UTBots@Home 2022 - Apollo (based on Pionner 3-AT) Service Robot - UTFPR

Piatan S. Palar[1], Marlon O. Vaz[2], Bruno Bock Silva[3], Gustavo F. Lewin[3], Gustavo F. Armênio[4], Hermann J. R. Reuter[5], Felipe P. Conter[6], André S. de Oliveira[1] and João A. Fabro[7]

***Abstract*— This TDP describes Apollo, a service robot under development at UTFPR (Federal University of Technology - Paraná - Brazil). This robot is being developed since 2013, and participated in the Brazilian an Latin American Robotics Competition (CBR/LARC) since 2014. The team participated in every Brazilian and Latin American competitions since, both in person (2013 2019, and again in 2022) and remotely (2020 and 2021). The team obtained the third place in both 2014 and 2022 Latin American competitions, and for the first time is applying for participation in the world RoboCup competition. The robot is based on the Pioneer 3-AT mobile base. In order to execute all the tasks proposed by the Robocup@Home initiative. "Apollo" has sensors, such as a LIDAR laser scanner, a Kinect depth sensor, and a camera, that provide information necessary for navigation, object and people recognition, environment mapping and auto-localization. An innovative human-robot interaction interface, using different facial expressions based on simulated emotions, is also presented. The robot is capable of executing the following Robocup@Home tasks: voice synthesis and recognition, objects and people recognition, navigation, and small objects manipulation. All the programming uses ROS (Robot Operating System) Noetic, executing on a NUC onboard computer, connected to two Arduino microcontrollers. The team achieved a 3rd place in the 2022 Brazilian and Latin American competition, among 9 competing teams, and thus is for the first time applying for participation in the world Robocup.**

I. INTRODUCTION

Robocup@Home is a competition that aims to foster the development of robots that will help people in domestic environment, being the biggest competition for service robots on the planet [1] [2]. In order to perform household tasks, the robotics fields of human-robot interaction, navigation, map construction on dynamic environments and computer vision are developed, among many others [1].

To accomplish these tasks, a research robot is under continuous development by team UTBots@Home at UTFPR[1]: "Apollo", presented in Figure 1. This robot is composed of a mobile base (Pioneer 3-AT), a manipulator with 4 degrees of freedom, a LiDAR range finder, a screen to present the "face" of the robot, speakers for verbal communication, and a depth/RGB camera.

[1]HomePage of the Team: `https://laser.dainf.ct.utfpr.edu.br/doku.php?id=utbots_at_home`

Fig. 1. "Apollo", composed of a Pioneer 3-AT mobile base, with its sensors (LiDAR, Kinect), the LCD display for simulated emotions, and the manipulator.

[1] André S. de Oliveira is a professor with CPGEI - Graduate Program of Electrical Engineering and Industrial Informatics, UTFPR - Federal University of Technology - Parana, Brazil - Campus Curitiba. Piatan S. Palar is a PhD student within this Graduate Program. `andreoliveira@utfpr.edu.br`, `piatan@alunos.utfpr.edu.br`

[2] Marlon O. Vaz is a professor with IFPR - Federal Institute of Parana, Campus Pinhais, Pinhais. He is also a PhD. student at CPGEI. `marlon.vaz@ifpr.edu.br`

[3] Bruno Bock Silva and Gustavo F. Lewin are students of Electronics Engineering at UTFPR. `{brunobock,lewin}@alunos.utfpr.edu.br`

[4] Gustavo F. Armênio is a student of Computer Engineering at UTFPR. `gustavofardoarmenio@alunos.utfpr.edu.br`

[5] Hermann J. R. Reuter is a student of Mechatronics Engineering at UTFPR. `hermannreuter@alunos.utfpr.edu.br`

[6] Felipe P. Conter is with UNILA-Latin American Integration Federal University - Foz do Iguaçu - Paraná - Brazil. `felipeconter.cc@gmail.com`

