# Design of a Quadrotor System for an Autonomous Indoor Exploration

M. Gäbel*, T. Krüger and U. Bestmann
University of Braunschweig – Institute of Technology

## Abstract

This paper describes the hardware components and the embedded software of a self-designed unmanned aerial system (UAS) to fulfill the task of an autonomous indoor exploration. The project, the paper is based on, was conducted at the University of Braunschweig - Institute of Technology by students and scientific researchers studying and working at the Institute of Flight Guidance. The paper is structured as follows. The first part deals with the development, selection, collocation and assembling of the appropriate hardware to obtain a reliable and robust system which suits the task of controlling the UAS, collecting information about the environment and providing the computational resources to execute the implemented algorithms. The second part deals with the software architecture that generates the control commands for the quadrotor UAS and ensures the execution of the mission by the stepwise fulfillment of single tasks.

## 1 Introduction

Investigating the interior of a building with UASs has gained interest during the past years. Especially the use of multirotor systems such as quadrotor UAS has been taken under consideration by many researchers and is still topic of current studies as they are highly maneuverable and thus especially suited to fulfill challenging indoor tasks. Moreover, compared to helicopters and flapping wings, the design of multirotor sytems allows to carry a wide range of scientific payload. Several studies, experiments and competitions [1],[2],[3],[4],[5], [6] have been undertaken and carried out to analyze and present the suitability of multirotor systems for indoor and outdoor tasks. As a result of the promising findings there is ongoing research to develop and implement new hardware and software to extend the degree of autonomy, the sensing abilities and the flight duration of multirotor systems. Especially in the case of a natural disaster UASs could be used to obtain an overview of the situation inside and around buildings to assist rescue units. This paper introduces an approach to deploy a quadrotor UAV for the task of an autonomous indoor exploration. The proposed quadrotor system is equipped with the Pixhawk autopilot, a Hokuyo laser scanner, an Optical Flow Sensor and an onboard computer with a low power consumption.

*Email address(es): m.gaebel@tu-braunschweig.de, th.krueger@tu-braunschweig.de, u.bestmann@tu-braunschweig.de

## 2 Hardware

In most cases for the purpose of doing scientific research including the operation of multirotor UAS the Astec Pelican is considered first as it satisfies the most users demands regarding scientific payload, endurance and computational capacity provided i.e. by the Astec Mastermind. Nevertheless for our demands the Astec Pelican has proven to get close to its limits regarding the payload and flight endurance. Therefore we designed and built a quadrotor UAS from scratch, which is able to carry the required payload, provides sufficient computational resources for the onboard calculations and guarantees a minimum flight time of 25 minutes. The quadrotor UAS with all its components and an detailed overview about how the single components are linked with each other is represented in Figure 1. In its final configuration the quadrotor UAS weights about 3 kilograms and has a maximum diameter of 76 centimeters.

