



ROBOtic Replicants for Optimizing the Yield by Augmenting Living Ecosystems

Supplementary information for deliverable D1.4

Open-source library incl. communication, motion & high-level control

Lead Beneficiary	CVUT
Delivery date	30.4.2025
Dissemination Level	PU
Version	2.0
Project website	www.roboroyale.eu





This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 964492

DELIVERABLE SUMMARY SHEET

Project number	964492
Project Acronym	RoboRoyale
Title	ROBOTic Replicants for Optimizing the Yield by Augmenting Living Ecosystems
Deliverable No	D1.4
Due Date	Project month M42
Delivery Date	30.04.2025
Name	Open-source library incl. communication, motion & high-level control
Description	Open-source library including communication, motion and high-level control functions
Lead Beneficiary	CVUT
Partners contributed	UDUR, UNIGRAZ, METU
Dissemination Level	Public

This document is not a deliverable, since the D1.4 is not a report, but a software release. We provide this document as a courtesy to the potential users of the software. If you are not the system user, please disregard this document and refer to the software itself.



Table of contents

Table of contents.....	3
Robotic Observation and Interaction System.....	4
Software architecture and description.....	5
Hive controller sensing and interaction modules.....	6
XY actuator.....	7
XY actuator firmware.....	7
Actuator ROS driver.....	8
Camera ROS driver.....	8
Camera optics ROS driver.....	9
WhyCode marker detection.....	9
Interaction agent driver.....	10
High-level control.....	10
The agent behaviour control.....	11
Hive controller data recording and reporting.....	12
Hive controller data management.....	14
Data splitters.....	14
Data uploader.....	15
Master computer.....	16
Configuration and support packages.....	17
Software installation.....	20
Master and controller.....	20
Hive controller.....	22
Hive master.....	23
Software license.....	24
Current users.....	24
References.....	24
Appendix.....	25



Robotic Observation and Interaction System

The presented software is part of the Autonomous Robotic Observation and Behavioral Analysis (AROBAs) system, comprising two vertical gantry robots with infrared cameras. These robots perform long-term data gathering on two sides of the observation beehive. Additionally, the software contains modules to control special manipulators capable of interacting with the honeybee workers and the honeybee queen. A detailed description is presented in [1], the software itself is available for download on the project website (<https://roboroyale.eu>) and on Zenodo (<https://zenodo.org/records/15294021>).

One of the main goals of the autonomous observation system is to monitor the queen honeybee and gather behavioural data while she is actively working within the hive. To achieve this goal, the system must track the queen during her periods of activity and shift focus to other parts of the hive when she is at rest. The mechanism developed for this purpose is a vertical gantry system to allow for horizontal and vertical movement of the camera along the observation hive plane. Two such systems are required to monitor both sides of the hive and ensure continuous data collection. Figure 1 illustrates a schematic of the system, highlighting its various components and how they function together to meet these objectives.

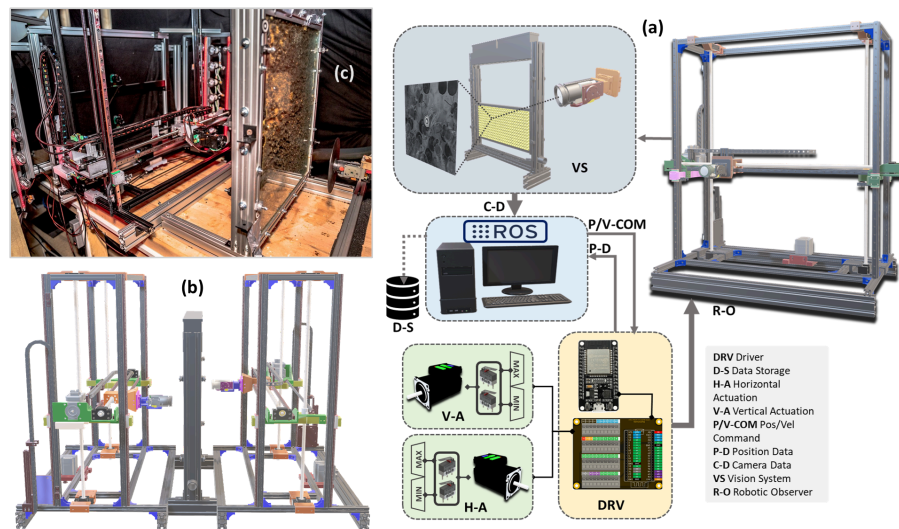


Figure 1. The autonomous robotic observation system. (a) The schematic diagram for different components and subsystems. (b) The CAD model of the experimental setup for beehive observation. (c) The constructed robot which is performing the observation. Courtesy of RoboRoyale Deliverable 1.2.

As illustrated in Figure 1, each mechanism incorporates a servo stepper motor for both the horizontal and vertical directions. In the horizontal axis, the motor is paired with a single ball screw and guided by two linear rails. For the vertical axis, motion is driven by two ball screws, each supported by two linear guides, forming a robust structure that ensures reliable load support. Each motor is equipped with an embedded controller for precise position and velocity control. The camera holder is designed to accommodate a custom manipulator equipped with biomimetic agents capable of directly interacting with bees.

Software architecture and description

The purpose of the software is to control the activity of the aforementioned robotic systems. The software is based on the Robot Operating System framework, complemented by service, maintenance and monitoring scripts developed for the Ubuntu 20.04 operating system. The software supports the management of several observation hives, each with its own robotic system. As previously described, each observation hive is served by two independent but cooperating robots controlled by a single computer called **(hive) controller**. The hive controller runs two sets of identical modules, each controlling the activity of a single robot monitoring one side of the hive. Additionally, each hive controller runs a set of tools aimed at automating data management and maintenance tasks. An overview of the software modules of one hive controller is provided in Figures 3 and 5. Several hive controllers running on the same network are monitored by a **master** computer, which continuously gathers information on their performance, runs diagnostics, and reports potential problems. The master also aggregates the state of the connected hive controllers and provides regular reports of their state. If the master detects problems with the data collection at some hive, it resets the relevant software modules. Furthermore, it controls the power delivery to hive controllers and hive actuators. Therefore, it can force a cold restart or even shutdown of the potentially malfunctioning robots of the hive observation systems. The architecture of the system with one master and several hive controllers is shown in Figure 6.

As the system generates 0.8TB of data per day and hive, a dedicated **network storage** was set up to hold the data for further processing. Each hive controller uploads the collected data in the form of rosbags complemented with metadata containing the types and numbers of the stored messages.

The network storage, master, and controllers are accessible remotely via an SSH connection. The control and recording software modules are running in terminal multiplexer sessions, enabling system users to monitor and interact with the individual modules, see Figure 2. This multiplexer is started automatically so that the data collection and interaction are initiated after the system (re)boots without user input. To enable rapid intervention, anomalous behaviours and potential faults are reported via a dedicated discord channel.

```
[ INFO] [1744607233.151771451]: Switched to Track-queen
[ INFO] [1744608005.280040762]: Other table sees the queen better
, mirroring it
[ INFO] [1744608032.803479604]: Switched to Mirror
[ INFO] [1744608032.834268311]: Queen spotted, starting tracking.
Total tracking time 147763 out of 285459. Tracking ratio 51%.
[ INFO] [1744608032.864922537]: Switched to Track-queen
[ INFO] [1744608258.279947997]: Other table sees the queen better
, mirroring it
[ INFO] [1744608346.678555005]: Switched to Mirror
[ INFO] [1744608559.306310488]: Next time of random publishing is
3233 seconds from now
[ INFO] [1744609026.966348515]: Queen spotted, starting tracking.
Total tracking time 147980 out of 286453. Tracking ratio 51%.
[ INFO] [1744609026.997172839]: Switched to Track-queen
, mirroring it
[ INFO] [1744607370.407841817]: Switched to Mirror
[ INFO] [1744607534.516098288]: Next time of random publishing is 1
585 seconds from now
[ INFO] [1744608004.699987330]: Queen spotted, starting tracking. T
otal tracking time 102810 out of 285433. Tracking ratio 36%.
[ INFO] [1744608004.730691076]: Switched to Track-queen
[ INFO] [1744608033.496872677]: Other table sees the queen better
, mirroring it
[ INFO] [1744608047.203366896]: Switched to Mirror
[ INFO] [1744608257.492551558]: Queen spotted, starting tracking. T
otal tracking time 102816 out of 285686. Tracking ratio 35%.
[ INFO] [1744608257.523186485]: Switched to Track-queen
[ INFO] [1744609027.496892979]: Other table sees the queen better
, mirroring it

data: "Track-queen"
---
data: "Track-queen"
---
boot-sess0:Cameras 1::Detection- 2::Motors 3::Operations* 4::Data 5::Diagnostics "/home/pc/workspace/sr" 07:37 14-Apr-25
```

Figure 2: Example terminal multiplexer session of the hive controller PC.



