UMASS BOSTON
ENGINEERING
DEPARTMENT OF ENGINEERING
COLLEGE OF SCIENCE AND MATHEMATICS

University of Massachusetts Boston
2020 Engineering Expo Proceedings

Experimental Unmanned Aerial Vehicle Platform for Indoor Simultaneous Localization and Mapping

Deqiang Xu, Lucas E. Lomba, Carlos Orta

Abstract

This paper presents a UAV-based platform with a Nvidia Jetson Nano performing tasks on Visual Simultaneous Localization And Mapping (SLAM), and Object Detection. We have managed all the hardware resources on the drone under the Robotic Operating System (ROS) installed on Nano, including the Crazyflie 2.1 for flight control, the stereo camera for real-time video streaming, the power sensor for battery level monitoring, the Bluetooth Low Energy (BLE) and LoRa module for communication between drones. We have organized all software in the form of individual ROS packages communicating effectively using ROS messages through the ROS internal communication protocol. Our system features a high degree of hardware/software flexibility/modularity and system extendibility, and it is suitable for conducting further visual-orientated research on UAV such as incorporating multi-sensor fusion for accurate mapping.

Deqiang Xu, Electrical Engineer, E-mail: *duncan7823@gmail.com*
Lucas E. Lomba, Computer Engineer, E-mail: *lucas.e.lomba@gmail.com*
Carlos Orta, Electrical Engineer, Email: *Carlos.Orta001@umb.edu*
Advisor/Client: Dr.Honggang Zhang, Associate Professor, University of Massachusetts Boston

1. Introduction

Simultaneous Localization And Mapping (SLAM) is a technology for constructing a map of an unknown environment while keeping track of the location of a moving robot based on the information collected from sensors. Visual SLAM mainly uses one or more cameras as the sensors to complete the SLAM tasks [1][2]. The resulting map can be in 2D or 3D based on the type of utilized sensors and the applied SLAM algorithm, and a map includes the sparse or dense information of explored spaces that is essential for providing localization and navigation services to robots or other agents.

Most of the visual SLAM algorithms have demonstrated their feasibility on ground robots, such as the floor cleaning robots and autonomous cars but few on the aspect of the unmanned aerial vehicle (UAV). The system proposed in [3] is a platform for connected and autonomous vehicles (CAVs) and this wheel-based robot carries widely used sensors such as cameras and Light Detection and Ranging (Lidar) sensors for remote control, autonomous driving, and map generation tasks. Comparing to those ground robots, a UAV has a wider view and a more agile nature, capable of exploring a more complex indoor or outdoor environment where ground robots are restricted or even inaccessible. Thus, we aim to build a UAV-based platform for conducting visual SLAM experiments and further research. The platform is also designed to be highly modular and extensible so that it can easily incorporate varied types of sensors to increase the perception ability of the robot and mapping accuracy through sensor fusion, and perform other visual tasks such as object detection by incorporating other software into the system.

Object detection is another prevailing research topic in mobile computer vision, dealing with providing semantic information of the objects in images or videos. The development of deep learning-based object detection algorithms such as the You Only Look Once (YOLO) [4] makes it possible for real-time processing. Many computer vision tasks are based on object detection, such as object tracking which enables the robot to track a specific object in the environment. While the integration of object detection and SLAM system enables the robot to build a map with both geometric and semantic information.

In this paper, we describe the process of building a modular drone with a large load capability based on the Crazyflie 2.1 system, the process of building hardware drivers for onboard sensors and communication modules, and the integration of all software to the Robotic Operating system (ROS) framework.

2. Background

The mainstream visual SLAM algorithms have been proposed since the 2000s and many of them have reached satisfying estimations and reconstructing results. With the maturing of SLAM algorithms and some successful applications, the investigating direction of SLAM moves more toward sensor fusion, specialization for Embedded system, and semantic SLAM that takes advantage of the deep neural networks. These require a system with a high degree of system extendibility and modularity so that it can be flexible to investigate performance improvement with varied types of sensors and software.

Referencing the system design in [3] , we have built our UAV-based platform inheriting the advantage of the system extendibility and flexibility for those research challenges. We have decided to incorporate the ORB-SLAM2 algorithm in our system for its open-source and modular nature, the open-source Crazyflie system for flight control, and two low power wireless communication protocols for communication between drones.

