Proceedings of the
1st International Conference and Exhibition on Future RFID Technologies
Eszterhazy Karoly University of Applied Sciences and
Bay Zoltán Nonprofit Ltd. for Applied Research
Eger, Hungary, November 5–7, 2014. pp. 23–34

doi: 10.17048/FutureRFID.1.2014.23

RFID from the air – Product localisation with RFID scanner, carried by autonomic drone, aided by visual markers*

István Fazekas, Péter Magyar, Tamás Gregus, Gábor Geda

Eszterhazy Karoly University of Applied Sciences
Institute of Mathematics and Informatics, Eger, Hungary
i.fazekas@agria.hu, magyarp@aries.ektf.hu,
gregtom6@gmail.com, gedag@aries.ektf.hu

Abstract

We investigate the possible applications of RFID technology. The focus of the examination is to define restrictions of RFID technologies, and to extend the potential application area by means of supporting hybrid technologies. Of all the difficulties it is the range what represents the hardest problem. A possible way to extend it, is to approximate the scanner to the tag with a robot, supported with image processing. The robot we chose is an in-door, unmanned aerial vehicle (drone, or quadrocopter). Hereby we would like to publish our experiences and observations so far.

Keywords: RFID, Drone, Contactless Card, UAV, Localisation, Hybrid technology

1. Introduction

RFID as stands for radio-frequency identification is a very promising technology in the range of nowadays numerous solutions of automatic identification. Not only pure recognition of an object, but also its tracking or even logging relevant data

*This work is connected to the RFID research project was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.2.C-11/1/KONV-2012-0014.

can be delivered with RFID. The tag itself can take many shapes, but as for our application it is a small size, light-weight sticker on the packaging.

The situation in our focus, is an extended warehouse, and the goal is to find the exact location of a specific product, fitted with an RFID tag, within this warehouse.

To retrieve information stored in the tags, a scanner is required. The scanners' reading principle varies depending on parameters of the technology and the operation circumstances given, however typically, cost effective RFID tags and scanners are fitted with digital communication interfaces hiding the overlaying deterministic beam characteristic. Therefore with these devices localisation cannot be implemented by measurement of analogue values. Sensitivity range of a scanner–tag combination is a well specified value, ranging from a few centimetres to as much as kilometres, but without known beam characteristic, there is no point to use long these scanners for localization.

As a conclusion, the solution we have chosen is small range RFID carried by robots. The very location of the tag is indicated by visual tags, and an RFID tag is brought in range of the scanner of the robot with processing the visual information by means of its camera.

In the first phase of the project the aim has been to set up a development environment to allow of team work.

2. UAV – Unmanned Aerial Vehicle

Land robots have been developed for a long time, but because running gear is a hugely problematic mechanism, we chose unmanned aerial vehicles (UAV).

Thanks to recent developments aerial vehicles are not expensive high-end technologies anymore. Very simple, cost effective, yet durable, off-the-shelf models are available. The remote controlled versions of these flying objects are often referred as drones these days.

Among hobbyists, fixed wing aeroplanes have been known for long. The simple structure can be easily built in small size. A single motor drive, and a relatively simple steering mechanism does not require complicated control procedures, so altogether it is easy to construct and fly. However, the up thrust on the wings is generated by the horizontal speed of the airflow, therefore indoor manoeuvring (e.g. in a warehouse) is most problematic.

Recently, several versions of rotorcraft machines were developed. Although steering is far more complicated with rotors, development of microcontrollers allowed to construct highly responsive, still cheep electronic control units. With the rotating wings up thrust is not generated by the speed of the aircraft, but the rotors, therefore rotor driven planes has good manoeuvring characteristic on low speed which is ideal for indoor applications.

Not all rotorcraft machines are this simple. The steering of double rotor crafts probably require even more complexity, namely, the torque of the spinning horizontal rotor tends to turn the body of the aircraft in the opposite direction, which is compensated by a vertical rotor. Moreover, the horizontal thrust on the ro-

tors is provided by cyclic rotation of the rotor blades' pitch. Fairly complicated mechanisms both to construct and to fly.

Latest development of 3, 4 or even 6 rotor units was the first helicopters with a simple mechanics. The rotation or velocity of the unit on one selected axes is generated by slim differences of the opposite side rotor speeds, keeping balance in the rest of directions with compensation of the other rotors. This is probably the most promising technology these days.

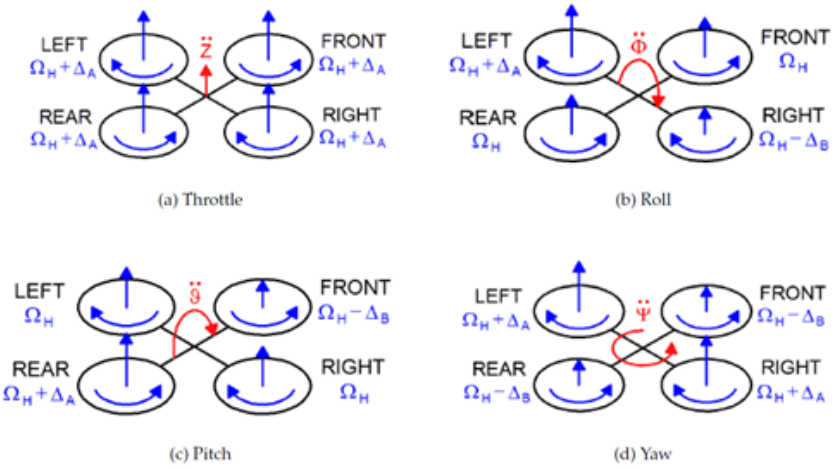

Figure 1: Basics of navigation)

3. Drone types

Several models of different operating principles, developers and manufacturers were examined. Most promising of all were the aircrafts developed on Óbuda University. However these models have been in experimental phase, reproduction requires the developers' skill level. It may result in lost of focus now, but these machines will most likely be the long term solution.

