

A Desktop-sized Platform for Real-time Control Applications of Pneumatic Soft Robots

Brandon J. Caasenbrood¹, Femke E. van Beek¹, Hoang Khanh Chu¹, and Irene A. Kuling¹

Abstract—In recent years, there have been many modeling and control advances in the field of soft robotics, which resulted in a growing interest in practical, real-time applications. To further enable these developments, we present a desktop-sized testing and development platform intended for fast, precise, and reliable (closed-loop) control of pneumatic soft robots. The Soft Robotics Control-unit (SRC) is fully compatible with Matlab, Simulink, and Unity – allowing for real-time control tasks of various complexity. The performance was tested in three use-cases: model-based controller design, soft haptic feedback in virtual-reality, and tele-operation of a soft gripper. To promote the use of the SRC in the soft robotics community, all presented material is fully open-sourced such that it can be reproduced by students or researchers from any technical background.

I. INTRODUCTION

In the past decade, the diversity in soft robotic design has been growing rapidly, ranging from the popular PneuNets [1] and McKibben muscles [2] to a vast collection of bio-inspired systems [3]–[6]. Although soft robotic design and actuation is diverse in nature, a large bulk of these systems rely on fluidic or pneumatic actuation. There is a stark contrast between the ease with which soft actuators can be designed and the burden of developing a tailored pneumatic control system. Often, researchers are forced to develop a custom control interface [3], [7], as commercially-available solutions are limited. In particular, with the increasing development in model-based controllers for soft robotics, there exists an increasing demand in versatile platforms that allow for complex control objectives that are nowadays standardized in rigid robotics. Some examples include: motion control, impact and robust control, locomotion, tele-operation, haptic feedback, and machine and reinforcement learning. We believe that such control strategies for soft robotics could advance faster if researchers would be able to use a standardized development platform.

In addition to enabling faster development, a standardization of control systems would allow for a better description of the performance of new robotic designs. Currently, descriptions of new soft robots often present numbers on for instance response times [8]. However, these performance values partially depend on soft robotic design, the hardware, and control software (e.g, PD [9], [10] and bang-bang control [11]–[13]) that is used to control the pneumatic networks. To aid in these aims, we present a fully integrated solution for real-time control of pneumatic soft robotics systems.

¹B.J. Caasenbrood, F.E. Beek, H. Khanh Chu, I.A. Kuling are with the Faculty of Mechanical Engineering, Dynamics and Control Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands. Email: b.j.caasenbrood@tue.nl, f.e.beek@tue.nl, h.chu.khanh.hoang@student.tue.nl, and i.kuling@tue.nl



Fig. 1. Overview of the Soft Robotics Control-unit (SRC) connected to a PneuNet bending actuator (SmoothSil 945). Through a TCP/IP server, the pneumatic control unit can communicate with a host-pc running MATLAB/Simulink in real-time – thus allowing for fast (closed-loop) control and simultaneous data acquisition.

A. State-of-the-art

The development of pneumatic control architectures is not new, and the aim of this work is not to compete with its predecessor, but to focus on fast and closed-loop control applications. A popular option for pneumatic control, especially for newer researchers in the field of soft robotics, is the 'Soft Robotics Toolkit' [9], [10]. This platform has been a major way to democratize soft robotics, including a vast catalogue of soft robotic designs, models, and low-level controls. Other democratization initiatives are the FlowIO development platform [12], [13] and the Programmable-Air platform [14]. Although these initiatives offer great prototyping capabilities, they are limited in more advanced control settings where onset latencies, high flow-rates, and computational flexibility are essential for real-time control applications. Booth et al. have produced a very small, fast, and wearable-robot control system [11]. Although their onset latencies do come close to our requirements, and their solution is very elegant for on-robot-wearable form factors, their flow rate (6 L/m) is not sufficient for fast actuation of larger-sized robots.

Rather than focusing on small-scale and autonomous soft robots in which compactness is key, we focus on a class of soft robotic systems that are suitable in a classic industrial setting, e.g., manipulation, pick-and-place, and general motion control. We also envision use cases in haptics and

tele-operation, which must satisfy tight delay constraints and high-bandwidths. The haptic sense (i.e., sense of touch) reliably detects latencies exceeding 50 ms between motor actions and haptic feedback [15], and tele-operation systems suffer instabilities in the presence of delays [16]. Similar instability issues might occur in model-based control strategies for soft robotics, where small delays can diminish the intrinsic passivity of soft robotics systems.

B. Contribution

In this work, we present an open-source Soft Robotics Control Unit (SRC) for pressure-driven soft robots, with a particular focus on fast and accurate closed-loop applications. Compared to previous hardware, our main contributions include:

- Desktop-sized, modular control platform for real-time control of pneumatic soft robots;
- Sensor support via standardized I²C Qwiic connection (up to individual 6 sensor channels)¹;
- Low-level controller suited for piezo-actuated pressure valves – RMS errors ≤ 1 kPa and delay ≤ 20 ms.
- High-frequency control loop with bandwidths up to 500 Hz (up to 12 individual pneumatic channels);
- Several options for fast and easy software interfacing via MATLAB, Simulink, and Unity.

