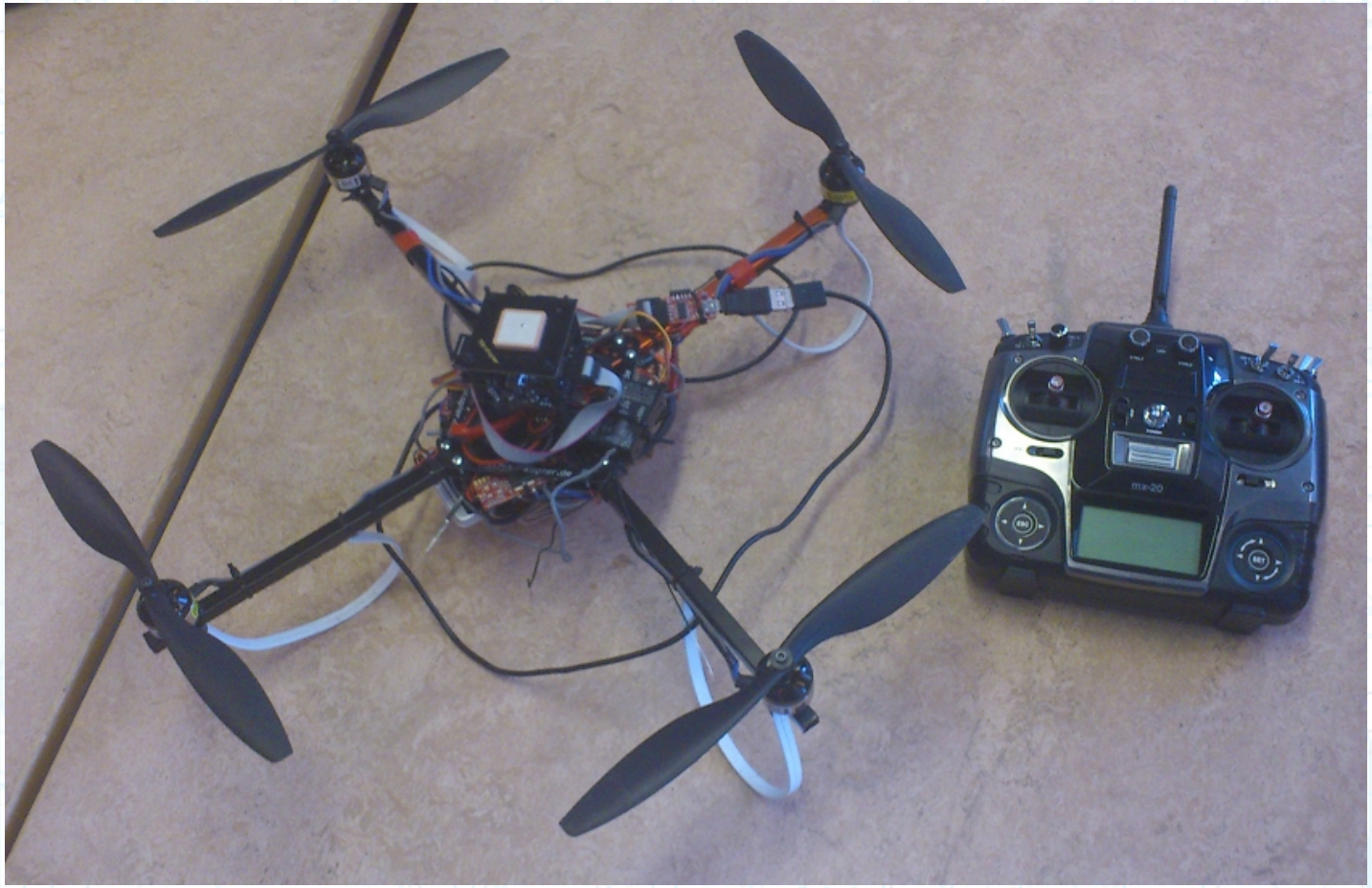
Distributed data collection with an UAV

Emanuel Wahlqvist, Adrian Dudau, Adrian Caragea, Egemen Taskin



Project outline

Navigate a UAV to collect data samples of remote sensor stations.

