Micro Aerial Disaster Communication Systems

Tobias Simon
Andreas Mitschele-Thiel

Integrated Communication Systems Group
Ilmenau University of Technology
Ehrenbergstrasse 29
98693 Ilmenau, Germany
tobias.simon@tu-ilmenau.de
mitsch@tu-ilmenau.de

Abstract: In this work, an autonomous micro-aerial vehicle (MAV) system prototype is presented, which is specifically designed for controlled mobility in disaster communication networks. Our requirements and our motivation for designing and building such a system differ from traditional approaches both in hardware and software aspects, as rapid MAV system prototyping is a crucial asset for us. Our system is able to perform navigation tasks in predefined outdoor environments, including autonomous take-off, GPS-based navigation and landing. Movement decisions are computed online, depending on the requirements of the communication scenario. We propose a work-flow for the operator-MAV interaction and present an example reconnaissance measurement campaign, which forms the base for our next steps.

1 Introduction

In our technology-centric society, existing terrestrial communication infrastructure is always threatened by natural and man-made disasters. Major disaster scenarios include earthquakes, floods, forest fires, warfare and terrorist attacks. As network traffic increases in a disaster due to a huge number of phone calls, overload situations can amplify the already existing network failures, which are caused by physically destroyed communication infrastructure or by power outages. Since search and rescue coordination primary depends on the ability to communicate in a proper way, network failures prevent rescue actions from being well-coordinated, especially in geographically wide areas [ST01].

Since placing and configuring these network nodes manually consumes too much human resources and time, different alternative approaches have been proposed [iT06]. One way to overcome this problem is to add *additional network nodes* to the existing *disaster network* in order to restore its functionality or even to extend it. These additional nodes behave as service providers in close cooperation with the still operational parts of the terrestrial communication infrastructure. An intelligent multi-MAV network, as we envision it, has to implement the following desirable features:

- *self-organized* decision-making *without* a central coordinator
- *adaptation* to *changing* communication *demands*
- *fault-tolerance* and *scalability*

In such a scenario, micro-aerial Vehicles (MAVs) are an ideal platform for carrying mobile communication technologies. Recent advances in microelectronics, MEMS sensors, brushless motors and lithium polymer batteries lead to small multi-rotor helicopter designs (40x40cm) with payloads up to 500 grams and flight times up to 20 minutes. In theory, these systems can take off, fly above obstacles and place themselves on good spots for providing network connectivity. Equipped with powerful RISC processors, MAV's are smart enough to restructure the network in case the rescue forces are shifting their operation area.

Designing an autonomous outdoor MAV system is, of course, not an easy task. Several dependencies between the MAV and its environment have to be considered in advance. Figure 1 gives an overview on these dependencies. Every decision is influenced by communication (inter-MAV signaling, user data), by the state of the environment (wind, obstacles) and by the current mobility (state machine, flight speed and direction) of the system. The implications of such a tight coupling is here only shown on a very high level of abstraction, but it exists on several levels, leading to systems which are hard to predict.

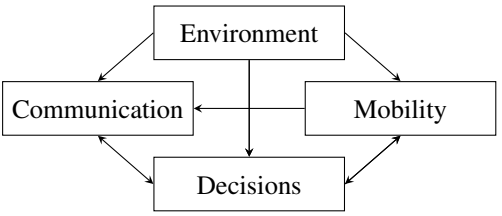


Figure 1: Dependencies and interactions in multi-MAV systems. An arrow from A to B indicates "A influences B".

In general, it is desirable to design a (multi-)MAV system which is able to support different kinds of communication for building rescue networks: *Real-time* and *delay-tolerant* communication. Real-time communication is usually required for bidirectional audio/video links which require high bandwidth, low latency (milliseconds) and is geographically restricted. In contrast, delay-tolerant data in form of e-mails, SMS or voice mails is transported with higher delay in the scale of seconds or minutes.

The remainder of this paper is structured as follows: First we will provide a brief overview on the related work in Section 2. In Section 3 we define the requirements of our MAV system. Based on these desirable features, we describe our design with respect to hardware (Section 4) and (Section 5). We present a possible system usage scenario by a human operator in Section 6. Finally, we conclude our work in Section 7 and give an outlook about our next steps.