The paper is organized as follows. First, we give an overview of the control platform, detailing the hardware, software, and its compatibility for interfacing with other software. Next, we discuss a low-level controller to improve the system's performance. Lastly, we show the performance of our SRC in three soft robotic use cases with different levels of control complexity.

II. PLATFORM OVERVIEW

Here we briefly detail the hardware and software components of the SRC. To support the soft robotics community, all material presented in this work (i.e., hardware, software, and use cases) is made publicly available at the following repository². We would like to stress that, although some components are custom-made, our open-source resources make it easy for researchers or students from any technical background to reproduce the SRC.

A. Hardware

The SRC contains open-source hardware designed for closed-loop control of pressure-driven soft robots. The system consists of a single-board computer (Raspberry Pi 4, 2GB RAM), a proportional piezo-actuated pressure regulator (Festo, VEAB-L-26-D7-Q4-V1-1R1) with an active range of $-0.1 < P(t) \leq 0.1$ MPa differential pressure and flow rate of $0.075 \leq Q(t) \leq 0.33$ l/s, a custom Raspberry Pi Hat that interfaces between the single-board computer and two pressure regulators. The enclosure is fully 3D-printed, and can be fabricated using a standard FDM printer.

¹The Raspberry Pi 4 supports up to 6 unique I²C busses. Each I²C bus can support multiple sensors; however, each sensor ICs must have a unique I²C address on the same bus to avoid cross-talk.

²github.com/chukhanhhoang/SoftRoboticSetupFesto



Fig. 2. Image of the SRC real-time controller hat compatible with the Raspberry Pi (3/4/Zero). The board has two Qwiic I²C ports for additional sensors, two ports for the Festo VEAB pressure regulators, and an onboard 24V buck-boost converter. The board is also stackable upto six RPi hats.

The custom circuit board has a small footprint of 30x60 mm and can be stacked as a Raspberry Pi Hat. To interface with the regulators, the circuit board has two Digital-to-Analog converters (MicroChip, MCP4725) and two Analog-to-Digital converters (Texas Instruments, ADS1014) that send and receive data from the regulators, respectively. Given the specifications of the DACs and ADCs, the system has an optimal control and measurement resolution of $e = \pm 0.048$ kPa. Both the ADCs and DACs are paired with a non-inverting opamps (Texas Instruments, LM358S) such that the voltage-levels (in this case 3.3V) are compatible with various of board computers, e.g., the Raspberry Pi Zero, Raspberry Pi 3 and 4, and the Jetson Nano. The array of ADCs and DACs on the board allow for easy communication via I²C. To avoid the ICs sharing the same I²C address, we exploit the available six I²C busses on the Raspberry Pi – thus allowing for connecting twelve regulators simultaneously. Furthermore, each board is equipped with two I²C Qwiic connectors that allow for additional sensor communication at 400 kHz. The design files are open and can be found on the repository, to enable reproduction. To be more specific, the custom board can be ordered from most PCB manufacturing facility that allows for in-house PCB fabrication and component placement.

B. Software architecture

The main software of the setup, written in Python, is designed in three layers to optimize for speed, convenience, and versatility. The setup can read multiple sensors and control several VEAB regulators simultaneously, as well as communicate with other devices via TCP/IP. An illustrative diagram of the software architecture is given in Figure 3.

The first layer of the software provides two base classes: *baseSoftRobot.py* and *baseSensor.py*. The class *baseSoftRobot.py* sets up the multi-processing environment that handles the TCP/IP communication for the array of VEAB regulators and sensors. Multi-processing is used here to process data in parallel, improving not only the speed of the system but also the timing precision. The class *baseSensor.py*

