodes, as well as their network interfaces. Each such node is formed by a widely-used Arduino UNO microcontroller, which is equipped with a nRF24L01+ RF module [1]. Some of these nodes take over the role of a controller, whereas

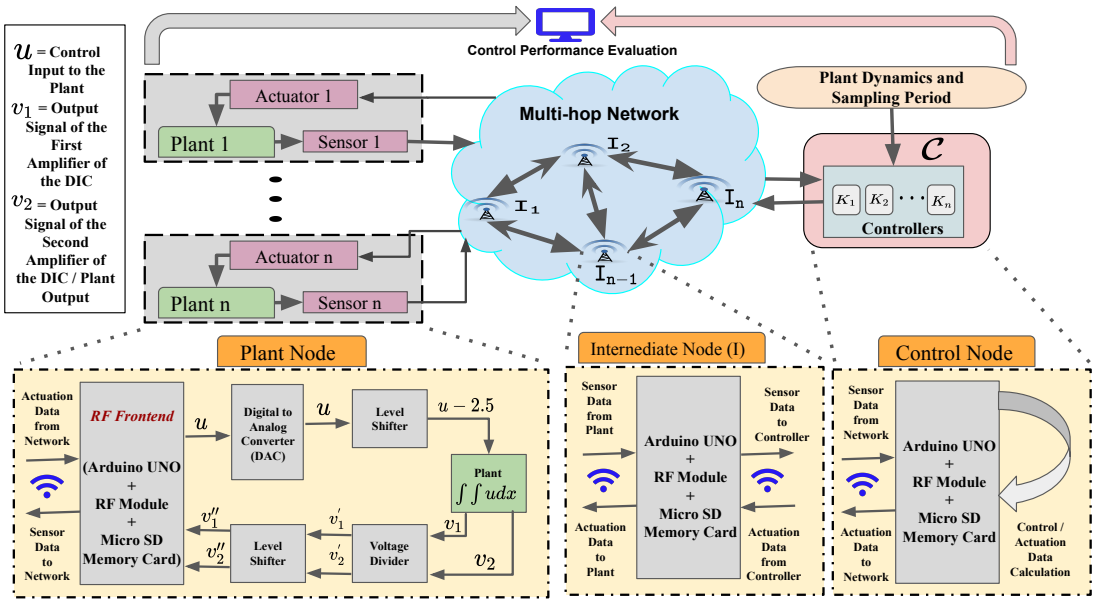


Figure 1: Overview of our proposed CPS-testbed.

others are equipped with some physical setup to be controlled (the physical plant), or act as intermediate nodes that relay information. The firmware images are deployed over-the-air, which reduces the effort of testbed deployment. After these steps, a testbed that implements the specified CPS configuration and failure characteristics is ready to use. Further, during the experiments on the resulting CPS, every node records and stores data. After the experiment, this data is wirelessly transmitted to the PC, on which our proposed tool orchestrates data collection, aggregation, and visualization. The data can be *replayed* for analyzing events of interest (e.g., specification violations, unstable behavior, etc.). The salient features of our testbed are summarized as follows.

1) The testbed supports specifying and analyzing a CPS configuration in an easy-to-use manner using a novel GUI-driven software tool with minimal design effort.
2) Artificial faults can be injected by specifying the probability distributions for packet loss and other errors.
3) Experimental data and execution statistics of each node can be retrieved with zero effort. Based on them, the platform's and controller's behavior can be evaluated.

The testbed thus targets educational and teaching purposes. The extremely low design effort greatly reduces the development cost and time. The testbed allows studying various CPS concepts by carrying out different experiments through an easy-to-use interface. The required hardware consists solely of widely-available off-the-shelf components (e.g., Arduino), thereby making the testbed accessible and low-cost.

Paper Organization: The rest of this paper is organized as follows. In Sec. II, we describe the relevance of our testbed in CPS education. In Sec. III, we present our proposed testbed in detail. Next, in Sec. IV, we demonstrate multiple teaching experiments for learning different aspects of real-world CPS, and present the concluding remarks in Sec. V.