Figure 1. Drone on fight test

Crazyflie The Crazyflie [5] is a lightweight, open-source hardware and software platform based on a small quadcopter. It is developed by Bitcraze, and it provides a broad and versatile platform that allows rapid development to be made on top of it. Research and development with the Crazyflie platform are possible since it provides various expansions for it, allowing it to increase the functionality and performance with minimal decrease in flight time. These expansions called Expansion decks [5], can add wide functions from obstacle avoidance, known as OA Deck to pre-programmed flight paths through the Flow Deck. When adding the expansions, the increment in weight and power has to be taken into consideration. It is possible to expand the Crazyflie to allow for heavier motors through the BigQuad Deck. The Crazyflie communicates through a 2.4 GHz radio USB dongle that allows for control implementation at the user's device.

Robotic Operating system The Robot Operating System [6] is a flexible framework for writing robot software. It collects robot-related tools, libraries, and conventions so that it can simplify the task of creating a complex and robust robot across a wide variety of robotic platforms. Each software package in ROS is executed and managed as a ROS Node and they can communicate efficiently with each other through ROS messages implemented through the inter-process protocol. ROS messages can be single string information, varied types of sensor data, processed information, motion commands, or even customized messages. ROS is naturally designed with distributed computing [6] so that the system can run across multiple machines and realize efficient communication and cooperation between machines.

ORB-SLAM2 ORB-SLAM2 system builds upon the ORB-SLAM which is an ORB feature-based SLAM system with a monocular camera for sparse mapping of the surrounding environment in real-time [7]. The performance of ORB-SLAM is degraded due to scale drift and potential failures when doing pure rotation. By applying a stereo camera or RGB-D camera, ORB-SLAM2 solves those issues and provides the relocation and map reuse services. It mainly composes three threads when operating the system: tracking of the camera movement and pose, local mapping, and loop closing for global map optimization [7].

Wireless communication Wireless communication has become a necessary part of various systems. With a wide range of choices, this system focuses on low power consumption as its key aspect. Bluetooth Low Energy (BLE) is well known for its low energy consumption, able to run for days to months on a single cell battery and its small size profile. It is a common technology that is well documented and seen throughout various industries. LoRa (Long Range) is another low power wireless communication that is able to transmit in a much longer range. BLE has a theoretical max range of 350 meters within line-of-sigh, but a more effective operation range closer to 30 to 50 meters [8] while LoRa is able to reach ranges of 10km [9], depending on area and blockage.

3. Design

In our design, we have considered the system design on drone mechanics, hardware, and software. The drone mechanics describes the choices on drone hardware considering the requirement of the load capacity. The hardware portion describes the on-board components of our system shown in Figure 2. Our software design includes the building and integration of various sensor drivers and software to the ROS framework shown in Figure 3.

3.1 Drone Mechanics

Drone Frame The drone is equipped with the Iflight IX5 V3 frame, which weights in at about 32.2 grams. This allows it to protect the circuit components from any static built up and be resistant to the moving impact, all while having little impact on the overall weight of the drone. It has several build-in cuttings and custom made holders for attaching components.