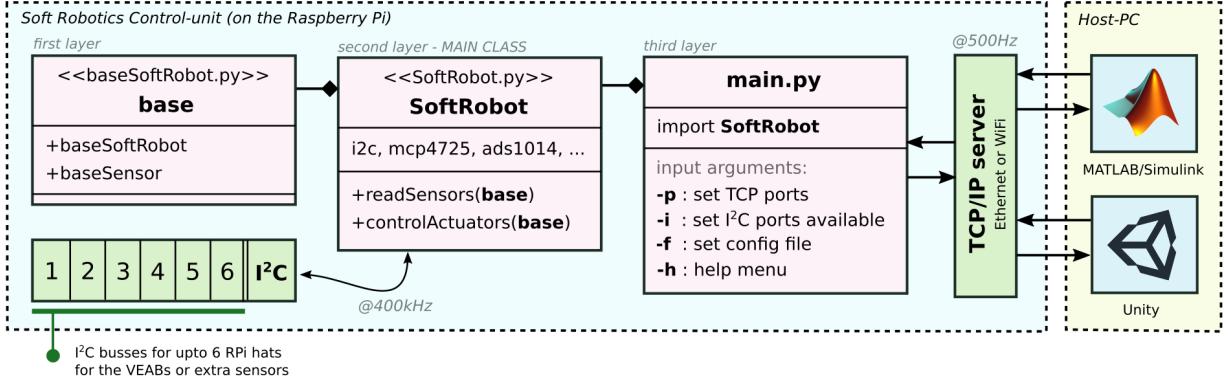


Fig. 3. Schematic overview of the software architecture running on the Raspberry Pi 4, and its interfacing options. The software architecture consists of three consecutive layers that build upon each other. All sensor and hardware I²C communication occurs in the second layer; whereas the last layer acts as an optional layer for changing configuration parameters. Lastly, the python code starts a TCP/IP server to interface with Matlab/Simulink or Unity.

serves as a wrapper for other sensor classes. Both classes act as parents for other classes in the next layer(s).

The second layer facilitates all communication between the Raspberry Pi and the SRC real-time controller hat. First, necessary libraries are imported. Next, sensor classes are written as children of the *baseSensor.py* class, and in each sensor class, a function called *ReadSensor* must be implemented when using sensors. Finally, the class *SoftRobot*, inheriting all functions of *baseSoftRobot.py*, is assembled from all software parts relating to the TCP/IP communication, regulators, and sensors. To some extent, the class *SoftRobot* acts as the main class of the software architecture that encompasses the user's requirement.

The third layer is dispensable and added for the convenience of users. This layer is a Python script that takes the arguments and initialized a *SoftRobot* object accordingly. Changing the parameters of the software (i.e., number of sensors and actuators, sampling frequency, setting the TCP port) is no longer an arduous task of re-writing the code.

C. MATLAB, Simulink, and Unity interfaces

One of the platform's key features is its versatility in terms of a programming interfacing. First, in MATLAB, we developed an Object-Oriented class SRC.m, which can be found on the repository. The class allows for easy communication with the hardware platform with through an array of integrated functions. Alternatively, the Simulink interface provides the same functionality as the MATLAB interface, yet it provides GUI-based programming. The base sets up TCP/IP communication with the SRC; and additional functionalities can be added using Simulink blocks.

For our human-robot interaction use cases, we wanted to enable communication with the Unity game engine. Therefore, we designed a Unity package for TCP/IP communication based on the Weiland's work [17]. Important parameters are exposed to Unity's interface, so they can be set by users without the need for changes in the code.

We provide working examples of the three interface options (i.e., MATLAB, Simulink, and Unity) to further support

the use-cases presented in section III. However, let it be clear that any interface that allows for TCP/IP communication is compatible with minimal changes to the software.

D. Low-level controller design

To enhance the pneumatic performance of the SRC, we extend beyond the standard PD- and bang-bang control architecture in previous literature [9]–[13]. As such, we propose a novel low-level controller that ensures a low tracking error and delay w.r.t. a dynamic reference. As a first step, the dynamics of the intrinsic VEAB pressure regulation together with the pneumatic network of the soft robot can be approximated using a first-order dynamical system:

$$\Sigma : \begin{cases} \dot{x} = Ax + \phi(u), \\ p = x, \end{cases} \quad (1)$$

where x denotes the internal pressure state of the pressure regulator, u the control input, p the pressure output of the system, and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ a nonlinear input map due to the piezo actuation of the VEAB regulators. To be more specific, the (monotonic) mapping $\phi(u)$ describes the activation of the regulators for pressure signals that are close to ambient pressure. This behavior (similar to a dead zone) is important to include in the model, as it is the lead cause of tracking errors when switch from positive to negative differential pressures. Now, the control objective is to ensure that the VEAB regulator closely follows the reference trajectory, denoted here by p_d . As such, we wish to ensure that the tracking error goes to zero after sufficient time passes, i.e., $\lim_{t \rightarrow \infty} p \rightarrow p_d$. To achieve this, we propose a combination of feed-forward (FF) and feedback (FB) control:

$$u = \hat{\phi}^{-1}(\nu), \quad (2)$$

$$\nu = F_d \dot{p}_d + p_d - K_p(p - p_d) \quad (3)$$

where K_p is the proportional gain of the feedback controller, F_d the differential gain of the feed-forward controller, and $\hat{\phi}^{-1}$ the approximation of the inverse mapping to counteract the dead-zone such that it satisfies $\phi \circ \hat{\phi}^{-1} \approx 1$.