For navigation of the UAV, a smartphone should be used along with a simulation of data collection.

All software should be written in modules to allow change of UAV/Phone/Data collection

Scenario description

A cold morning in the north of Sweden a researcher prepares for his weekly round of data collection from his ice cracking sensors out in the wild mountains.

He loads the phone with a predefined waypoint set, and power up the UAV. After helping the it to take off, he pours a cup of coffea and finishes it just in time to recieve the UAV with the freshly collected data that it got from all the remote sensors places months before.

As always, he wonder how this little thing can find all the sensors, all by itself and then return home.

Truly amazing.

Project milestones

- Establish communication between the phone and the UAV
- Navigate the UAV using GPS waypoints
- At every waypoint, collect and store a data sample

The smartphone

- HTC Hero
  - Android 1.5 (API 3)
  - Modified kernel with USB Host mode

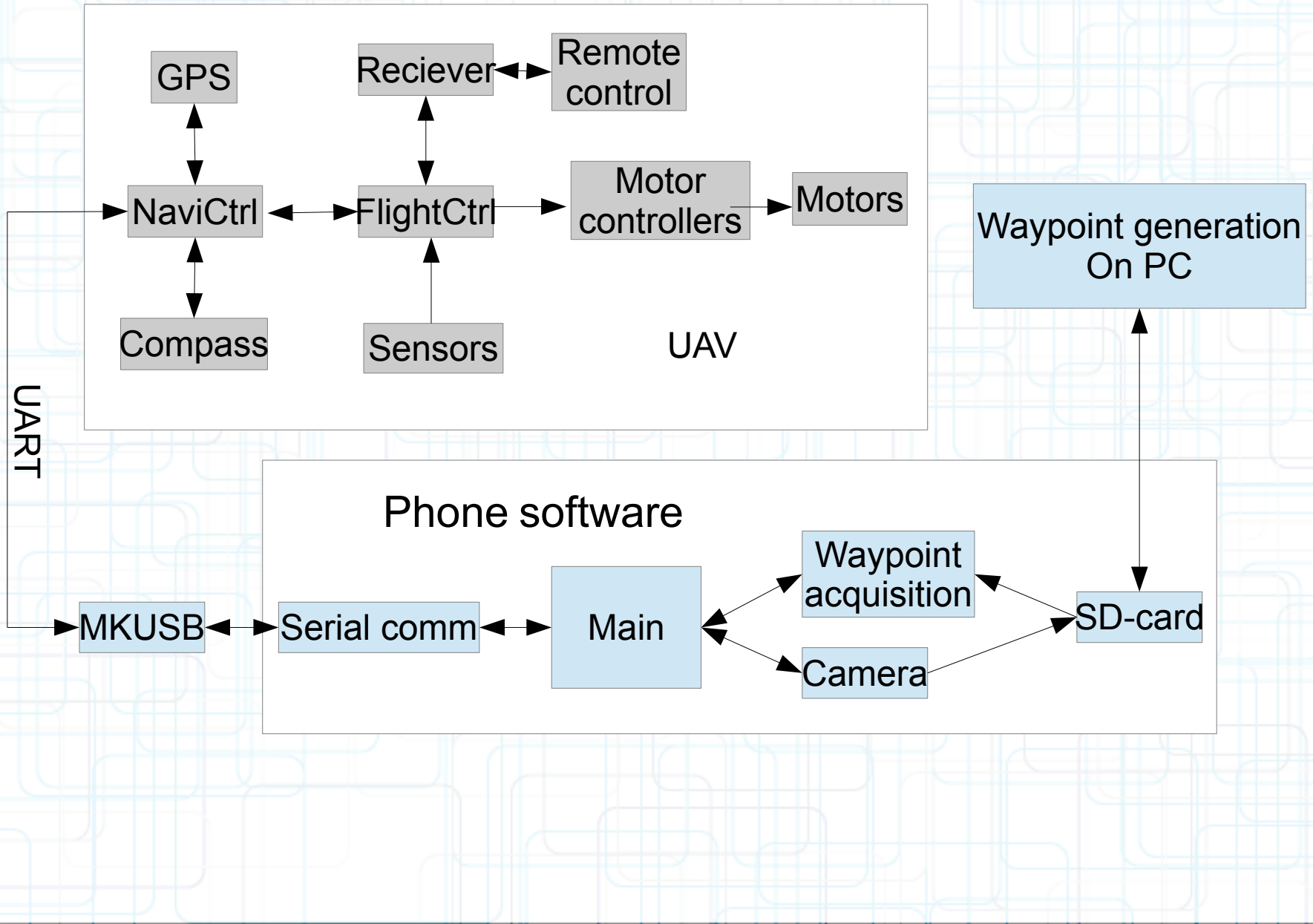
The UAV

- UAV from mikrokopter.de
  - Open source software (almost)
  - GPS Navigation with waypoint flight
  - Position hold
  - Altitude hold

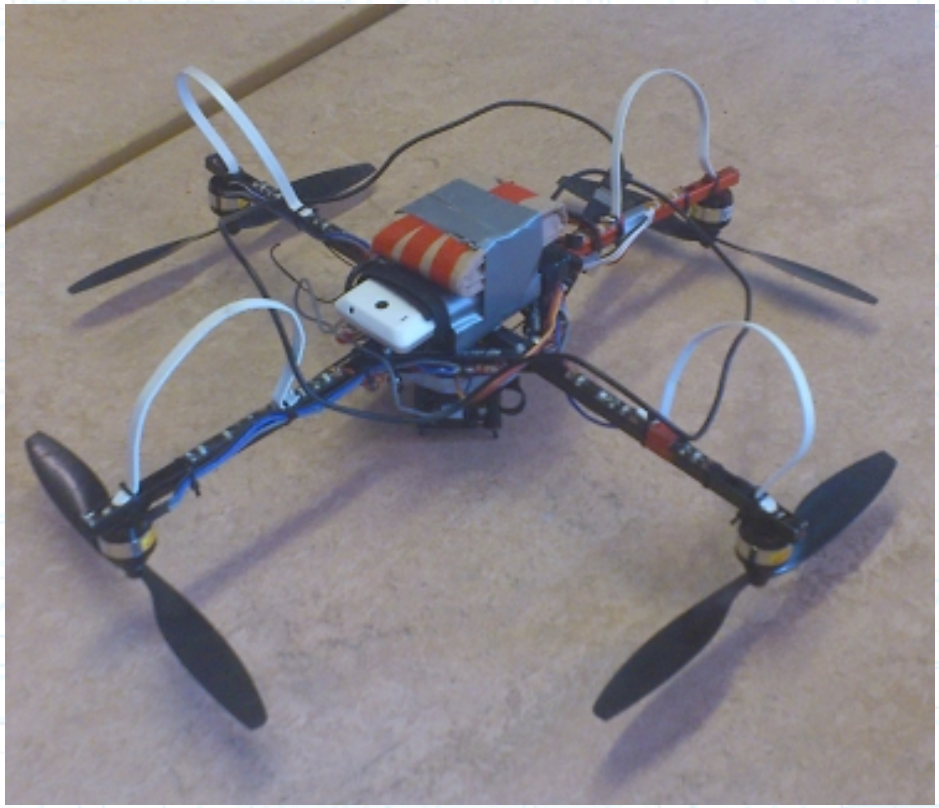
NaviCtrl

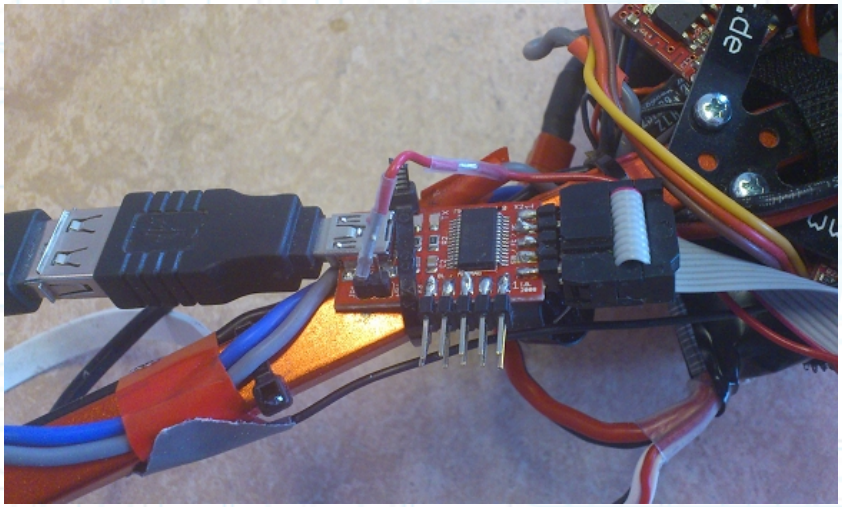
- Recieves waypoints over UART
- Sends flightcommands to the FlightCtrl
- When reaching a waypoint, it checks for further waypoints, if empty then it hovers