[7] João A. Fabro is a professor with the Graduate Program on Applied Computing - PPGCA, Informatics Department - DAINF - at UTFPR. He is also the leader of the Laboratory of Embedded Systems and Robotics (LASER-http://laser.dainf.ct.utfpr.edu.br) `fabro@utfpr.edu.br`

The applications are running with ROS (Robot Operating System), a set of software libraries and tools that help to develop applications for robotics [3], that run on a Intel NUC [5] embedded computer inside the robot. The control system of the robots is responsible for acquiring and interpreting various sensors and, from that data, make decisions to fulfill the tasks. Experiments are being developed using a NVIDIA Jetson Nano[2] board, that will be included in the robot to process just objects and person recognition tasks.

In this document, the partial results and expectations of projects under development for the robot are presented. Section 2 presents a description of the robot's hardware, including specifications and characteristics. The software already developed, and what is still in development, is described in section 3. Conclusions and future perspectives are presented in section 4.

II. HARDWARE

A. Robot Specifications

The robot is based on a Pioneer 3-AT research robot: this base weights only 12 Kg, with 12 Kg of payload, and maximum speed of 0.7 m/s. Its autonomy is of 2 hours, with 3, 12 volts, standard batteries. Its dimensions are: 50 cm wide, 50 cm deep and 28 cm high. The robot connects to an external embedded computer via a serial-USB interface allowing its connection to a ROS system (Figure 2).

A redesign is being planned, and a Jetson Nano [4] will be coupled to the robot, in order to process computer vision and other GPU-focused processes. The idea is to integrate the Jetson Nano to the ROS network on the embedded NUC computer [5].

B. "Neck" Support

A support built in aluminium (as seen in Figure 1) is the "neck" of the robot. This support is 1.2 meters long, allowing the positioning of the LCD Display and the Kinect sensor to a height comfortable enough to allow interaction with people (1.5 meters). In addition to that, an Android device is also present, needed to capture the voice commands.

C. Sensors

A Kinect V.1 depth sensor has been incorporated, in order to capture images and depth information. These data is used to identify silhouettes of persons and objects, among other functions [6]. In order to establish the communication between the Kinect sensor and ROS the *freenect* package was used. "Apollo" also has a YDLIDAR X4, which is a low cost LIDAR sensor capable of 360 degrees distance measurements [7]. The data generated by the Kinect and the laser sensor are published in ROS topics.

D. Manipulator

The original model of the manipulator used in the robot is a Beckman Coulter ORCA Robotic Arm [8], a planar manipulator that has three rotational joints and can be seen in Figures 1 and 2. To make the structure respond to the desired commands, a process of retrofitting was carried out, where the motors and encoders were connected to Arduino Mega microcontrollers that allow the open programming of the mechanism.

In addition to the three motors that define the degrees of freedom of the structure, each joint is also coupled to a quadrature encoder, that allows for the microcontrollers to independently control each joint is in motion.

The quadrature encoder emits two pulses slightly out of phase. This allows to infer the direction in which the motor is rotating. Arduino microcontrollers use interruptions to sense these pulses and infer the position of each joint.

Joints 1 and 2, respectively related to the manipulator's shoulder and elbow, are controlled by an Arduino that has an H-bridge for each engine. Joint 3, related to the handle, and the gripper are controlled in the same way by another Arduino. Each joint has a PID control system that aims to maintain the angle of the joint in the desired position in opposition to external forces, in addition to allowing smoother movements between one position and another. Both Arduinos are connected to USB ports of the NUC computer,

Fig. 2. Side view of the robot subsystems mounted on top of the Pioneer 3-AT base.

[2]https://developer.nvidia.com/embedded/jetson-nano-developer-kit

via a hub. Through this connection, ROS commands are received.

Customized ROS messages were implemented for sending and receiving robot information. While the microcontrollers receive the messages containing the desired angle in the topic of the respective joint, it sends messages allowing the monitoring of the pose of the structure.

As the encoders do not have memory of the location where the joint stopped and also do not allow monitoring the movements that occur when the structure is off, the robot needs to perform a startup routine every time the system is powered. This makes all the joints assume an initial position and the control is done from these already known angles.