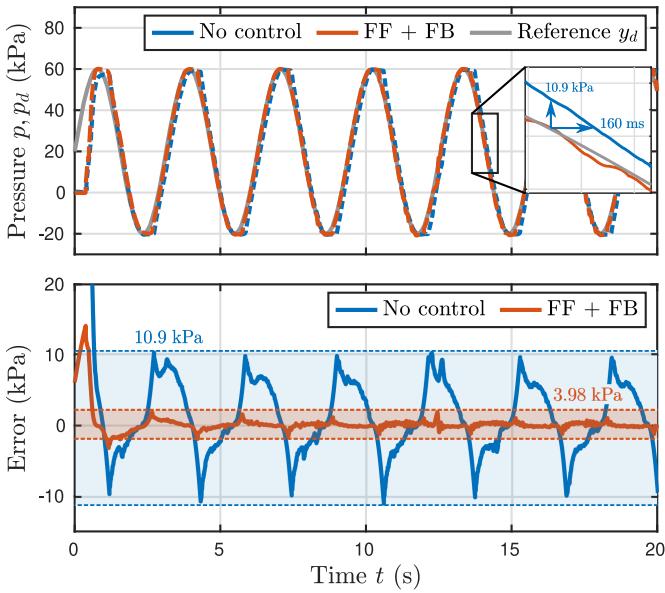


Fig. 4. Performance results of the control unit under dynamic testing in which a feed-forward (FF) and feedback (FB) controller is employed for accurate and fast pressure regulation. Here we observe a substantial performance increase using the proposed low-level control law (2)-(3).

E. System performance measurements

To test the performance of the full system, we connected the SRC to the PneuNet shown in Figure 1, and considered a dynamic pressure trajectory $p_d = 20 + 40 \sin(t)$. To illustrate the effectiveness of the low-level controller, we also compared the open and closed-loop performance. The open-loop controller is equivalent to setting the input of the controller to the reference, i.e., $u = p_d$. The performance results of the low-level controller are shown in Figure 4, where the RMS error and maximum error are ± 0.35 kPa and ± 1.91 kPa, respectively. In contrast, the open-loop system has a RMS error and maximum error of ± 3.98 kPa and ± 10.93 kPa, respectively. Regarding the phase, we observe an estimated time-delay of ± 45 ms and ± 160 ms in closed and open-loop, respectively. Given these results, we show that the low-level controller reduces both the tracking error and delay substantially. Please note that the system parameters of Σ are unique – and depend for instance on the tube length, volume, and the type of regulator. For optimal performance, tuning of the gains might be required for every soft robot.

III. USE CASES

In this section, we highlight the performance of the SRC in terms of precision and delay, and its versatility. To do so, we explore three distinct use cases: *i*) system identification and model-based control of a soft robot, *ii*) tele-operation using a soft gripper, *iii*) and soft haptic feedback using Virtual Reality (VR). To improve transparency, all the study-cases can be viewed in the supplementary material of the paper, and all software is made publicly available at the repository under the folder `./src/examples`.

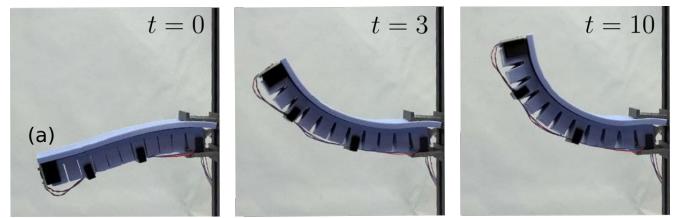


Fig. 5. A few snapshots of the PneuNet when subjected to a regulated sinusoidal pressure signal $p(t)$. Left: (a) Placement of the inertial sensor for real-time curvature sensing that are communicated via I²C to the SRC.

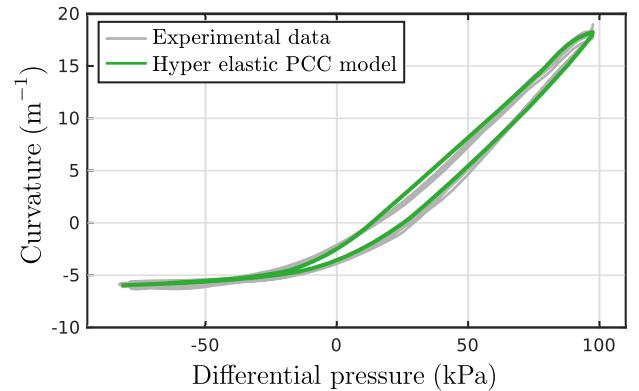


Fig. 6. System identification results of the PneuNet actuator using experimental data – and the model-fit of the hyper-elastic PCC model. The system is subjected to a periodic sine input, we observe a hysteresis loop, hyper-elasticity, and self-contact.