System overview



HW setup

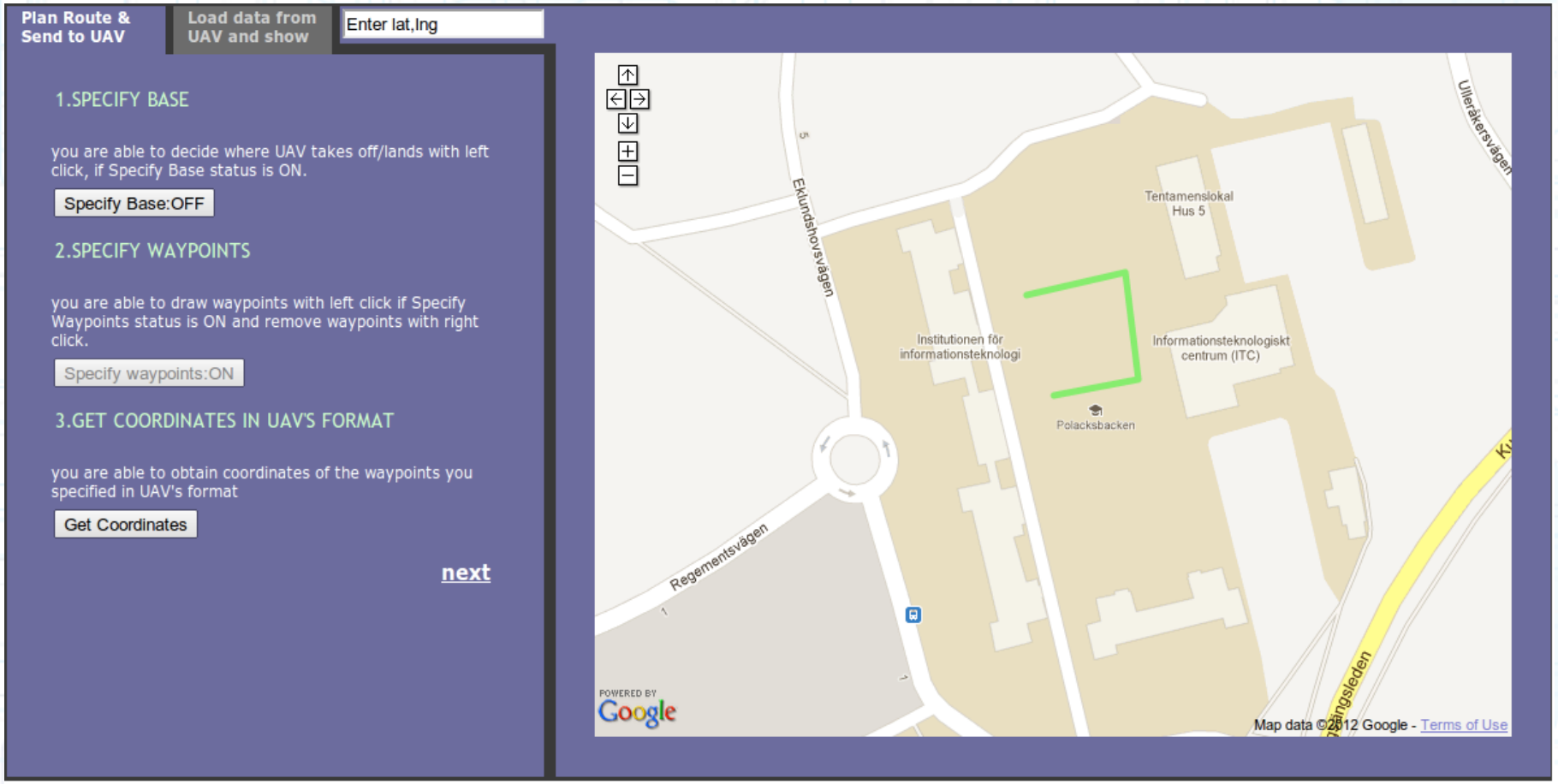




Waypoint generation

- Specify waypoints from Google maps
- Waypoints include
  - Latitude, longitude and altitude
  - Holdtime, tolerance radius, tag, event type
- Outputs xml file to put on the phones SD-card

WP generation interface



Waypoint acquisition

- Parse the XML file on the SD card
- Returns a list of waypoints

Serial communication

- Background thread that continiously communicate with the UAV
- Signals the main app when waypoint is reached
- Waits for signal from main app before sending new waypoint
- Returns when all waypoints are done

Camera

- Used to simulate data collection
- Triggered at every waypoint
- Automatically takes a picture and store on SD
- Signals to main app when picture is taken
- Instance is terminated after picture is stored

Main

- Reads waypoints from SD-card
- Initiates the serial communication
- Starts a thread for data acquisition
- Passes waypoints to serial comm thread
- Maintains synchronization between serial comm and data acquisition threads

Difficulties

- NaviCtrl software not open source
  - No change in behaviour
  - Harder debugging
- Hardware failure on UAV at demo time
  - Very unstable hovering
  - Sometimes GPS does not work properly

Result

- It recieves and flies the waypoints, and waits for data collection
- It triggers and collects data correctly
- Dirty pictures due to unstable hovering
- Movie time!
  - Three waypoints marked with bags