II. CPS Testbed for Education

Fig. 1 shows the overview of our proposed testbed. A set of n physical plants are controlled by n controllers running in parallel on a dedicated node, called the *control node*. Intermediate nodes connect plants and the control node by realizing a muti-hop wireless network. Each plant can be a real-world physical system or a Simulink model. Currently, our tool supports Double Integrator Circuits (DICs) as a physical plant. However, support for other physical plants can be added easily in the future. A DIC is an operational amplifier-based circuit that computes the integral over the integral of its input signal. We have chosen a DIC as our first supported plant, since it forms a standard control-theoretic benchmark that is widely used in control engineering.

As already mentioned, the proposed testbed targets educational purposes. We have therefore implemented the following five primary features that make it particularly suitable for CPS education and training.

Low-Cost Components: Many of the existing CPS testbeds target research applications [8]. Here, the cost of the components is not a key aspect, since often, a single setup is sufficient. In contrast, hands-on educational formats are more cost-sensitive, since a larger number of course participants require multiple setups in parallel. Therefore, our testbed is built upon the off-the-shelf Arduino UNO microcontroller equipped with a nRF24L01+ RF module [1]. Besides the wide global availability of these components, they lead to a low cost of ≈ 30 USD per intermediate- or control node, and a cost of ≈ 45 USD per plant node. Overall, a CPS setup having one plant node, five intermediate nodes and a control node incurs ≈ 225 USD, which easily suits the typical budgets of most educational institutions.

Versatility: A testbed used for education and training should be versatile enough to support a large number of different experiments, which will convey knowledge on various aspects

of CPS. The proposed testbed can easily realize different CPS configurations by simply adjusting the following parameters and configuration options.

1) **Plant Parameters:** Number of plants and their types, given by their mathematical descriptions.
2) **Control Parameters:** Sampling periods and specifications of the different discrete controllers.
3) **Network Parameters:** Number of nodes, network topology, routing paths of all plant-control loops, number of parallel channels, number of re-transmissions of packets by each node, etc.

Realistic Behavior: The behavior of a CPS testbed should be as close as possible to that of a real-world setup. In particular, it should be suitable for studying the robustness of a CPS design in the presence of non-ideal behavior. Hence, it is inevitable that a testbed exhibits non-idealities such as packet drops, network delays, etc. Our proposed testbed is complex enough to exhibit such non-ideal behavior. In addition, it comes with a programmable fault insertion methodology, using which users can insert artificial faults that correspond to the errors described above. Hence, the robustness of e.g., a given control algorithm, can be tested easily.

Easy-To-Use GUI: A feature that makes our testbed unique compared to existing ones is its easy-to-use GUI-based interface. Based on this GUI, the following tasks can be controlled. 1) Automatized CPS configuration, 2) generation of firmware for each node, 3) wireless deployment of the firmware onto each node, 4) real-time execution and orchestration of the CPS, and 5) data aggregation and visualization. In other words, a user can specify its configuration in a simple fashion, and our tool will generate a CPS testbed realizing these specifications on-the-fly. This allows for studying CPS testbed configurations that are actually different CPS, while using one physical setup. With our approach, manually integrating plants, controllers and the communication network is no longer required.

Easy Development and Scalability: The proposed testbed can be developed with minimalistic development effort and is highly scalable. Extra hardware can be added or existing devices can be upgraded easily, without requiring a time-consuming software redesign. For example, a small network can be easily transformed into a larger network by adding additional nodes. The only required step is the GUI-driven configuration, specification and automated regeneration of the software. Furthermore, different network topologies, e.g., single- and multi-hop, can be generated easily.

The proposed system has already demonstrated its practicality for teaching at universities. In particular, it has been used in the courses entitled 1. *Computational Foundation of Cyber-Physical Systems*, 2. *Introduction to Programming Intelligent Physical Systems* at the Indian Institute of Technology (IIT) Kharagpur in 2019. More than 50 students from several disciplines, including CS, EE, and others attended both the courses. Many of them did not have any prior background in control theory and/or networking. Based on their feedback, we can claim that the participants found the setup helpful for understanding different aspects of CPS, e.g., communication, distributed controllers, different design options and performance issues under non-ideal conditions.

III. Proposed CPS Testbed

In this section, we describe our proposed software tool and the corresponding hardware details of the testbed.

A. Tool for Automated CPS Generation and Operation

The core of our proposed testbed is a software tool, which runs on a PC that is wirelessly connected to the CPS setup. Our software tool mainly fulfills the following two purposes.