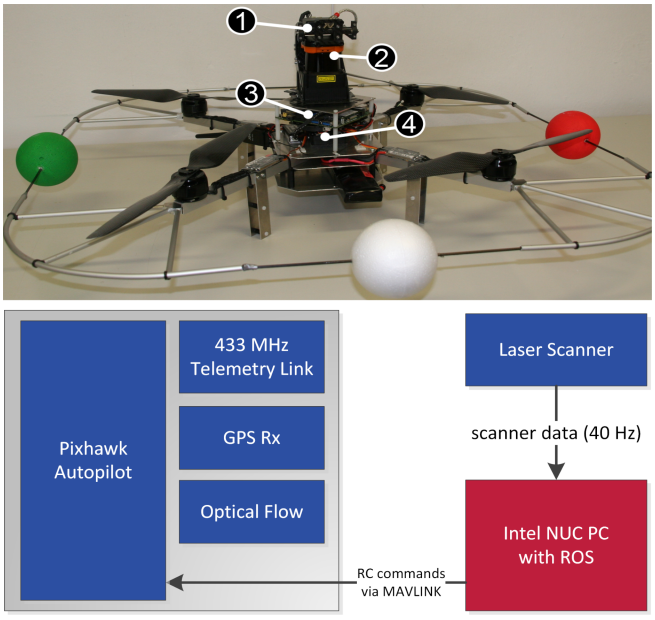



Figure 1: Quadrotor UAS

### 2.1 GNSS Unit

The GPS unit is mounted on top of the laser range finder and shown in Figure 1 marked with number 1. It consists of an ublox LEA-6H satellite receiver and the Honeywell HMC5883L 3-axis digital compass. The ublox satellite receiver is capable of handling signals from the Russian GLONASS or the American GPS and is able to achieve a horizontal dilution of precision of 2.5 meters assuming optimal conditions. Furthermore the ublox LEA-6H satellite receiver is able to process the signal of a satellite based augmentation system to enhance the accuracy of the calculated position solution up to a horizontal dilution of precision of 2 meters. The update rate of the position can be modified and set to a frequency of 1 or 5 Hz. Right after starting the satellite receiver it takes about 26 seconds to obtain a first position solution. In case the GNSS unit has just been switched off for a few seconds (hot start) or a signal of a satellite based augmentation system (SBAS) is available a first position can be obtained within a few seconds. The built in Honeywell magnetometer guarantees a compass heading accuracy with a maximum deviation of 2 degrees. Either devices, the GNSS receiver and the Honeywell magnetometer, are supplied with 3.3 Volts by the Pixhawk autopilot.

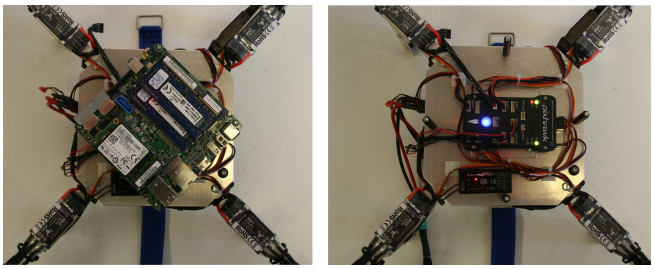
### 2.2 Hokuyo UTM-30LX

The Hokuyo UTM-30LX is shown in Figure 2 and marked with the number 2. The laser range finder device is mounted on the quadrotor UAS to sense the surrounding environment of our quadrotor UAS. With a scanning range from 0.1 up to 30 meters, a field of view of 270 degrees covered within 25 milliseconds, a weight of 210 grams and a power consumption of only 8 Watts it is the main sensing device. Intending to travel at a constant altitude that should not be occupied or blocked by free standing obstacles, i.e. furniture, the implementation of a device sensing obstacles just in a two-dimensional plane is suitable to fulfill the task of exploring the interior of a building while staying at the same floor. Besides the purpose of sensing obstacles, the information gathered by the laser range finder are required to perform the simultaneously localization and mapping (SLAM) and to reveal the unknown space inside the building.

Figure 2: Hokuyo UTM-30LX

### 2.3 Intel NUC i5 Board

The multirotor system has been equipped with the Intel D54250WYB NUC board shown in Figure 3 a). With its 4th generation Intel Core i5, 120 GB SSD hardrive and 8 GB of memory it provides the required computational power to perform the high-level task such as localization, mapping, obstacle detection, path- and motion planning and moreover provides enough disk space to log data and install the required software. Further advantages of the NUC board are its compact dimensions of 11 x 11 x 3 centimeters (w x b x h) and the low required power consumption within the range of 9 Watts at idle and a maximum of 24 Watts at full load. In addition the NUC board is equipped with a wireless local area network (WLAN) module with integrated Bluetooth to establish a reliable link between the onboard PC and the ground control station outside the building. Mainly, the link serves the purpose to monitor the onboard processes and visualize the progress of the mission.



a) b)

Figure 3: Image conversions

### 2.4 Pixhawk autopilot

The Pixhawk autopilot, depicted in Figure 3 b), is a state-of-the art autopilot equipped with the quite powerful STM32F427 Cortex M4 168 MHz processor with 256 KB of RAM and a flash memory of 2 MB. In addition to the main CPU the autopilot is equipped with a STM32F103 processor to avoid the loss of the UAV in case the main processor fails. The firmware of the Pixhawk autopilot, which can be set up with the open source ground control QGroundcontrol, can be adjusted for different kinds of UASs. Furthermore the autopilot's architecture allows to replace the orignal Pixhawk firmware by another compatible autopilot firmware i.e. Ardupilot. The autopilot itself integrates a 16 Bit gyroscope, a 14 Bit accelerometer and magnetometer both manufactured by STMicroelectronics and a MS5611 barometer which can be used as an altimeter and variometer with a minimum resolution of 10 centimeters. To extend the degree of autonomy the autopilot provides several interfaces to connect additional sensors like an optical flow sensor, airspeed sensor or a GPS unit via USB, SPI (Serial Peripheral Interface), CAN (Controller Area Network) and $I^2C$ (Inter Integrated Circuit). In addition there are interfaces for two telemetry modules and a Spektrum DSM receiver.