A. System identification and model-based control of PneuNet

In the first study case, we primarily focus on model-based strategies for the soft robot seen in Figure 1. Our aim here is two-fold: *i*) to employ system identification to derive a dynamic model, and *ii*) to utilize the dynamic model for control. As such, fast and low-latency pressure regulation is important. To measure the state of the system, an Inertial-Measurement Unit (InvenSense, MPU9250) is attached to the end-effector of the soft robot (see Figure 5). The sensor is connected via I²C to the SRC and the data acquisition is programmed in the python class `SoftRobot.py` (see Section. II-B). Based on preliminary tests, we propose an ansatz for the (nonlinear) dynamics using an in-extensible planar Piece-wise Constant Curvature (PCC) model that accounts for hyper-elastic stiffness and elastic self-contact. It is worth mentioning that there exists a rich literature on PCC models, and the reader is referred to [4], [18]–[20] for more detail. Let us introduce the state vector as $q := \kappa$ with κ the curvature. Then, the dynamic model takes the form:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + P(q, \dot{q}) + G(q) = Hp, \quad (4)$$

where $M(q)$ is the generalized inertia matrix, $C(q, \dot{q})$ the Coriolis matrix, $P(q, \dot{q}) = k(q)q + \mu\dot{q}$ the hyper-elastic and visco-elastic forces with $k(q)$ a nonlinear stiffness and $\mu > 0$ a damping coefficient, $G(q)$ the gravitational forces, H a pressure map, and p the control input (i.e., pressure). To model the hyper-elastic stiffness and (soft) self-contact, we

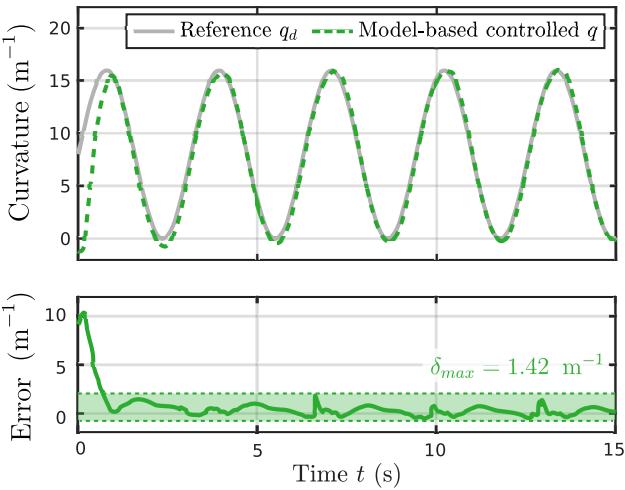


Fig. 7. Experimental results of the proposed model-based controller in (6). We observe that the model-based controller can provide good tracking performance – the steady-state error is $\delta_{\max} \leq 1.42 \text{ m}^{-1}$.

propose the following nonlinear state-dependent function:

$$k(q) = \alpha_1 e^{-\min(\alpha_2 q, 0)} + \alpha_3 (q - q_c)^{\alpha_4} \cdot \min(q, 0), \quad (5)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are hyper-elastic stiffness coefficients, and q_c the joint position when self-contact of the soft body occurs. In the identification procedure, we subject the soft robot to a periodic pressure input $p = 10 + 80 \sin(2t)$ (kPa). The resulting dynamics converges to a periodic solution, on which we fit our dynamic model. The system identification results of the ansatz in (4)-(5) using the periodic experimental data is shown in Figure 6, showing a good match between model and measurement.

Now, using the aforementioned model, we propose a model-based controller to control the soft robot's curvature:

$$p = H^{-1} \left[P(q, \dot{q}) + G(q) - K_p e - K_i \int e(\tau) d\tau \right], \quad (6)$$

where K_p, K_i are proportional and integrator gains, respectively; and $e := q - q_d$ the tracking error between the state and the desired trajectory. The model-based controller is a combination of gravity-stiffness compensation together with a PI-controller. The model-based controller above is programmed using MATLAB/Simulink and communicates via TCP/IP with the control unit (at 200 Hz). Considering a described dynamic trajectory $q_d = 8 + 8 \sin(2t)$, the tracking performance of the model-based control law is shown in Figure 7, where we observe good tracking performance with a maximum steady-state error of $e = \pm 1.42 \text{ m}^{-1}$. These results demonstrate the SRC's versatility as a platform for the development and testing of model-based controllers.

B. Tele-operation of soft gripper

In the next use case, we focus on a tele-operation scenario in which a user's hand movements are mapped to a remote three-fingered soft PneuNet gripper (DragonSkin 10) as shown in Figure 8. In this particular use-case, minimizing latency outweighs tracking performance, as a latency between

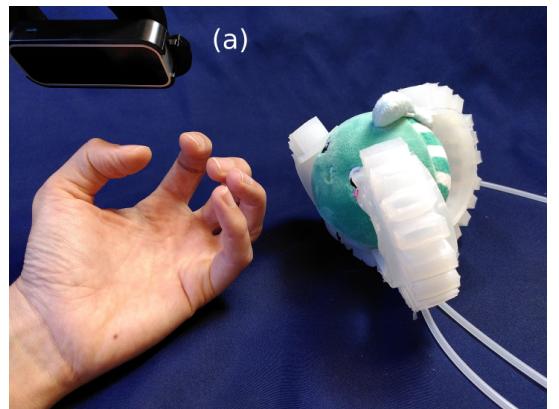


Fig. 8. Overview of the tele-operation system in use case 3. The angles of the user's thumb, index and middle finger are captured by the Leap Motion camera, and mapped to pressures of the three-finger PneuNet gripper.

