Hardware & software interface for robot programming by manual guidance

The module for robot programming by manual guidance enables fast and intuitive programming of new robot tasks without using traditional robot programming systems. Traditional robot programming approaches require the user to guide and program the robot with its controlling interface. By handling either a touchscreen or a joystick, the user moves the robot to a desired position and defines the type of motion to that configuration. This method turns out to be relatively slow and requires a certain proficiency in robot handling. Our programming by demonstration modules is based on manual guidance (i.e. kinesthetic teaching), where the operator physically guides the robot through the desired robot configurations and movements in order to program a sequence of robot movements that lead to a successful task execution. Kinesthetic teaching can be used for the programming of point-to-point movements or whole movements (trajectories). The robot has to be equipped with necessary sensing equipment which enables manual guidance, i.e. it should enable control in gravity compensation mode. Joint torque sensors or 6-D force-torque sensor mounted at the end-effector are also be beneficial for kinesthetic guidance.

A Graphical User Interface (GUI) is provided that facilitates this skill acquisition process. In the developed system, skills are saved in a MongoDB database available in ROS. A general overview of all main components of this module is shown in Fig. 1. The module uses ROS (Robot Operating System), which provides the database MongoDB, high level communication, and other functionalities. While many features and tools are included in ROS, a subset of them is used to realize manual guidance within the cell: ROS nodes, MongoDB database, topics, services, action and parameters servers, messages, etc.

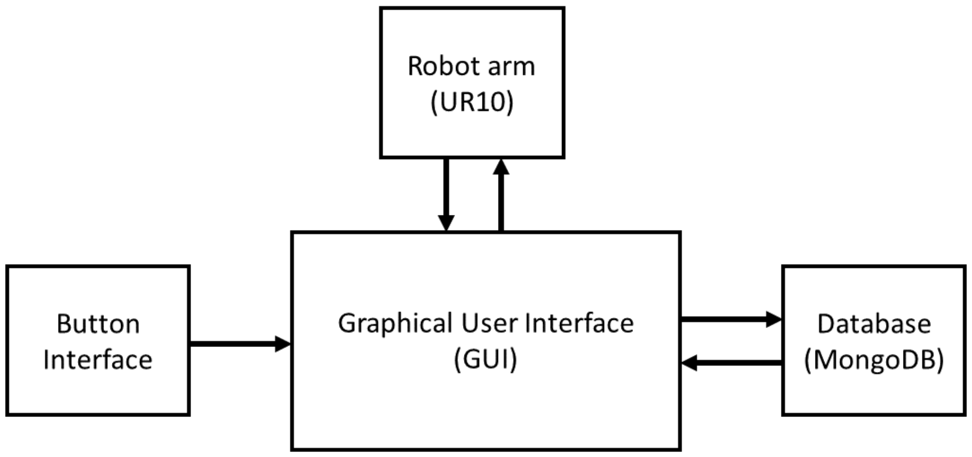


Figure 1: Overview of main module components. The GUI is at the core of the module. It receives triggers from the button interface, which are used to change the control mode of the robot, e.g. put the robot into gravity compensation mode, and initiate saving and reading the data from the database (MongoDB-based). The button interface is connected to the GUI via the robot's I/Os.

A **button interface** can be mounted on the robot arm. It eases the acquisition of data from manual guidance, as it lowers the effort and shortens the time needed for teaching. The cover houses 2 buttons and 2 switches which are used to trigger various functions that support manual guidance. An example cover, suitable for Universal robot UR10, can be seen in Fig. 2. The default settings for this module maps the switch 1 to the gravity compensation control, switch 2 to the tool exchange system, while both buttons are used to trigger save events. The tool exchange system is an optional hardware module which enables automatical exchange of the robot's end-effectors. As the buttons are connected through the robots's I/Os, their states can be read by observing the states on the robot. This is done through the ROS framework.

