Univerza v Ljubljani
Fakulteta za strojništvo

Ian Hafner

Erasmus+ TET project

Exploring capabilities of the Hiwonder JetMax robotic arm for detecting and moving objects

Škofja Loka, 2026

1. Introduction

Robotic arms are an important part of modern mechatronic systems, as they enable the automated detection, movement, and sorting of objects. Their operation is based on the integration of mechanical, electronic, and software components, such as actuators, sensors, a camera, a control unit, and software.

This seminar paper presents practical work with the Hiwonder JetMax robotic arm. Its structure, the hardware and software used, and the development of programs for detecting and moving objects are discussed. Particular emphasis is placed on the use of computer vision, the ROS system, the robotic gripper, and the Python programming language.

The procedures for establishing access to the robotic system, analyzing an existing ROS project, creating a custom package, and developing programs for detecting AprilTag markers, picking up objects, and assembling a tower are also presented. The problems that arose during the work and the methods used to resolve them are described as well. The purpose of the paper is to demonstrate the practical possibilities and limitations of the JetMax robotic arm when performing robotic tasks.

2. Robot Hiwonder JetMax

Slika 2.1: Robot JetMax

2.1 Basic Description

Hiwonder JetMax is an educational robotic manipulator intended for learning about robotics, automation, computer vision, and artificial intelligence. It was developed by the company Hiwonder, and the robot is based on an NVIDIA Jetson Nano computer. Due to its open-source design and support for the ROS operating system, it is particularly suitable for students, researchers, and developers who wish to test various robotic applications.

2.2 Robot Structure

JetMax consists of a robotic arm, a gripper, control electronics, and a camera. The camera is mounted on the end of the arm, allowing the robot to detect objects within its working space. The robotic arm enables the gripper to be moved into different positions, while the gripper allows the robot to pick up and move lighter objects. The structure is primarily intended for educational and demonstration projects, rather than demanding industrial tasks.

2.3 Programming and Control

An important part of the system is the use of the Python programming language. With it, the user can prepare programs for motion control, image processing, and communication with robotic components. JetMax also supports ROS, or Robot Operating System. ROS enables a program to be divided into multiple nodes that exchange data with one another. For example, one node can process an image from the camera, another can calculate the position of an object, and a third can control the motors of the robotic arm.

2.4 Computer Vision

The JetMax camera enables the performance of computer vision tasks. The OpenCV library can be used for image processing, enabling the recognition of colors, shapes, faces, and various gestures. Based on this, it is possible to develop an application in which the robot automatically searches for an object, determines its position, and moves it to another location using the gripper. Such a system combines environmental perception, data processing, and the mechanical execution of a task.

2.5 Kinematics of the Robotic Arm

Inverse kinematics is particularly important in learning about robotics. This is a process in which the angles of the individual joints of the robotic arm are calculated from the desired position and orientation of the gripper. In this way, the robot can reach a specific point in space. JetMax is therefore useful for studying forward and inverse kinematics, trajectory planning, and the control of robotic systems.

2.6 Advantages and Limitations

The advantages of the JetMax robot include its flexibility and the extensive amount of educational material available. The user can carry out projects such as sorting objects by color, object tracking, facial recognition, and automated object picking. Its limitation, however, is that it is not intended for industrial use, heavy loads, or very high precision.

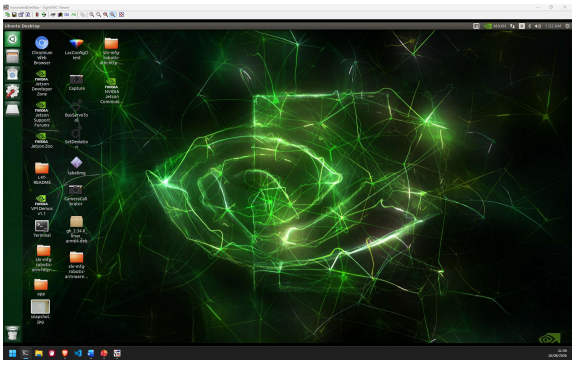
2.7 Conclusion

JetMax represents a useful development platform, as it combines the mechanical, electronic, and software aspects of robotics in a single system. It enables students to gain practical knowledge of robotic control, computer vision, artificial intelligence, and the ROS system. It is therefore particularly suitable for seminar, laboratory, and research projects.

