An Internet of Things Testbed for Education and Community Research

John Swaim
Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR, USA
jeswaim@uark.edu

Chad Workman
Electrical Engineering Department
University of Arkansas
Fayetteville, AR, USA
cmworkma@uark.edu

Jia Di
Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR, USA
jdi@uark.edu

Xiaojiang Du
Electrical and Computer Engineering Department
Stevens Institute of Technology
Hoboken, NJ, USA
xdu16@stevens.edu

***Abstract*—The testbed presented in this study supplies various devices to emulate a smart home. The paper highlights how devices can be connected and programmed to perform functions using an application programming interface. Remote-controlled robots in the testbed enable a user to manipulate, monitor, and configure home-based Internet-of-Things (IoT) technologies. The paper describes the equipment used in the testbed, including a wireless security camera, a smart lock, a climate sensor, and two types of robots. Security measures implemented in the testbed are also discussed. Several application scenarios are presented and analyzed on how they were accomplished to demonstrate the functionalities. The smart home testbed is a useful resource for education and development, as it allows for sufficient performance using a single control point.**

Keywords—Testbed, Home Automation, IoT, Robot

I. Introduction

THE Internet of Things (IoT) is a network of objects with sensors and the capability to connect to the Internet [1]. This network of connected devices communicates either to each other or to a centralized server. When connected the devices will share information gathered by sensors as well as take commands from a server [2][3]. This technology has been quickly making its way into our homes and places of work. With the drastic growth in the number of these devices for a variety of different commercial premises, there is a push for a more interchangeable and reliable control system [4]. An application programming interface (API) enables these devices to be connected and programmed to perform various functions. If two devices do not share the same interface, they cannot communicate directly. Currently, most devices meant for households have their proprietary interface and are controlled separately [5][6]. Only devices specifically designed around a base control system have the benefits of direct communication.

Industry also uses IoT devices to control their systems inside manufacturing plants. Recently, there have been advances in edge computing on IoT devices used in business settings [7-10]. Edge computing devices are more like a network of computers rather than a single computer with many sensors [11]. They can process information on their own and send fewer data signals to the main control point, freeing up the bandwidth for more devices on the same network. This form of control network is seen as ideal for performance. However, this type of system is unreasonable for a household where the number of devices being used at the same time is much less as compared to an enterprise setting. The best control scheme for home use is usually from a single access point so that all devices are connected to the main API and can be seen and accessed remotely during their use [12]. This scheme is more affordable and simpler to install in homes. As edge computing is getting into households, the testbed presented in this paper is a valuable tool for educational and development purposes where a single control point will better fulfill their requirements[13-15].

Most current IoT devices are controlled via an app that is made specifically for that device. This would lead to smart homes that require the user to have a different app for each device they own. Besides being inconvenient, the devices will not work effectively together as a smart home due to the disconnected nature of individual application programming interfaces [16].

In this paper, a smart home testbed system is introduced, which is capable of manipulating, monitoring, and configuring home-based IoT devices and systems. Section II describes the structure of the smart home testbed based on technologies like microcontrollers, TCP sockets, network broadcasts all centered around a single controlling server. Section III details the security and accessibility of the testbed network. Section IV analyzes several scenarios and breaks down how they were accomplished to demonstrate functionality. Finally, Section V contains a synopsis of the smart home testbed system and is concluded by considering future work.

II. Testbed Overview – Equipment

The testbed is centered around a single access point server, which establishes a network topology where all devices and components are managed through a centralized server. This access point server serves as the primary point of control, enabling users to monitor and configure all aspects

