Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 8833–8838

Low-cost vision-based embedded control of a 2DOF robotic manipulator

Jasper Juchem * **Michiel De Roeck** * **Mia Loccufier** *

* *Department of Electromechanical, Systems and Metal Engineering, Ghent University, 9052 Ghent, Belgium (e-mail: {Jasper.Juchem, Michiel.DeRoeck Mia.Loccufier}@UGent.be).*

Abstract: Simulations are frequently used to validate underactuated control of the planar two-link manipulator, a typical benchmark system. These new control strategies need to be evaluated on a mechanical setup in order to further research their validity. In this work the benchmark system is specifically designed and built to fit on a table and to be as low-cost as possible. One of the large discrepancies between the simulator and the real setup is the presence of friction. With the help of a carefully designed bearing topology and by using image vision to measure the states of the system, as opposed to mechanical sensors, this friction can be kept to a minimum. The image vision requires a computer that is capable to process the images and send the control effort to the actuator in a timely manner. For this purpose a Raspberry Pi 4 is used to interface with the camera. The architecture consists of a traditional Raspbian operating system, which is expanded with Robot Operating System (ROS) and Xenomai. With the latter, tasks can be done in hard real-time, which is needed to achieve a fixed sampling time. In this case, a sampling time of $9ms$ is reached. An open-loop and closed-loop experiment with PD-controller are performed to validate the software architecture.

Keywords: Vision-based, Raspberry Pi, Xenomai, disturbance rejection, embedded control

1. INTRODUCTION

In underactuated control, a planar two-link manipulator is a well-known benchmark system (Liu and Yu (2013)), which is often used to research underactuated control strategies (Mahindrakar et al. (2006); De Luca et al. (2000); He et al. (2016); Huang et al. (2022)). Simulations of such benchmarks can verify theoretical findings, as it allows to ignore effects of damping due to friction, measurement noise and other practical aspects of a real setup. Furthermore, it allows to compare with other research results that have been obtained on the same benchmark. However, building the benchmark system can be helpful for didactic purposes or for real-world verification of control strategies.

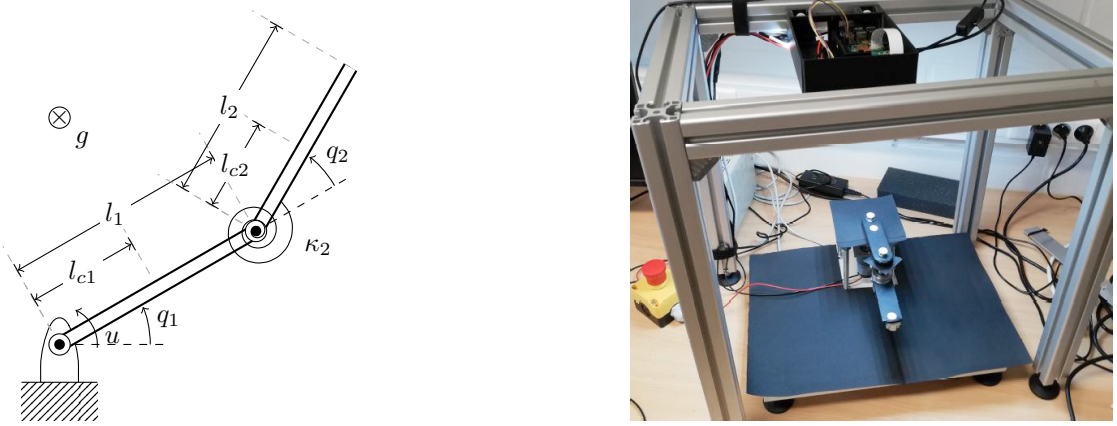
Sensors can be very invasive from a dynamical point of view, especially if setups become small-scale. They can add mass to the setup, influencing the resonance frequencies of the setup, or they can add damping due to friction in a mechanical sensor. Furthermore, the data transfer from moving sensors to a static computing unit can be complicated (Fraden (2016)). These problems can be solved by using machine vision to implement vision-based control. This was first proposed in Shirai and Inoue (1973). Back then, the large-scale roll-out of this innovative idea was hindered due to technological issues, like low resolution, low frame rate, and slow processing time. Currently, these problems are no longer an issue for mechanical applications (Zhao et al. (2016)). Vision-based sensors give rise to a fairly large field of view, a rather high spatial and temporal bandwidth and good controlability. The sensor does not influence the dynamics of the systems (i.e. additional mass, friction). A disadvantage is the sensitivity to a structured environment: a more complex and unpredictable environment will lead to loss of robustness (Hashimoto (2003)).