1) Configuration and Firmware Generation

The GUI-based front-end is shown in Fig. 2a. It is used for specifying the CPS that is to be realized. Among others, aspects that can be configured are the network topology, routing paths of the control loops, reference values of the plant outputs, and properties related to fault injection.

The upper part marked '*A*' of Fig. 2a shows the main configuration window, which can be used for specifying the length of routing paths, the number of nodes, and the injection of artificial faults. In the middle Part *B*, inputs for specifying the routes of the control loops are provided. More specifically, the wireless channel is subdivided into multiple time-slots, and each slot can be used for transmitting data between a certain pair of sender and receiver on a certain wireless channel. The user can specify the set of nodes that form a route and the corresponding set of slots used.

The lower Part *C* of the figure is used for specifying probabilities of simulated packet loss and for specifying the target values of plant outputs (i.e., reference values). Based on these inputs and more elaborate specifications from configuration files, the software-tool generates and wirelessly deploys a firmware image for each node.

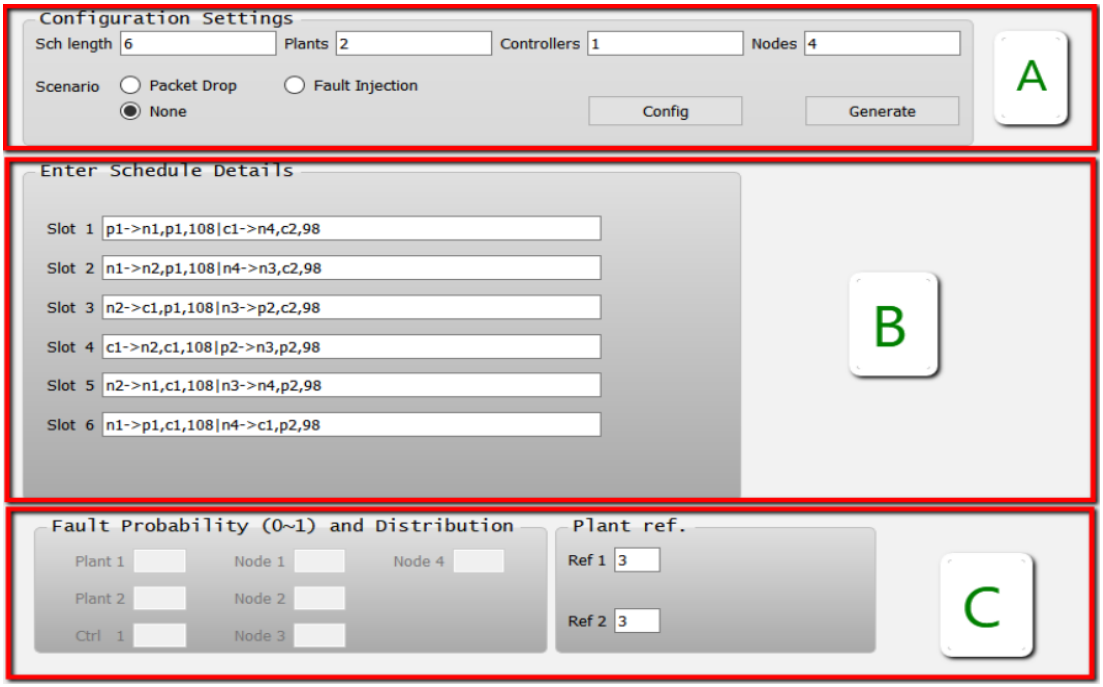
2) Data Analysis

After an experiment of the CPS has been carried out, the data is transmitted wirelessly to the PC and can then be analyzed using our software tool. Our tool orchestrates data collection and aggregation and provides functionality to visualize and analyze the CPS behavior. The front-end for data analysis is depicted in Fig. 2b. It offers the following functionality.