  - Too short delay at waypoints to be stable

Improvements

- Initialization of the serial comminication
- Automatic uploading of data at return
- Wireless startup of application
- Wireless user interface
- Integration with obstacle avoidance

Applications

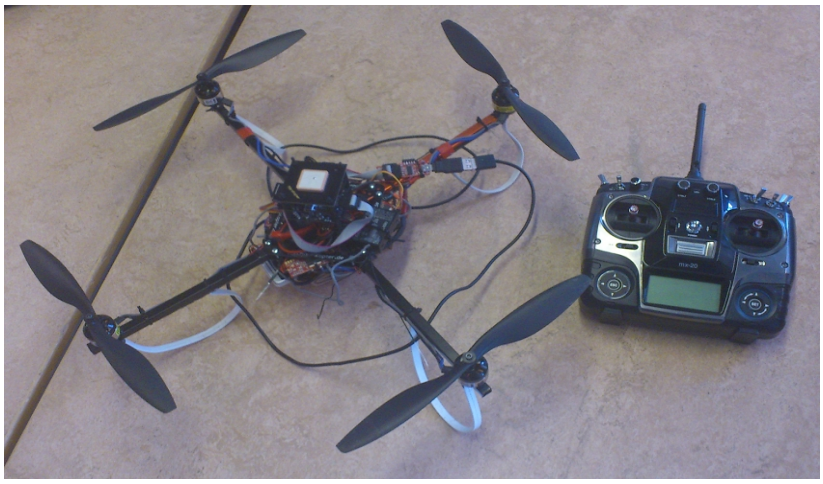
- Mapping of catastrophe areas
- Data collection at research sites
- Log data collection from off-shore power sites when communication fails
- Filming for TV news
- Follow wild animals with GPS trackers
- Crowd monitoring at festivals/demonstrations/events

Questions?

WCNES

Introduction

This project aims to navigate a UAV using the HTC Hero smartphone and at certain locations, collect a data sample. The UAV is a quadcopter from mikrokopter.de provided by the IT department. The purpose with this project is to prove if the concept is feasible for collecting data from remote sensors to research stations.



The MikroKopter comprises four brushless-DC motors each with its own speed controller. These communicate via I2 C bus to the central flight controller board. The flight control board version 2.1 is based on an Atmega 1284 processor running at 20 MHz which implements the state estimator, control loops, decodes the pulse stream from the radio control receiver, and also receives commands over a serial data port and transmits status information. The flight control board holds a triaxial MEMS accelerometer and gyroscope, and a barometric pressure sensor.