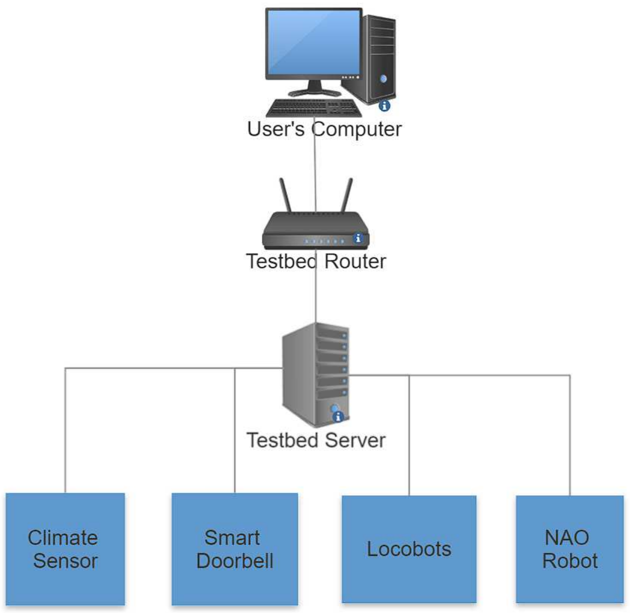

Fig. 1. Remote users can access the testbed through a wireless connection.

of the testbed, including device configuration, data collection, and analysis. By centralizing control in a single server, users can simplify the management of the testbed and ensure that all devices are working together in a coordinated manner, reducing the potential for errors or inconsistencies. In addition, having a single access point server provides users with more granular control over the testbed, enabling users to configure the testbed to meet specific research objectives and evaluate the performance of different devices and components under various conditions. The topology of the network including the server and other devices can be seen in Fig. 1, which supplies a visual representation of how the testbed is configured and how the devices are connected to the access point server.

The network connecting all testbed smart devices to the main server takes advantage of Static IP addresses and MAC address binding. By default, each device has a dynamic IP address so it would change approximately every week. This

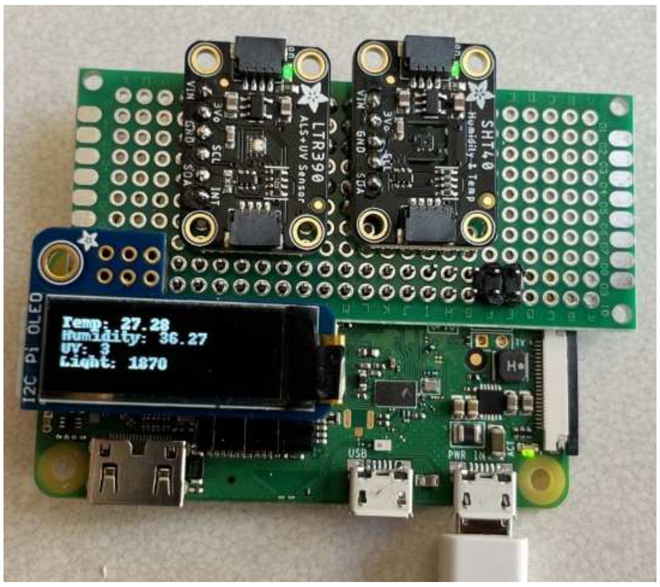

Fig. 2. The climate sensor, implemented on a Raspberry Pi Zero, transmits data to the server and displays it on the onboard screen.

would prevent the server from accessing other devices in the testbed. Static IP addresses and MAC binding resolve this by supplying a fixed and unchanging address for each device on the network. This is vital for the testbed because devices must communicate with each other and the server in real time. The code written to accomplish this task contains a string with the IP address of the other device. Since each device is always found at the address specified in the testbed documentation, user code will connect to the correct devices every time.

The testbed functionality tests were performed using a Wansview W6 wireless security camera mounted in the ceiling. It is integrated into the testbed by the Wi-Fi network to allow communication with all other devices on the network. It is used to collect data and supply insights into the behavior and activities of robots and other devices within the testbed. This data can be analyzed to detect anomalies or patterns that could be useful in improving the efficiency and effectiveness of the testbed. The way this data was analyzed is by broadcasting a livestream that was digitally processed by a Python script using Open-Source Computer Vision Library (OpenCV) [17]. Processed video frames were then utilized to control other devices in the testbed.