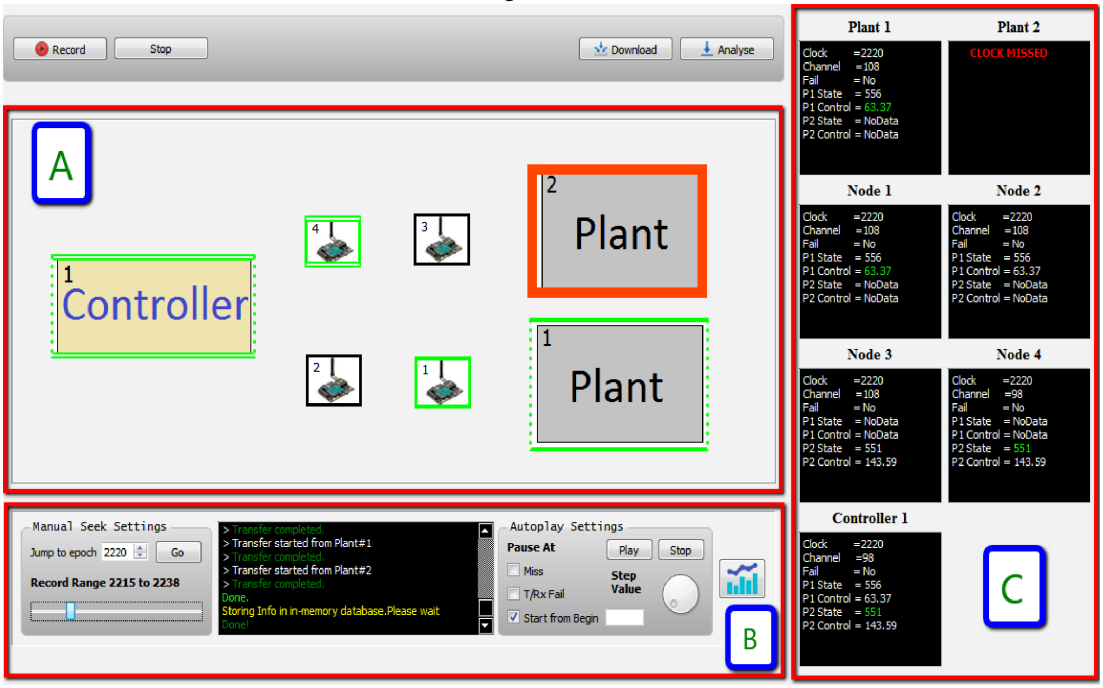
Record and Store Data: During an experiment, each node stores information on every network transaction on a SD memory card. After the experiment, this data is transmitted to the PC for analysis.

Visualize and Analyze: The upper Part *A* of Fig. 2b visualizes the CPS. Suitable icons represent plants, intermediate nodes and controllers. On request, the tool animates the actual CPS operation, based on the recorded data. As shown in Fig. 2b, Plant 1 and Node 1 are highlighted using green color, indicating a successful transmission between them. In the depicted example, Plant 1 is in reception mode, which is annotated by the "dotted" line. Similarly, Node 4 successfully transmits to the controller, which is indicated by the green double-frame. In contrast, Plant 2 faces a transmission failure as indicated by the red color.

On the left of the lower Part *B* of Fig. 2b, the user can select the point in time of the experiment to be examined. In the middle of Part *B*, a console depicts various messages. On the right of Part *B*, the *Autoplay Settings* allow for automatically



(a) Configuration front-end.



(b) Analysis Front-end.

Figure 2: The GUI-driven software tool.

advancing the point in time under consideration. This enables a record and playback - style of operation. Since an experiment typically generates a large set of data, the *Autoplay Settings* also provide a discontinuous and fast mode of replay. Here, the playback is interrupted whenever certain events, such as packet loss or communication errors, occur. The right Part *C* of the figure shows the internal state of each node at the point in time under consideration. For each node, the current timestamp, the corresponding channel, the most recently received data from each plant, the controller output and the failure/success status of the last executed transmission are depicted.

B. Hardware Implementation

We next describe key aspects of the testbed hardware. The different hardware components forming the testbed are depicted in Fig. 3.

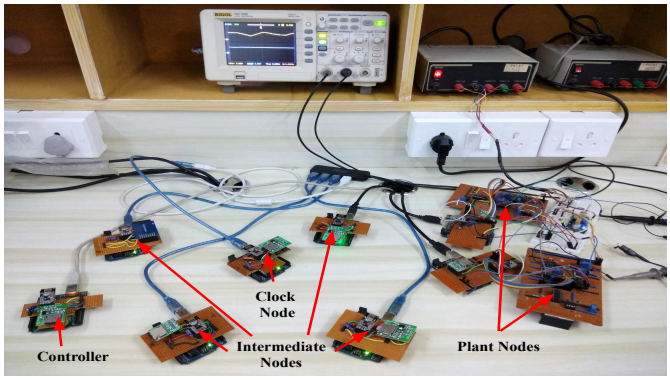


Figure 3: The CPS testbed consisting of a control node, four intermediate nodes and two physical plants.