3. Software Used

3.1 Software on the Robot

3.1.1 Operating System



Slika 3.1: Ubuntu 18.04 LTS (NVIDIA verzija)

The Hiwonder JetMax robot uses the Ubuntu 18.04 LTS operating system, which is based on Linux and installed on an NVIDIA Jetson Nano computer. The operating system is part of the NVIDIA JetPack software package, which includes the necessary drivers and libraries for using the graphics processor.

3.1.2 Robot Operating System

Robot Operating System (ROS) is a software framework intended for the development of robotic systems. It does not represent a conventional operating system, but rather provides standardized mechanisms for communication between various software and hardware components of a robot. Its use enables modular development, as individual functions, such as image acquisition, data processing, motion planning, and motor control, can be developed separately and then connected into a complete system.

The basic building block of ROS is a node (angl. node), which performs a specific task. Nodes exchange data through topics (angl. topics), through which they send and receive structured messages (angl. messages). Functionalities and the corresponding files are organized into packages (angl. packages), which facilitates the reuse of software and the organization of the project.

In a project involving the Hiwonder JetMax robotic arm, ROS could connect the program for controlling the arm with the camera or other sensors, as well as with actuators that perform movements of individual joints. For example, the camera could provide data to an object detection node, which would then transmit the results via messages to the motion planning and control node. Whether

ROS was actually used in the specific implementation of the Hiwonder JetMax robotic arm must be supplemented according to the software used and the architecture of the project.

3.1.3 Programming Languages and Development Environment

The robot is programmed primarily using Python and C++. Python is suitable for developing programs for computer vision, artificial intelligence, and simple control, whereas C++ is often used for more demanding and time-sensitive robotic tasks. The combination of these programming tools enables the development of applications for computer vision, robotic arm control, and the automation of various tasks.

3.2 Accessing the Robot via the VNC System

The VNC (Virtual Network Computing) system was used for remote control of the Hiwonder JetMax robot. It enables access to the graphical desktop of the NVIDIA Jetson Nano computer built into the robot from another computer. In this way, the user can remotely monitor the robot's desktop and manage its programs, settings, and control without being directly connected to it. The connection is established through a local network, whereby a VNC server must be installed and running on the robot, while a VNC client must be installed on the user's computer. This method of access simplifies development, testing, and troubleshooting during the operation of the robot.

3.3 Software Environment

I used Visual Studio Code (VS Code) to write and edit the program code. It is a text editor that enables the clear writing, editing, and organization of program files. I used it primarily to develop and edit programs for controlling the Hiwonder JetMax robotic arm.

VS Code enables the use of extensions that facilitate work with the selected programming language, such as syntax highlighting, automatic code completion, and clearer project organization. The programs were divided into individual files or sections according to their function, which enabled easier testing and troubleshooting. A large part of the practical work was carried out in this environment, as I wrote, modified, and checked the program code in it.

3.4 File Transfer and Secure Shell

When working with the Hiwonder JetMax robot, I used the WinSCP program for secure file transfer. The program uses the SFTP protocol, which is based on the SSH (Secure Shell) protocol and enables an encrypted connection between the computer and the robot. Through the graphical interface of the WinSCP program, it was possible to transfer Python programs, configuration files, and other necessary files to the robot and, when necessary, transfer them back to the computer. Establishing the connection required the robot's address, username, password, and SSH port. WinSCP thus simplified file management on the robot without using the command line.

The SSH (Secure Shell) protocol enables a secure remote connection to the Hiwonder JetMax robot over a network. During the connection, the user logs in with a username and password, while communication between the computer and the robot is encrypted. SSH enables the remote execution of commands, file management, and the launching of programs on the robot. In combination with the WinSCP program, it is used primarily for secure file transfer via the SFTP protocol, which is based on SSH. Both the login credentials and the content of the transferred files are protected in this process.