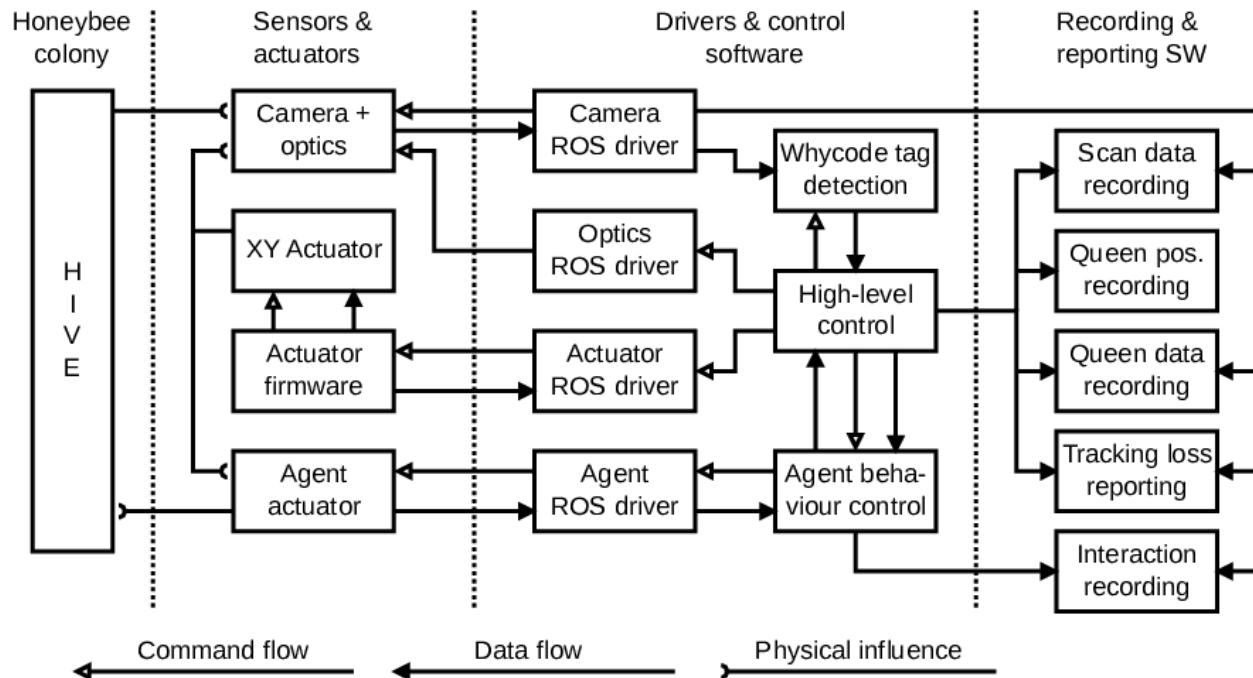


Figure 3: Software modules controlling a single robot of the observation system.

Hive controller sensing and interaction modules

Each hive controller controls two robots, which monitor and interact with the left and right sides of the observation hives. Each robot consists of an XY actuator, carrying a camera and (optionally) an agent to interact with the bees, see Figure 3. A dedicated control board runs **XY actuator firmware**, which is aimed at minimising vibrations that could disturb the bees. The camera contains an independent optical system capable of software control of the iris, zoom, and focus. The **camera**, its **optics**, **XY actuator**, and **agent actuator** have their respective **ROS drivers**, which translate their protocols into ROS-compatible messages. The **queen tag detection** node runs a whycode [2] method to identify the marker on the queen and/or special markers used to calibrate the system. The **high-level control** contains a complex state machine that processes the inputs from the tag detection, and it commands the XY actuator to move to track the queen or scan the honeybee comb. The **high-level control** modules running on the left and right sides of the observation hive inform each other of their current state to coordinate their activity. This is useful when tracking the honeybee queen as she transits from one side of the hive to the other. Moreover, if the honeybee queen rests for an extended period, the robot that does not track her initiates a high-detail survey of the comb to gather information about the brood. In specific situations, the **high-level control** activates the agent behavioural control, which then interacts with the queen or the worker bees.

XY actuator

The XY actuator system comprises a vertical motion module and a horizontal feed drive. The vertical motion module employs two synchronized ball screws (SFU1204) driven by a single iHss573610 servo stepper motor. Each ball screw is supported by two linear guides, ensuring stability and precision. The servo stepper provides 1,000 pulses per revolution, resulting in a linear resolution of approximately 4 μm . This vertical actuation unit is designed to carry the entire horizontal axis assembly along with the attached payload. The horizontal motion unit consists of a single ball screw coupled to another servo stepper motor and supported by two linear guides. It is designed to handle a 2 kg load over a 550 mm travel range, achieving a maximum speed of 10 mm/s. The linear guides minimize deflection and prevent buckling of the ball screw during operation.

To evaluate system performance, a vision-based measurement system was employed during repetitive motion tests. In each test cycle, a camera identified four reference tags located at the corners of the workspace and issued a repeated sweeping motion command to the mechanism. After each cycle, position drift was quantified by comparing the camera-measured position of the first tag. The long-term evaluation, conducted over eight hours, demonstrated high positional accuracy and repeatability, with a maximum drift of less than 8 mm after traversing a total distance of 1 km.

XY actuator firmware

The **RoboRoyale Xy Table Driver** firmware is an ESP32-based control system that manages two stepper motors for precise positioning. The system uses FreeRTOS tasks to handle motor control, command processing, and communication separately. The motor control uses hardware timers to generate pulse and direction signals for stepper drivers, with a state machine that manages acceleration, constant speed, and deceleration phases. Each axis has limit switches for safety and automatic homing.

The system accepts commands through two channels: UDP network communication and serial UART. UDP provides real-time position commands and status feedback, while serial offers three command types: position/speed control, position reset, and full parameter control including acceleration settings. All commands include checksum validation and the system responds with current position data. The communication is designed for integration with external control systems.

The implementation uses hardware timer interrupts for accurate pulse generation and includes watchdog timers for system reliability. The code is modular with separate classes for motor control, command handling, and WiFi management which is used for developmental purposes only and is off in real deployment. Configuration parameters are easily adjustable for different table sizes, and the system includes comprehensive safety features through limit switch monitoring and automatic homing sequences.



Actuator ROS driver

The actuator driver is a **ROS node** written in Python programming language in a package ***rr_xytable***. The main purpose of the driver is to integrate the XY gantry mechanism into a ROS framework. Thus, the driver acts as a bridge between ROS and a microcontroller controlling connected via a USB interface

The driver communicates with the XY table's microcontroller using a udev-rules set virtual serial device operating at 115200 baud rate. The driver accepts position commands over a ROS topic of `joint_state` type, with optional information on maximal speed and acceleration limits. These are converted into a fixed-sided packet, complemented with a header and a checksum and sent over to the controller board. With every received command, the driver responds with a response packet, which provides the current XY positions of the gantry, the status of the limit switches, and motor electrical current values. While the positions are provided over a `pose_stamped` message, the limit switches, motor currents and other diagnostics are provided in a custom message.