1) Plant Node

As mentioned earlier, we consider a DIC as the first physical plant supported by our testbed. It consists of two operational amplifiers (UA741) connected in series, thereby computing the integral over the integral of the input signal of the first amplifier. The control objective is maintaining a given target voltage (i.e., the *reference value*) at the output of the second amplifier by adjusting the input voltage of the first amplifier. An overview of a plant node is depicted in the lower-left part of Fig. 1.

2) Control and Intermediate Nodes

Controller- and intermediate nodes consist of a microcontroller, RF interface and SD memory card. The microcontroller is entirely programmed using our software tool. On a control node, the microcontroller receives plant output, computes the control input using standard control design techniques such as pole placement [3], and then wirelessly transmits control input.

3) Global Clock Node

The clock node consists of the same hardware and provides a clock signal to which all nodes are synchronized. With a period equivalent to the slot length, a clock signal is broadcasted to all nodes. In addition, the clock node is also responsible for orchestrating the collection of experimental data over the network.

4) Communication Network

Given the GUI-based CPS specification, firmware for each node is generated and deployed by the software tool. This firmware is also responsible for controlling the connectivity. The wireless communication is built on top of the *Enhanced ShockBurst* protocol, which operates in the ISM band (2.400 - 2.4835 GHz). We use an over-the-air data rate of 250 kbit/s. As a congestion control mechanism, the NRF24L01P RF module implements up to 6 logical addresses (called *pipes*) for listening in parallel. A module can only transmit to one pipe at a time. In our setup, three pipes are maintained by each RF module for handling the global clock, plant/control and log data. Channel arbitration is done based on time-multiplexed media access. Time is subdivided into 10 ms-slots and the clock node transmits a synchronization pulse to all nodes every 10 ms. Upon receiving the clock pulse, each node first checks if there is any data to be processed. A node may initiate a transmission, whenever the current slot is assigned to this node for transmission. Similarly, a node will listen for incoming transmissions from other nodes during the appropriate slots. The clock node can also initiate data logging on a node, or request the transmission of the previously logged data.

IV. Demonstration of Teaching Experiments

In this section, we demonstrate how important curricular activities like CPS- and controller design, execution, performance evaluation, and refinement can be easily performed using our testbed. Five different teaching experiments, which can be used to study different aspects of CPS, are exemplified below. In each experiment, the testbed is executed for 15 s. We initially set the output voltage of both plants to 0 V, whereas the reference value is 3 V. Upon starting such an experiment, the controller will attempt to bring the output voltage to the reference value. The standard measure of quality of control in CPS is minimizing the *settling time*, which is the time until the 2% envelope around the reference value is reached.

1) Experiment 1 - Simple CPS

We demonstrate a simplistic CPS platform, in which a single-hop wireless network connects the two plants and the shared control platform. Here, each plant is sampled with a period of 20 ms and pole placement-based controllers are employed for both plants. Fig. 5 depicts the output signal of Plant 1, which is measured using an oscilloscope. As can be seen in Fig. 5, the system settles quickly within 3.303 s for Plant 1.

2) Experiment 2 - Multi-hop CPS

Here, we demonstrate a more involved CPS model, in which plants and controllers exchange messages through a multi-hop wireless network. Due to the limited receiver sensitivity, platform level data loss occurs (e.g., $\sim$1% in our example). We realize the topology of the wireless network depicted in Fig. 4, with two plants P_1 and P_2, and two controllers running on the shared control node C. The routing paths for the pair of control loops are depicted by the dotted lines between P_1 and C and between P_2 and C. As mentioned in Sec. III-A1, in each slot, one transmission between one pair of nodes per channel is possible. Recall that transmissions occur with a slot length

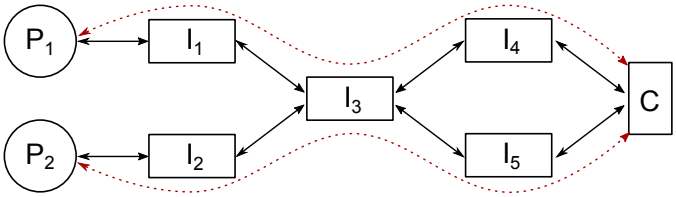


Figure 4: Network graphs for Experiments 2, 3, 4 and 5.