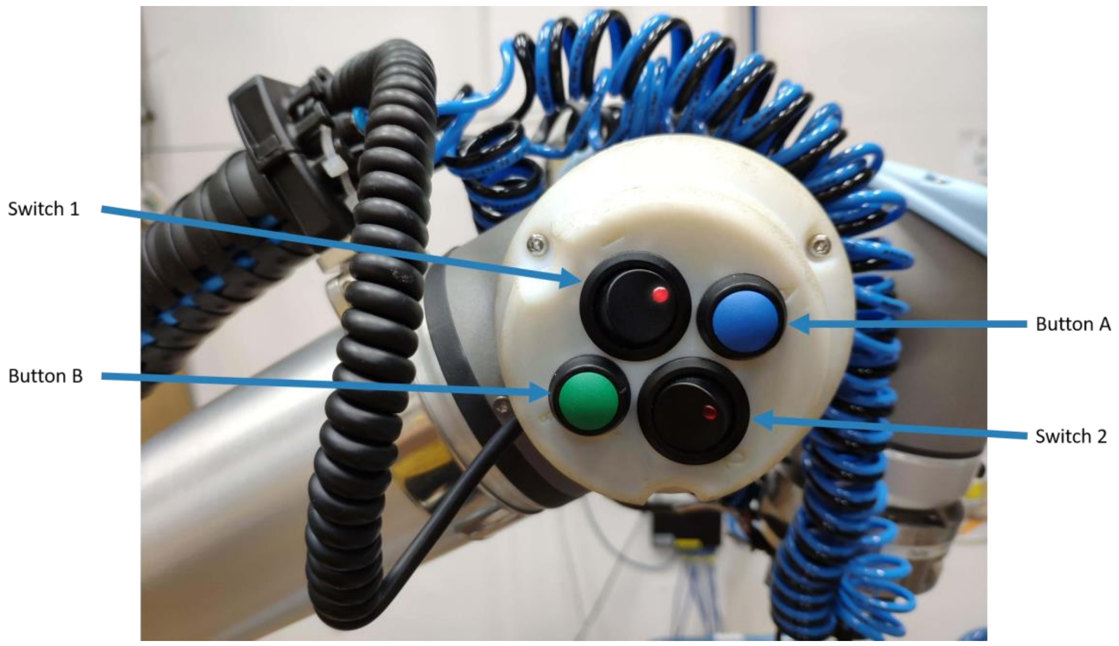
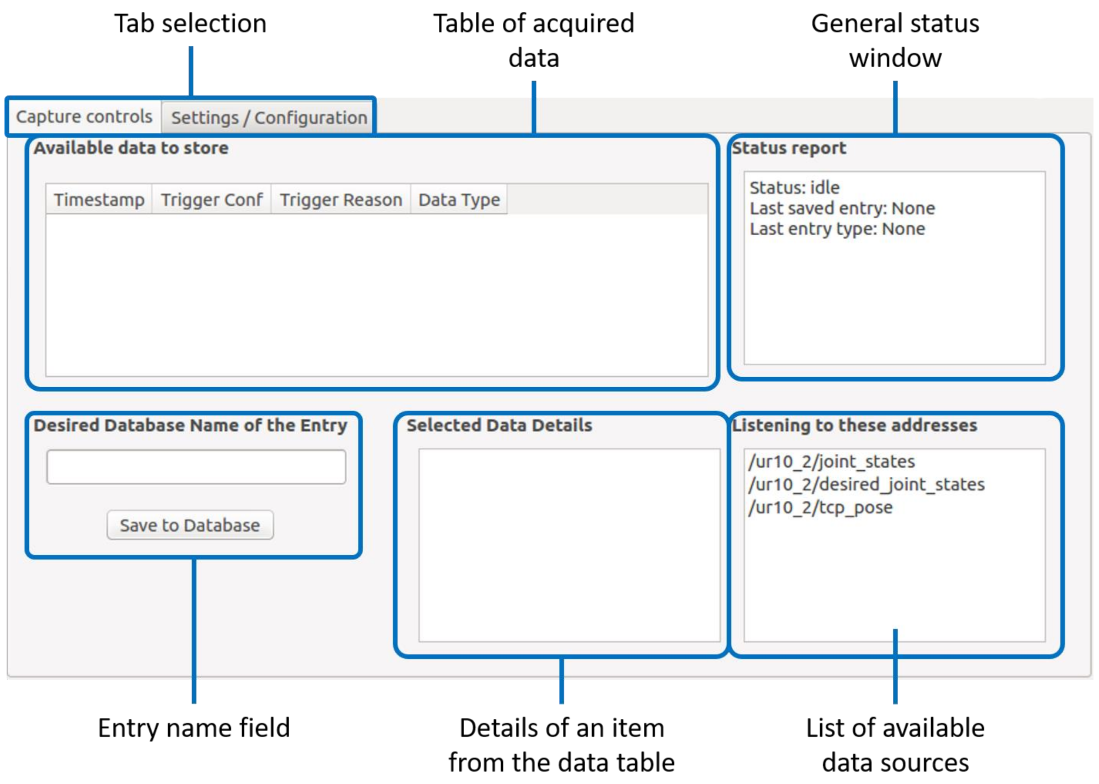


Figure 2: The button interface used for manual guidance. The button interface replaces one of the standard UR-10 joint covers. The button states are connected through the robot controller I/Os.

The **database** component is at the back-end of the module. The data acquired by the manual guidance module should be accessible throughout the entire workcell software framework, which is based on ROS. For this reason the mongodb_store ROS package is integrated, as it allows all ROS nodes in the network to access the database. The MongoDB database usually runs on the same computer that runs the ROS core. After the data for robot programming have been acquired by manual guidance module and stored in the database, the required motions to execute the task can be computed.