E. Connections

The general connections among the hardware components are detailed in Figures 3 and 4.

III. SOFTWARE

A. ROS - Robotic Operating System

The robot is controlled via a Nuc computer with the "Noetic" version of ROS, on the operating system Ubuntu 20.04 LTS. ROS allows the application of several tools, libraries and conventions that simplify the development of complex and robust tasks for robots [3]. Topics are data buses through which nodes exchange messages, and multiple nodes may subscribe to each topic (receiving its data) or publishing to it (sending data).

B. Emotion Simulation

Emotions are simulated by the robot through faces, displayed according to the present situation, for a better visualization of internal states of the robot, and better interaction between the robot and people (Figure 5).

Emotions were based on Plutchik's wheel of emotions [9]. A LCD screen running the ROS Image Viewer application displays these faces by subscribing to the custom ROS *\face_emotion* topic. The emotions system is based on the custom *ros_display_emotions* package [10], which runs a ROS node that subscribes the *\emotion* topic and publishes a image message to the *\face_emotion* topic.

Figure 6 illustrates an emotion being displayed by the robot through the screen.

C. Simultaneous Localization and Mapping

Odometry errors can lead to uncertainty in navigation. It is important that the robot is able to correct its location based on the feedback from sensors in real time. SLAM (Simultaneous Localization and Mapping) [11] algorithms can achieve this. Currently, the YDLIDAR readings and the robot's wheels odometry are feed to the standard ROS navigation stack and *move_base* is utilized to send navigation tasks to the robot. By constructing a map of the environment at the same time as it is updating the robot's position, the robot estimates it's position and updates wheel velocity. To accomplish this, SLAM has a number of tasks: extraction of reference points, data association, state estimation, status update, and reference point update. Parameters adjustments and tests under different SLAM configurations are crucial for improving navigation.

D. Voice recognition and synthesis

For speech synthesis, a custom ROS package called *voztts* [12] has been implemented to interface custom nodes with the MaryTTS Text-To-Speech System [13]. This package works by running a ROS node that subscribes to the topic *\tts*, and performs voice synthesis based on the specified parameters. The configuration of the speech parameters encompasses several characteristics, such as speech speed, volume, pitch (lower or higher), the space between words, age and intonation variation. By altering these parameters, the speech characteristics can be defined in accordance with the emotion displayed on its "face". Therefore, the voice synthesis is affected by the *ros_display_emotions* package, which reconfigures the speech parameters according to the current emotion. This process depends on the *dynamic_reconfigure* package, which allows you to change parameters on ROS nodes without having to restart them.

E. Object Recognition

For object recognition, the *darknet_ros* package [15] performs the integration of the YOLOv3 neural network [14] with ROS. Upon training the YOLOv3 model, a variety of objects can be grouped under a certain class - for example, we can label the same class to different instances of apples, with different sizes and colors (green and red apples). The *darknet_ros* package works by subscribing to the Kinect RBG image topic, so it can perform real-time detection, publishing bounding-box and class messages.

The classes detected and the accuracy of the detection depends on the number and quality of labeled images of the dataset used for training. To improve the dataset quality, an external apparatus that allows capturing of a large number of images of each object has been developed [16]. This system has two Logitech C920 cameras (stereovision) and an ASUS Xtion depth sensor, which allows the capture of 2,600 RGB images by the camera and 2,600 point cloud images, each one viewed by a slightly different angle and distance. Therefore, it is possible to generate a large dataset of every object, as well as a general 3D format of it.

For manipulation tasks, it is essential to estimate the object 3D position as well. The applied estimation method is based on the intersection of 3D arrays in the environment's 3D space [17]. Its most significant singularity is that it requires no prior knowledge about the 3D shape of the target object.

By identifying a point in the center of the visible face of the object, and a point on the RGB-D sensor, a 3D array is defined. By moving the robot, other 3D arrays are defined. The method makes use of those 3D arrays to estimate a point that represents the object's 3D center. Experiments have shown that this method performs with a mean euclidean distance of 9.4 cm between the true 3D center of the object and the estimated coordinate.