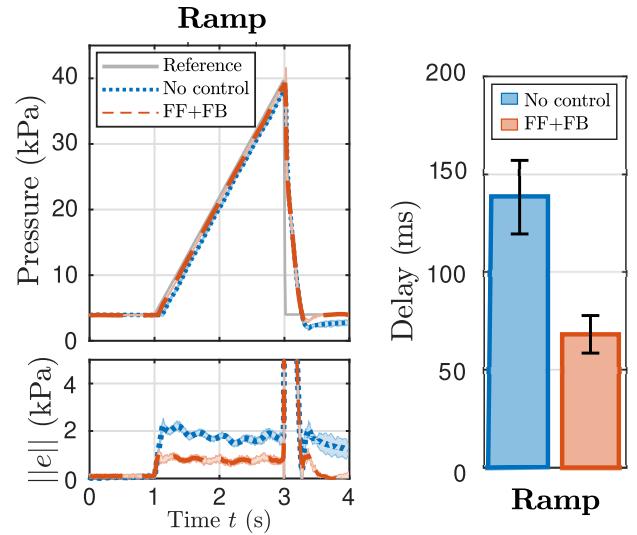


Fig. 9. Left: Performance for tele-operation in Unity. For each Unity frame, the desired pressure is sent out and a measured pressure is received from Simulink. The top row shows performance without (blue) and with our low-level controller (red). The bottom row shows the difference (i.e. vertical error) between the reference (gray) and measured signals (blue/red). In both figures, the 95% confidence interval is indicated with shaded regions, but the 10 repetitions show such comparable responses that it is hard to see the intervals. Right: Time delays (i.e. horizontal error), indicating the horizontal distance between the reference and measured pressures shown on the left. Time delays for the low-level controller are much smaller than open-loop control delays.

the user moving and the system responding compromises system stability [16] and will reduce transparency between the user and remote robot [21]. To track the user's hand movements, we use Leap Motion tracker (Ultraleap) positioned on a stand above a desk. The joint-angles of the user's thumb, index and ring finger are registered by Unity, and sent to MATLAB/Simulink using the TCP/IP communication package.

To match the Leap Motion system, the refresh rate is chosen at 120 Hz. To measure the effect of the additional communication layers in this scenario in a repeatable way, a ramp is used as a representative reference signal, which is

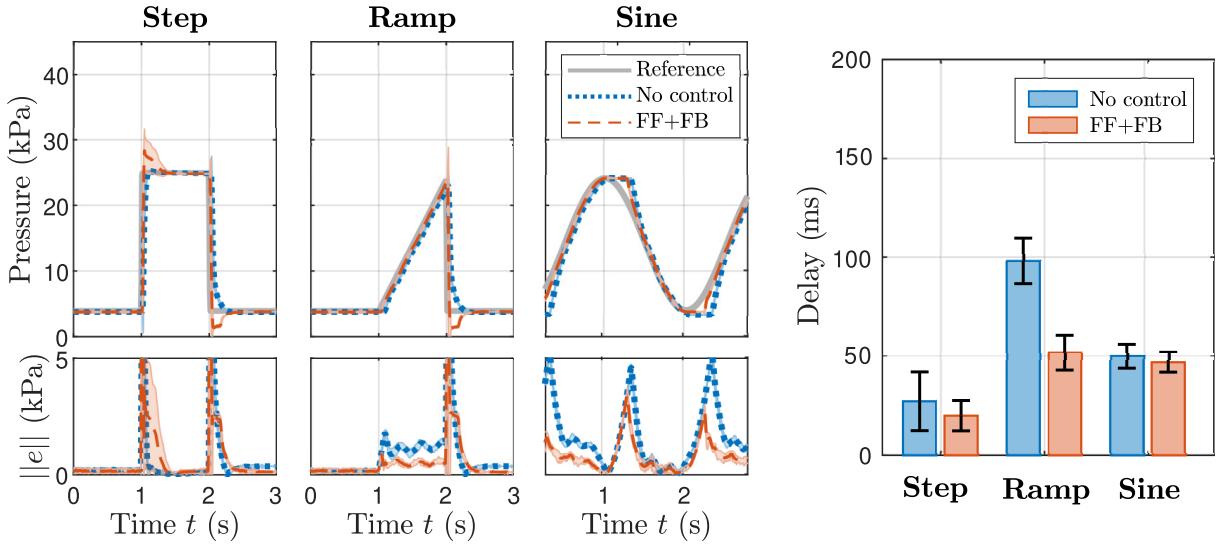


Fig. 10. Left: performance of the Haptic VR Feedback use case primitives, as recorded in Unity. For the full caption, see Figure 9. The controller is tuned to minimize delay, which results in larger overshoots compared to no control. For visibility, the tops of the peaks are cut off. Right: The time delays. For all primitives, the low-level controller reduces the delays below the perceptual detection threshold.