The **graphical user interface**, called The Helping Hand, is at the center of the manual guidance module. The GUI has two tabs: the main tab called *Capture controls* and the secondary one called *Settings/Configuration*. The main tab of the GUI can be seen in Fig. 3. Through this tab, all the software components are called and triggered. There are two types of skill acquisition: Single Point Acquisition and Whole Trajectory Acquisition. If a single point is stored in the database it can later be used for generating point-to-point movements. For the acquisition of whole robot movements, a dense or even continuous sequence of robot points and corresponding times is stored. It can be used to record Cartesian space trajectory or a joint space trajectory. The trajectory is post processed to an extent, as is being encoded as a Dynamic Movement Primitive (DMPs).



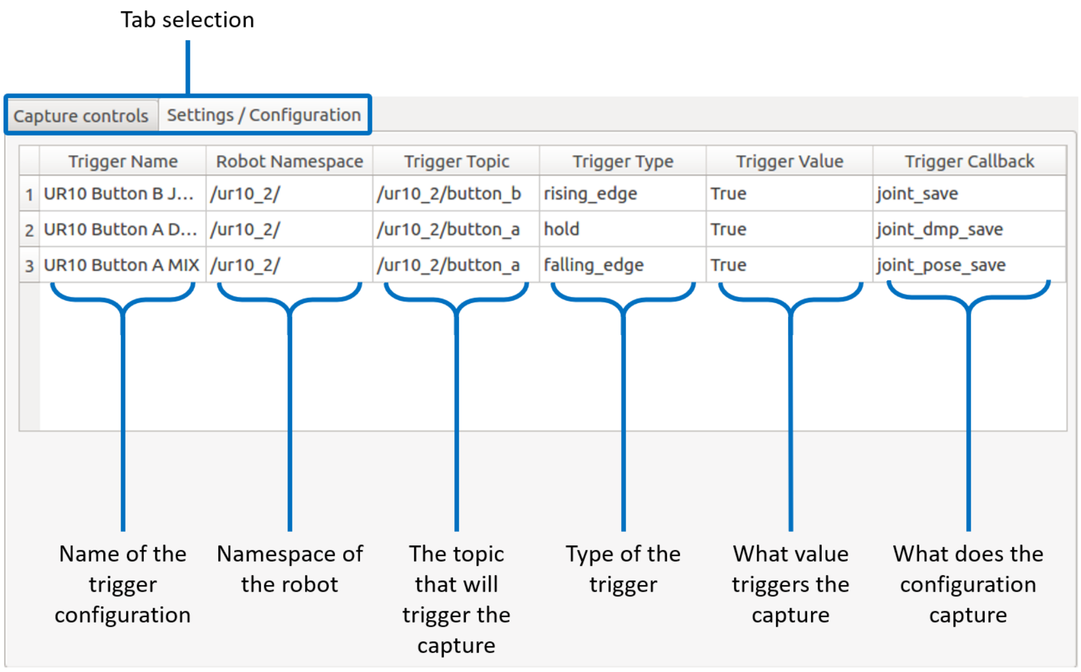
Status: idle
Last saved entry: None
Last entry type: None

/ur10_2/joint_states
/ur10_2/desired_joint_states
/ur10_2/tcp_pose

Figure 3: The Helping Hand's main tab called Capture controls.

As mentioned, this module uses the button interface to trigger save events by default. Nonetheless, to allow the user to have custom signals that trigger custom save events, configuration file is used. The secondary tab of the GUI can be seen in Fig. 4 and is used to check the configuration of triggers defined in the YAML configuration file. For the default UR10 module option, no changes to the configuration file should be necessary. If changes are needed, e.g. different topics are used to listen to buttons on the robots, the YAML configuration file needs to be modified. Besides UR robots, we also implemented the interface for Franka Emika Panda robots.

In order to use the GUI, it needs to be installed with required dependencies. To do that, a package called Helping Hand needs to be installed. This package contains various tools to facilitate programming robots by manual guidance. However, the main tool is the provided GUI. Installation instructions and the open-source package (under a three-clause BSD license) for this module can be found on the repository.


Tab selection
Capture controls
Settings / Configuration
Name of the trigger configuration
Namespace of the robot
The topic that will trigger the capture
Type of the trigger
What value triggers the capture
What does the configuration capture

	Trigger Name	Robot Namespace	Trigger Topic	Trigger Type	Trigger Value	Trigger Callback
1	UR10 Button B J…	/ur10_2/	/ur10_2/button_b	rising_edge	True	joint_save
2	UR10 Button A D…	/ur10_2/	/ur10_2/button_a	hold	True	joint_dmp_save
3	UR10 Button A MIX	/ur10_2/	/ur10_2/button_a	falling_edge	True	joint_pose_save

Figure 4: The Helping Hand's configuration tab. Default settings are displayed in this example.