3.5 AI Tools

The Kilo Code tool, which is based on artificial intelligence, was used when creating program code for the Hiwonder JetMax robot. The tool functions as a programming assistant and enables the generation, completion, and explanation of program code. It was particularly useful when preparing commands for controlling the robotic arm, moving the robot, and performing individual tasks.

Kilo Code also helped with finding and correcting errors in the program. Based on a description of the problem, it can suggest code changes, improve its clarity, and explain the operation of individual commands and functions. This made it easier to understand the connection between the written program and the operation of the Hiwonder JetMax robot.

The tool was used as a supporting aid, not as a complete replacement for one's own programming. The suggested code had to be checked, adapted to the software used, and tested on the robot. In this way, it was possible to determine whether the robot responded correctly and whether the program fulfilled the intended tasks.

4. Establishing Access to the JetMax Robotic System and Analyzing the ROS Project

4.1 Establishing Remote Access

To improve access to the graphical interface, I searched for a more suitable solution on the internet. I discovered that a VNC interface was already installed on the system, but it had not been enabled. I therefore activated it and then checked the operation of the remote access.

The VNC interface was successfully enabled, and remote access to the graphical interface functioned without any major difficulties. This allowed me to access the JetMax system through another computer, which simplified further work with the robotic system.

I then wanted to establish a connection between the JetMax robotic system and a personal computer using Visual Studio Code as well. For this purpose, I used the Remote Host option, which enables remote editing of files and execution of programs on another computer. The connection could not be established successfully because of a mismatch between the version of the standard C library in the robot's operating system and the environment used by Visual Studio Code for the connection.

After that, I tried using another method of connecting to the JetMax robotic system. I did this using the SCP, or Secure Copy, system. After establishing the connection, I was able to access the files stored on the robot. I could view and manage the files from the personal computer.

4.2 Checking the Operation of the Robotic System

After establishing access to the files, I ran several demonstration programs prepared by the manufacturer of the robotic arm. The programs started successfully and operated correctly, confirming the operation of the robotic system and the correctly established connection to it.

4.3 Transferring and Reviewing the ROS Project

After successfully testing the robotic system, I saved the entire ROS project to a USB flash drive. The purpose of saving it was to transfer the project to another computer, where I could review and analyze

it in greater detail.

First, I reviewed the files in the ROS project, read the program code, and tried to determine the purpose of the individual files. I encountered difficulties because of my lack of experience with the ROS system, as well as because the program code was unclear and insufficiently commented. Due to these difficulties, I obtained too little information about the content and operation of the project during the initial review.

To facilitate understanding of the project, I used the Kilo Code extension in Visual Studio Code. The extension read the entire ROS project and analyzed its program code. Based on the analysis, it created a README.md file describing the programs included in the ROS project and explaining their content and purpose. The file helped me better understand the structure of the project and the operation of its individual program components.

4.4 Creating a Custom ROS Package

After reviewing the existing project, I created a new package named hafi on the robotic arm within the ROS system, as this is a username that I frequently use.

The hafi ROS package was successfully created on the robotic arm. I then began saving my own programs in it. I added the programs by creating new files containing program code within the package. In this way, I prepared the basic structure for developing custom solutions that could later be run on the robotic arm.

5. Programs

This chapter presents the programs developed and used during the practical work with the Hiwonder JetMax robotic arm. Their structure, purpose, and method of operation in determining the camera's position, detecting AprilTags, controlling the robotic arm, and automatically moving objects are described. The programs were developed in the Python programming language and connected to the ROS framework, which enabled communication between the camera, robotic arm, gripper, and graphical interface.

5.1 `cam_pos.py`

`cam_pos.py` is a ROS node (`cam_pos_estimator`) for the JetMax system. Its primary task is to calculate the camera's position and orientation from the robot's tool coordinates.