2 Related Work

Autonomous Disaster Management Systems Most existing disaster communication approaches uses mobile, self-propelled "LAN droids", which are implemented as unmanned ground or airborne vehicles. These communication platforms create a mobile ad-hoc network (MANET), which is supposed to reestablish fundamental communication services. The "UAV Aided Intelligent Routing approach", for example, forms a hierarchical two-hop heterogeneous WLAN consisting of MAV network nodes [GPL+00]. A similar, but non-hierarchical approach is established by "AUGNet", which uses MAVs to improve ad-hoc WLAN connectivity and throughput between ground stations and mobile ground units [BAD+04]. These approaches are focused on technical aspects of MANETs, but not on self-organization techniques. Self-organized, adaptive networks for military applications were investigated in [CDB04]. The most recent work done on self-organized ad-hoc mesh-based networks is currently done by DARPA. The project "LANdroids" started in 2008 and uses battery-powered, mobile communication platforms to support combat missions. The system creates a flexible, fault-tolerant network with respect to the special circumstances of complex non-line-of-sight (NLOS) urban environments [USA02]. The research direction clearly turns towards ad-hoc flexible and reliable mesh networks with no predefined structure formed by LAN droids. Due to the MANET-inherent decentralization, special distributed services like the distributed DNS "CoDoNs" are required. The aim of such distributed services is to spread information over a certain set of independent systems to reduce the risk of information loss in the case of disappearing nodes. Several research groups work on indoor/outdoor multi-MAV testbeds for MAV coordination and communication [CAKH03, KS03].

Available Open Source MAV Systems Recent development in autopilot software for autonomous navigation is reflected in several open source autopilot hardware/software projects. The systems provides several levels of abstraction, depending on the usage of operating system mechanisms. OpenPilot [Ope12] for example, uses FreeRTOS, which is an embedded preemptive real-time operating system. As the projects Mikrokopter [Mik12], AeroQuad [Aer12], ArduPilot[Ard12], and Paparazzi [Pap12] do not provide an operating system, they have to be programmed using C/assembly. Pixhawk [Pix12] follows a hybrid approach, in which one controller is equipped with an operating system (computer vision) and the other one uses an embedded controller for sensor data acquisition, sensor data fusion, and control. Unfortunately, they do not operate a general purpose operating system. To summarize this section, available systems have the following drawbacks, which make it hard to prototype systems in a fast and flexible way:

- Monolithic embedded autopilot systems do not provide flexibility
- Physical contact is required for programming the controllers
- Embedded C programming does not provide a high level of abstraction

3 MAV System Requirements

Several drawbacks of existing MAV systems have been identified in section 2, which our approach tries to mitigate.

In contrast to the available monolithic approaches, our MAV hardware concept uses of several *highly specialized*, *lightweight*, *orthogonal* modules. This makes it easy to exchange defect modules and allows to add redundant modules, if required. Such components are for example the central "computer-on-module" (COM), the "Attitude and Heading Reference System" (AHRS), USB-hubs, USB-serial converters, and USB-I2C-converters,

As hardware flexibility is always tightly coupled with the availability of corresponding *bus and device drivers* in the used operating system (OS), a *POSIX*[1]-compatible operating system like Linux should be used on the COM. For the operating system, the The COM has to meet several minimal performance (CPU clock, main memory) and hardware requirements (e.g. availability of a memory management unit).

Besides device access, the POSIX operating system provides *multi-threading*, *file system abstraction* and it supports *multiple programming languages/environments* like C/C++, Python and Java. As a positive side-effect, this also enhances the software lifetime since the POSIX abstraction makes the software COM hardware-independent. From the programming perspective, a *self-contained*, *non-graphical build environment* should allow programming using every device (PC, notebook, tablet, smart phone) capable of establishing a SSH connection using the MAVs WiFi interface. This minimizes maintenance of cross-compiling environments, as the MAVs COM is fast enough to compile software natively.

Based on the operating system, a lot of open source technologies including programming languages and libraries should be used for gaining rich functionality with a minimal effort. As data *logging* and *streaming* is essential for debugging and monitoring the performance of the MAV, several data sinks like the local file system or a remote software like MATLAB should be supported. With regard to the operated communication or disaster network, *flexible routing* is required so that the communication system of the MAV is able to deploy AODV routing.