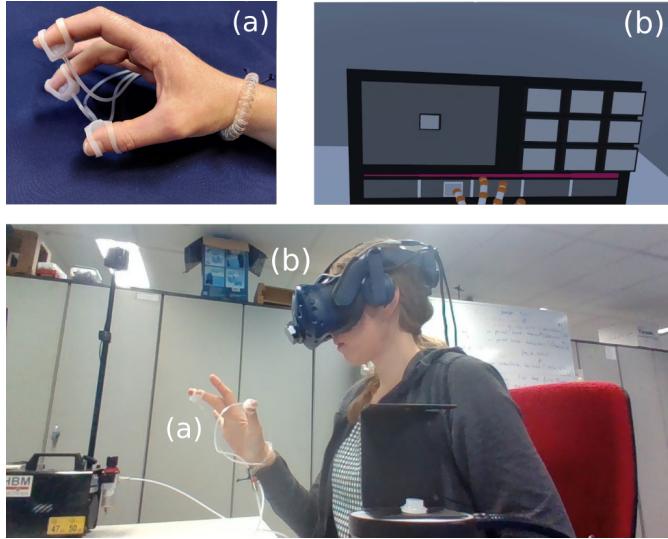


Fig. 11. Bottom: User wearing a headset while exploring an interface in use case 2: haptic feedback in VR. The interface provides visual, auditory and haptic feedback. Top: (a) The soft robotic tactile feedback units, which provide (real-time) pressure feedback to the user's finger tips. (b) The VR interface, as shown in the HMD, which provides congruent visual, auditory and haptic feedback to the user.

communicated from Unity to the SRC. The measurement was repeated ten times, with and without the low-level controller (2)-(3). The controller was tuned a-priori and the same controller gains are used for the next use-case. As illustrated in Figure 9, the open-loop latency is considerable (138 ± 19 ms) for the current setup. However, with the low-level controller, the delay is substantially smaller (68 ± 9 ms), albeit still above the human perceptual limit. This illustrates that our system, under the appropriate control conditions, is a possible candidate for exploring soft robotic tele-operation applications in lab-based settings.

C. Haptic feedback in Virtual Reality

In this use case, we employ soft robotics to provide haptic (i.e., touch) feedback to users of a virtual interface. Adding haptic feedback to Virtual-Reality (VR) scenarios has the potential to create better immersion and interaction with virtual objects [22]. However, such benefits only arise if the feedback is perceived as being congruent with the other signals in the scene. Therefore, the aim here is to test if the SRC is fast for the additional complexity of VR, and to test actuators with volumes that were very different from the Pneunets used in the previous use cases. We again used a Leap Motion tracker, however, it is now attached to a VR headset (HTC, Vive Pro VR) for hand tracking. For soft robotic haptic feedback, small cylinders with a single air chamber are designed (11x7 mm) and made from Dragonskin 10. Silicone finger sleeves (Xutong) are used to attach the cylinders to the thumb, index and middle finger of the user's right hand, as shown in Figure 11. In the VR environment, users could interact with the virtual interface, and haptic effects are generated based on finger positions. The array of haptic effects are composed of ramps, steps and sine-wave primitives.

To match the sampling rate of the VR headset, a refresh rate of 90 Hz is chosen. The same procedure as described in the previous use case is used to test the haptic primitives in this use case ten times per controller setting. The results in Figure 10 show that the VR use case operates below perceptually noticeable delays (i.e., ≤ 50 ms). We do see peaks in (vertical) errors when the references show large discontinuities, due to tight control gains for minimal system delay rather than small vertical errors. Nevertheless, these peaks are small and short compared to the reference, thus having little effect on the user experience. Together, these results show that SRC is a viable option for haptic feedback in a VR setting.

IV. CONCLUSION

In this work, we present a desktop-sized testing and development platform purposefully designed for real-time control applications of pneumatic soft robots. Using a low-level controller that compensates the intrinsic nonlinear dynamics of the piezo-actuated regulators, we further enhance the base performance of the closed-loop pneumatic system – in both response time (≤ 50 ms) and tracking error (≤ 1 kPa). One key contribution of this work is the platform’s versatility allowing up to twelve regulated pressure ports and easy-accessible sensor communications via standardized I²C. To demonstrate its versatility and the system’s performance in an advanced control setting, the system is experimentally corroborated using three different study-cases: model-based control of a soft robot, soft haptics in VR, and remote teleoperation of a soft gripper.