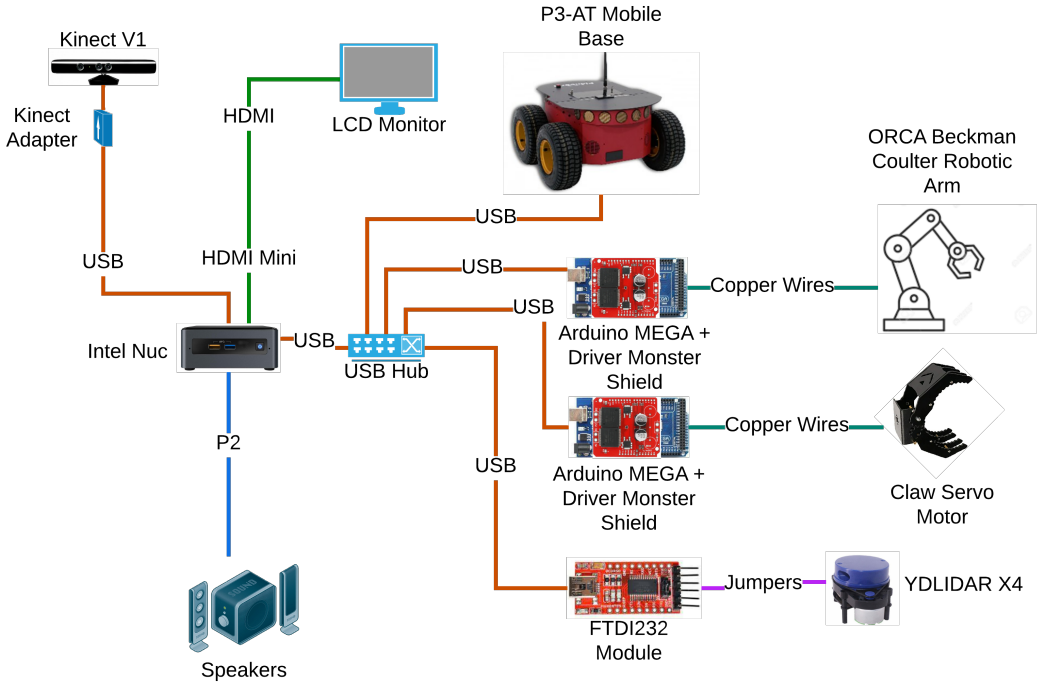

Fig. 3. Diagram for the logical communication connections of the hardware components