The use of image vision requires a computing unit which is able to extract relevant features from images. This is done with image processing techniques. With the advent of low-cost single-board computers, e.g. Arduino and Raspberry Pi, hardware/software interfacing, sensor measurements and steering of actuators became more accessible. This has lead to a large community of developers and scientists who provide open-source documentation, software and examples (Jolles (2021); Kondaveeti et al. (2021)). These development boards have become more advanced with regard to computing power and hardware over the years, which allows embedded vision control (Ortmeyer (2014)).

Many software architectures can be chosen for robotic systems. Ahmad and Babar (2016) gives an elaborate overview of the many possibilities that exist. In Delgado et al. (2019) an example is given on a real-time (RT) control architecture for a service robot. A combination of Robot Operating System (ROS) with a dual-kernel approach for RT, using Xenomai, is proposed. This strategy is well-documented and proves to be feasible for their application. Note that ROS has its own real-time control library (ros2_control). Nevertheless, in this work the above architecture is chosen to explore its capabilities in the proposed setup.

In this work a Raspberry Pi 4 with custom Linux distribution, including Xenomai and ROS as described in Delgado et al. (2019), is connected to an ArduCam© OV9281

10.1016/j.ifacol.2023.10.073



(a) (b)

Fig. 1. The planar (perpendicular to the gravitational force denoted by g) two-link manipulator; (a) a schematic overview indicating all parameters: q the relative angles, l_i and l_{ci} the links' lengths and distance to the center of mass respectively, u the torque provided by the motor, and κ_2 the spring constant; (b) the actual setup as described in this work.

CMOS camera to observe a small-scale, low-cost planar two-link manipulator. The design is described in Section 2. In Section 3, an open-loop experiment is discussed that shows the impulse response by disturbing the unactuated coordinate of the system. To prove that it is able to work in closed-loop a PD-controller is implemented and the results are compared to the open-loop response.

2. SETUP

The predetermined design objectives were to design a table-top, low-cost planar two-link manipulator (see Fig. 1a) to test control strategies. The two-link system is underactuated, as it can only be actuated at the base of the first link. To monitor the performance and to be able to close the loop, the angular positions and velocities of both joints need to be measured. The setup will be utilized to assess controller performance for vibration mitigation: an impulse response is applied to the unactuated, second link. Therefore, it is crucial to minimize the internal damping of the system, as this can complicate quantification of the controller performance. To avoid damping due to angular position sensors or to avoid changing the dynamical behavior due to additional mass from accelerometers, a vision-based sensor is selected. This means that a camera needs to be interfaced with a programmable computer platform. In addition to that, for control purposes a fixed sampling time is necessary.

This section discusses the design choices made to meet aforementioned requirements. The hardware is discussed first, followed by the software. The software section expands on image processing, operating system and real-time features.

2.1 Hardware

The links are made from aluminum bars with cross section $30 \times 5mm$. Table 1 lists the dimensions of the two-link system. The links experience a dynamic radially-oriented force due to the centrifugal force from the rotary movement and a static moment due to gravity exerted at the center of mass. The latter induces both a radial and axial force in the bearing. The moment is countered using two deep groove ball bearings (SKF 61800-2RS1 and SKF 619/8-2Z, lubricant: WD-40) at each rotation point. The torsional spring at the second joint consists of two symmetrical, pretensioned springs. The springs are guaranteed to operate in both directions by the symmetry, and the pretension prevents a dead band in the torque for small angles. In the first joint a DC-motor (RB-35 GM Type 01, 1.2W) is connected.