Since the gantry mechanism is equipped with incremental position sensors only, the driver also provides a ROS service to (re)set the internal position of the gantry head. This is used during system calibration.

Camera ROS driver

The ***rr_usb_camera*** package is a ROS camera driver for Harrier zoom USB cameras. It extends the standard `cv_camera` driver with features needed for multi-camera robotic systems. The driver uses OpenCV to capture video from USB devices and publishes the images as ROS messages at 1920x1080 resolution and 30Hz frame rate.

The system has three main parts: a Capture class that handles the camera hardware, a Driver class that manages settings and timing, and both regular node and nodelet versions for different deployment needs. It includes camera calibration support, automatic device reset when cameras fail, and frame rotation for different mounting angles. The driver can handle MJPEG and UYVY video formats and works with standard ROS image processing tools.

The deployment setup uses udev rules to automatically find cameras by their USB serial numbers and creates consistent device names. Each camera gets its own ROS namespace based on the robot's hostname (like `hive_0/xy_0`), and all settings are stored in YAML files. This makes it easy to deploy the same system across multiple robots while keeping each camera's settings separate. All camera calibrations are also stored in this package to assure backtracability of used parameters.



Camera optics ROS driver

The ***rr_lens_controller*** package is optics ROS driver is a Python-based ROS node designed to control the zoom, focus, and iris of a Harrier camera lens module. It communicates with the camera hardware via USB using the Harrier USB SDK and implements the VISCA protocol, which is a standard for serial communication with camera systems. The primary purpose of the driver is to allow the camera optics control via the ROS framework.

In particular, the driver allows you to adjust the focal length in order to control the focus distance to ensure sharpness of the details at different distances as we need to monitor the bees, the comb surface, and the comb cell contents. Finally, the driver allows to control the aperture size to balance the image sharpness and brightness levels. The driver abstracts away low-level communication details and exposes control functions for high level (ROS-based) control nodes.

The driver is organized into layers: the hardware communication, which handles the USB interface with the camera, the protocol layer, which implements the VISCA protocol, the control layer to provide an API for the zoom, focus, and iris, and application layer, which handles the driver operation.

The driver receives commands over separate topics, each related to zoom, focus or iris control. Moreover, the driver reports the state of the zoom, iris and focus over a custom ROS message.

WhyCode marker detection

The ***rr_whycode*** is a ROS package enabling tracking of a marker and by extension the queen.

To ensure the vision system operates smoothly and reliably during critical tasks, the environment is augmented with black-and-white fiducial markers. These markers are detected by a dedicated module that processes camera images in real time. The module is based on the WhyCode tracking method, which accurately estimates both the position and orientation of each marker.

Each side of the hive is equipped with four calibration markers. In addition, the honeybee queen is fitted with a circular fiducial marker attached to the dorsal side of her thorax. This marker consists of two concentric black-and-white rings with a flexible circular encoding pattern, enabling six-degree-of-freedom tracking of the queen. Owing to its circular design, the marker conforms better to the queen's thorax compared to square fiducial markers. Square markers, which typically cover the entire thoracic area, either fail to utilize the available surface efficiently or have protruding corners that worker bees may chew, potentially degrading tracking performance.



The WhyCode detection method is highly computationally efficient, capable of locating a tag in under one millisecond. This enables the system to rapidly respond to the queen's movements. Empirical data show that while the actuator tracks the walking queen, her mean distance from the camera's optical axis is 3.78 mm, with a standard deviation of 3.20 mm. To minimize motor vibrations that could disturb the bees, actuator velocity is software-limited to 1.0 cm/s. Although the queen can occasionally move faster, this speed limit has not resulted in any tracking loss. Furthermore, to avoid unnecessary actuator motion caused by minor displacements of a resting queen, the system initiates re-centering only when her marker shifts more than 1.0 mm from the optical axis.

The performance of the WhyCode tracking module was evaluated using 16,230 manually annotated images, of which 6,414 contained the queen. The system achieved an average localization error of 0.847 pixels (56.7 μm), with a maximum error of 2.23 pixels (149 μm). The detection precision reached 1.0, recall was 0.952, and the resulting F1 score was 0.975.

Interaction agent driver

The *rr_poker* package is a ROS-based control system for a Versatile Linear Actuation Device (VLAD) that manages Dynamixel servo motors. The system uses Python nodes to handle different input methods including joystick, keyboard, and programmatic control, all feeding into a common control module that runs PID controllers for precise position control. The architecture separates input handling from control logic, allowing multiple control interfaces to work with the same underlying motor control system.

The control system communicates with Dynamixel servos through serial ports using the *dynamixel_interface* package, with configuration managed through YAML files that set servo parameters and safety limits. A specialized z-axis ramp controller provides smooth velocity changes for safe operation. The system uses ROS parameter server for runtime settings and implements proper safety limits on position, velocity, and torque to prevent damage to hardware.

High-level control

The main package controlling all system behaviour is *rr_xy_operator*.

The high-level control module coordinates the motion of the XY gantry and the camera's zoom, focus, and iris settings based on data received from the WhyCode detection module, motors, optics driver, and the high-level control node managing the opposite side of the hive. It integrates these inputs to compute the global position of the tracked queen, which is then recorded for later analysis. The module also determines whether the current camera feed should be published to topics relevant to comb scanning, queen tracking, or worker bee behavioral studies. Additionally, it monitors queen tracking losses, triggering specific system behaviors designed to quickly locate the queen.



At startup, the high-level control moves to four calibration markers located at the hive's corners to calculate the transformations between the motor coordinate system and the hive-side reference frames. It then retrieves the timestamp of the last comb scan to decide whether upcoming focal site observations and comb scans can be interrupted if the queen is detected. Next, the system performs focal site observations at predefined locations, capturing multiple high-resolution videos before proceeding with a full comb scan. If the queen is detected during a scan that is marked as interruptible, the actuator begins physically tracking her, keeping her centered within the camera's optical axis.

The high-level control module tracking the queen transmits her time-stamped positions to the control module of the other side. When not calibrating or performing continuous scanning, the second actuator mirrors these movements. This setup ensures that if the queen crosses to the opposite comb—by crawling over an edge or through a gap—the mirroring actuator can detect her and take over her tracking, while the first actuator switches to mirroring mode.

Using the queen's transmitted positions, the mirroring system estimates the probability that the queen will remain resting for the next ten minutes, based on historical resting data. If this probability exceeds 95%, the mirroring actuator begins focal site observations and subsequently scans the entire comb. High-resolution scans older than 24 hours and standard scans older than 8 hours are designated as uninterruptible tasks. Otherwise, scanning halts once the queen resumes movement, allowing the second actuator to resume mirroring the tracking actuator.

If the queen becomes occluded or both actuators fail to detect her for over a minute, the system recalibrates by docking at the calibration markers. After recalibration, it reassesses the interruptibility of upcoming observations and scans before resuming focal site imaging and comb scanning while attempting to reacquire the queen.

The agent behaviour control

The high-level control module can interact with an agent behaviour control node, which is implemented as a ROS action server. This module can be activated if a set of conditions, depending on the actual interaction scenario, are met. Once activated, the agent behaviour module can take temporary control over the high-level control node, and operate the XY gantry mechanism, the camera optics, as well as the interacting agent and its supporting devices. Since the primary purpose of our software is observation, we provide an example action server, which can be extended depending on the particular interaction scenario or experiment. The interface of the action server allows the high-level control module to terminate the agent control in case other, high-priority actions have to be performed.