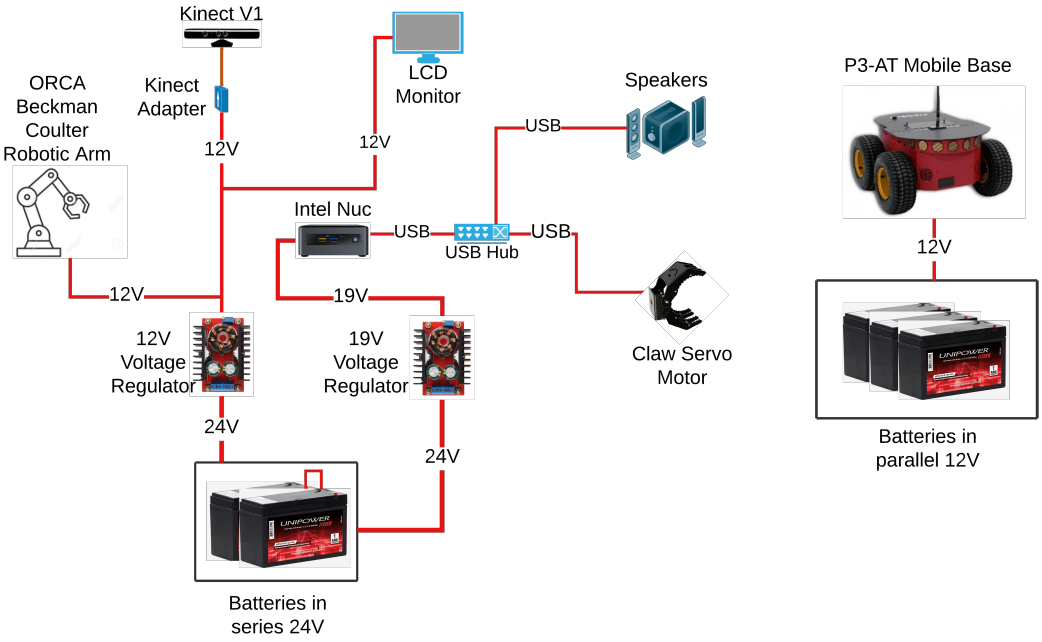

Fig. 4. Diagram for the power connections that feeds every hardware component

Fig. 5. Examples of faces developed.

The intention is to run these processes on the Jetson, placed inside the robot, so the detection and 3D pose calculations are faster.

F. Person Recognition

To perform the person recognition tasks, the RBG layer of the Kinect can be evaluated by YoloV3 through *darknet_ros*, using weights trained to detect people. For some tasks, the custom *mediapipe_track* package implements MediaPipe Pose [18] with ROS messages. The MediaPipe Pose processes body pose information and tracks the general "skeleton". The *mediapipe_track* package [19] performs tracking and 3D position estimates by subscribing Kinect RGB and Depth messages, and can publish the data to be processed by other nodes.

Fig. 6. Simulated emotion presented on the face of the robot.

Some future work involves crowd processing (instances that more than one person can be seen in the frame). Considering that MediaPipe is limited to tracking one person only, tracking a crowd can be done by processing every *darknet_ros* person bounding box. To identify a specific person amongst a crowd, it is possible to extract a person's skeleton information beforehand through MediaPipe Pose and then iterate every person bounding box comparing the parameters. Testing and research is being done so this ideas are improved and can be implemented.

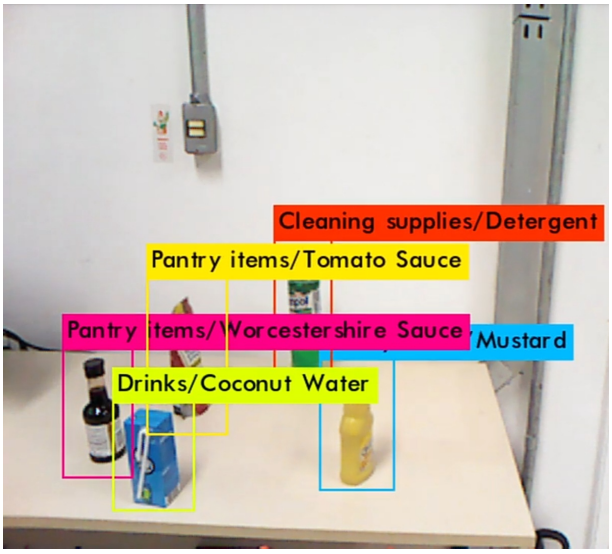

Fig. 7. YOLO v3 model detecting a custom set of objects real-time with ROS.

G. Manipulation

In a new project involving optimization and alternative robotic kinematics techniques, the retrofit process of the manipulator arm was carried out. The manipulator has three rotational joints and a claw-like end effect for manipulating objects.

Considering that the mean error of the position estimation algorithm is too big to allow for a correct pick-up of objects by the claw of the robot, it is planned to fix a Realsense F200 RGB-D sensor (which records distances at a resolution of 480p [21]) near the arm's handle, through a 3D printed support. Thus, this sensor can observe the operation of the claw, and perform new position estimates as the arm moves. The movement creates new challenges related to the position of the camera in the 3D space and perspective issues, that must be considered.