Table 1. Parameters of the planar two-link manipulator setup.

l_1 $[mm]$	l_2 $[mm]$	κ_2 $[Nm/°]$
91	85.5	0.144

To keep the system cost-optimal and flexible a Raspberry Pi (RPi) 4B - 4GB RAM is chosen as the computer. In addition to the fact that it is relatively cheap, it allows to connect the camera module, ArduCam© OV9281 (1280 × 800 at 120 fps), to the CSI camera port. Furthermore, GPIO pins can be used to interface with the motor driver, Cytron© MD10C. All components can be found in the low-cost hobby world and are intensively discussed on online forums.

In Fig. 1b the setup as-build is shown. On top of the frame a 3D printed casing is attached to hold the RPi and the camera. Furthermore, an LED-strip is glued to the frame to increase the image quality. Below the frame, the two-link setup is shown. The links have the same black color as the background, except for the white dots that are tracked to obtain the joint angles, which is explained below.

2.2 Software

On the RPi, a Linux distribution (Raspbian) is used as its Operating System (OS) (Raspberry Foundation (2021)). Furthermore, the open-source robotics middle-ware suite, Robot Operating System (ROS), is implemented. This is

not an OS, but a set of software frameworks for robot software development. It allows to abstract the algorithm to smaller blocks of code, in different programming languages, and facilitates communication between these different blocks (ROS (2021)). As a fixed sampling time needs to be achieved, the OS should have real-time (RT) capabilities. Unfortunately, the RPi's OS is non-deterministic, meaning that there is no control on the timing of individual processes assigned to the kernel. Therefore, the kernel is expanded to include Xenomai, which provides hard RT computing (Gerum (2019)). This will create a dual kernel topology, such that the default Linux kernel stays active to deal with non-real-time (NRT) tasks (e.g. ROS nodes), and the second kernel focuses on high-priority tasks to ensure precise timing and reliability. This strategy is preferred over the single kernel configuration as it has a lower latency (Barbalace et al. (2008)). In this topology, an Adaptive Domain Environment for Operating Systems (ADEOS) layer enables multiple kernels to have access to the hardware. It is priority-based such that RT tasks are prioritized over NRT tasks. The RT computing environment that is implemented is given in Fig. 2.

As mentioned in Delgado et al. (2019) a Cross-domain datagram protocol (XDDP) needs to be used to achieve two-way communication between Xenomai RT tasks and NRT Linux threads and processes. This is done using the sockets that are connected to a pseudo device file in the standard Linux.

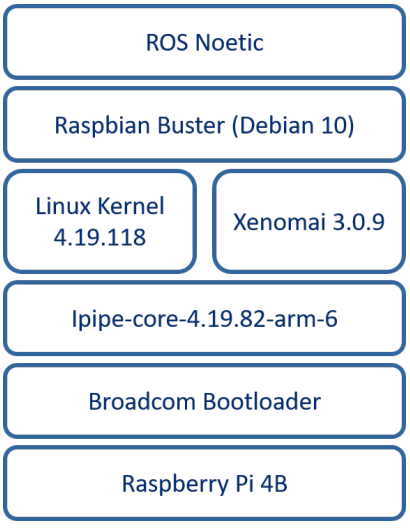


Fig. 2. Raspberry Pi 4B with its real-time environment.