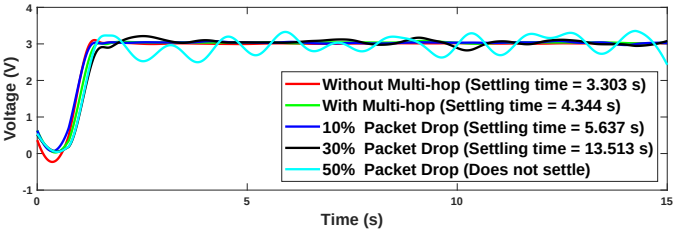


Figure 5: Output of Plant 1 for all Teaching Experiments.

of 10 ms. For the control loop between P_1 and C, a round-trip delay of 80 ms is induced by the 2×4 hops. For the loop between P_2 and C, Node I_3 will be busy with relaying data for P_1 for two slot lengths, and hence, the round-trip delay for this loop becomes 100 ms. We therefore choose sampling periods of 80 ms and 100 ms for controlling P_1 and P_2, respectively, and apply pole-placement based controllers. As can be seen from Fig. 5, the settling time for Plant 1 has increased to 4.344 s as a result of the increased sampling period of 80 ms. The settling time of Plant 2, as reported in Table I, exhibits a similar behavior.

3) Experiment 3 - Impact of Different Controllers

Experiment 2 makes clear that the multi-hop network degrades the control performance. In this experiment, we study how this degradation can be mitigated by applying different controllers. Whereas we have used a pole placement-based controller in the previous experiments, we switch to a controller based on the linear quadratic regulator [3] optimal control design technique for Plant 2. Table I clearly shows the improvement in settling time on applying this new controller.

4) Experiment 4 - Effect of Non-Ideal Multi-hop CPS

In order to explore more practical situations, we now consider packet drops during transmission. We therefore specify packet drop rates for Node I_3 (cf. Fig. 4), which are realized by regenerating the firmware of Node I_3 using our tool. Fig. 5 shows the output response of Plant 1 for different packet drop rates. As can be seen, such transmission loss significantly reduces the control performance. For Plant 2, similar effects in terms of increasing settling time and eventual unstable behavior are reported in Table I.

5) Experiment 5 - Impact of Re-Transmission of Packets

The effect of packet drops can be reduced by tuning the number of re-transmissions. In particular, our tool can configure the number of re-transmission attempts to be automatically made by each node whenever its transmission has failed. Settling time values considering this refinement are given in Table I. As can be seen in Table I, a larger number of re-transmissions improve the control performance.

Table I: Settling Time (in s) of Plant 2

No Loss		**Packet Loss**		
Single-Hop	Multi-Hop	10%	30%	50%
3.285	4.979	6.248	14.144	unstable
Performance Improvement by Changing the Controller				
2.896	4.162	5.229	13.059	unstable
Performance Tuning by Enabling Re-Transmission				
3.138	4.517	5.183	9.361	unstable

V. Concluding Remarks

1) Learning Objectives

Each of the experiments provides opportunities for imbibing students with the salient features of CPS design. The first two experiments will help students to learn the effects of network parameters (e.g., routing path, delay) and control parameters (e.g., sampling period) on the control performance. Experiment 3 will teach how the degradation in system performance caused by network-delay can be mitigated by designing the controller appropriately. Experiment 4 will help students to understand the impact of environmental non-idealities and faults on the system behaviour. In Experiment 5, students will learn how the effect of such non-idealities (e.g., packet drops) can be mitigated by re-configuring some network parameters, e.g., tuning the number of re-transmissions. This experiment also helps students to understand how a vulnerable node in the network can cause performance degradation. Such directions can potentially motivate a student to undertake more involved work in the domain of fault-tolerant, distributed CPS design, fault-diagnosis, mitigation, etc.