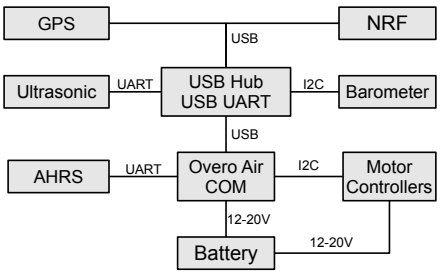
4 Hardware Overview

The MAV system we propose is called ARCADE (Airborne Robots for Communication and Autonomy in Disaster Environments). Our hardware design is based on a Mikrokopter [Mik12] chassis, including standard riggers, batteries, center plates, power distribution PCB and Robbe motors. Based on this foundation, additional lightweight hardware has been designed according to the hardware requirements defined in Section 3. The central component of the MAV is an "Overo Air" [ove12] COM from Gumstix [gum12]. This COM is very small (58 mm x 17 mm x 4.2 mm) and features a 600MHz Cortex A8 pro-

[1] Portable Operating System Interface

cessor with NEON FPU, 512MB primary memory, MicroSD card as secondary memory, 802.11 b/g WiFi, Bluetooth, I2C and SPI master, 2x USB host, as well as several ADC's, GPT's and GPIO's. ARCADE's brushless motor controllers are directly connected to the COM's I2C bus. The main sensor, a CHR-6DM attitude and heading reference system from CHRobotics is connected directly to the COM's UART. Figure 2b shows the schematic electric wiring between further MAV components like GPS, NRF (inter-MAV link), ultrasonic sensor for landing and a barometric sensor for altitude stability. Using this equipment, the MAV is able to navigate in predefined outdoor environments. It is currently not equipped with additional collision avoidance sensors, as cognitive robotics is currently not in our research focus. Figure 2a shows one of the current quadrotor prototypes used for disaster communications. The electronic components shown in Figure 2b follow the requirements specified in section 3.

(a) Quadrotor Prototype



(b) Electronic Components

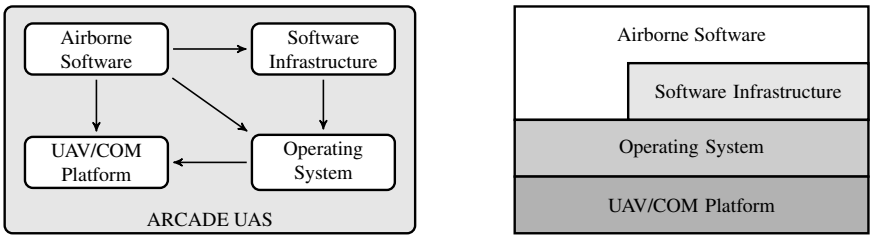
Figure 2: UAS Hardware Overview

5 Airborne Software

On the software side, our approach is to reduce embedded C programming to a minimum, since debugging capabilities are limited, memory protection is not available and low-level programming consumes a lot of time.

We decided to implement the whole system using the general-purpose operating system Linux with PREEMPT_RT real-time scheduling extensions. Although the scheduling implies some uncertainty in task execution, it turned out that the system is stable enough even to run the basic stability controllers at a frequency of 300Hz.

In general, computation-intensive and real-time enabled software components are implemented in C, while software which is typically on a higher layer of abstraction is implemented in the Python programming language. Python provides a huge variety of extension modules and C library bindings. In order to support IPC within the operating system, we use a software infrastructure called SCL (signaling and communication link),



(a) Dependencies of UAS software components (b) Layer structure of UAS software components

Figure 3: Software system overview

which implements configurable inter-component message passing based on open source libraries such as ZeroMQ [zer12] for message-passing, YAML [yam12] for configuration file processing and Protobuf [pro12] for data serialization. The dependencies between airborne software, software infrastructure, operating system and the underlying hardware are shown in Figure 3a. The final layered system structure which forms the base of our system is depicted in Figure 3b. Here, the airborne software uses the operating system and the software infrastructure directly, but it interacts only in an indirect way with the COM hardware, which makes the system portable to new hardware. Figure 4 shows the main components involved in ARCADE's software infrastructure two of them, "CORE" and "ICARUS" are discussed in detail. The main inter-MAV communication protocol is based on the MAVLink marshalling library and QGroundControl is used for multi-MAV visualization [aut12].

The *Core Component* is a highly specialized real-time component responsible for controlling the MAV system. It is implemented in C and reads several sensor inputs, fuses these inputs to create global position / speed estimates, and controls the MAVs actuators according to given position set points. The software requires various sensor inputs like AHRS (including 3D Accelerometer, 3D Gyroscope and 3D Magnetometer), GPS (Global Positioning System), Ultrasonic Sensor, and a Barometric Sensor.