The navigation control board is connected to the flight control trough UART bus and can communicate data from GPS and compass and forwards commands. With this GPS-System you can use the function like PositionHold, ComingHome, CareFree, the Waypoint-flight as well as the fallowMe-function. The NaviCtrl-PCB has got besides the ARM9 Micro-Controller and the Micro-SD-Card-Socket two additional extension plugs that can be used for additional connections. The second UART connection on the NaviControl board is used to connect the UAV with the phone and send waypoints data in flight.

Initial conditions

- The UAV has built in functions that can provide hovering without any controlling of it
- The UAV can handle GPS waypoints send serially using the provided Koptertool
- The waypoints can be considered present on the phones SD-card
- The phone has a patched kernel that can work in USB master mode

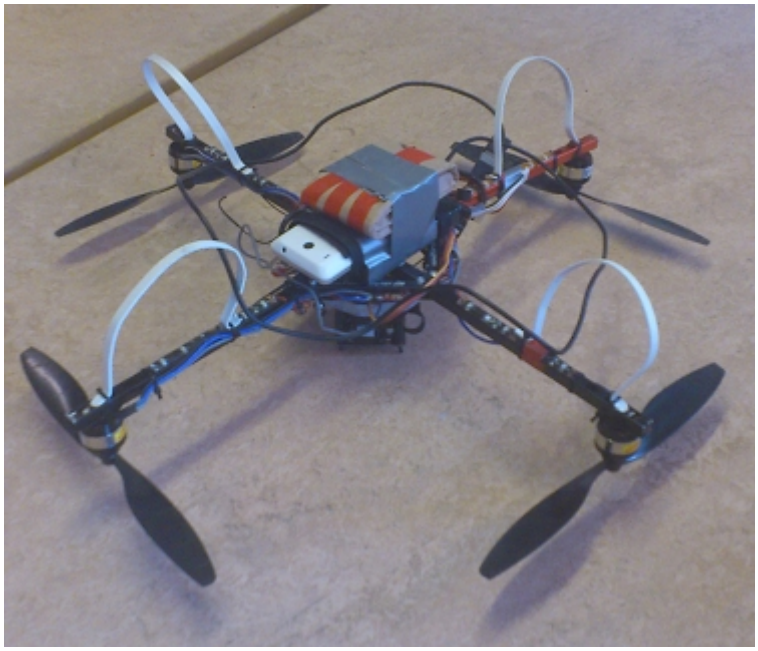
Milestones

- Establish communication between the phone and the UAV

- Navigate the UAV using GPS waypoints
- At every waypoint, collect and store a data sample

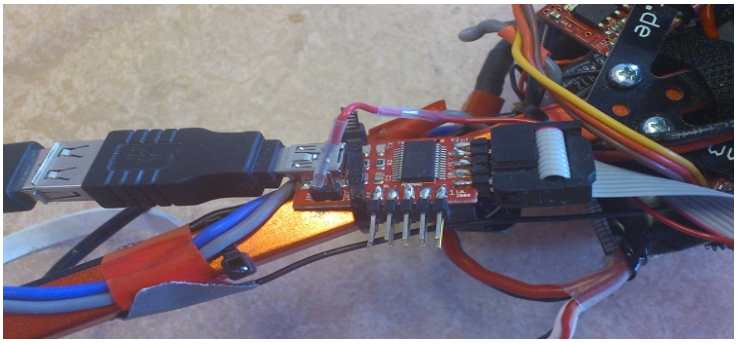
Hardware

The phone was mounted on the bottom side of the UAV, with a cover made of “wellpapp” to protect the phone in case of a crash. This turned out to be sturdy enough for our prototype flying, with the only drawback that the phone had to be removed from the UAV every time the UAV battery was changed.