Hive controller data recording and reporting

The controller also runs a **queen data recording** and a **scan data recording** script, which record information and images of the queen and the comb, respectively. Both scripts operate as a bash-based loop invoking a rosbag recording module with parameters ensuring proper naming of the recorded files according to the particular hive, side, date, and time of data acquisition. The name is chosen in accordance with the data management plan of the RoboRoyale project. A cron-based job interrupts these recording scripts on a regular basis. As the recording scripts execute in a loop, the recording is resumed immediately after termination, resulting in the data being split into rosbags covering regular intervals - an hour in the case of the queen data and a day in case of the scan data. Two dedicated files contain version-controlled names of the topics recorded by the aforementioned scripts.

A separate **queen position recording** script records the queen position in a text format. The script is based on a rostopic tool, which outputs a single record per line with data organised as a comma-separated list with several columns related to the queen position published by the high-level control module. Similarly to the queen and scan data recording, the queen position script is terminated and restarted every midnight, ensuring that the position logs size remain manageable by standard text editing tools. This facilitates easy analysis of the queen motion without the need to download the full data rosbags, which are typically over 30 GB in size. Each log file contains a header with descriptions of the contents of the individual columns constituting the file.

A separate **tracking loss reporting** module stores the last 2 minutes of the images containing the queen movement. In case the queen tracking is lost, the module creates a video documenting the events that resulted in the loss of queen tracking, and publishes the video over a dedicated discord channel, see Figure 4. The videos contain the events preceding the tracking loss, the process of the tracking loss, and the subsequent actions performed by the system to search for the queen.

An **interaction recording script** is tailored to store information about the agent-queen or agent-worker bee interaction. As the interaction topics are scenario-dependent, the interaction script provided in the software shows an example of such topic selection.

The aforementioned recording and reporting modules store information about both sides of the observation hive. Therefore, they are not duplicated for the right and left hive side. We include them in Figure 3 to better illustrate the data and command flows in the hive monitoring system.





Figure 4: A detail of the video report of the queen tracking loss event.

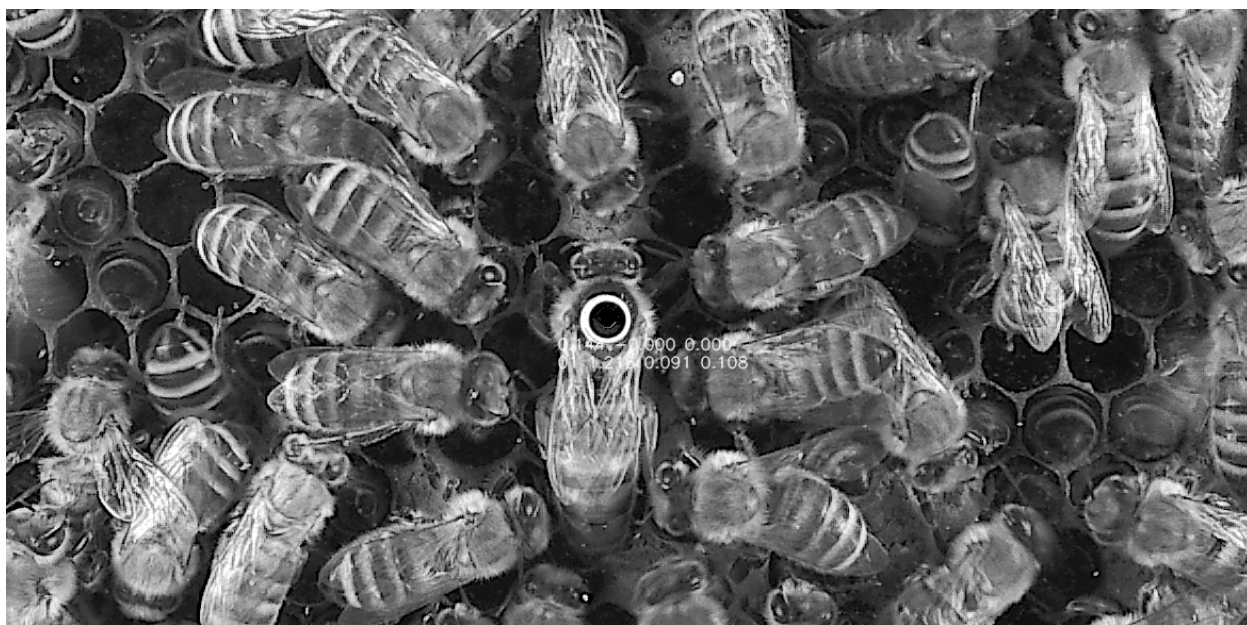


Figure 5: A detail of the queen with WhyCode marker detection results.

Hive controller data management

Data splitters

The data stored by the aforementioned modules are handled by a set of scripts that are invoked by the controller's cron subsystem on a regular basis. As mentioned before, the **queen data splitter** terminates the **queen data recording** on an hourly basis, causing it to save the queen-related data in a rosbag format. Immediately after the termination, the **queen data recording** starts to record another rosbag again. This causes the queen data to be split in hour-long rosbags. The same procedure is performed by the **scan** and queen **position data splitters** on a daily basis.

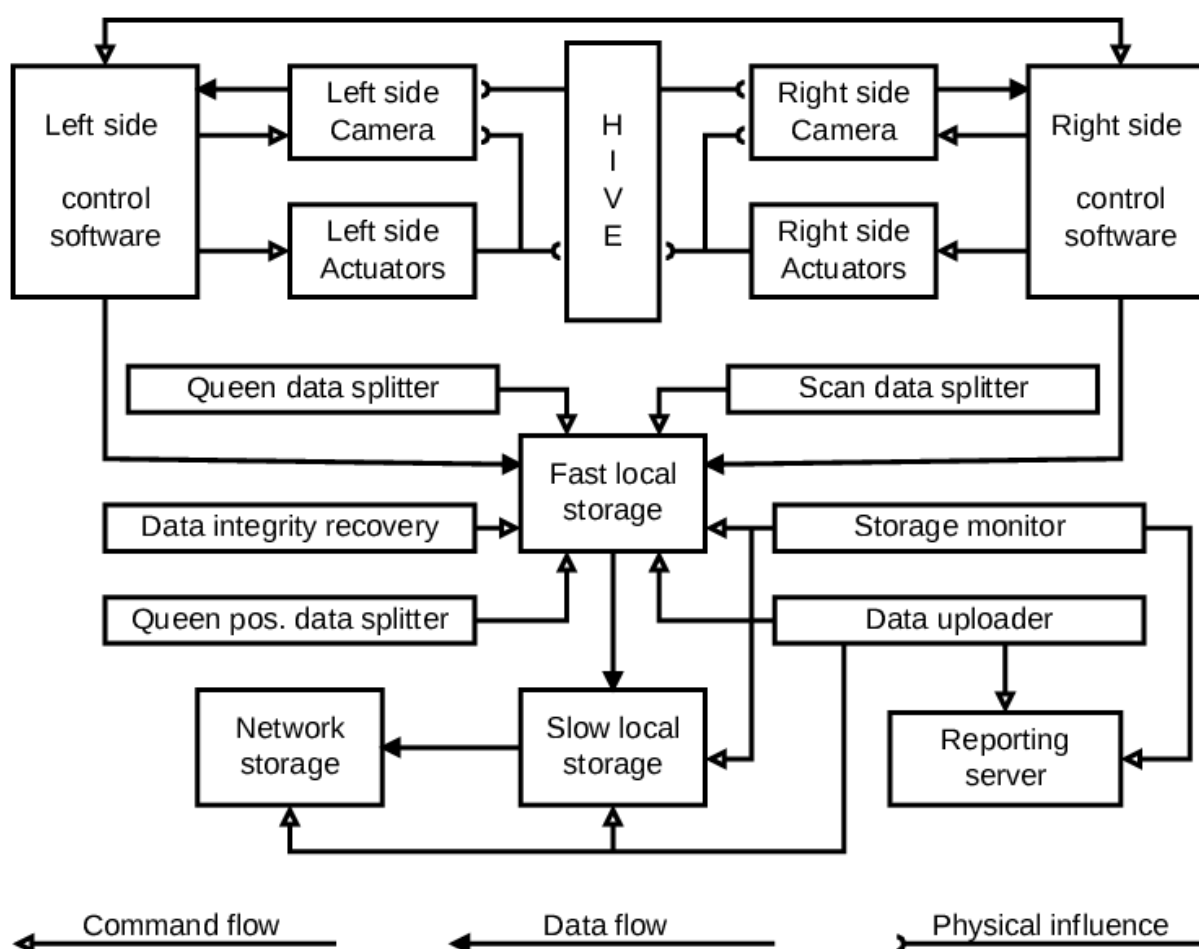


Figure 6: Hive controller software modules. The modules constituting the Right and left side control software are displayed in Figure 3.