The *ICARUS Component* (Intelligent Command Arbitration and Reaction on Unforeseen Situations) is a proxy for the mission planner, managing and evaluating the system's flight state machine, checking the battery level and arbitrating mission requests according to remaining battery power. A mission is not like in other systems a list of actions and way points, but it consists of a Python script which dynamically decides where to go and what action has to be performed. Basic commands like autonomous take-off, GPS-based navigation and landing are decomposed by this component into sub-commands sent to the core component.

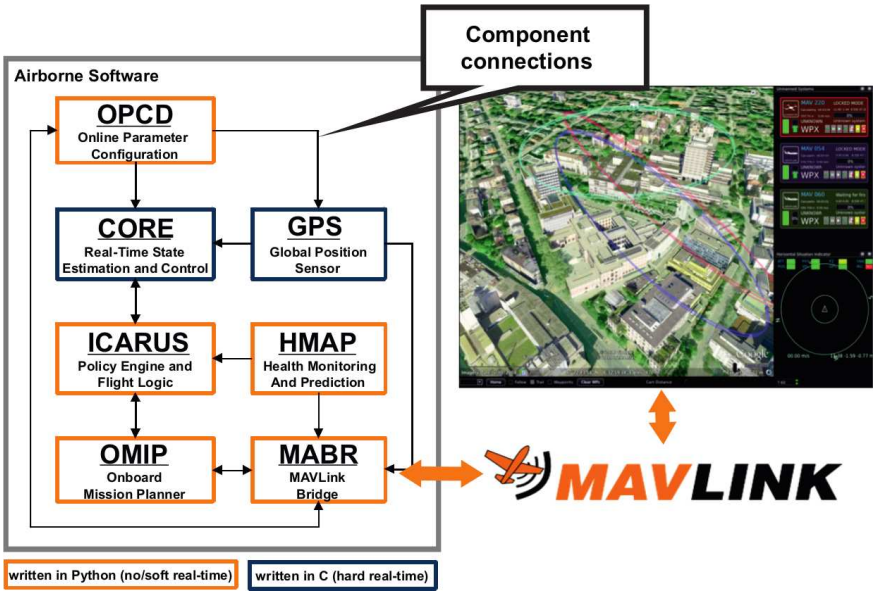


Figure 4: ARCADE software architecture

6 Possible Usage Scenario

In a rescue scenario, the proposed work-flow for the operator-MAV interaction should consist of the following sequence, which minimizes the MAV operator's effort for running the system:

1. A specially trained MAV operator takes out the MAVs, powers them and places them next to each other on the ground.

2. The operator defines the rescue area, in which the systems should analyze the existing communication infrastructure and other communication peers. Other peers could for example be people trapped in collapsed buildings, trying to communicate using their mobile phones.

3. The operator starts the system using the QGroundControl GUI. After this command, no further activity from his side is required and the system state/performance may be monitored using the GUI.

4. The MAVs take off and perform a coordinated reconnaissance flight, in which a cooperative map of localized communication peers is created. Additional functionality would also include image analysis for identifying destroyed buildings, flooded areas and damaged infrastructures.

5. When the map is complete, the multi-MAV system starts to split into groups responsible for different goals.

One goal for the system could for example be to collect messages from mobile phones and tag them with GPS positions, depending on text analysis for keywords like “help”. If there is someone sitting in a cluttered building, rescue forces could be assisted by providing such information Another goal would be to create a *high-bandwidth ad-hoc network* in the rescue area, for audio and video transmission between rescuers and the rescue headquarters in a peer-to-peer fashion.

We have already conducted a fully autonomous outdoor reconnaissance mission for localizing WiFi network nodes, where a localization error below 5 meters was achieved. A signal strength map created using real measurements is shown in Figure 5.