## 3 SOFTWARE

The developed and implemented software shall meet the major task to guarantee the safe operation of the quadrotor UAS at every point of the mission. Since this task must be achieved in a limited amount of time, at first a look around for applicable open source projects in the field of robotics was conducted. As a result the Robot Operating System (ROS) has been considered as a candidate to push the project forward. The Robot Operating System has been developed at Stanford Artificial Intelligence Laboratory for the Stanford-AI-Robot (STAIR) project in 2007 and evolved since then to a platform suitable for a wide range of robotic applications. Today ROS's further development is driven by Willow Garage[1] and the nonprofit organization Open Source Robotics Foundation (OSRF)[2]. The success of ROS is based on the large number of industrial, scientific and common co-contributors and the quite active community. Furthermore the modular design of the available software packages and the corresponding tutorials facilitates the first steps into ROS and the implementation of the algorithms required for the own project. At the moment there are more than 3000 packages available ranging from SLAM- and navigation algorithms to camera and laser drivers. In addition ROS integrates also external libraries like OpenCV or the PointCloud Library (PCL). Since ROS already provides drivers for several devices and moreover provides additional software with a variety of functionality, we decided to use ROS and to integrate the developed software in the typical ROS manner as standalone packages. The Robot Operating System and the software packages run on the NUC board described in the hardware section. The implemented software consists of the algorithms performing laser data processing, SLAM, exploration, path planning, path tracking, mission control and the communication interface between ROS and the Pixhawk autopilot. A flow chart for the exploration task integrating the hardware and the algorithms is illustrated in Figure 4. The essential algorithms and their interactions will be explained in the following subsections.

### *3.1 SLAM*

For the task of a precise indoor navigation a robust localization of the robot is quite essential. Overall the UAS is equipped with three different kind of sensors that could be used for this task. The mounted optical flow unit is directly connected to the autopilot to provided complimentary information about the current velocity and altitude of the quadrotor UAS in its body axis system. Thus the sensor data of the optical flow unit might be already out of time due to the transfer latency when processed by an algorithm running on