The tasks in the NRT part are subdivided in sub-algorithms, called Nodes, as shown in Fig. 3. The camera module sends a RAW8 monochrome image to the image processing algorithm. This picture contains the two links, which have the same color as the background. Four filled circles are drawn on the links to track relevant coordinates: one on each joint, one at the end of the second link, and an extra one on the first link. To avoid long processing time the current position and velocity of the circular markers are estimated to crop the full image **A** to the relevant areas **B** and **C**, as shown in Fig. 4. Based on the relative coordinate (R_{B2} or R_{C2}) of the marker within the cropped image and the absolute coordinate of the cropping window (R_{B1} or R_{C1}), the absolute coordinate of the marker

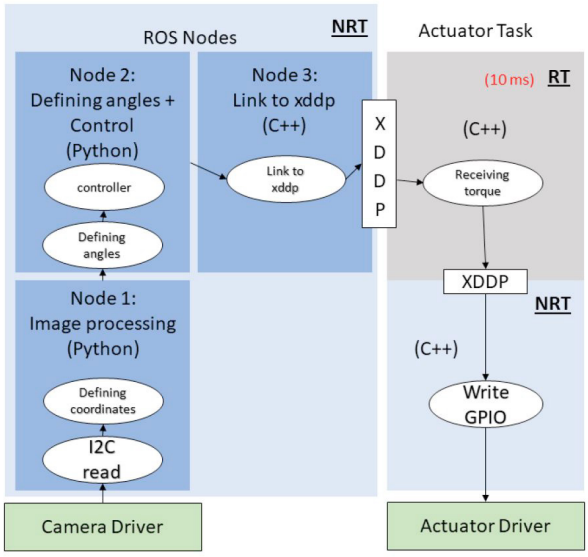


Fig. 3. A schematic overview of the different blocks of the software algorithm.

can be found. To predict the velocity of the marker the following relation is used:

$$\dot{\hat{q}}_{i,kT_s} = \frac{\Delta q}{\Delta t} = \frac{q_{i,kT_s} - q_{i,(k-1)T_s}}{T_s} \tag{1}$$

with T_s the sampling time, i the joint number, and $k \in \mathbb{N}$ the time step. Now the angles can be estimated

$$\hat{q}_{i,(k+1)T_s} = q_{i,kT_s} + \dot{\hat{q}}_{i,kT_s} T_s. \tag{2}$$

This can then be used to estimate the position of a marker, based on the lengths of the links. Afterwards, *thresholding* transforms the cropped gray scale images to black and white. This results in a black image with a white circle at the location of the markers. Then the coordinate of the marker is found using the *Hough transform* algorithm that tries to approach a circle's radius and center based on image processing (Yuen et al. (1990)). This is done in Python using the *openCV* module.

The measured coordinates of the markers are sent to Node 2, where the coordinates are transformed into joint angles using the trigonometric relations of the system. The angles are then used to calculate the control effort, a torque, that has to be exerted on the first link by the motor. This Node is programmed in Python as well.

The motor torque is transferred to a third node, which is written in C++, as XDDP is only available in this programming language. So the sole purpose of this node is to open the XDDP communication port and transfer the numerical value of the torque from the NRT kernel to the RT kernel.

Next, the RT kernel receives the torque. Here, it converts the torque to the appropriate input voltage. The conversion from torque to input voltage can be found in the data sheet of the motor. It transfers the voltage exactly every sampling time to the NRT kernel through a second XDDP port to be able to communicate with the General Purpose Input/Output (GPIO) pins of the RPi.

It takes $9ms$ to process an image and output a motor voltage. This needs to be taken into account when designing

the two-link system, as the eigenfrequencies of the system cannot be too high to avoid aliasing.

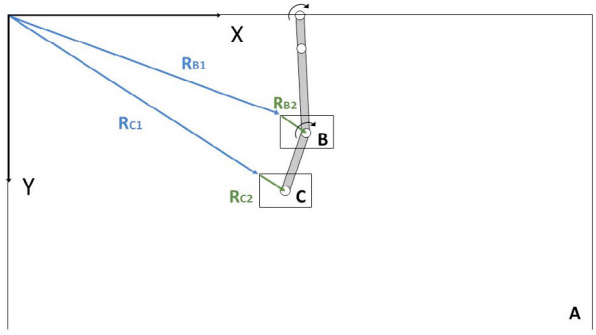


Fig. 4. The strategy to decrease processing time by processing only a relevant part of the image.