The testbed comprises a variety of devices, each interacting with the testbed in unique ways and collecting diverse types of data. One of these devices is an August smart lock connected via Wi-Fi to the main server. This allowed users to grant or revoke access to the lock based on specific conditions set by the controlling API. Additionally, the lock can be configured to alert the server when interacted with, providing users with valuable data on access patterns and usage. Another device controlled by the server is a Raspberry Pi climate sensor as seen in Fig. 2. The Raspberry Pi is a low-cost, compact computer that is used as a platform for IoT devices. Attached to the Raspberry Pi via Inter-Integrated Circuit are sensors to allow users to read the climate data using Python or other programming languages. For example, climate sensor data can be used to evaluate the effectiveness of different cooling strategies on device performance or to identify the impact of temperature and humidity on wireless communication and energy consumption. Due to the implementation of the climate sensor as a Raspberry Pi, its functionality can be expanded to meet future testing protocols and user needs. The two types

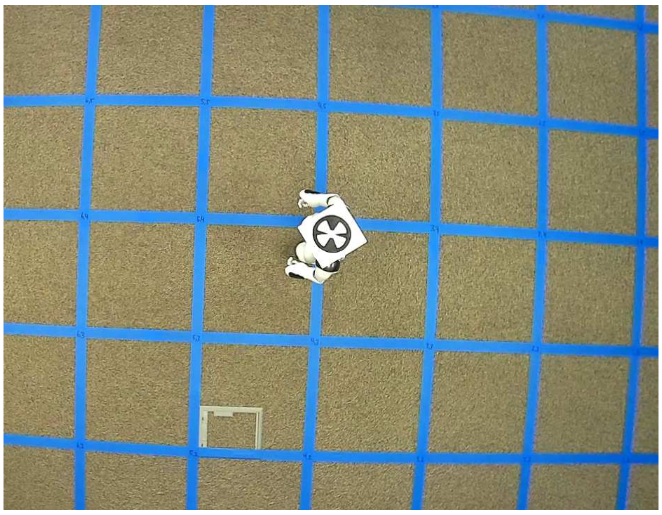

Fig. 3. Cameras were placed around the testbed for motion detection protocols, and remote users to view the testbed.

of robots used in the testbed are the LoCobot produced by Interbotix and NAO developed by the Aldebaran robot company. LoCobot moves using two wheels while NAO used servo-controlled legs to move. Both types of robots were evaluated for how accurately they could travel to a location within the testbed. All testing was performed on a 3×3m grid divided into 36 squares of 0.5×0.5m as shown in Fig. 3. Corrections to the movement protocol were made by measuring the error between the actual position and target position using a metric tape measure. With this data, adjustments were made to the W6 camera, LoCobots, and NAO to improve their movement accuracy.

III. Security

The server offers remote accessibility using NoMachine, which is a remote desktop software that allows users to connect to the server from outside of the testbed. This is particularly useful for users who need to access the testbed from outside of the university campus, as it allows them to connect securely and work on their projects as if they were physically present in the testbed area. For security purposes, robust passwords are in use, providing enhanced integrity against unauthorized access. Moreover, each user must have an account with the university to have an account on the server. Current university students and researchers are able to login with the credentials they already possess. This separates all directories so users can customize their experience and the programs they use. This also ensures the integrity of all user data from malicious or accidental actions. To further enhance the security of the testbed, regular hard drive backups are performed to ensure that user data is safe and can be easily restored in case of incidents. The schedule of such backups depends on the frequency of users committing changes to the server. Hard drive backups are stored in a secure location to prevent unauthorized access and data breaches.

The testbed server is a subnetwork of the campus network which means that it benefits from the same level of protection and security measures as the larger network. This includes firewalls, intrusion detection systems, and antivirus software that are in place to prevent unauthorized access and protect against potential cyber-attacks. In addition to the campus security measures, the testbed server has its own security protocols. The most relied upon network, the Wi-Fi, is secured with WPA2 PSK with 128-bit AES encryption.

Additionally, the network is not discoverable to outside devices, which means that only authorized users with the correct credentials can connect to it. To further ensure the security of the wireless network, it is purposely designed to have a short range to prevent any device outside of the testbed work area from connecting to the network. This prevents any outside interference, unauthorized access to the network, and users accessing devices outside the scope of the testbed.

IV. Scenarios

A doorbell system was used in the testbed for motion detection, object identification, and tracking when the door was opened. One scenario using the doorbell is as follows: NAO actuated the doorbell pressure plate which activated the camera. An unlock signal was sent to the smart lock when NAO was recognized by the doorbell. The testbed achieved this result by broadcasting a video stream after the doorbell was activated. This stream was processed by the main server using OpenCV an object detection protocol [16].