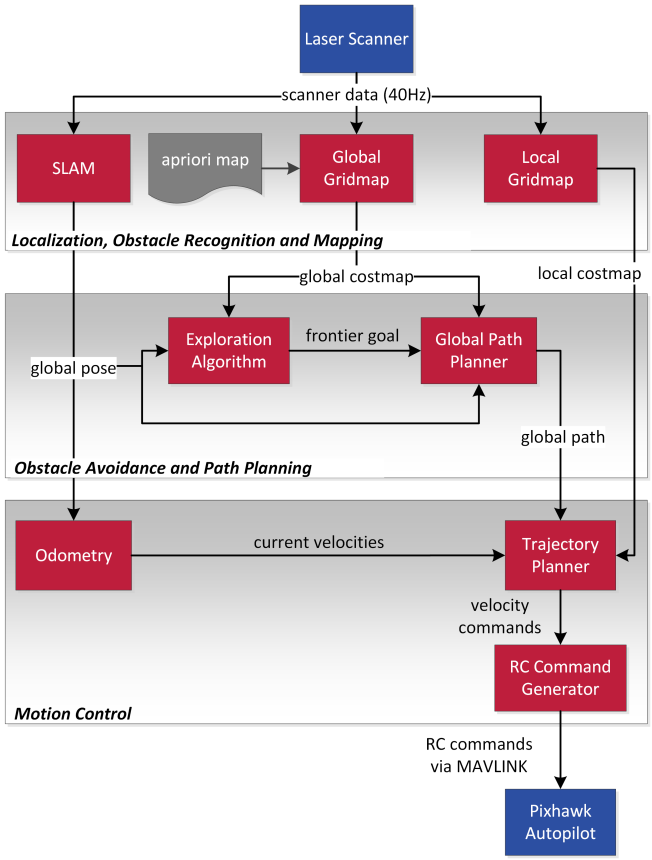



Figure 4: Exploration software configuration

the NUC board. The IMU is not appropriate for the task as well since suffering the same problem besides its biased measurement results. Therefore another approach has been taken into consideration. As the quadrotor UAS is equipped with the Hokuyo laser range finder directly connected to the NUC board, a SLAM [7] algorithm has been implemented to localize the robot and recognize its movement while mapping the environment. To fulfill this task the Hector SLAM algorithm [8] has been implemented and extended to provide additional information about the linear velocities and angular rate in the x-y plane of the 3 dimensional, global coordinate frame. Among the available LiDAR-based (Light Detection and Ranging) SLAM algorithms the Hector SLAM algorithm is preferred since the algorithm is open source and already implemented in ROS as a standalone package. The Hector SLAM estimates the position and orientation of a UAS by trying to match the incoming laser scans. Therefore the Scanmatcher algorithm aligns the latest laser scan with the map generated by all the previous laser scans. An example of a map generated by the Scanmatcher is shown in Figure 5. In order to work, compared to other SLAM algorithms like GMAPPING, the Hector SLAM algorithm does not require any odometry information. In addition the algorithm

[1] Willow Garage was founded 2006 by Scott Hassan "to accelerate the development of non-military robotics and advance open source robotics software." http://www.willowgarage.com/

[2] "The mission of OSRF is to support the development, distribution, and adoption of open source software for use in robotics research, education and product development." http://osrfoundation.org/

can be even used on 6 DoF platforms whereas others like the GMAPPING algorithm, which do not take into consideration pitch and roll angles, might fail without modifications of the underlying code. Moreover the Hector Slam algorithm is able to cope with the high publish rates of the Hokuyo laser range finder. The Hector SLAM algorithm is a typical frontend SLAM, which focuses on estimating the robot movement in real-time and does not make use of any kind of pose graph optimization, i.e. loop closing. Nevertheless the algorithm has proven to work reliable and accurate in many small scale scenarios and therefore it has been chosen to perform the 3D localization of the UAS in the unknown environment inside the building. For the task of indoor exploration the 3D pose estimate is passed to the exploration algoritm and both gridmaps. Furthermore the odometry algorithm keeps track of the global pose changes to compute the current velocities of the UAS. The map generated by the Hector SLAM algorithm is only used to verify the overall performance of the SLAM algorithm. The mapping result itself is not fed into the costmaps, which are only updated by the incoming laser scanner data. The algorithms generating the maps, the scanmatcher map and the gridmaps, can be adjusted to filter the laser scanner data in terms of range measurements. Thus the information stored in the scanmatcher map and global gridmap can differ from each other depending on the range limits for the corresponding filter.