2) Conclusion and Future Extension

The experiments demonstrate that our proposed testbed is versatile and facilitates rapid prototyping of CPS configurations with different numbers of control loops, network links, routing configurations etc. It can provide hands-on experimental support for teaching courses in CPS that involve control theory and the implementation of control algorithms on embedded platforms. On the other end of the spectrum, more complex configurations of the testbed with a larger number of nodes can be used for a course project or a thesis, where the students need to develop intelligent control strategies, wireless protocols, and their combination exploring the co-design aspect of CPS. In future, we plan to add support for further physical plant types, e.g., inverted pendulums and also firmware generation support for other popular low power wireless platforms. Further, we intend to make this software tool available in a free repository online, along with instructions for interfacing intermediate nodes and physical/simulated plants. This will allow others to easily replicate and make use of the proposed testbed, leveraging our versatile but easy-to-use software interface.

REFERENCES

[1] https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF24-series.

[2] R. Alur, A. D'Innocenzo, K. H. Johansson, G. J. Pappas, and G. Weiss. Compositional modeling and analysis of multi-hop control networks. *IEEE Transactions on Automatic Control*, 56(10):2345–2357, 2011.

[3] K. J. Åström and B. Wittenmark. *Computer-controlled systems*. Prentice-Hall, Inc., 1997.

[4] M. Hashemi, W. Si, M. Laifenfeld, D. Starobinski, and A. Trachtenberg. Intra-car multihop wireless sensor networking: a case study. *IEEE Communications Magazine*, 52(12):183–191, 2014.

[5] P. Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer, 2011.

[6] J. Stankovic, J. Sturges, and J. Eisenberg. A 21st century cyber-physical systems education. *ACM Computer Magazine*, 50(12), 2017.

[7] M. Törngren et al. Education and training challenges in the era of cyber-physical systems: beyond traditional engineering. *Workshop on Embedded and Cyber-Physical Systems Education*, 2015.

[8] X. Zhou, X. Gou, T. Huang, and S. Yang. Review on testing of cyber physical systems: Methods and testbeds. *IEEE Access*, 6:52179–52194, 2018.

Sumana Ghosh Sumana Ghosh is currently a postdoctoral fellow in the Chair of Real-Time Computer Systems at the Technical University of Munich, Germany. Her current research interests include cyber-physical systems, fault-tolerant embedded control systems, cyber-security, and formal methods. She received her Ph.D. from the Indian Institute of Technology Kharagpur, India, in 2019.

Arnab Mondal Arnab Mondal is currently an MS (by research) student in the Advanced Technology Development Centre at the Indian Institute of Technology Kharagpur, India. His research interests include cyber-physical systems, embedded system design, and embedded control systems. He obtained his BE in Electrical Engineering from IIEST, Shibpur, India, in 2015.

Philipp H. Kindt Philipp H. Kindt is a postdoctoral researcher at the Technical University of Munich (TUM). His research interests span wireless communication, distributed embedded systems, and the Internet of Things. He received his Ph.D. in electrical and computer engineering from TUM in 2019.

Prateek Sharma Lt Cdr Prateek Sharma is serving in the Indian Navy as a Technical Officer and has been working in the field of embedded systems security. He was awarded the degree of Masters of Technology from IIT Kharagpur in the year 2017.

Yash Agarwal Yash Agarwal is currently a final year Bachelor student, pursuing a B.Tech degree in instrumentation engineering, in the Department of Electrical Engineering at the Indian Institute of Technology Kharagpur, India. His research interests include embedded system design, control systems, cyber-physical systems and robotics.

Soumyajit Dey Soumyajit Dey (Member IEEE, ACM) is currently an Associate Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology Kharagpur, India. His research interests include cyber-physical systems, automated reasoning, and GPGPU optimizations. He obtained his PhD from IIT Kharagpur in 2011.

Alok Kanti Deb Dr. Alok Kanti Deb (member IEEE, societies: CSS, CIS; local section chair 2016) received his Ph.D in Electrical Engineering from IIT Delhi, India in 2006. He is currently an Associate Professor with the Department of Electrical Engineering, IIT Kharagpur, India. His research interests include control systems, computational intelligence and automotive diagnostics.

Samarjit Chakraborty Samarjit Chakraporty (Senior Member, IEEE) is currently a William R. Kenan, Jr. Distinguished Professor in the Department of Computer Science at the University of North Carolina at Chapel Hill, USA. His research interests include distributed embedded systems, embedded control systems, energy storage systems, electromobility, and sensor network-based information processing. He obtained his PhD from ETH Zurich in 2003.