Data uploader

The **data uploader** script then searches for the completed rosbags on the fast storage and extracts and stores their metadata. Then, it calculates their MD5 signature and moves them from a fast SSD data storage to a slower, rotating HDD storage. Finally, it copies the rosbags to the network storage and invokes MD5 calculation on the network storage computer. If the signatures match, the files are marked as uploaded correctly and moved to a separate folder on the controller HDD, which has enough space to contain two weeks of data.

The system user can invoke **hdd cleanup** scripts, which verify if the data were uploaded to the network storage and remove these data from the controller to free space. A single day of data produced by the controller occupies approximately 0.8 TB. Power outages, system failures, forced restarts, and similar events can cause the rosbags recorded by the data recording modules to be incomplete and unreadable.

The **data recovery module** detects these data, performs their recovery through the rosbag reindexing, and prepares them for upload. Similarly to the aforementioned modules is invoked through the cron subsystem.

The **storage monitor** regularly checks the storage status and reports it via a discord channel. In case of low storage, warnings are sent through another dedicated channel with high-priority messaging.

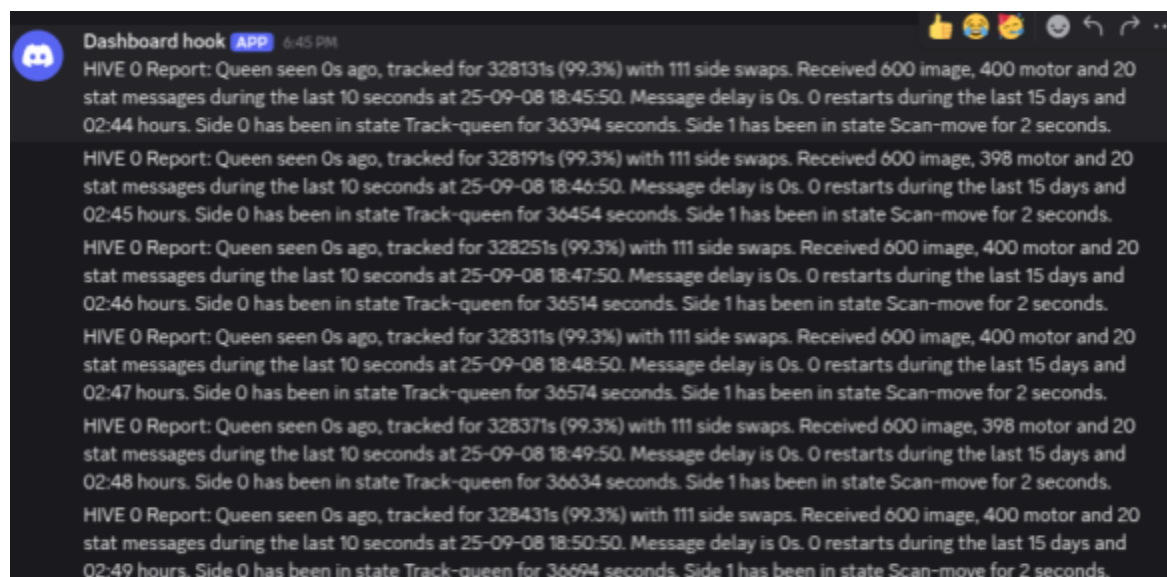


Figure 7: Example of the hive state monitoring report.

Master computer

The system is meant to perform data gathering from several beehives simultaneously. To provide a central point to access and manage the information provided by the individual hives, a separate computer runs a set of diagnostics modules, which aggregate the information from the individual hive controllers, see Figure 6.

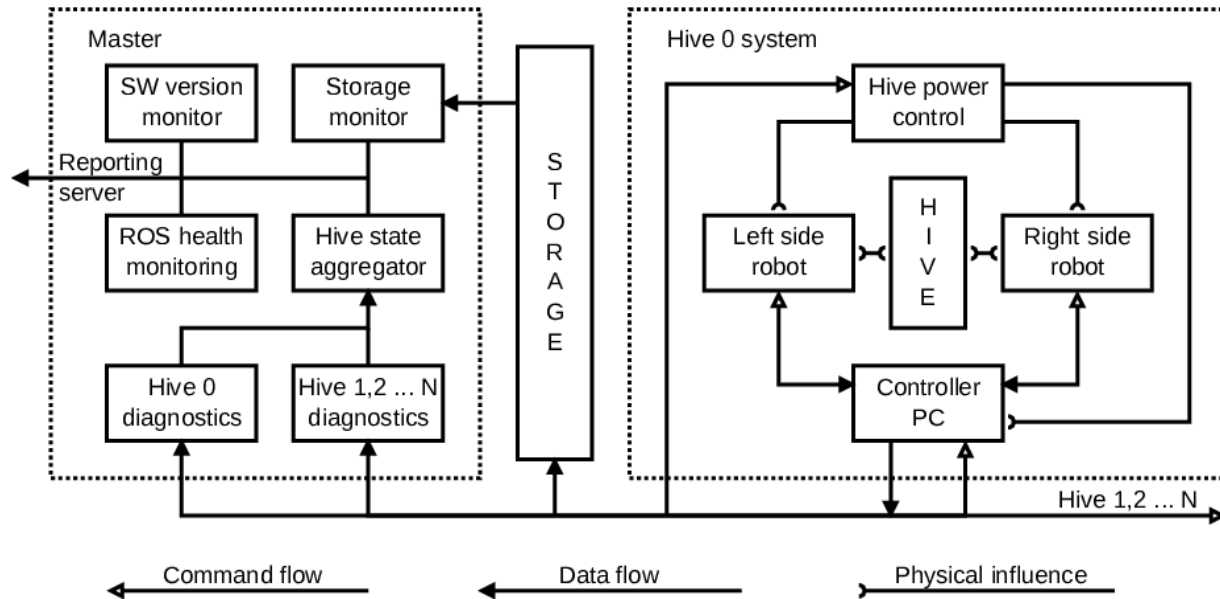


Figure 6: Core modules of the master computer and hive controllers and their interactions.

The computer also monitors the status and response of the core module of the Robot Operating System with its **ROS health monitoring** module. The **storage monitoring module** regularly reports the status of the data collection of the hive controllers; see an example report in Table I. A **hive diagnostics** module aggregates data from the left and right robots interacting and monitoring a given hive, and reports the state over a dedicated discord channel, see Figure 5. The master computer runs one of these modules per monitored hive and displays its output in a single screen of a terminal multiplexer. Furthermore, the **hive diagnostics** modules report the state of the hives to the reporting server once per minute. The **hive state aggregator** processes the outputs of the diagnostics modules, gathers images from the hives, and publishes a short graphical summary every 10 minutes, see Figure 7. In case the data from the hive diagnostics or ROS monitoring modules indicate an issue with some of the hives, a report is sent, and the master will attempt to restart the relevant software modules. If the restart does not resolve the issue, e.g., in case of hardware-related faults, the master will use its **power management** modules to temporarily cut off the power to the faulty controllers in order to force a reboot. System users are notified using a high-priority channel as the given hive might need maintenance.