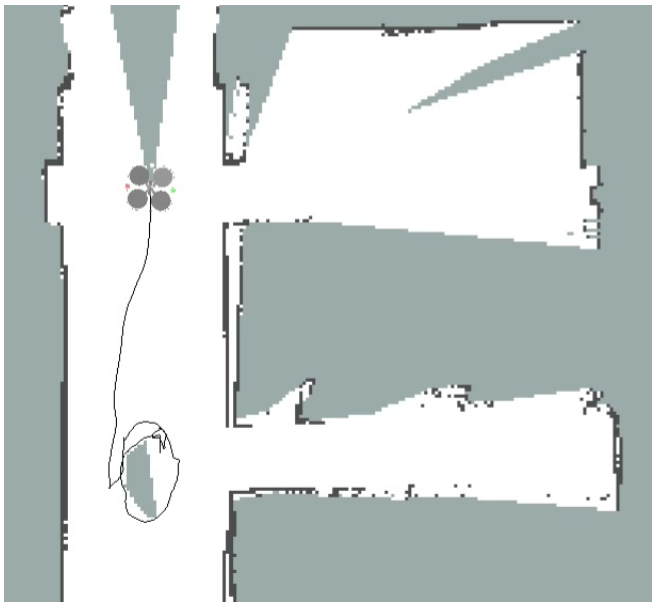


Figure 5: Scanmatcher map

### *3.2 Exploration*

The implemented 2D Exploration algorithm has been proposed for the first time by Brian Yamauchi in 1997 [9]. To suit the defined objectives the grid-based exploration algorithm is initialized with an apriori map of the building. To obtain a suitable map for the indoor exploration task an image of the building's contour is required. The image is manipulated in such a way that only the interior of the building will be considered unknown which will force the robot to explore only the inside of the building. The SDL (Simple DirectMedia Layer) library is used to convert the manipulated image into a binary file containing the gray scale information. Incorporating the binary file, the information about the image resolution, the size of a grid cell and the gray scale thresholds, each cell of the grid is categorized into free, occupied or unknown. The result of combining all these information is a scaled gridmap also called occupancy grid. An example of the steps for generating such a map is depicted in Figure 6. Figure 6 a) presents the top view of our Institute which is used to derive a simple image of the Institute's shape shown in Figure 6 b). In a next step the interior of the contour with its bounding walls is shaded in gray to be recognized by the algorithm as the unknown space. Figure 6 d) presents the grid-based costmap as the final output of the algorithm. The unknown space is still represented by the gray shaded cells. The walls are now classified as obstacles and shown in red. Moreover a predefined area around the obstacles is marked in pink to indicate the space occupied by the inflated obstacles.

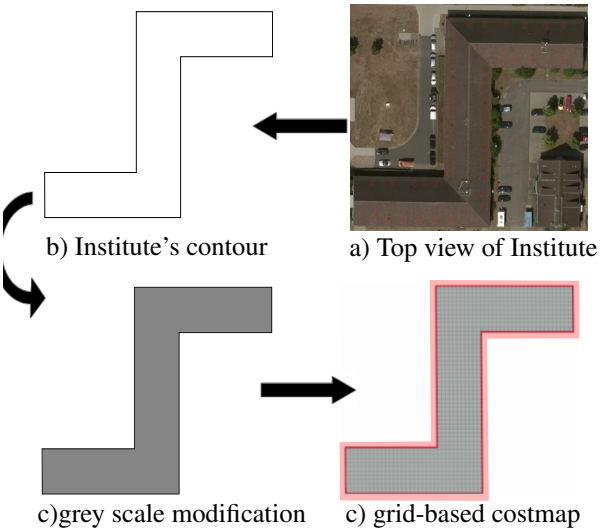



Figure 6: Image conversions

The exploration algorithm aims at cleaning out all the unknown space using the laser scanner data to update the information about the unknown cells and set them either to free or occupied. To do so the algorithm sets accessible goal poses by determining the frontiers of the free to the unknown space. For the purpose of frontier-based exploration a frontier is defined as a set of adjacent cells which share at least one edge with an unknown cell. In addition such a frontier must be wide enough to accommodate the exploring UAS. If both conditions are met the examined frontier is considered

accessible. In a next step a frontier goal is placed a predefined distance apart from the accessible frontier within the free space to preclude the collision with an unrevealed obstacle that might be hidden in the unknown space. All frontier goals are saved and updated every time new information have been added to the map. Among the current frontier goals the closest one to the robots position is submitted to the navigation algorithm which finally plans an obstacle-free path to the goal pose starting from the UAS pose. In case for a specified time no progress towards a goal pose is achieved, the currently pursued goal pose (frontier goal) will be put on a blacklist. Then, among the remaining frontier goals a new selection is performed by determining the frontier goal with the minimum euclidean distance to the UAS's position. Afterwards the determined closest frontier goal is passed to the navigation algorithm. As soon as all frontier goals are processed the quadrotor UAS will return to its initial point at the starting time of the exploration. Figure 7 presents the frontier goals identified by the exploration algorithm, the planned path towards the pursued frontier goal, the path traveled so far, the underlying costmap and the current position of the quadrotor UAS.