The node subscribes to the `/jetmax/status` topic, which provides the current coordinates (x, y, z) and the robot's rotation angle (*yaw*). When it receives a message, it performs a coordinate transformation using the `tool_to_camera()` function: it shifts the coordinates from the tool frame to the camera frame using fixed offsets (`CAMERA_OFFSET_A`, `CAMERA_OFFSET_Z`). The transformation uses cylindrical coordinates (`cartesian_to_cylindrical`) to shift the radial component, after which it converts the coordinates back to Cartesian coordinates.

The orientation (*yaw*) is converted into a quaternion representation (`yaw_to_quaternion`) for publication in the `PoseStamped` message. The result – the camera position in meters (`/camera_pose`) and the rotation angle in degrees (`/camera_yaw`) – is published on the corresponding ROS topics, enabling other nodes to track the camera's position in real time.

5.2 center_tag.py

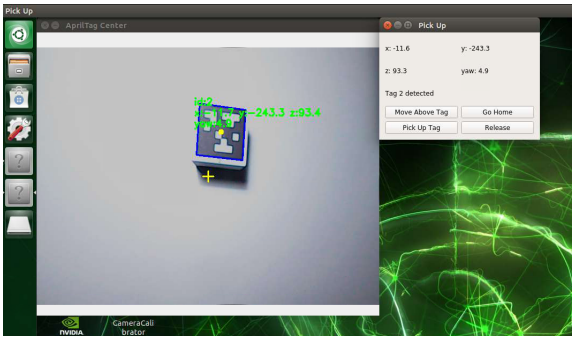
The program center_tag.py is a ROS node that detects an AprilTag (tag family tag36h11) in camera images in real time and determines its position and rotation in the world coordinate system.

The node subscribes to camera data and, using the calibration parameters (matrix K, rotation R, and translation T), calculates the position of the tag in the world (in millimeters) and publishes it on the /center_tag/detections topic in JSON format. At the same time, it visualizes the detected tag using OpenCV (outline, center, and coordinate data).

For the transformation from camera coordinates to world coordinates, it also uses data on the camera's position and rotation, received via the /camera_pose and /camera_yaw topics and through the cam_pos.py process. It also includes a position correction (POSITION_CORRECTION) and a shutdown mechanism for the subprocess.

5.3 pick_up.py

pick_up.py is a ROS node for controlling the JetMax robotic arm, enabling the automated picking up of objects using a vacuum suction cup. The program operates through two main components:



Slika 5.1: GUI and OpenCV

1. Marker detection – starts center_tag.py as a subprocess, which determines the position of the object using an ArUco marker (coordinates x, y, z, and *yaw*). The data is received via a ROS topic and stored in a thread-safe structure (threading.Lock, threading.Event), enabling simultaneous communication with the GUI.

2. Graphical interface (PyQt5) – displays the marker's current coordinates to the user and provides buttons for:

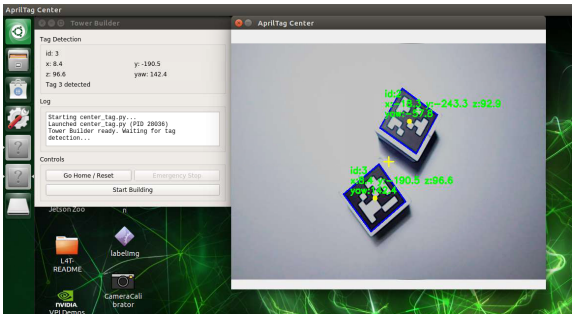
   - moving the arm above the marker (Move Above Tag),
   - returning to the initial position (Go Home),
   - picking up the marker with the vacuum suction cup (Pick Up Tag) – approaching from above, lowering to the configured height pick_up_z (default 95 mm), activating the suction cup, and lifting,
   - releasing the object (Release).

Communication with the robot takes place via ROS publishers (/jetmax/command, /jetmax/end_effector/sucker/command) and a service proxy (/jetmax/go_home). The program stops if it does not detect a marker within a specified time, and it properly terminates the subprocess upon exit. The configurable pickup height (pick_up_z) enables adaptation to different scenarios.