Graz Master PC Bot - 4:55 AM

[4:55 AM] The last h0xy rosbag on NAS is /data/nas/h0xy_queen_2025-04-14-03-00-02.bag sized 34527 MB with 107969 queen images. h0xy /home/pc/rosbags has 0 queued bags and 704 GB left, enough for 14 hours. h0xy /data/hdd has 0 queued bags and 13064 GB left, enough for 261 hours.

[4:55 AM] The last h1xy rosbag on NAS is /data/nas/h1xy_queen_2025-04-14-03-00-02.bag sized 49312 MB with 108054 queen images. h1xy /home/pc/rosbags has 0 queued bags and 717 GB left, enough for 14 hours. h1xy /data/hdd has 0 queued bags and 12979 GB left, enough for 259 hours.

[4:55 AM] The last h2xy rosbag on NAS is /data/nas/h2xy_queen_2025-04-14-03-00-02.bag sized 64444 MB with 108052 queen images. h2xy /home/pc/rosbags has 0 queued bags and 735 GB left, enough for 14 hours. h2xy /data/hdd has 0 queued bags and 13003 GB left, enough for 260 hours.

[4:55 AM] The last h3xy rosbag on NAS is /data/nas/h3xy_queen_2025-04-14-03-00-02.bag sized 57758 MB with 107981 queen images. h3xy /home/pc/rosbags has 0 queued bags and 694 GB left, enough for 13 hours. h3xy /data/hdd has 0 queued bags and 13388 GB left, enough for 267 hours.

Table 1: Example report of the master storage monitoring module

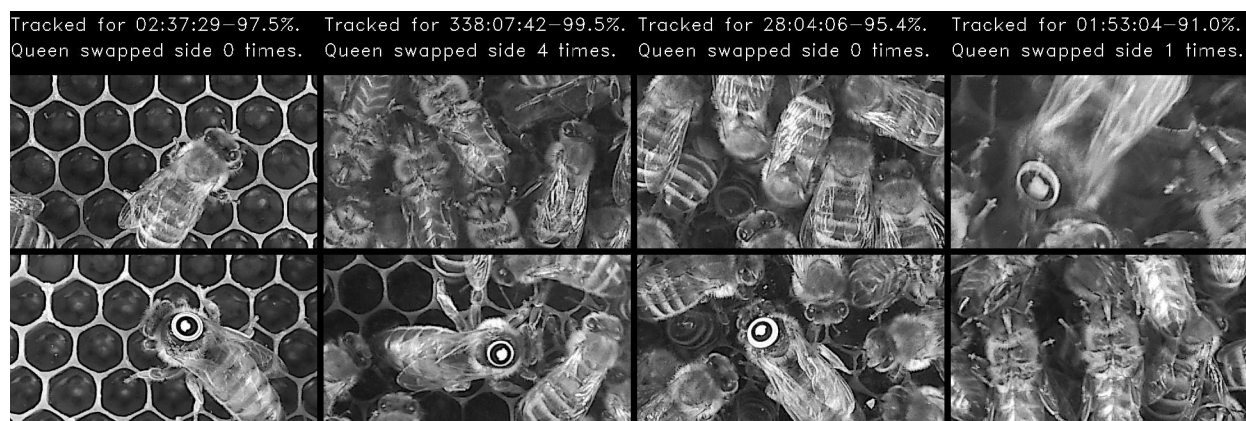


Figure 7: Hive state aggregator report example.

Configuration and support packages

The software running on the individual hive controllers and master requires a specific setup, e.g., `fstab` entries for network storage, `udev` rules for attached devices, `cron` tables etc. These configuration files are part of the **`rr_os_setup`** package. Furthermore, the **`rr_msgs`** package contains definitions of the ROS messages, configurations, and services tailored to the system.



Captain Hook APP – 10:28 AM

HIVE 1 WARNING: Subscriber /hive_1/xy_1/camera/image_raw_status did not obtain images during the last 16 seconds. Camera problems!

HIVE 1 WARNING: Subscriber /hive_1/xy_1/camera/image_raw_status did not obtain images during the last 26 seconds. Camera problems!

HIVE 1 WARNING: Subscriber /hive_1/xy_1/camera/image_raw_status did not obtain images during the last 36 seconds. Camera problems!

HIVE 1 WARNING: Subscriber /hive_1/xy_1/camera/image_raw_status did not obtain images during the last 46 seconds. Camera problems!

HIVE 1 WARNING: Side 1 devices were idle for 56 seconds!

HIVE 1 WARNING: PC is off, setting its power to 0 - response is 1 (1 = fine)

HIVE 1 WARNING: Side 1 devices were idle for 76 seconds!

HIVE 1 WARNING: PC is off, setting its power to 0 - response is 1 (1 = fine)

HIVE 1 WARNING: PC is booting for 20 seconds, setting its power to 1 - response is 1 (1 = fine)

HIVE 1 WARNING: PC is booting for 80 seconds, setting its power to 1 - response is 1 (1 = fine)

Table II: Example warning of the hive diagnostics detecting a camera issue and performing a cold restart through the power management modules.



Table III: Overview of the individual modules, the package they are part of, the computer running them, the terminal multiplexor window they are started in, and the initials of the main contributors.

Module	Package	System	Where	Major contributors
ROS camera driver	rr_usb_camera	Hive controller	tmux 0	JU
ROS optics driver	rr_lens_controller	Hive controller	tmux 0	JU
queen tag detection	rr_whycode	Hive controller	tmux 1	JU, TK
actuator firmware	RoboRoyale Xy Table Driver	Actuator MCU	-	FRB
actuator ROS driver	rr_xytable	Hive controller	tmux 2	FRB
High-level control	rr_xy_operator	Hive controller	tmux 3	TK
queen & scan recording	rr_xy_operator	Hive controller	tmux 4	TK
tracking loss reporting	rr_xy_operator	Hive controller	tmux 5	TK
agent ROS driver	rr_poker	Hive controller	tmux 6	TR
agent behavioural control	rr_interaction	Hive controller	tmux 6	TR
interaction recording	rr_interaction	Hive controller	tmux 6	TR
data integrity recovery	rr_os_setup	Hive controller	cron	TK, TR, GB
controller storage monitor	rr_os_setup	Hive controller	cron	TK, TR, GB
data uploader	rr_os_setup	Hive controller	cron	TK, TR, GB
master storage monitor	rr_os_setup	Master	tmux 2	TR
ROS health monitor	rr_os_setup	Master	tmux 0	GB, TK
hive state aggregator	rr_datacollection	Master	tmux 2	TK
hive diagnostics	rr_datacollection	Master	tmux 1	TK
power management	rr_power_plug	Master	tmux 3	JU



Software installation

Here, we provide a coarse step-by-step installation guide for both master and hive controller PCs. We assume that the user has already downloaded the `rr_os_setup.zip`, `rr_controller.zip` and `rr_master.zip` files from zenodo.