3. RESULTS AND DISCUSSION

To validate the performance of the setup, two experiments are conducted. First, an open-loop experiment is done to verify the software/hardware architecture and to observe the dynamics of the mechanical setup. Secondly, a closed-loop experiment with a PD-controller is performed. This is done to show that a controller can affect the dynamic behavior of the setup real-time.

3.1 Open-loop measurement

The open-loop experiment is the result of a short hit on the second link close to its joint. This is to approximate an impulse response. The measurement of the first and second joint angle are given in Fig. 5. The impulse force is applied after 9 seconds.

The first link does not have a restoring force, which means that it has a range of equilibrium points. In absence of friction the link would keep rotating, once in motion (zero eigenfrequency). However, this is a real setup so the bearings and the gearbox of the DC-motor experience friction. That explains why the first link comes to a halt after approximately $1s$. Hereafter, it remains at its current position. The second link has a restoring force due to the spring κ_2, which creates an isolated equilibrium at $q_2 = 0°$. Again, due to friction the equilibrium is reached after $4s$. The second link undergoes an oscillatory movement before coming to a halt.

The measurements represent qualitatively the movement of the two link systems as observed by the researchers. The sampling time is sufficiently small to observe the full dynamic behavior of the setup. Also, the noise-to-signal ratio is very low. Sufficient dynamic behavior is present in the setup such that control strategies can be tested to improve the impulse response.

3.2 Closed-loop: PD control

In this section a traditional Proportional-Derivative (PD), the equivalent of adding a virtual spring and damper, controller

$$C(s) = K_p\,(1 + K_d s) \tag{3}$$

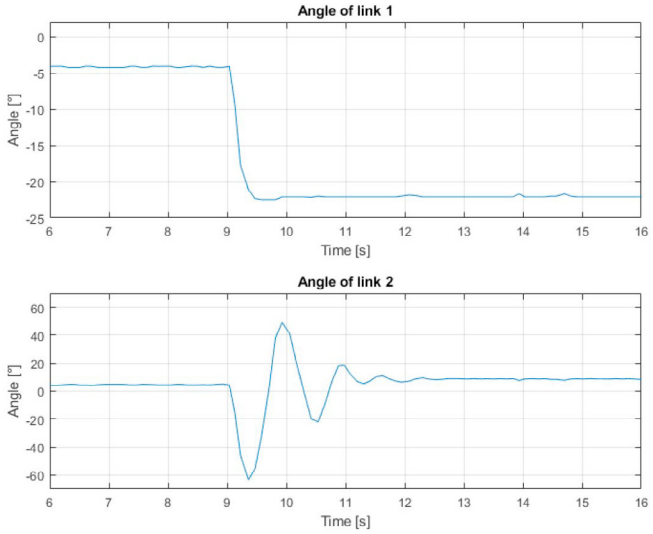


Fig. 5. Measurement of the open-loop angular position of the first (*top*) and second joint (*bottom*) for an impulse disturbance on the second link.

is implemented to validate the closed-loop possibilities of the setup. No I-action is included as it is already present in the motor model. Notice, that the two-link can only be actuated at the first joint and for stability reasons collocated control is chosen. This means that only the states of the first joints can be used in the controller. The states of the second link are merely measured for logging purposes.

The controller is tuned as follows. First, the system is split in two substructures: the first link with a virtual spring K_p is decoupled from the second link with its spring and is rigidly connected to a virtual ground. Both linearized subsystems have an eigenvalue:

$$\begin{cases} f_1 = \sqrt{\dfrac{\hat{I}_1}{K_p}} \\ f_2 = \sqrt{\dfrac{\hat{I}_2}{\kappa_2}} \end{cases} \tag{4}$$

with $\hat{I}_i = I_i + \frac{m_i l_i^2}{4}$ for $i \in \{1, 2\}$, I_i is the moment of inertia for a rotation around its center of mass, and m_i, l_i the mass and length respectively of link i. Now, K_p can be chosen such that both eigenfrequencies are equal. This means that

$$\begin{aligned} K_p &= \frac{I_1}{I_2}\kappa_2 \\ &= 9.14e\text{-}3 \end{aligned} \tag{5}$$

Next, root-locus is used to determine K_d, again on the linearized system including the P-controller. The value of K_d is chosen such that the real parts of the closed-loop poles are as negative as possible. This is the case for $K_d = 1.60e\text{-}2$.