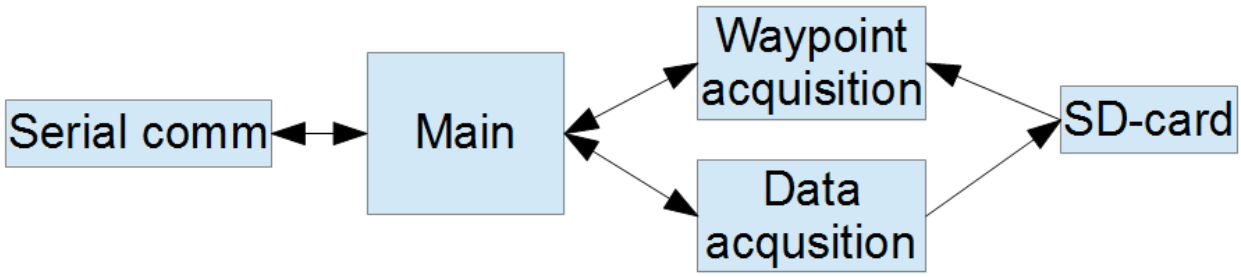
MKUSB

To convert the USB on the phone to UART which is used on the UAV an adapter called MKUSB was used. This is connected to the phone as a slave, with the phone in master mode. Since the phone can not provide power from the USB port the adapter is powered from the UAV:s 5V supply circuit. The picture shows the adapter mounted on the UAV, with the external power cable connected. Since there already is a ground wire present in the serial cable from the Navi control, there is no need for an external one.



The phone software

The software was written in three separate modules, the serial communication module, the data acquisition module and the waypoint acquisition module. These modules are knotted together in a main application. This makes it easier to adapt the software to be used with different kinds of data collection hardware and also with different UAV:s.



Data acquisition

Since the goal of the project was aimed at navigating the UAV, only a simulation of data collection was done. To achieve this in a manner as realistic as possible, the camera on the phone was used to take a picture at each waypoint. The application utilizing the camera was written to be invoked with the Android specific ”Intent” class. This means that it can be called from any main app and it can also return a result code when finished, making it possible to handle a lot of different failure scenarios as well as successful ones. The “Intent” class also allows for additional information to be included in the call as text strings. This makes it very easy to extend the software with more data acquisition scenarios.

When the camera applications is invoked, it displays what is called a preview, which is basically a stream of images from the camera shown on the screen. This can seem to be unimportant since the phone is mounted on the UAV where no one can see it, but is actually necessary for capturing the image, as the same feed is used by the android function used to take pictures, called takePicture.

Since the above mentioned “takePicture” function doesn't in any way synchronize with the feed, it is possible to execute the command before the feed is properly initialized. This causes all pictures to be black. To solve this issue, an AsyncTask is created after the preview is called that synchronizes with the preview, and the calls the takePicture function after an arbitrary delay. This not only solves the issue of synchronizing, it also provides an excellent simulation of data acquisition, since the delay can be varied, and thus making it more realistic.

Waypoint acquisition

It is explained in the end-user software part by issuing Java classes running on android. Please check.

Serial communication

At startup, this module initializes the serial communication with a series of terminal commands with super user rights on the phone. Then, a new thread is spawned where a continous link to the UAV is kept open, where messages is sent and recieved according to the protocol in ”NCSerialProtocol.java”. The first message to be sent is an invalid waypoint, which clears the UAVs ram of any residing waypionts. After that, the only message sent over the link is the new target waypoint struct every time the main app decides that the UAV should move. The UAV however, continiously sends messages over the serial link. Some of them are ignored, but the ones that are received and processed are:

- Waypoint
  - An 8 bit integer showing the number of waypoints in the UAVs ram
- Error message
  - Contains error messages from the UAV. Only used for debugging.
- NaviDataStruct
  - Contains all information about the current location, heading and state of the UAV. Especially interesting is the NC_FLAG_TARGET_REACHED flag which is true if the UAV has arrived at the target waypoint.
- WayPointStruct
  - Contains the total number of waypoints in the UAVs ram, which WP it is currently regarding as target WP along with that targets waypoint struct.
- EchoPattern
  - A test pattern to ensure that the link is still open.

Every time a waypoint is reached, the thread signals the main application and then waits for a signal back to send the next target waypoint struct.

The end-user software

The end-user software is a PHP(ajax) web application functioning for mainly two purposes with the help of Google maps:

1-To Create way-points with specified features and output it in a language the UAV understands.