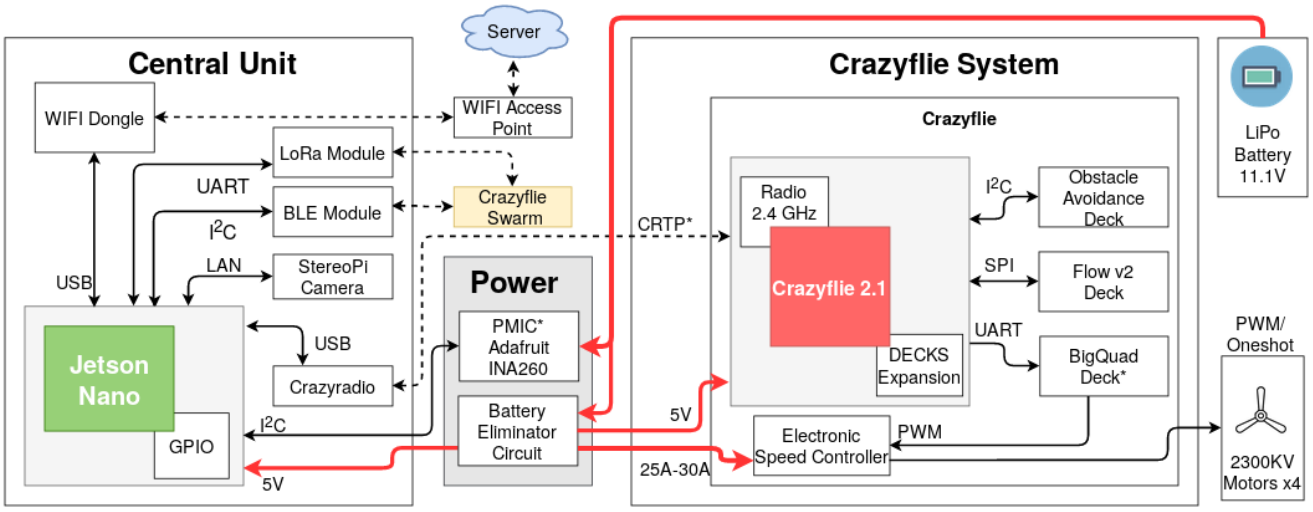


Figure 2. Hardware System Design Diagram

Power Source and Motors The drone is powered by an 11.1V rechargeable Lithium-ion Polymer battery with a capacity of 1800 milliampere-hour(mAh). The applied motors are the EMAX RS2205S and have a KV rating of 2300. A drone with larger propellers requires motors with a lower kV rating, as the propellers get larger they produce more torque and thus produce more thrust per revolution than a smaller propeller. For our system, a combination of a high kV rating and a battery which is able to discharge energy quickly is crucial to ensure that the system operates successfully.

Table 1. Table of Weights for On-Board Components

Components	Weight
Jetson Nano + StereoPi	289g
Crazyflie 2.1 + BigQuad + Flow Deck	32.4g
Motors + Battery	313.2g
Others*	40g

Others is the approximate combined weight of several small components (PMIC, BLE, LoRA and propellers, BEC).

3.2 Hardware

Jetson Nano The Nvidia Jetson Nano [10] has been selected for our system for its balance of performance, cost, and power. The 40 General Purpose Input/Output(GPIO) pins and multiple ports enable the connections to multiple onboard hardware components through varied types of communication protocols. Its CPU and GPU combination provides sufficient computing power that allows it to perform real-time visual SLAM and object detection tasks while only consuming 5 watts. As the brain of our system, the Jetson Nano uses the incoming data from the stereo camera and other sensors to perform several computing and processing tasks in real-time, and issues motion commands to the Crazyflie 2.1 or the Crazyflie swam through its communication modules.

Crazyflie The Crazyflie in our system acts as the flight controller in our system. The BigQuad Deck allows for the Crazyflie to control much larger motors through PWM signals and the OneShot protocol. The Flow Deck tracks the moment of the UAV through a floor facing camera and a laser sensor to maintain a constant flight distance from the ground. This aforementioned deck being one of the vital hardware components of the UAV which provides its autonomy. The Obstacle Avoidance (OA) Deck, has five laser sensors that allow it to maintain a distance of 2 meters from walls or obstacles in the way. Through a Proportional-Integral-Derivative (PID) controller, the Crazyflie can manage its response during flight. The PID parameters have to be adjusted as more weight and different components are added.

With these decks, the Crazyflie's internal system deals with the motor control, PID algorithm and more, allowing the Nano to focus on the visual computing tasks.

Power Monitoring (PMIC) The Adafruit INA260 is a voltage and current monitoring breakout board, which relays sensor data via I2C protocol to the Jetson Nano. It uses an integrated precision shunt resistor which can measure up to 36 volts with a continuous flow of current to up to 15 Amps, making it a reliable sensor as some the systems on the UAV can discharge upwards of 30 Amps[11]. This senor has been incorporated into our system to monitor the usage of the battery for the concern of safe flight.

StereoPi StereoPi is an open-source low-cost solution for a stereo camera set. It is based on two Raspberry Pi P1 cameras and a Raspberry Pi board. With its S.L.P image, the StereoPi can steam the captured frame through a network by protocols such as UTP and RTSP but not supporting USB streaming. The StereoPi has been directly connected to the Nano via the Ethernet cable, forming a small local network for real-time video streaming.