REFERENCES

- [1] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, “Pneumatic networks for soft robotics that actuate rapidly,” *Advanced Functional Materials*, vol. 24, no. 15, pp. 2163–2170, 2014.
- [2] F. Schmitt, O. Piccin, L. Barbé, and B. Bayle, “Soft robots manufacturing: A review,” *Frontiers in Robotics and AI*, vol. 5, 2018.
- [3] A. D. Marchese, R. K. Katzschmann, and D. Rus, “A Recipe for Soft Fluidic Elastomer Robots,” *Soft Robotics*, vol. 2, no. 1, pp. 7–25, 2015. [Online]. Available: <http://online.liebertpub.com/doi/10.1089/soro.2014.0022>
- [4] R. K. Katzschmann, C. D. Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, “Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model,” *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, no. February, pp. 454–461, 2019.
- [5] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, “Exploiting the Dynamics of Soft Materials,” vol. 5, no. 3, pp. 339–347, 2018.
- [6] V. Falkenhahn, T. Mahl, A. Hildebrandt, R. Neumann, and O. Sawodny, “Dynamic Modeling of Bellows-Actuated Continuum Robots Using the Euler-Lagrange Formalism,” *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1483–1496, 2015.
- [7] P. Polygerinos, S. Lyne, Z. Wang, L. F. Nicolini, B. Mosadegh, G. M. Whitesides, and C. J. Walsh, “Towards a soft pneumatic glove for hand rehabilitation,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, p. 1512–1517.
- [8] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, “Multigait soft robot,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011. [Online]. Available: <https://www.pnas.org/content/108/51/20400>
- [9] A. A. Zheng Wang, Panagiotis Polygerinos and A. Campo. Fluidic control board. Last visited: 20-09-2021. [Online]. Available: <https://softroboticstoolkit.com/book/control-board>
- [10] D. P. Holland, C. Abah, M. Velasco Enriquez, M. Herman, G. J. Bennett, E. A. Vela, and C. J. Walsh, “The soft robotics toolkit: Strategies for overcoming obstacles to the wide dissemination of soft-robotic hardware,” *IEEE Robotics and Automation Magazine, Special Issue on Open Source and Widely Disseminated Robot Hardware*, vol. 24, no. 1, pp. 57–64, 2017. [Online]. Available: <http://dx.doi.org/10.1109/MRA.2016.2639067>
- [11] J. W. Booth, J. C. Case, E. L. White, D. S. Shah, and R. Kramer-Bottiglio, “An addressable pneumatic regulator for distributed control of soft robots,” *2018 IEEE International Conference on Soft Robotics, RoboSoft 2018*, no. April, pp. 25–30, 2018.
- [12] A. Shtarbanov. Flowio platform. Last visited: 20-09-2021. [Online]. Available: <https://www.softrobotics.io>
- [13] ———, “Flowio development platform – the pneumatic ‘raspberry pi’ for soft robotics,” in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, no. 479. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3411763.3451513>
- [14] Programmable-air. Last visited: 20-09-2021. [Online]. Available: <https://www.programmableair.com>
- [15] T. Kaaresoja, S. Brewster, and V. Lantz, “Towards the temporally perfect virtual button: Touch-feedback simultaneity and perceived quality in mobile touchscreen press interactions,” *ACM Trans. Appl. Percept.*, vol. 11, no. 2, 2014. [Online]. Available: <https://doi.org/10.1145/2611387>
- [16] T. Sheridan, “Space teleoperation through time delay: review and prognosis,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.
- [17] T. Weiland. Tcp-udp networking. Last visited: 29-09-2021. [Online]. Available: <https://github.com/tom-weiland/tcp-udp-networking>
- [18] B. J. Caasenbrood, A. Y. Pogromsky, and H. Nijmeijer, “Dynamic modeling of hyper-elastic soft robots using spatial curves,” in *IFAC-PapersOnLine*, vol. 53, 2020, pp. 9238–9243.
- [19] A. D. Marchese and D. Rus, “Design, kinematics, and control of a soft spatial fluidic elastomer manipulator,” *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 840–869, 2016. [Online]. Available: <http://journals.sagepub.com/doi/10.1177/0278364915587925>
- [20] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, “Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment,” *International Journal of Robotics Research*, vol. 39, no. 4, pp. 490–513, 2020.
- [21] A. Toet, I. A. Kuling, B. N. Krom, and J. B. F. van Erp, “Toward enhanced teleoperation through embodiment,” *Frontiers in Robotics and AI*, vol. 7, 2020.
- [22] B. Hannaford and A. M. Okamura, *Haptics*. Springer Berlin Heidelberg, 2008, pp. 719–739.