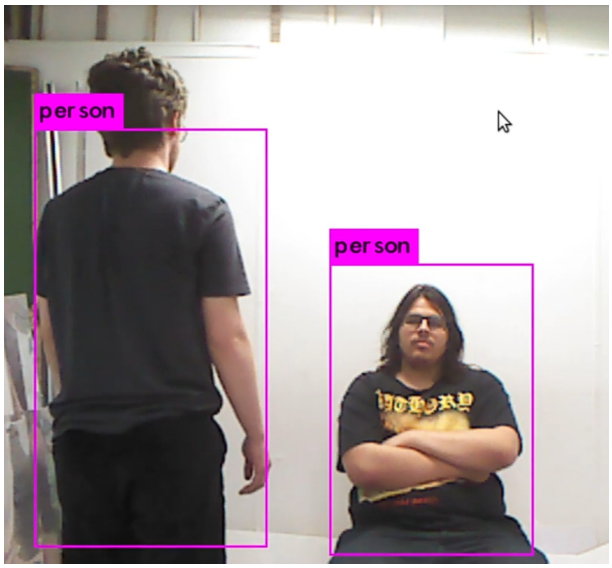

Fig. 8. YOLO v3 model detecting people real-time with ROS.

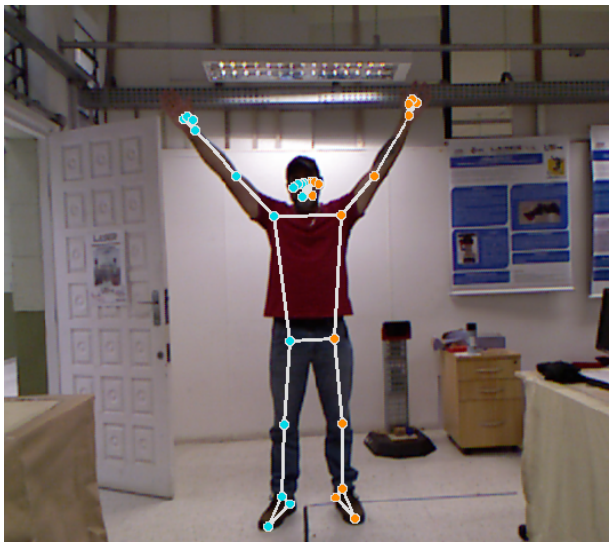

Fig. 9. MediaPipe Pose detecting the person's skeleton with ROS.

The attached RealSense F200 is enough to detect different surfaces in the manipulator's working space. This data will be sent as a PointCloud via ROS network.

The manipulator movements are measured through quadrature encoders, limited by limit switch sensors and are also sent through the ROS network.

Set-point angle commands for each joint are sent through ROS where forward and inverse kinematics calculations are computed using Clifford's Algebra. Such techniques seek to implement more efficient calculation tools using the algebraic advantages of the dual quaternion, which are a subset of Clifford's Algebra. Clifford's Algebra is also used to relate the objects detected by the camera to points and planes that constitute a collision and manipulation control system.

H. Simulation

A scene containing the Apollo robot and a set of obstacles was developed for the CoppeliaSim simulation platform (version 4.2.0) [3], which allows the interfacing of all ROS nodes at an entirely virtual environment. Figure 10 shows the 3D model of the robot.

The robotic arm Beckman Coulter ORCA was replicated in a 3D model, and put together with the Pioneer 3-AT mobile base model already available at the simulator. In this

[3]CoppeliaSim robotics simulator can be download, free of charge for non-commercial applications at http://www.coppeliarobotics.com

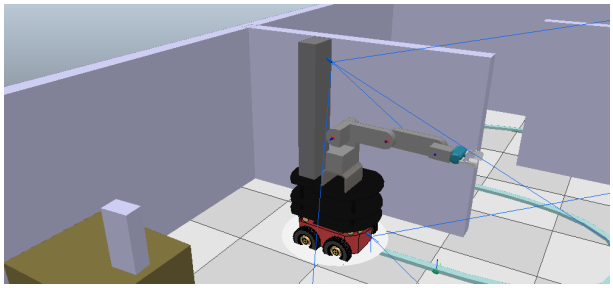

Fig. 10. Apollo's simulation on CoppeliaSim.

case, 3 joint motors were applied (shoulder, elbow and wrist). They receive angular values from ROS topics responsible for the movement of the arm's joints.