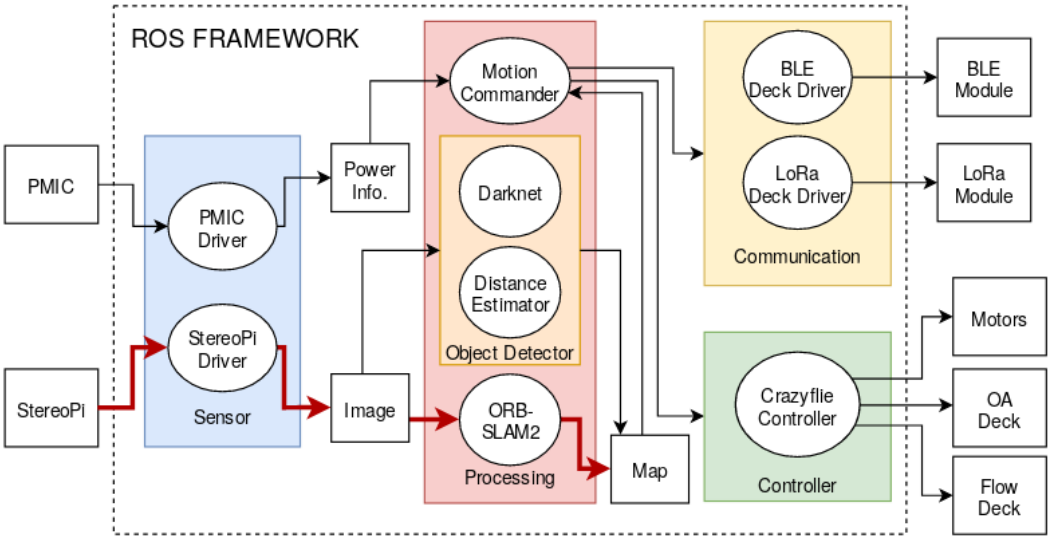


Figure 3. Software Design Diagram

BLE Deck and LoRa module A custom BLE Deck based on the NRF52832 SoC has been applied in our system and it utilizes the I2C protocol for inter-processor communication. The BLE Deck is originally designed to be fit into the Crazyflie 2.X as a functional Deck to resolve the communication problem between Crazyflies. In our system, the BLE Deck has been directly connected to the Nano as a communication module. In addition to the BLE, a Lora module has been added and attached to the Jetson Nano to realize long range communication and it uses the UART protocol for communication. These two wireless communication modules enable the UAV to exchange information with the Crazyflie swarm.

Customized PCB The Crazyflie's BigQuad Deck enables the control of the brushless motors through the usage of several GPIO pins. However, one of the IO ports conflicts with the Flow Deck's PMW3901 chip selection pin for its SPI bus. This restricts the usage of both the BigQuad and the Flow Deck together on the Crazyflie. We have designed two custom PCB boards that allowed us to bypass these requirements to avoid damaging the decks during the physical modifications.

3.3 Software

Crazyflie ROS Controller Developing upon the python interface of the Crazyflie system, the controller creates a connection to the Crazyflie 2.1 via the Crazyflie dongle when launched, and continuously listens to the motion commands which can direct the drone to lift, and maneuver through the air and safely land on the ground. It also publishes the sensor data taken from the OA deck which tells the distance to the obstacles in four directions at 1Hz.

PMIC ROS driver The INA260 uses micropython code which is a version of python optimized to run on microcontrollers. This enables the PMIC to communicate via I2C protocol with the Jetson Nano. The INA260 library uses python3 which is not compatible with our version of ROS, therefore a sub-process technique has been applied to fully integrate the INA260 to our ROS framework. This driver gets the real-time current draw from the battery to calculate the remaining energy and alerts the drone to perform a safe landing when the battery is lower than a preset threshold.

BLE Deck ROS driver This driver enables the communication between the Nano and the BLE Deck. The BLE Deck ROS driver utilizes the python SMBus module to configure the I2C bus on the GPIO pins of Nano. As a receiver, the Nano can get information from other Crazyflies or Bluetooth devices, process them and push that information into the ROS message flow. As a central motion commander, the Nano can send motion commend and other configuration information to direct the movement of other Crazyflies through the BLE protocol.

LoRa module ROS Driver The LoRa module communicates with the Jetson Nano through this driver. The LoRa ROS driver uses the pySerial module to establish serial communication with the LoRa module. As the LoRa module receives the packets, it stores the information until full and transmits them to the Jetson Nano. The data is then published to the ROS message flow. This driver allows for both the Jetson Nano to send information to the LoRa and vice-versa.

StereoPi Driver We have applied a video streaming node [12] based on the OpenCV implementation as our stereoPi driver node which steams the videos taken by StereoPi through RTSP protocol in real-time. Through this node, the frames taken by the left and right camera are merged into one big raw image. Static IP address, 10 FPS, and 640*480 pixel resolution have been set to reduce the streaming latency of this node, which ensures the real-time performance of our system.

