# Problems

## GENERAL PROBLEMS WITH THE FLIGHT CONTROLLER

## Project-Management

### Time- and Team-Management

At first our team consisted of nine people which was reduced to eight during working on the project. Due to the nature of our curriculum we weren’t able to effectively coordinate dedicated times to work on the project. This was especially a problem in the fifth semester (second half of working on the project), where every student had to choose three elective courses which took place at different times. It was almost impossible to find a fixed weekly date where everyone had time to work on the project as a group, especially with a team with so many members. This made synchronization of knowledge about the state of the tasks being worked at amongst team members a problem.

### SCRUM

Scrum is a very good management system for software development. Development with SCRUM is an agile, iterative and incremental procedure. Sadly, SCRUM didn’t yield the productivity that our team hoped for. SCRUM is normally meant to be used by small teams that are working together closely on a daily basis to reach a common goal. Our curriculum however does not allow for daily work on the project. We were only able to dedicate two or three days a week to the project, which rendered SCRUM less effective than it could have been. SCRUM meetings took up a lot of time with our group of eight to nine individuals. It was hard to stay focussed in the meetings with so many people as discussions often drifted into regions that were hardly relevant for the tasks at hand. The more people a group consists of, the more thoughts go into a discussion and sometimes it takes too much time to evaluate each and every opinion which generates a time-consuming overhead.   
Sprint planning meetings at the beginning of each sprint were held to get an overview about what should be the output of each sprint. Each sprint planning meeting took up several hours of time for the whole team that could have been dedicated to actively working on the project. Creating user stories and tasks took up too much time. This was because in sprint planning meetings the whole sprint, which was about six weeks of work in our case, had to be planned in meticulous detail. Task-priorities were determined in a democratic fashion with votes from every team member. These sprint planning meetings led to further discussions which took up even more time than the daily SCRUMs. Although SCRUM is meant to reduce time spent on planning the procedure of development, in our case it had the opposite effect and increased overhead.

### JIRA

We used a project management tool called JIRA for managing our tasks and keep track of priorities and dependencies. After each sprint planning meeting, the SCRUM-Master had to update JIRA with new tasks and priorities for the sprint. The upkeep of JIRA took away a lot of time for the SCRUM-Master which could have been better spent in actively working on the project, especially because most team members didn’t actively use JIRA. Not using JIRA wasn’t because of laziness though. The team was well aware about what had to be done, even without JIRA. JIRA was simply an inconvenience for our project and was dropped later on.

### Git

It was decided to use Git as the management system for resources like documents, datasheets and source code of our project. Further it was decided to use *SourceTree* from *Atlassian* as a Git-client. SourceTree has an easy to use graphical user interface and has a faster learning curve than Git’s terminal-like interface. Although easy to use, some team-members had problems getting SourceTree to synchronize their files reliably on their personal computers. Git also behaved in a weird way when installed the computers in the laboratory (REF GENERAL PROBLEMS WITH THE FLIGHT CONTROLLER). Another problem was incompatibility with the Eclipse IDE-Environment. Sometimes team-members simply forgot to pull to have the most recent resources which could have been circumvented with a feature that pulls from the repository automatically every five minutes.

## Linux and Buildroot

We were a bit late in choosing members that specialized exclusively in understanding the Embedded Linux system and the VM that is used to cross compile code for ARM. It sometimes happened that one of our Linux specialists was not in the laboratory when a quick update of code should have been tested on the embedded OS which led the development to a grinding halt at times. This problem stemmed from the fact that our curriculum was so diverse in the fifth semester.   
The team underestimated the time it would take to understand the system and get enough experience to work with it reliably. Many stumbling stones occurred during development like getting stable WiFi working or realizing that the date had to be set before using secure copy (SCP) to the ARM-system.