First thing an end user shall do is to choose a way-point and enter appropriate data according to instructions of the end-user software. Also, end-user can (re-)specify normal way-points and the base way-point by using the button residing on the left side of menu and mouse clicks. End-user can delete way-points by right click and add way-points by left click on the google map.

After the end-user creates way-points according to entered data. Only thing end-user shall do is to click “Get Coordinates”. Then, a specific xml file is obtained as seen below.

```
<coordinates>
      <coordinate>
           <id>0</id>
           <lat>59.84065385614278</lat><lng>17.647411823272705</lng>
           <picture/>
           <data/>
           <altitude>5</altitude>
           <tag>picture</tag>
           <holdtime>30</holdtime>
           <toleranceradius>2<toleranceradius/>
       </coordinate>
        … n “coordinate” objects
</coordinates>
```

Root object, “coordinates” may contain as many “coordinate”s as end-user wants.
Each “coordinate” consists of properties stated below?

- Id is a unique number to identify the way-point and it grows incrementally as end-user adds

a new waypoint.

- Lat and Lng represent latitude and longitude of a way-point, respectively. In other words, X and Y coordinates.
- Altitude represents the altitude of a way point as it is understood from the property name. In other words, Z coordinate.
- Tag specifies what the UAV shall do when it reaches destination. For now, main android application just supports picture but end-user software supports any type of tag.
- Picture specifies the name of taken picture when the UAV reaches destination if “picture” is assigned to Tag.
- Data specifies the other format of data collected when the UAV reaches destination if “data” or any other name is assigned to Tag. This is intended for future use, such as collecting data from any wireless device staying on the ground.
- Hold-time specifies how long UAV will stay on the way-point in seconds.
- Tolerance radius specifies tolerable distance of the UAV from destination way-point in meters.

Note: picture and data property will be initially empty because this represent requests from end-user. However, after UAV collects some data or takes pictures, these properties will be filled accordingly.

Then, end-user stores store the XML file into the SD card and inserts it to the smartphone's SD card slot. Main application (through Parser.java in uav.xml.parserconstructor library) reads the XML file from the SD card and parses the way-points and convert them to way-point Java objects. Main application uses these objects and reaches destination points according to information provided by obtained objects.

2-After UAV finishes collecting some data according to the directions specified in the XML file, the main application converts(through Constructor.java in uav.xml.parserconstructor)way-point java objects to an XML file and saves it onto the SD card with taken pictures. Then, the SD card is inserted into any kind of computer's SD card slot and the XML file is uploaded with taken pictures into end-user software and then what user needs to do is just clicking way-points to get the collected information. Also, any number of XML files can be uploaded and managed (deleted) as wanted.

Future work and limitations

The initialization of the serial communication sometimes doesn't work after the UAV has been turned on and then of again. Since it somehow interfere with the data acquisition, it can't just be reinitialized but the application must be restarted. This should be further investigated and solved.

After the UAV has completed its path, and returns back to the base station, one has to manually get the SD-card out of the phone to extract the collected data. This could be done automatically by uploading the data to a server or similar at arrival using wifi, or even 3G to do it while flying between waypoints.

The application is it is now does not feature any user interface since it is always mounted on the UAV. This could be improved by adding a wireless interface, which could include functions such as:

- Start a collecting path
- Erase all data on SD-card

- Upload data from SD-card
- Display current waypoints

Depending on the nature of the data that is collected, some calculations could be performed on the phone while flying between waypoints since the phone is basically not used at all (except for receiving commands from the UAV), and the battery life of the phone vastly outperformes the battery life of the UAV.

Also, this is a typical implementation of location dependent data collection, and perhaps one of the algorithms developed by the other groups could be implemented. It should also be possible to generate waypoints depending on the location and the data received. For example, if flying around in a city with the purpose of finding and filming large crowds and at every road intersection there is a couple of sensors that counts the number of persons that passes by, the UAV could make intelligent decisions depending on these values, and set waypoints and places of interest accordingly.

If more than just prototype testing is to be done, a more permanent mounting solution has to be constructed. Preferably, one could integrate a mounting frame with a new landing gear, to provide more safety for the equipment and be able to change the battery without removing the phone.