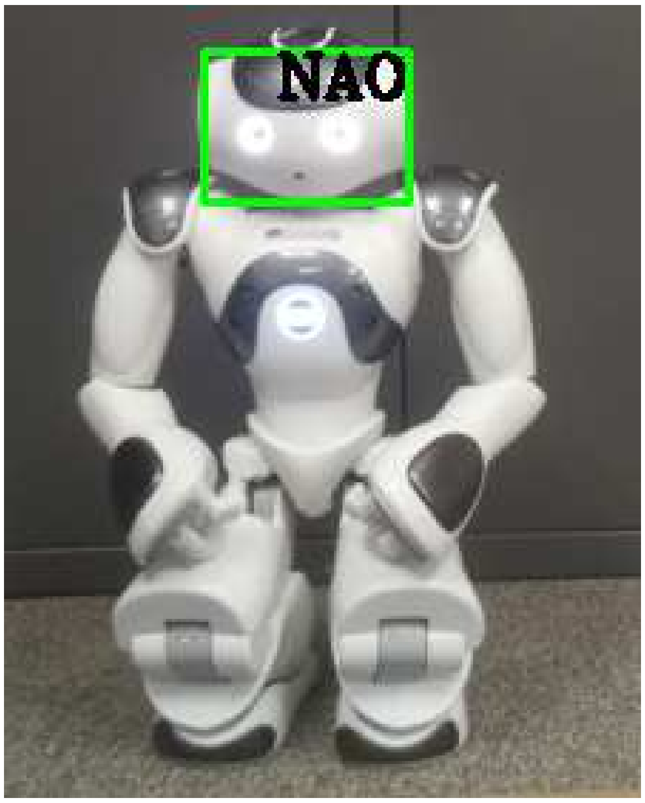

Fig. 4. The doorbell can detect objects based on the object detection protocol.

OpenCV utilizes Haar cascades to detect objects in a video stream. Harr cascades work by analyzing the pixel values in an image to identify patterns that correspond to a specific object. To achieve this, the Haar cascade algorithm utilizes a set of pre-trained models that serve as templates describing the visual characteristics of a particular object, such as NAO in this case as shown in Fig. 4. The server recognized NAO as the entity which activated the doorbell and sent a signal to a smart locking system to unlock the door.

A second scenario proved how the doorbell platform was expandable and interfaced with other IoT devices. A magnetic contact switch (door sensor) was added to the doorbell to monitor the opened or closed status of the testbed door. When the door is opened, the climate sensor would measure the UV light, brightness, temperature, and humidity of the room. This result was achieved using the Raspberry Pi General Purpose Input/Output (GPIO) pins and editable networking scripts. The door sensor connected to a custom soldered circuit board and acted as a switch connecting the Raspberry Pi 5V power pin to the ground pin. The GPIO5 pin was connected in parallel to the ground pin to receive input on door status. The doorbell monitored the GPIO5 pin for a changing voltage. The doorbell measured logic 1 from the pin with the door closed and logic 0 with the door open. The status was relayed to the server over a Transmission Control Protocol (TCP) network socket. Once the server sensed a change in door status, it would run a bash script through an SSH connection to the climate sensor. The climate sensor would report data back to the server through another TCP socket.

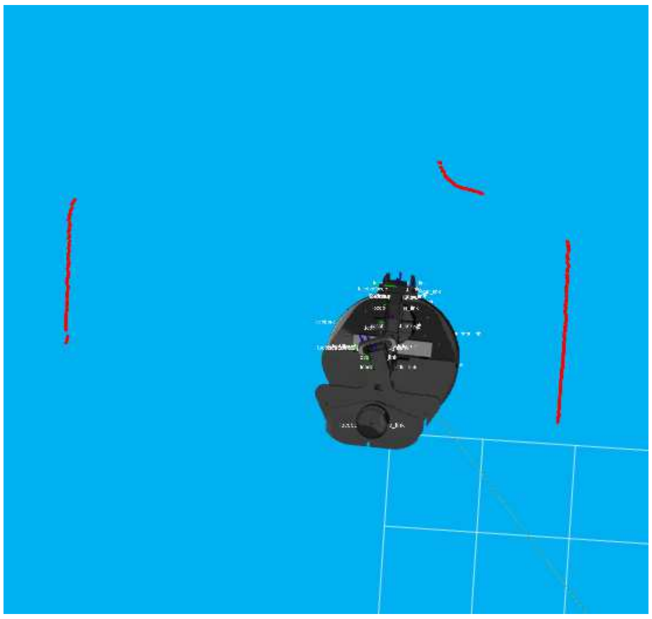

Fig. 5. The LoCoBot can detect its surroundings using a lidar sensor and process this data to show shapes and distances in RViz.