At first we thought that we could implement a WiFi-driver for Embedded Linux on ARM ourselves. After two weeks of research and work we scrambled this idea as it took up too much resources. We had to look into what integrated drivers Buildroot offered and chose a WiFi-dongle with a supported chip accordingly. It has to be stated that even if Buildroot offers a driver it still doesn’t mean that everything works out of the box if implemented into the kernel. Configuration of the WiFi-stick (especially the WPA-supplicant part) took up a considerable amount of time.   
Working with Buildroot wasn’t easy. Although it is meant to be easier to use than similar tools for building custom Linux distributions, it has still a steep learning curve. The design of the Buildroot user-interface is not very self-explanatory and sometimes check boxes for customization options are hidden behind dependencies. We weren’t able to find a driver for our WiFi-stick before checking some, at first seemingly unrelated, options in Buildroot. The steps to bring an updated image of the operating system onto an SD-card have to be followed precisely and it was hard get everything right when not working with Buildroot in a daily basis. Things that were clear three weeks in the past had to be re-learned at times because of the unintuitive nature of the system. The tutorials from Mr. Strahnen in the Docs folder of the VM were a great help for working with Buildroot.

## Development Tools from Altera

## USB-Controller and 3D-Mapping

To get the multi-copter to the point where it is able to map its environment in 3D is a very hard task. A lot of time was spent finding a suitable USB-controller that matches all the requirements (pin count, linux drivers, data transfer rate) of our system. At one time a suitable controller was found but it was only available for the automotive sector so availability on the free market had also be taken into consideration.   
But even after finding a suitable USB-controller, problems still bubbled up. The raw computing power needed for 3D-mapping from two Kinects in real time is very high. Mr. Steiper stated that on his private multicopter, a dedicated mobile Intel i7-CPU is doing this task solely and it is still only able to process the image at a low framerate. The assumption that the ARM-CPU on the DE1-SoC-Board would suffice for this task had to be corrected. New 3D-cameras that need less performance have to be found. But even if performance wasn’t a problem, finding Linux drivers for our ARM-processor could pose quite a challenge as 3D-cameras haven’t penetrated mainstream in a big way at this point in time and custom drivers for ARM are rare. Own drivers would have to be written if no Linux drivers are to be found which could take up a whole to semester project for a group of students (if even possible).   
In the end, the Task of getting 3D-mapping to work had to be dropped as it would have taken up most of the time left for the project. The problems that showed up while working on the USB-controller are documented in the corresponding chapter in more detail (REF zum USB Controller Gedöns).

# Lessons Learned

## GENERAL PROBLEMS WITH THE FLIGHT CONTROLLER

## Project Management

* A team of eight to nine students is too big to effectively work on one project. Even if the team is divided into smaller groups, communication suffers from the size of the whole team.
* Reducing overhead by choosing the right method to manage the project is of high importance. SCRUM was not the right choice for our project. It is probably way better suited for teams that are working on projects on a daily basis. Managing User Stories and Tasks takes up too much time when the group is only able to work actively on the project on two to three days a week. But even with the drawbacks of SCRUM for the X-Copter project, there was an undeniable learning effect in having to work with such a system.

## Linux and Buildroot

* It is necessary to choose someone or even better a small group that specializes itself in working with linux.
* It is essential to get to know Buildroot when adding drivers to the Kernel. The Buildroot webpage has a great manual that (REF Buildroot Anleitung von der Seite <http://buildroot.uclibc.org/downloads/manual/manual.html>) describes everything in detail. The Documents from MR. Strahnen located on the VM are also a great help.
* Some basic knowledge about how makefiles and (cross-)compilation in general works should be acquired before developing for the ARM-system.

## Development Tools from Altera

## USB-Controller and 3D-Mapping

* This task will take up a huge amount of time. A group of five people working exclusively on this problem for one or two semesters could be sufficient to get this feature to work.
* Kinect cameras are not suitable for the DE1-SoC system. The demanded computation power for two Kinects is simply not available.
* Alternatives for Kinect will have to be chosen. One major point in considering 3D-cameras is the computing power that is needed to handle the 3D-image data.
* It is no easy feat to choose the right USB-controller that is compatible with the DE1-SoC Board and checks all the boxes on the list of features that are needed to use USB-driven 3D-cameras.