The controller is implemented in the real system and the impulse response is shown in Fig. 6. A short impact has been applied close to the second joint to mimic an impulse on the second link. The magnitude and duration of impact are measured and are similar to the impact for Fig. 5. The impact is applied at $6.7s$. The first link overshoots to $-20°$

after which it returns to ±0°. The P-controller clearly introduces a restoring force to the system and creates a new isolated equilibrium at 0°. The first link gets actually stuck at −4°. This is because of an artificial dead band on the motor's torque, which avoids endless small controller torques due to small measurement errors. The first link reaches the equilibrium after 1s. The second link first deviates 60° from its equilibrium, which is similar to the open-loop experiment, but for the second peak only 30° is reached, compared to 50° in the open-loop experiment. The second link is stabilized in 2s, which is a reduction of 50%. This clearly shows that the closed-loop time behavior outperforms the open-loop response. This means that the PD-controller has a clear effect on the dynamics of the system.

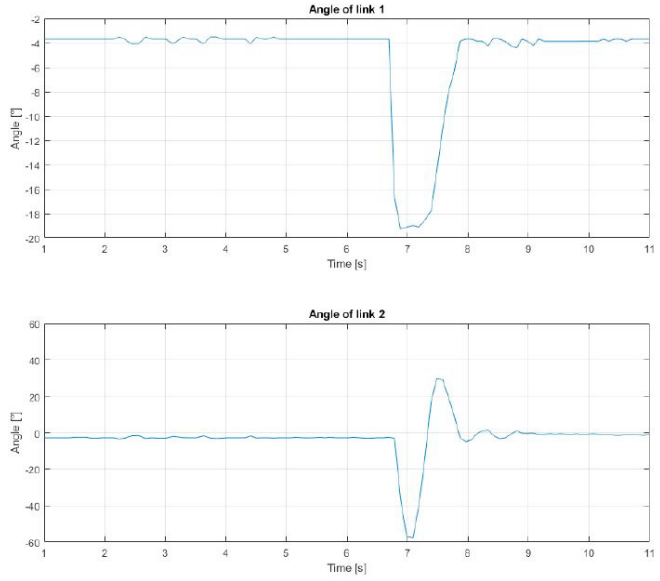


Fig. 6. Measurement of the angular position of the first (*top*) and second joint (*bottom*) for an impulse disturbance on the second link in a closed-loop configuration (PD-controller).

4. CONCLUSION

A mechanical setup of the classical underactuated benchmark system, the planar two-link manipulator is realized in a table-top version. To stay as close as possible to the actual benchmark the internal damping due to friction needed to be as minimal as possible. This is realized with two measures: a double deep groove ball bearing design is implemented to have smooth rotation at the joints and a camera is used to measure the angles of the joints. The latter avoids damping due to friction of mechanical sensors and due to wires going from the sensor to a data-logger. This also does not influence the dynamics of the system due to the addition of mass from the sensor.

To extract the angles from the images and to use these angles to steer the motor a Raspberry Pi is used. The OS is complemented with ROS to allow for subdividing the entire algorithm in sub-algorithms which can be developed in any programming language. Next, the output of each subroutine can easily be transferred to the next one. Furthermore, a dual kernel topology is introduced using Xenomai to discern between real-time and non-real-time tasks. The real-time is needed to assure a fixed sampling time. To allow both kernels to access the hardware an ADEOS layer is added as well. To achieve communication between RT and NRT tasks XDDP is used. The complete setup reaches a sampling time of 9ms.