Image Pre-Processor The image pre-processor behaves as a bridge between the raw image streamed from the StereoPi Driver and other nodes for visual takes. Based on the input requirement of the ORB-SLAM2 node and object detector, the front end of the image pre-processor evenly divides the raw image into the left and right frame and publishes the intrinsic parameters of each camera as camera-info messages. The back end of the image pre-processor is the stereo-image-proc node [13] implemented by the ROS image-pipeline project, which takes the image frame and camera-info of each camera to produce a rectified image pair.

ORB-SLAM2 ORB-SLAM2 ROS node [14] is a ROS implementation of the ORB-SLAM2 system. Comparing to other SLAM ROS package, it is more mature and easily fit into our whole ROS framework. It takes the rectified image pair as input and generates the messages of camera pose and trajectory, processed image that indicates the extracted key points, and a point cloud of the environment.

Object Detector The object detector composes of two nodes. One is the off-the-shelf Darknet ROS Node [15] which implements the You Only Look Once (YOLO) object detection algorithm. Taking the Deep Learning approach, Darknet Node can predict the objects in an image and give a bounding box to its predicting edges based on a trained model. Another node in the detector does the estimation to the target object. As a byproduct of applying the stereo camera, we can easily get the disparity map from the left and right frame. The distance estimation node collects the disparity value inside the target's bounding box, performs the translation from disparity to depth, and gives the distance from the camera to the target through running average.

4. Results

We have successfully built the modular drone, as shown in Figure 4. The placement of the hardware components has been selected to maximize area usage and the protection of the onboard components. The Jetson Nano was placed on the top of the drone to allow its heat sink to emit heat without radiating to other components. The battery was fastened to the bottom of the drone to protect the rest of the components if a malfunction were to occur. Four 3-D printed legs allow for an 80cm clearance for more space and to provide a stable platform for takeoff and landing. The stereoPi is not shown in the picture and it should be placed in the front top of the drone. We have evaluated our system in terms of a series of flight tests, and the maps generated from our environments.

4.1 Drone Flight Test

We have performed two types of flight test, which at first the drone was controlled by the gamepad controller in the early stages of development to test the responsiveness of the motors and the flight stability and then its autonomous flight capabilities was tested.

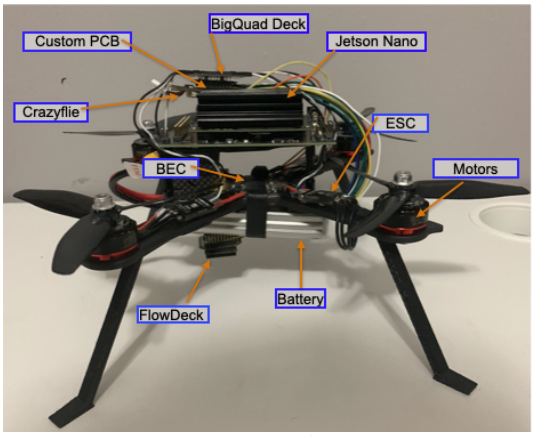


Figure 4. The Big Drone

Manual Flight Test Without the sensor data provided by the Flow Deck, the drone is limited to the input commands of the pilot via the gamepad controller. In this mode of flight, the Big Quad Deck was the only enabled peripheral deck allowing the Crazyflie to use larger motors but with no autonomous functions of flight. In the test, the drone was taken off and flew through space stably.

Autonomous Flight Test Enabling the Flow Deck, the drone can move autonomously through space with commands from the Jetson Nano without manual control. In the test, the Flow Deck help stabilize the drone and the drone could maintained a constant distance from the floor with the help of laser senor on the Flow Deck when took off. The drone successfully took off and landed without any human interference.

4.2 Camera Calibration

The camera calibration is an essential process to obtain the intrinsic parameters of the camera including focal lengths, principle points, distortion information etc. Those parameters are used when performing ORB-SLAM2 and generating disparity map tasks. The quality of the obtained intrinsic parameters directly influences the accuracy of the distance estimation result and the resulting map. The calibration of the stereoPi camera was done through the camera-calibration ROS package with an 8x6 chessboard.

4.3 Visual SLAM Results

Figure 5 shows the generated map with our system after performing the visual SLAM task to a typical house consisting of a living room, a dining room, a study, and a kitchen. White dots are the landmarks extracted from the objects in each keyframe for motion tracking and scene recognition. The blue triangle represents the current pose of the camera and the red arrow indicates the place that the camera looked at.