Master and controller

1. Install `Ubuntu 20.04 LTS`, use safe graphics mode, setup locale `US`, location `YOUR-TOWN`, without update, without third party drivers
2. Setup the networking for a given pc
3. Disable unattended-upgrades: `sudo aptitude -y purge unattended-upgrades`
4. Init upgrade:
 - a. `sudo apt-get update`
 - b. `sudo apt-get -y install aptitude`
 - c. `sudo aptitude update`
 - d. `sudo aptitude -y full-upgrade`
 - e. `sudo aptitude -y install openssh-server`
5. Nvidia driver and cuda toolkit
 - a. `sudo ubuntu-drivers install nvidia:535`
 - b. `sudo aptitude -y install nvidia-cuda-toolkit`
6. Verbose boot
 - a. `sudo sed -i '/GRUB_CMDLINE_LINUX_DEFAULT/ s/quiet splash//' /etc/default/grub`
 - b. `sudo update-grub`
7. Install tools
 - a. `sudo aptitude -y install mc vim tmux htop iotop bmon powertop net-tools dnsutils nmap curl git traceroute valgrind ncdu`
 - b. `sudo aptitude -y install tshark`
 - c. `sudo usermod -a -G wireshark <user>`
 - d. `sudo aptitude -y install guvcview v4l-utils v4l-conf libv4l-dev libv4l2rds0`
 - e. `sudo aptitude -y install lm-sensors hddtemp`
 - f. `sudo sensors-detect --auto`



8. Set preference to IPv4

- a. Uncomment line 54 of /etc/gai.conf: `sudo vim /etc/gai.conf`, change `#precedence ::ffff:0:0/96 100` to `precedence ::ffff:0:0/96 100`
- b. `sudo sed -i '54 s/^#//' /etc/gai.conf`

9. Setup miniforge3

- a. `wget -P Downloads/ https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Miniforge3-23.11.0-0-Linux-x86_64.sh`
- b. `bash Downloads/Miniforge3-23.11.0-0-Linux-x86_64.sh`
- c. `conda config --set auto_activate_base false`

10. Setup ROS Noetic

- a. `sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`
- b. `curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -`
- c. `sudo aptitude update`
- d. `sudo aptitude -y install ros-noetic-desktop-full`
- e. `echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc`
- f. `source ~/.bashrc`
- g. `sudo aptitude -y install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential`
- h. `sudo rosdep init`
- i. `rosdep update`
- j. `sudo aptitude -y install python3-catkin-tools`

11. Nvtop (after allowing IPv4 preference as apt-add-repository hangs on IPv6)

- a. `sudo add-apt-repository -y ppa:flexiondotorg/nvtop`
- b. `sudo aptitude -y install nvtop`

12. Add addresses of relevant machines to etc hosts

- a. `sudo vim /etc/hosts`
 - i. `xxx.xxx.xxx.xxx master`
 - ii. `xyy.xxy.xxy.xxy h0xy`



Hive controller

1. Get os setup correctly, ssh keys and so on
 - a. `unzip rr_os_setup.zip ~/`
 - b. `ln -s ~/rr_os_setup/vimrc ~/.vimrc`
 - c. `ln -s ~/rr_os_setup/tmux_startup.sh ~/tmux_startup.sh`
 - d. `ln -s rr_os_setup/tmux/tmux.sh-controller tmux.sh`
 - e. setup ssh for passwordless logins to the network storage
 - f. `rm .bashrc`
 - g. `ln -s rr_os_setup/bashrc/dotbashrc-controller .bashrc`
 - h. `source .bashrc`
 - i. `croninstall`
 - j. `sudo cp ~/rr_os_setup/rc.local.master /etc/rc.local`
2. Download prerequisites
 - a. `sudo aptitude install python3-pip`
 - b. `pip install pyserial`
 - c. `pip install numpy==1.24.1`
 - d. `pip install scipy==1.10.0`
 - e. `pip install control==0.9.4`
3. edit the sudoers file by adding the line
 - a. `sudo vim /etc/sudoers`
 - i. `pc ALL=(ALL) NOPASSWD:/usr/bin/usbreset,/usr/sbin/reboot`
4. Setup folders and permissions
 - a. `mkdir -p ~/workspace/src`
 - b. `mkdir -p ~/rosbags`
 - c. `sudo usermod -a -G video pc`
 - d. add fstab entry to have /data/hdd to point to HDD
 - e. add fstab entry to point /data/nas to network storage
5. Setup workspace
 - a. `cd ~/workspace/src`
 - b. `unzip controller.zip`
 - c. `cd ~/workspace`
 - d. `catkin build`
6. Connect to the reporting server
 - a. create discord rooms for individual reporting modules
 - b. create a webhook for each room
 - c. add the webhook to the reporting modules



Hive master

1. Setup OS

- a. `unzip rr_os_setup.zip ~/`
- b. `ln -s ~/rr_os_setup/vimrc ~/.vimrc`
- c. `ln -s ~/rr_os_setup/tmux_startup.sh ~/tmux_startup.sh`
- d. `ln -s rr_os_setup/tmux/tmux.sh-master tmux.sh`
- e. `ln -s rr_os_setup/bashrc/dotbashrc-master .bashrc`
- f. `source .bashrc`
- g. `croninstall`
- h. `sudo cp ~/rr_os_setup/rc.local.master /etc/rc.local`

2. Setup workspace

- a. `mkdir -p ~/workspace/src`
- b. `unzip master.zip`
- c. `cd ~/workspace`
- d. `catkin build`

3. Edit logs

- a. `sudo vim /etc/rsyslog.d/50-default.conf`: uncomment `cron.* /var/log/cron.log`
- b. `sudo service rsyslog restart`
- c. `sudo service cron restart`

4. Connect to the reporting server

- a. create discord rooms for individual reporting modules
- b. create a webhook for each room
- c. add the webhook to the reporting modules



Software license

For open research and education purposes, the software is licensed under the Apache 2.0 licence. For commercial use, the users have to seek consent of the copyright holders. Attribution information was added to the NOTICE and CITATION files. Concrete contributor names and copyright holders were extracted from the software repositories.

Current users

Currently, the software is being used by a team based at the Vienna “GRG 19 - School in Nature” secondary academic school (*Realgymnasium and grammar school*). Their team is replicating the autonomous observation system without the interaction components. They will follow the documentation of the system and the software manual and provide feedback regarding the improvements that will facilitate easier setup, use, and maintenance of the observation system.

References

- [1] J. Ulrich, M. Stefanec, F. Rekabi-Bana, et al. Autonomous tracking of honey bee behaviors over long-term periods with cooperating robots. *Science Robotics*, 9(95), 2024, eadn6848.
- [2] J. Ulrich, J. Blaha, A. Alsayed, T. Rouček, F. Arvin, T. Krajník. Real-Time Fiducial Marker Localisation System with Full 6 DOF Pose Estimation. *SIGAPP Appl. Comput. Rev.* 23(1), 2023, pp. 20–35.



Appendix

This appendix contains example data as captured using the software described. The data originate in the XY actuator ros driver, which passes them to the high-level control module. The high-level control module recalculates the motor position using a homography matrix established during the last calibration step. This is used to estimate the position of the camera head in the comb coordinate system. This position is then combined with the WhyCode system output, which provides the position of the queen tag in the camera coordinate frame based on the images provided by the system camera. The relative position of the tag is converted to a global coordinate system of the comb and passed to the motor controller, which then moves the camera head so that the queen tag is kept in the optical centre of the camera image. The XY actuator ROS driver ensures that the commands are executed in a way that does not produce excessive vibrations that would disturb the honeybee colony. The position of the queen and the camera head (robot) over a period of 5 hours are shown in Figure A.