CoppeliaSim's built-in model BaxterGripper was used in order to simulate the real gripper. As for the LIDAR sensor, a pre-existing model was used (*hokuyo*). It allows the detection of distances to nearby obstacles.

Aiming to simulate the operator's vision and reproduce the executed processes with the highest precision, a RGB camera was placed following all dimensions used in the physical robot.

IV. CONTRIBUTIONS TO THE COMMUNITY

The main contributions to the Robocup@Home community are listed bellow:

- An innovative HRI (Human-Robot Interface) composed of packages to display a face with "simulated emotions" [10];
- A system that allows for the creation of a dataset of image and depth measures of several objects of interest for Robocup@Home [16];
- A ROS package that allows for the use of the MediaPipe "skeletization" solution with ROS [19];
- A complete simulation environment that allows for experiments with our robot and its sensors and actuators [20].

V. CONCLUSION AND FUTURE WORK

The team is capable of fulfilling several core tasks of the Robocup@Home category successfully. Among the work under development are more accurate recognition of voice commands, more sophisticated person recognition methods and position estimate corrections, as well as improvements in the manipulator arm. Furthermore, correctly integrating the robot's subsystems is an ongoing challenge as its effectiveness, and complexity, grows. Finally, the simulation ambient has been an important addition that enables multiple simultaneous testing without over-stressing the robot.

ACKNOWLEDGMENT

The authors would like to thank the financial support of UTFPR.

REFERENCES

[1] Official homepage: Robocup@Home. Available at: http://www.robocupathome.org/. Accessed June 2022.

[2] IOCCHII, L.; HOLZ, D.;del SOLAR, J.R. ; SUGIURA, K.;van der ZANT, T. RoboCup@Home: Analysis and results of evolving competitions for domestic and service robots. Artificial Intelligence. 229, C (December 2015), 258-281. DOI: https://doi.org/10.1016/j.artint.2015.08.002

[3] About ROS Official Homepage. Available at: http://www.ros.org/about-ros/. Accessed June 2022.

[4] Jetson Nano Module. Available at: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed June 2022.

[5] Intel Nuc. Available at: https://www.intel.com.br/content/www/br/pt/products/details/nuc.html. Accessed June 2022.

[6] MICROSOFT, Develop Network: Kinect Sensor. Available at: http://msdn.microsoft.com/en-us/library/hh438998.aspx. Accessed June 2022.

[7] Lidar Scanner Ydlidar X4. Available at: https://acroname.com/sites/default/files/assets/ydlidar_g4_datasheet_0.pdf. Accessed June 2022.

[8] BECKMAN COULTER ORCA Manipulator. Available at: https://americanlaboratorytrading.com/lab-equipment-products/beckman-coulter-orca\-control-module-with-robotic-arm-13916. Accessed November 2022.

[9] R. PLUTCHIK, R.; KELLERMAN, H. (Ed.). Emotion: theory, research and experience. New York: Academic press, 1986.

[10] PALAR, P. S. ROS package ros_display_emotions. Available at: https://github.com/kpiatan/ros_display_emotions. Accessed September 2022.

[11] S. RIISGAARD, S.; BLAS, G. F. SLAM for Dummies. Available at: http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf. Accessed June 2022.

[12] MARTINELLI, D. voztts. Available at: https://github.com/dmartinelli1997/voztts. Accessed November 2022.

[13] MaryTTS. Available at: http://mary.dfki.de/. Accessed November 2022.

[14] REDMON, J.; FARHADI, A. YOLOv3: An Incremental Improvement. arXiv, 2018.

[15] BJELONIC, M., YOLO ROS: Real-Time Object Detection for ROS. Available at: https://github.com/leggedrobotics/darknet_ros. Accessed June 2022.