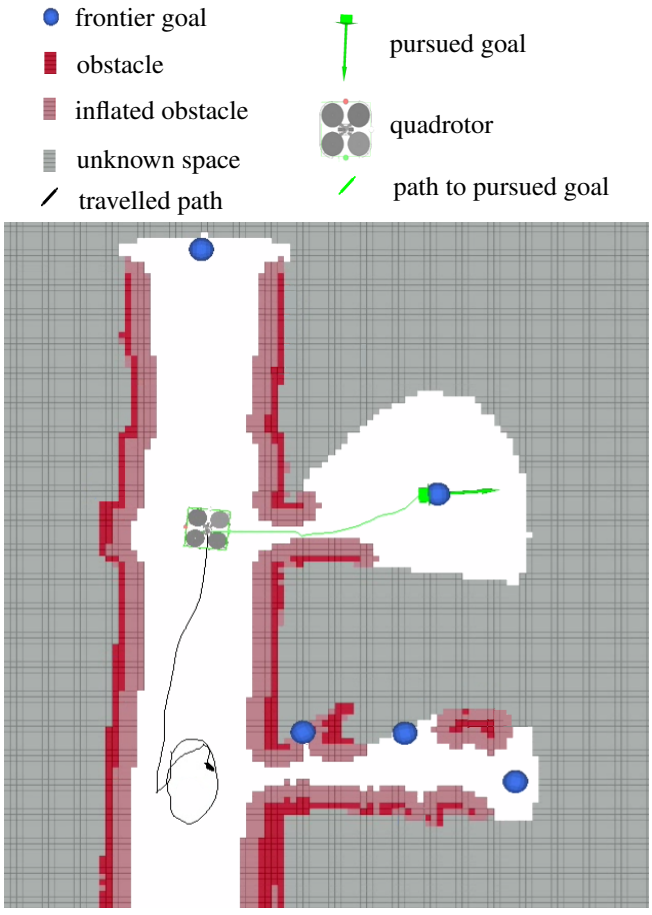



Figure 7: Exploration algrithm

### *3.3 Navigation*

The navigation task is fulfilled by the combination of a global and a local planner. The global planner plans the global path from the UAS's current position towards the goal to pursue, avoiding the recognized obstacles, the area within a predefined distance around the obstacles and all cells of the grid map marked as unknown. The underlying path planning algorithm is part of the ROS navigation stack and computes the path using a grid-based map of the environment, which is identical to the one used by the exploration algorithm, and the well known Dijkstra algorithm. Furthermore to move the quadrotor UAS from its current position towards the goal along the global path linear and angular velocities have to be generated. This task is performed by the local planner which relies on the sensor measurements and a local grid-based map of the environment. The local map is centered around the quadrotor UAS and moves as the quarotor UAS moves towards the goal. The map is centered around the UAS as the local planner just cares about obstacles within the area close to the quadrotor UAS. To determine the velocities two algorithms have been examined, whereas the Dynamic Window Approach has been favored over the Trajectory Rollout Planner as it samples from the set of achievable velocities for just one simulation step compared to sampling over the entire forward simulation period. Thus it is computational more efficient in generating the velocity commands [10]. Nevertheless both algorithms work in almost the same manner. The implemented Dynamic Window Approach [11] samples the UAS's three- dimensional control space $(\dot{x}, \dot{y}, \dot{\psi})$ and for each sampled angular or linear velocity the algorithm performs a forward simulation from the UAS's current state to predict what might happen if the sampled velocity is applied for a short period of time. The obtained trajectory is scored evaluating parameters such as proximity to an obstacle, proximity to the goal, proximity to the global path and the velocity. All trajectories that would result in a collision with an obstacles are excluded from the set of feasible trajectories. In the end, among the remaining trajectories, which correspond to a set of sampled velocities, the one with the highest score is chosen. Finally the velocities are used to derive the corresponding radio control commands, which are then submitted to the autopilot. Another feature of the Dynamic Window Approach is its ability to adapt the velocities to the environment. In the absence of obstacles and assuming the UAS can fly straight towards the goal, the algorithm will generate velocity commands that correspond to the UAS achievable maximum velocities. Whereas in an environment cluttered with obstacles appropriate linear and angular velocities are generated to avoid collisions by trying to maximize an objective function. This objective function includes a measure of progress towards the goal, the forward velocity of the UAS and the distance to the next obstacle on the path. Taking all three parameters into account, the UAS balances the objective to move fast towards the goal against the objective to

maneuver around the obstacles. The result of this behavior is a quite robust collision avoidance algorithm.