4.4 Object Detection Results

Figure 6(a) shows the result of object detection in our system with a pre-trained model. A sofa was successfully detected in the image. Since there is not an object class called pumpkin

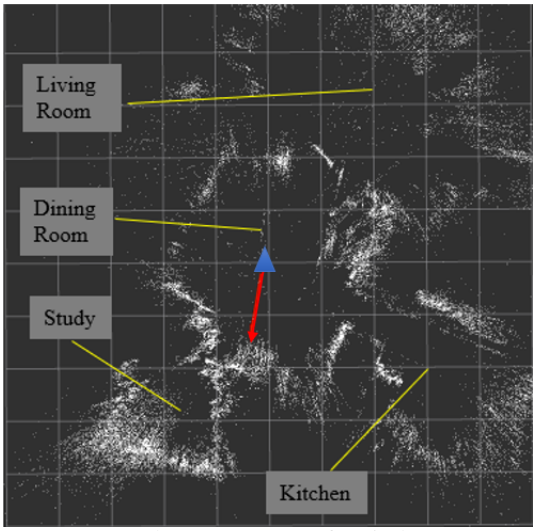
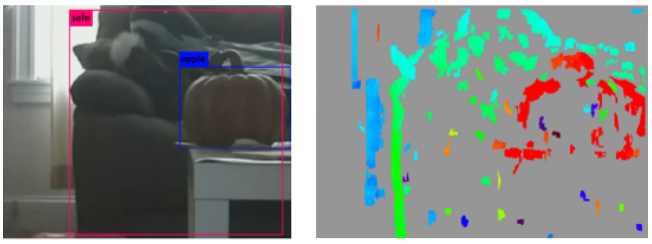


Figure 5. The sparse map of an indoor space

in the trained model, the pumpkin was predicted as an apple. Each detected object was bounded in a rectangle box to show the edges of the object with a predicted name that has the highest confidence. Figure 6(b) shows the generated disparity image based on the left and right frame when performing image rectification. The disparity image is color-coded where deeper color represents a smaller distance while the lighter color represents a larger distance from the camera. The pumpkin was found closer and the sofa was found further away from the image.



(a) (b)

Figure 6. Result of the object detector

5. Conclusion

We have implemented, tested, and proven the feasibility of our proposed design, including the development of a large drone, the programming of drivers for flight controller and drone communication, and the integration of all ROS packages for performing real-time visual SLAM and object detection tasks.

Acknowledgments

This project was supported by the Department of Engineering and the Mobile and Pervasive Computing Lab at the University of Massachusetts - Boston. Special thanks to Dr. Honggang Zhang, Dr. Filip Cuckov, and Andrew Davis.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7747236/

[2] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, Dec. 2017. [Online]. Available: http://ipsjcva.springeropen.com/articles/10.1186/s41074-017-0027-2

[3] "Hydra." [Online]. Available: https://thecarlab.org/hydraone/

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," pp. 779–788, Jun. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7780460/

[5] "Home | Bitcraze." [Online]. Available: https://www.bitcraze.io/

[6] "ROS.org | Powering the world's robots." [Online]. Available: https://www.ros.org/

[7] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7946260/

[8] M. Woolley, "Bluetooth 5go faster. go further." [Online]. Available: https://www.bluetooth.com/wp-content/uploads/2019/03/Bluetooth_5-FINAL.pdf

[9] J. B.-V. M.-D. C. Ramon Sanchez-Iborra, Jesus Sanchez-Gomez and A. F. Skarmeta, "Performance evaluation of lora considering scenario conditions," Mar. 2018.

[10] "Jetson Nano Developer Kit," Mar. 2019. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[11] "Adafruit ina260." [Online]. Available: https://www.adafruit.com/product/4226

[12] "video_stream_opencv - ROS Wiki." [Online]. Available: http://wiki.ros.org/video_stream_opencv

[13] "stereo_image_proc - ROS Wiki." [Online]. Available: http://wiki.ros.org/stereo_image_proc

[14] "orb_slam2_ros - ROS Wiki." [Online]. Available: http://wiki.ros.org/orb_slam2_ros

[15] "leggedrobotics/darknet_ros," May 2020, original-date: 2017-03-31T17:25:58Z. [Online]. Available: https://github.com/leggedrobotics/darknet_ros