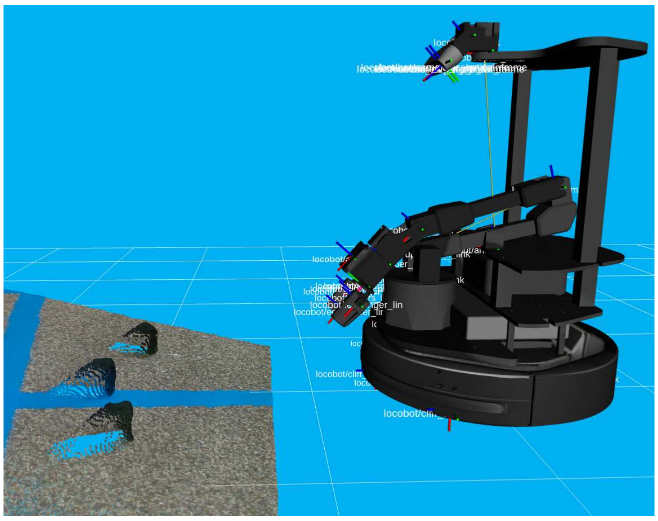

Fig. 6. RViz can convert the Intel RealSense infrared bit map into a 3D image that LoCoBot can use to process its surroundings.

The LoCobot platform was used for navigation and object interaction within the testbed. One scenario was tested where the LoCobot navigated an obstacle course to reach the desired position. Once at the position, the LoCobot picked up objects from the floor and placed them into a box. LoCobot movement was achieved using the ROS Visualization (RViz) remote monitoring software and the MoveIt package from the Robot Operating System (ROS) GitHub repository [18]. RViz ran on the testbed server, displaying LoCobot servo positions, the live camera feed, and lidar range data as shown in Fig. 5. The user relied on the RViz display in conjunction with the overhead camera feed to navigate the LoCobot and avoid obstacles.

LoCobot picking up and moving objects was accomplished with the perception pipeline package from the ROS GitHub and the equipped Intel RealSense depth camera [19-20]. The user ran an initialization script on the LoCobot which made the depth camera project infrared light onto the field of view. The depth camera captured the feedback using stereoscopic sensors and an RGB module. RViz displayed the infra-red depth map in three dimensions overlayed with the full-color camera feed shown in Fig. 6. The user then ran a Python script on the LoCobot which identified objects within the depth map and saved their positions. The LoCobot manipulator arm maneuvered to each position, picked up the object, moved the object to a user-specified location, and dropped the object.

V. Conclusion

This Internet of Things-based testbed and control system was conceived with the intent to progress development in motion detection protocols and the application programming interface that controls these device networks. The designed system works effectively, can be accessed from remote, and can be recreated affordably for personal or educational use. The motion detection analysis system is unique in low-cost testbeds, with which the accuracy of the system performs greater than the sum of its components. Based on the most common smart devices found today, this single access point testbed will work much more effectively to simulate a smart home than the state-of-the-art edge computing testbeds do.

The testbed was designed such that more smart devices could be added to fully emulate a smart home and current commercially available ecosystems. A microphone and speaker will be added to the Raspberry Pi doorbell so that users can communicate with the entity pressing the doorbell. Another planned device is a smart outlet that monitors power consumption and can be switched on remotely. Future devices will be designed using the Raspberry Pi platform and other low-cost commercially available components.

Acknowledgment

This work is supported by the NSF grant No. 2016485. The authors would also like to thank G. Tomlinson, L. Vaughan, and J. Hansen for their help to set up and evaluate the testbed.

References

[1] K. Ashton, "That Internet of Things thing", RFiD J., vol. 22, no. 7, pp. 97-114, 2009.