An open-loop impulse response is measured to validate the dynamics of the system. The sampling time is found to be sufficiently small to measure the dynamical behavior, without any risk of aliasing to occur. Next, a PD-controller is tuned to the theoretical properties of the main system. The controller is implemented and showed to reduce the settling time drastically compared to the open-loop case. This shows that the controller is able to improve the impulse response of the closed-loop system.

For future work, it would be interesting to get an idea on the accuracy of the angle measurements. Now, a qualitative approach is used to validate the image processing, but it would be interesting to quantify the measurement error. Furthermore, it will be interesting to implement more advanced controllers. Finally, identification should be done on the actual setup. This would be useful to compare the controller performance in simulation with the measured time response.

ACKNOWLEDGEMENTS

The authors would like to thank ir. Frédéric Tarras for his contribution to this work during his Master thesis research.

REFERENCES

Ahmad, A. and Babar, M.A. (2016). Software architectures for robotic systems: A systematic mapping study. *The Journal of Systems and Software*, 122, 16–39.

Barbalace, A., Luchetta, A., Manduchi, G., Moro, M., Soppelsa, A., and Taliercio, C. (2008). Performance comparison of vxworks, linux, rtai, and xenomai in a hard real-time application. *IEEE Transactions on Nuclear Science*, 55(1), 435–439.

De Luca, A., Mattone, R., and Oriolo, G. (2000). Stabilization of an underactuated planar 2r manipulator. *International Journal of Robust and Nonlinear Control*, 10, 181–198.

Delgado, R., You, B.J., and Choi, B.W. (2019). Real-time control architecture based on xenomai using ros packages for a service robot. *Journal of Systems and Software*, 151, 8–19.

Fraden, J. (2016). *Handbook of Modern Sensors - Physics, Designs, and Applications*. Springer, Cham, Switzerland, 5 edition.

Gerum, P. (2019). Xenomai. URL `https://xenomai.org`. Accessed: 18-02-2022.

Hashimoto, K. (2003). A review on vision-based control of robot manipulators. *Advanced Robotics*, 17(10), 969–991.

He, G.P., Wang, Z.L., Zhang, J., and Geng, Z.Y. (2016). Characteristics analysis and stabilization of a planar 2r underactuated manipulator. *Robotica*, 34, 584–600.

Huang, H., He, W., Li, J., Xu, B., Yang, C., and Zhang, W. (2022). Disturbance observer-based fault-tolerant control for robotic systems with guaranteed prescribed

performance. *IEEE Transactions on Cybernetics*, 52(2), 772–783.

Jolles, J.W. (2021). Broad-scale applications of the raspberry pi: A review and guide for biologists. *Methods in Ecology and Evolution*, 12(9), 1562–1579.

Kondaveeti, H.K., Kumaravelu, N.K., Vanambathina, S.D., Mathe, S.E., and Vappangi, S. (2021). A systematic literature review on prototyping with arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, 40, 100364.

Liu, Y. and Yu, H. (2013). A survey of underactuated mechanical systems. *IET Control Theory and Applications*, 7(7), 921–935.

Mahindrakar, A.D., Rao, S., and Banavar, R.N. (2006). Point-to-point control of a 2r planar horizontal underactuated manipulator. *Mechanism and Machine Theory*, 41, 838–844.

Ortmeyer, C. (2014). Then and now: A brief history of single board computers. *Electronic Design Uncovered*, 6, 1–11.

Raspberry Foundation (2021). Raspberry pi. URL `https://github.com/raspberrypi/`. Accessed: 15-11-2021.

ROS (2021). Robot operating system. Accessed: 10-10-2021.

Shirai, Y. and Inoue, H. (1973). Guiding a robot by visual feedback in assembly tasks. *Pattern Recognition*, 5, 99–108.

Yuen, H.K., Princen, J., Illingworth, J., and Kittler, J. (1990). Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1), 71–77.

Zhao, Y., Gong, L., Huang, Y., and Liu, C. (2016). A review of key techniques of vision-based control for harvesting robot. *Computers and Electronics in Agriculture*, 127, 311–323.