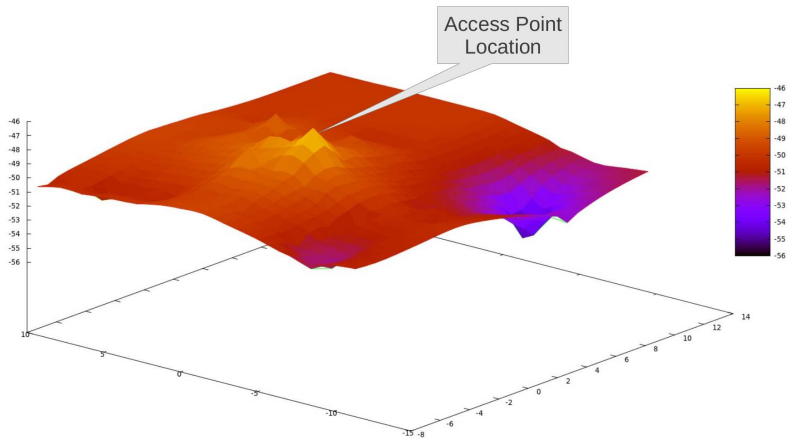


Figure 5: Airborne localization of a WiFi base station. Horizontal coordinates denote the MAV’s 2d position while the vertical coordinate indicates the received signal strength at the corresponding position.

7 Conclusions and Further Work

In this paper we have presented the requirements as well as hard- and software design of our ARCADE MAV rescue support system. Our system is able to perform navigation tasks in predefined outdoor environments, including autonomous take-off, GPS-based navigation and landing. Movement decisions are computed on-line, depending on the requirements of the communication scenario and depending on the system’s remaining battery lifetime. Besides proposing a general MAV-based rescue network creation life-cycle, we have given example goals for the disaster network. Also, an airborne measurement campaign with the goal to localize WiFi access points has been conducted. Further investigations will investigate multi-MAV system scenarios, where several MAVs will coordinate to achieve a common goal and create a disaster network demonstration.

References

[Aer12] AeroQuad Website. Website, 2012. https://www.aeroquad.com.

[Ard12] ArduPilot Website. Website, 2012. https://code.google.com/p/ardupilot/.

[aut12] QGroundControl authors. QGroundControl Website. http://www.qgroundcontrol.org, 2012.

[BAD+04] Timothy X Brown, Brian Argrow, Cory Dixon, Sheetalkumar Doshi, Roshan george Thekkekunnel, and Daniel Henkel. Ad Hoc UAV Ground Network (AUGNet). In *Proc. AIAA 3rd âUnmanned Unlimitedâ Technical Conference*, 2004.

[CAKH03] With Timing Constraints, Mehdi Alighanbari, Yoshiaki Kuwata, and Jonathan P. How. Coordination and Control of Multiple UAVs. In *in The American Control Conference*, pages 4–6, 2003.

[CDB04] K. Chandrashekar, M. R. Dekhordi, and J. S. Baras. Providing full connectivity in large ad-hoc networks by dynamic placement of aerial platforms. In *Military Communications Conference, 2004. MILCOM 2004. IEEE*, volume 3, pages 1429–1436 Vol. 3, 2004.

[GPL+00] Daniel Lihui Gu, Guangyu Pei, Henry Ly, Mario Gerla, Beichuan Zhang, and Xiaoyan Hong. UAV Aided Intelligent Routing for Ad-Hoc Wireless Network in Single-area Theater. In *In Proceeding of IEEE WCNC 2000*, pages 1220–1225, 2000.

[gum12] Gumstix Website. Website, 2012. http://www.gumstix.com.

[iT06] iDirect Technologies. Disaster Management: Communications Solutions for First Responders, 2006.

[KS03] H. Jin Kim and David H. Shim. A flight control system for aerial robots: Algorithms and experiments. *Control Engineering Practice*, 11:1389–1400, 2003.

[Mik12] Mikrokopter. Website, 2012. http://www.mikrokopter.de.

[Ope12] OpenPilot Website. Website, 2012. https://www.openpilot.org.

[ove12] Gumstix Overo Air. Website, 2012. https://www.gumstix.com/store/product_info.php?products_id=226.

[Pap12] Paparazzi Website. Website, 2012. https://www.paparazzi.enac.fr.

[Pix12] Pixhawk Website. Website, 2012. http://pixhawk.ethz.ch.

[pro12] Protocol Buffers. Website, 2012. http://code.google.com/p/protobuf.

[ST01] R. Simon and S. Teperman. The World Trade Center attack. Lessons for disaster management. *Crit Care*, 5:318–320, Dec 2001. Simon, RTeperman, SReviewEnglandCritical care (London, England)Crit Care. 2001 Dec;5(6):318-20. Epub 2001 Nov 6.

[USA02] DARPA USA. BAA 07-46 Proposer Information Pamphlet, 2002.

[yam12] YAML. Website, 2012. http://www.yaml.org.

[zer12] ZeroMQ. Website, 2012. http://www.zeromq.org.