### *3.4 Mission control*

Since the entire mission consists of several task an algorithm is deployed to initiate the single actions and switch between the several task regarding the feedback received from the autopilot or other algorithms, i.e. the exploration algorithm. The mission control program keeps track of the current state of the UAS, i.e. airborne or at ground, and switches between the available tasks for the current state in a predefined order. To initiate the switching between two consecutive tasks a feedback received from an algorithm or the autopilot is required, that the current task, i.e. the sensor initialization, the exploration or the landing procedure is accomplished. The exchange of information about the current progress of a task is handled using the ROS communication architecture. Since ROS is just running on the NUC board and thus it can be just used for the communication between the algorithms running on the NUC board as well another approach has been implemented to enable the communication between NUC board and the Pixhawk autopilot. This approach is described in the following section.

### *3.5 MAVLINK*

To control the quadrotor UAS's motion by sending commands to the flight controller and to get the current state of the quadrotor i.e. for switching between different task, a communication link between the Pixhawk autopilot and the Robot Operating System must be established. As either of them the Pixhawk autopilot with its different firmwares and the Robot Operating System support the Micro Air Vehicle Communication Protocol (MAVLINK) it is implemented to establish the communication link between both. The MAVLINK protocol is an open source, header-only marshalling library which is already implemented in many modern autopilots to communicate between an UAS and its control station [12]. But likewise it can be used to enable the communication by wire between an onboard computer and an autopilot to exchange information or submit control commands. Currently the MAVLINK protocol can support 255 aircraft at once. The maximum message length is constrained to 263 bytes, whereas 8 bytes are reserved MAVLINK frame consisting of an message ID, component ID, system ID, the packet start sign, the payload length, the packet sequence and the checksum.

## 4 Conclusion

In this work an UAS system has been introduced which will be able to perform an autonomous indoor exploration. Therefore this system makes use of packages for navigation and path planning, which are already implemented and available for the Robot Operating System. This first version of the UAS system uses mainly a two-dimensional laser scanner for the environment perception and the exploration task.

## 5 Future Work

Though the designed system might be capable to sense and build a 3D map of the environment, it lacks the ability to plan a three-dimensional path. Furthermore it is not able to generate the appropriate commands to follow or track such a path. In addition to the abilities of planning and tracking an obstacle-free, three-dimensional path the sensor-based localization in a 3D environment might be considered in future to improve and enhance the autonomy of the UAS. Thus modifications and extensions of the current sensor configuration have to be made to sense the full 3D environment or performing a 3D-SLAM, i.e. by adding stereo vision camera. Furthermore an appropriate path planning algorithm and path tracking algorithm have to be implemented as well to enable the system to navigate in a cluttered 3D environment. The implementation and evaluation of the software and hardware is part of the ongoing studies and will be presented in future publications.

## References


[1] G. Andrews S. Ahrens, D. Levine and J. P. How. Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments. In *IEEE International Conference on Robotics and Automation*, 2009.

[2] R. He A. Bachrach, S. Prentice and N. Roy. Robust autonomous navigation in gps-denied environments. *Journal of Field Robotics (JFR 2011)*, 2011.

[3] R. He A. Bachrach and N. Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 2009.

[4] R. He A. Bachrach and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *International Journal of Micro Air Vehicles*, 2009.

[5] J. H. Kurz C. Eschmann and C. Boller. Cure modern - franco-german infrastructure inspection with unmanned air systems. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, 2013.

[6] A. Leber C.-H. Kuo, C.-M. Kuo and C. Boller. Vector thrust multi-rotor copter and its application for building inspection. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, 2013.

[7] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping. *IEEE Robotics and Automation Magazine*, 2006.

[8] O. von Stryk S. Kohlbrecher, J. Meyer and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011.

[9] B. Yamauchi. A frontier-based approach for autonomous exploration. In *International Symposium on Computational Intelligence, Robotics and Automation (CIRA)*, 1997.

[10] E. Marder-Eppstein and E. Perko. Navigation stack - the base loacal planner. `www.wiki.ros.org/base_local_planner`. Accessed: 2014-05-09.

[11] W. Burgard D. Fox and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 1997.

[12] Lorenz Meier. The micro air vehicle communication protocol. `www.qgroundcontrol.org/mavlink/start`. Accessed: 2014-04-30.