[2] S. N. Swamy and S. R. Kota, "An Empirical Study on System Level Aspects of Internet of Things (IoT)," in IEEE Access, vol. 8, pp. 188082-188134, 2020, https://doi.org/10.1109/ACCESS.2020.3029847.

[3] A. K. Gupta and R. Johari, "IOT based Electrical Device Surveillance and Control System," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-5, https://doi.org/10.1109/IoT-SIU.2019.8777342.

[4] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," CISCO White Paper, vol. 1, pp. 1–11, 2011

[5] J. Huang, Y. Meng, X. Gong, Y. Liu and Q. Duan, "A Novel Deployment Scheme for Green Internet of Things," in IEEE Internet of Things Journal, vol. 1, no. 2, pp. 196-205, April 2014, https://doi.org/10.1109/JIOT.2014.2301819.

[6] S. L. Keoh, S. S. Kumar and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," in IEEE Internet of Things Journal, vol. 1, no. 3, pp. 265-275, June 2014, https://doi.org/10.1109/JIOT.2014.2323395.

[7] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," in IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22-32, Feb. 2014, https://doi.org/10.1109/JIOT.2014.2306328.

[8] L. Catarinucci et al., "An IoT-Aware Architecture for Smart Healthcare Systems," in IEEE Internet of Things Journal, vol. 2, no.

6, pp. 515-526, Dec. 2015, https://doi.org/10.1109/JIOT.2015.2417684.

[9] S. Savazzi, V. Rampa and U. Spagnolini, "Wireless Cloud Networks for the Factory of Things: Connectivity Modeling and Layout Design," in IEEE Internet of Things Journal, vol. 1, no. 2, pp. 180-195, April 2014, https://doi.org/10.1109/JIOT.2014.2313459.

[10] A. Ciuffoletti, "OCCI-IoT: An API to Deploy and Operate an IoT Infrastructure," in IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1341-1348, Oct. 2017, https://doi.org/10.1109/JIOT.2017.2734068.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016, https://doi.org/10.1109/JIOT.2016.2579198.

[12] S. Lee, H. Choi, T. Kim, H. -S. Park and J. K. Choi, "A Novel Energy-Conscious Access Point (eAP) System With Cross-Layer Design in Wi-Fi Networks for Reliable IoT Services," in IEEE Access, vol. 10, pp. 61228-61248, 2022, https://doi.org/10.1109/ACCESS.2022.3181304.

[13] Y. Ouyang and T. Yan, "Profiling Wireless Resource Usage for Mobile Apps via Crowdsourcing-Based Network Analytics," in IEEE Internet of Things Journal, vol. 2, no. 5, pp. 391-398, Oct. 2015, https://doi.org/10.1109/JIOT.2015.2415522.

[14] K. Liu et al., "On Manually Reverse Engineering Communication Protocols of Linux-Based IoT Systems," in IEEE Internet of Things Journal, vol. 8, no. 8, pp. 6815-6827, 15 April15, 2021, https://doi.org/10.1109/JIOT.2020.3036232.

[15] C. Stolojescu-Crisan, C. Crisan, and B.-P. Butunoi, "An IOT-based Smart Home Automation System," *Sensors*, vol. 21, no. 11, p. 3784, 2021. https://doi.org/10.3390/s21113784.

[16] K. Liu et al., "Security Analysis of Mobile Device-to-Device Network Applications," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2922-2932, April 2019, https://doi.org/10.1109/JIOT.2018.2877174.

[17] OpenCV Official website https://opencv.org/

[18] Ros-Planning, "Ros-planning/moveit: The MOVEIT Motion Planning Framework," GitHub, https://github.com/ros-planning/moveit

[19] Interbotix,"Interbotix_ros_rovers/interbotix_ros_xslocobots/interbotix_xslocobot_perception at Main · Interbotix/interbotix_ros_rovers," GitHub, https://github.com/Interbotix/interbotix_ros_rovers/tree/main/interbotix_ros_xslocobots/interbotix_xslocobot_perception

[20] "Intel RealSense ID solution F400," Intel® RealSenseTM Developer Documentation, https://dev.intelrealsense.com/docs/intel-realsense-id-f450-module-and-f455-peripheral#datasheet