5.4 `build_tower.py`

The program file `build_tower.py` represents a solution for autonomously assembling a tower from two square cubes using the JetMax robotic system. The program is based on the ROS (Robot Operating System) framework and is connected to a graphical user interface developed using the PyQt5 library. The purpose of the system is to automatically detect, pick up, and place the cubes at a specified location in 3D space. For this purpose, it uses visual recognition of AprilTags.

The system consists of three main components. The first component is the graphical user interface, which enables real-time monitoring of the positions of detected tags, reviewing the event log, and controlling the robot. The user can move the robot to its home position, perform an emergency stop, or start the tower assembly procedure.



Slika 5.2: GUI and OpenCV

The second component is a ROS node responsible for communication between the program and the robot. It publishes commands to the `/jetmax/command`, `/jetmax/end_effector/sucker/command`, and `/jetmax/end_effector/servo1/command` topics. The first topic is used to move the robot, the second to control the vacuum gripper, and the third to adjust the servo motor. The node also uses the `/jetmax/go_home` service and subscribes to the `/center_tag/detections` topic, through which it receives the detection results.

The third component is the `center_tag.py` subprocess, which captures images from the camera, detects AprilTags, and calculates their 3D coordinates in the camera coordinate system. Since the camera and the robot use different coordinate systems, the `camera_to_tool` function converts the coordinates into the robot coordinate system. In doing so, it takes into account the offsets between the camera and the end of the tool, defined by the parameters A, Z, and C.

The procedure begins by checking for the presence of exactly two tags. The tags are sorted according to the y coordinate, after which the robot picks up the lower cube and places it at the target location. After rescanning, it checks its position, adjusts the angle of the `servo1` motor, and places the second cube on top of the lower cube. The program also includes an emergency stop, checking of the `building_running` flag, and the proper termination of the subprocess when the application is closed.

6. Conclusion

In this seminar paper, I presented practical work with the Hiwonder JetMax robotic arm and investigated its capabilities for detecting and moving objects. First, I familiarized myself with the structure of the robotic system, its software, and the method of communication between the individual components.

I established remote access to the robot, examined the existing ROS project, and created my own package for program development.

The developed programs enabled the determination of the camera position, the detection of AprilTag markers, the control of the robotic arm and vacuum gripper, as well as the automatic picking up and moving of objects. By using the ROS system, the Python programming language, and computer vision libraries, it was possible to connect object detection with the appropriate movement of the robotic arm. The successful operation of the demonstration and self-developed programs confirmed that the system functions correctly and is suitable for performing basic robotic tasks.

During the work, I also encountered various difficulties, particularly in establishing the connection, understanding the ROS project, determining coordinates, and ensuring the precise movement of the robotic arm. Through error analysis, program adjustments, and practical testing, I gradually resolved these difficulties.

I found that the Hiwonder JetMax is a useful educational platform for learning about robotics, computer vision, and the programming of robotic systems. Although, due to its structural limitations, it is not intended for demanding industrial tasks, it enables the development and testing of various applications. The project provided me with a better understanding of the connection between environmental perception, data processing, and the physical execution of tasks using a robotic arm.

7. Sources

Literatura

[1] Hiwonder. *JetMax: JETSON NANO Robot Arm ROS Open Source Vision Recognition*. Available at: `https://www.hiwonder.com/products/jetmax`

[2] Hiwonder. *JetMax v1.0 Documentation – Advanced Program Lessons*. Available at: `https://wiki.hiwonder.com/projects/JetMax/en/latest/docs/6_Advanced_Program_Lessons.html`

[3] Open Robotics. *ROS Documentation*. Available at: `https://docs.ros.org/en/humble/index.html`

[4] Hiwonder. *JetMax Tutorial Video*. YouTube. Available at: `https://www.youtube.com/watch?v=C6BlNbeU3fQ\&list=PLAjUtIp46jDcQb-MgFLpGqskm9iB5xfoP`

[5] YouTube. *Video about Computer Vision and Image Processing*. Available at: `https://www.youtube.com/watch?v=oXlwWbU8l2o`