[16] FABRO, J.; VAZ, M.; OLIVEIRA, A. Design and Development of an Automated System for creation of Image Datasets intended to Allow Object Identification and Grasping by Service Robots. January 2020, 1-6. DOI: https://www.doi.org/10.21528/CBIC2019-136

[17] COMTER, F. P. "Object detection and 3D pose estimation through the use of convolutional neural networks". Master's Thesis, Federal University of Technology - Parana, 2022.

[18] MediaPipe Pose. Available at: https://google.github.io/mediapipe/solutions/pose. Accessed November 2022.

[19] LEWIN,G. F.; ARMÊNIO, G. F. Mediapipe_track. Available at: https://github.com/UtBotsAtHome-UTFPR/mediapipe_track. Accessed November 2022.

[20] CERBARO, J.; MARTINELLI, D. UTBot@Home simulation environment. Available at: https://github.com/UtBotsAtHome-UTFPR/P3AT-D. Accessed November 2022.

[21] Intel Realsense. Available at: https://www.intel.com.br/content/www/br/pt/architecture-and-technology/realsense-overview.html. Accessed September 2022.

ANNEX

A. Hardware

- Computing
  - Intel Nuc (https://www.intel.com/content/www/us/en/products/details/nuc.html)
- Navigation
  - P3-AT Mobile Base (https://robots.ros.org/pioneer-3-at/)
  - YDLIDAR X4 (https://www.ydlidar.com/products/view/5.html)
  - FTDI232 Module (https://components101.com/modules/ft232rl-usb-to-ttl-converter-pinout-features-datasheet-working-application-alternative)
- Human-Robot Interaction
  - Generic Android Device
  - Generic LCD Monitor 10.1", HDMI Interface, 12V
- Vision
  - Microsoft Kinect V1
- Manipulation
  - ORCA Beckman Coulter Robotic Arm (https://www.ebay.com/itm/161939304997)
  - 2 Arduino MEGA (https://store.arduino.cc/products/arduino-mega-2560-rev3)
  - 2 Driver Monster Shield (https://protosupplies.com/product/vnh2sp30-dual-monster-motor-shield/)
  - MG996R Servo Motor (https://components101.com/motors/mg996r-servo-motor-datasheet)
- Power Source
  - 5 12V 9Ah Batteries
  - 12V Voltage Regulator
  - 19V Voltage Regulator

B. Software

- Navigation
  - move_base (https://wiki.ros.org/move_base)
  - ydlidar (https://github.com/YDLIDAR/ydlidar_ros_driver)
  - rosaria (https://wiki.ros.org/ROSARIA)
- Human-Robot Interaction
  - MaryTTS (http://mary.dfki.de/)
  - Ros Voice Recognition App (https://index.ros.org/p/jsk_android_apps)
  - utbots_home_voicerecog* (https://github.com/UtBotsAtHome-UTFPR/utbots_voice)
  - voztts* (https://github.com/UtBotsAtHome-UTFPR/voztts/tree/1c8446048c5d4a1000a5e4b1c4efc9d662c23069)
  - ros_display_emotions* (https://github.com/UtBotsAtHome-UTFPR/display_emotions)
  - dynamic_reconfigure (http://wiki.ros.org/dynamic_reconfigure
- Vision
  - freenect_launch (https://wiki.ros.org/freenect_launch)
  - darknet_ros (Yolov3) (https://github.com/leggedrobotics/darknet_ros)
  - Mediapipe Pose (https://google.github.io/mediapipe/solutions/pose)
  - mediapipe_track* (https://github.com/UtBotsAtHome-UTFPR/mediapipe_track)
- Manipulation
  - Arm and Claw Firmware* (https://github.com/UtBotsAtHome-UTFPR/utbots_manipulation)
- Simulation
  - CoppeliaSim (https://www.coppeliarobotics.com/)
  - Apollo 3D Model*(https://github.com/UtBotsAtHome-UTFPR/P3AT-D)