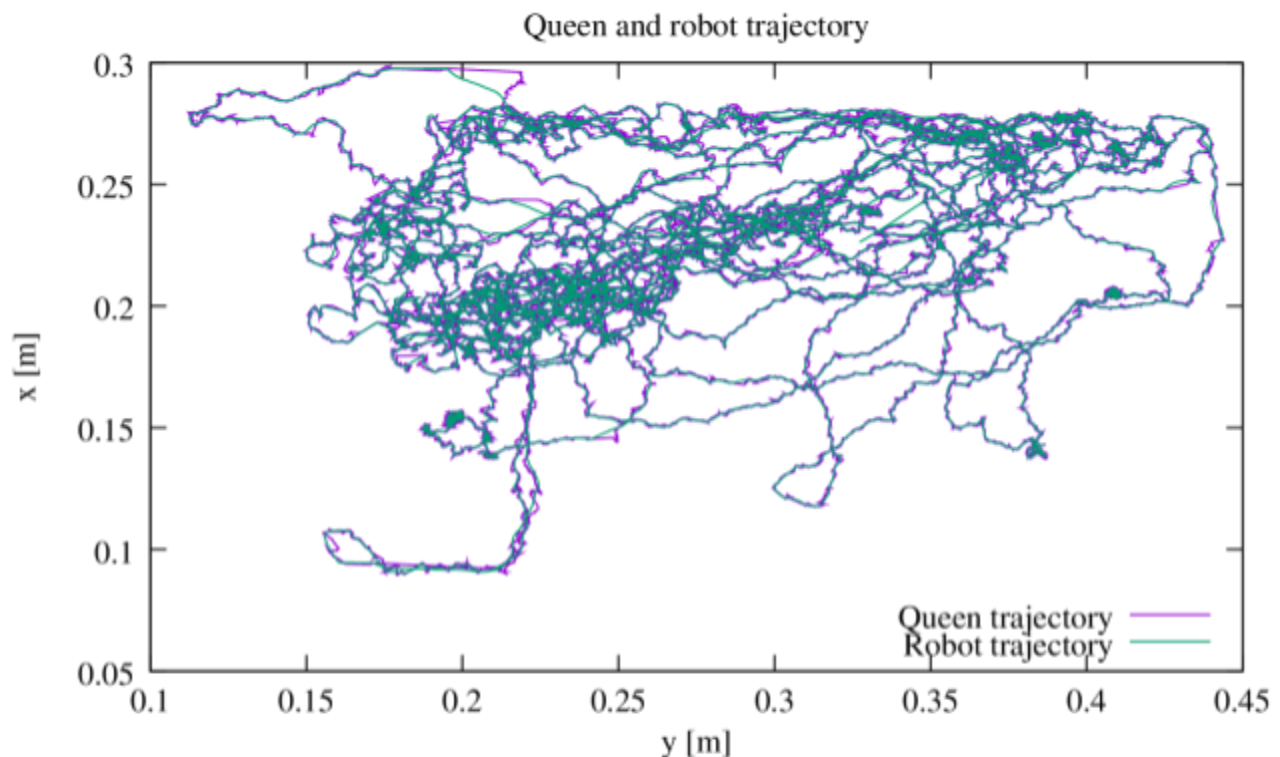


Figure A: Queen and robot trajectories over a 5 hour period as retrieved from the data provided by the queen position recording module. The data are produced by the high-level control module, which processes the XY actuator driver messages and the output of the WhyCode pattern detection module.

The evolution of the robot and queen position over time displayed on Figure A is also available as a video in a dedicated [playlist](#), where it includes the actual imagery.

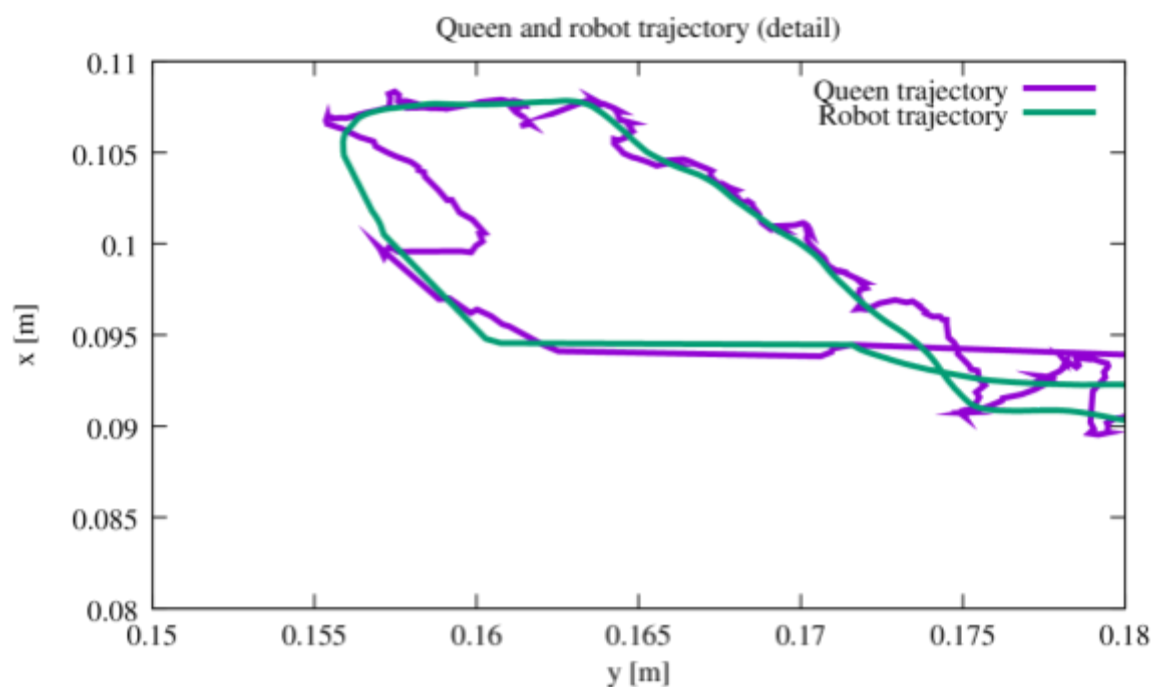


Figure B: Detail of the queen and robot trajectory from the bottom left of Figure A. Note that the robot trajectory is smoothed to minimise vibrations and disturbance to the hive.

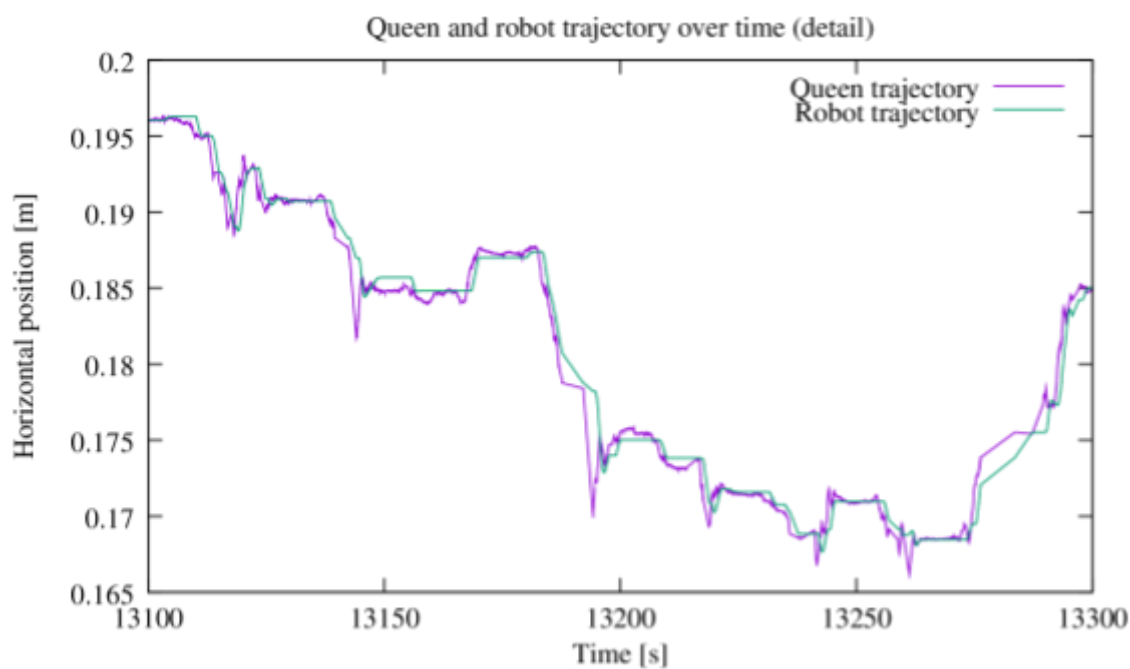


Figure C: The queen and robot horizontal position detail over time. The displayed position evolution corresponds to the detail provided in Figure B.