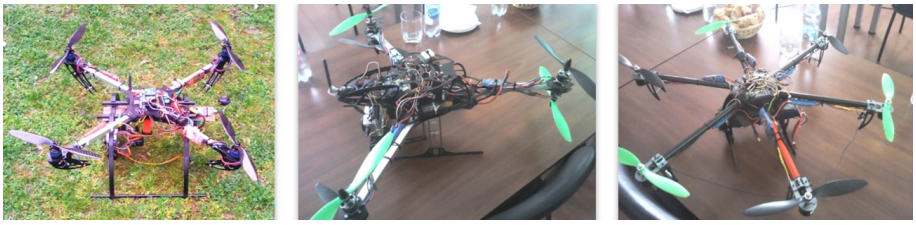

Figure 2: Drones of Óbuda University

Another type of a likely choice was DJI Phantom. Robust, stable, and with a high resolution camera it delivers the strictest specification. However the system of this drone is rather closed. There are neither public development tools nor API specification on the market. (Note: by the time of this article DJI announced to make an SDK available for outsiders.)

Figure 3: DJI Phantom

The best choice at present is AR Drone of French developer and manufacturer Parrot. Low cost, still stern mechanics. This model does not match the tough physical parameters of DJI Phantom, but the price difference is significant too. It is also important, that the system of AR Drone is an open embedded GNU-Linux version, with a precise API definition for both the SDK and the on board system.

Figure 4: AR Drone of Parrot

4. System components

The main units of the complex system are the land station, and the aerial unit. The aerial unit runs the on board system providing

- flight control
- IO system with on board sensors
- communication with the land station
- image capturing.

Where else the land station's responsibilities are

- flight instructions
- navigation
- tracking
- image processing
- database.

The development environment has to be able to write applications for both components, preferably on the same programming language.

5. Development environment

The above task means the development environment has to contain the following functions:

Integrated development environment (IDE)

- code editor,
- debugger,
- build system,
- download and run mechanism.

Software development kit (SDK)

- build time libraries ,
- runtime libraries.

6. IDE

As for the IDE we raised the following expectations:

- Possibly open source
- Configurable user interface
- Efficient resource management (C or C++)
- Long time support
- Platform independency
- Synchronicity with education.

Two main candidates seemed to fulfil the above requirements Eclipse with CDT plugin, and Qt. Both are multiplatform (OSX, GNU-Linux, Windows) with GCC as the C++ compiler engine. Also both environments are good in cross-compilation aided with device definition features. Plugin system gives extra functionality. Support and documentation is equally excellent.

7. Eclipse with CDT

- Pros: robust, very versatile system, with an open architecture.
- Cos: the plugin based system means non-uniform handling of functions. The ad-hoc structure carries huge overhead. It (and probably the Java programming language it is written in) results slow reactions on user controls, as well as a rather confusing user interface.

Graphic user interface is delivered with outsider class libraries. The most popular of these is GTK, however V3 was not supported in Windows at the time this article was being written.

8. Qt

- Pros: its IDE, Qt Creator is very fast, simple and logical. Written in C++ speed is impressive. It provides every expected function in a compact and neat way. Not only has it contained an IDE, but also a huge and consistent Q class library covering all from GUI to network handling.
- Cos: in comparison with Eclipse, Qt has a lot more rigid configuration when it comes to devices (kits). However configuration is a one time task, only to be completed prior the development process. Contrarily, the payback of the consistent class structure will be remarkable at a later stage.
- Conclusion: the IDE to be used is Qt Creator, and the SDK is to be developed as a library of Qt classes.

9. SDK

We evaluated available software development kits. We found several ones, just to find that each one of them is incomplete, or development has been stationary for too long, thus long time support cannot be trusted.

- ARDrone SDK: delivers nearly all requirements, but unfortunately the common opinion is that it is far too complicated to use, with numerous errors. Documentation is not satisfactory at all.
- Javadrone SDK: poor resource management.
- ADC (Open Drone Control): Scala programming language, not among the preferred languages.
- Python ARDrone: development was most likely stopped.
- Argus: Ruby: development was most likely stopped.
- Node-AR-Drone: javascript, poor resource management.

All the above systems conceal an even bigger problem: platform independency cannot be guaranteed using any SDK of a vendor specific product.

10. System hierarchy

The software system of the drone as delivered by the factory consists of two components. The drone's on-board flight control, and the remote control unit – for the AR Drone factory package it is some sort of Android, or iOS mobile equipment.

In our application the mobile remote control unit is entirely abandoned. The reason is, on the one hand we want computer controlled autonomic drone operation, on the other hand the concept of platform independency is corrupted if we would involve such element. Any vendor specific element should be closed in the bottom layer of the system, realising driver functionality. Later, mobile equipment's can be used, but rather than direct access to the drone, these will connect solely to the server to provide an extra user interface.

Figure 4 shows the overall system plan. The core of the drone's hardware is an ARM based unit, running embedded GNU-Linux operating system. The functionality is provided by an application that starts automatically after the boot process. This application delivers the sensor interface and the on-board fight control. Connection with the remote control is maintained through wifi connection, with the drone playing the role of the access point. The development host computer can connect to this wifi hotspot on a dynamic ip address, and can communicate with the on-board system on dedicated ports.

Since we need to add some extra functionality, namely RFID scanner, GPS or extra sensors that may not be factory installed on the drone (e.g. side and top

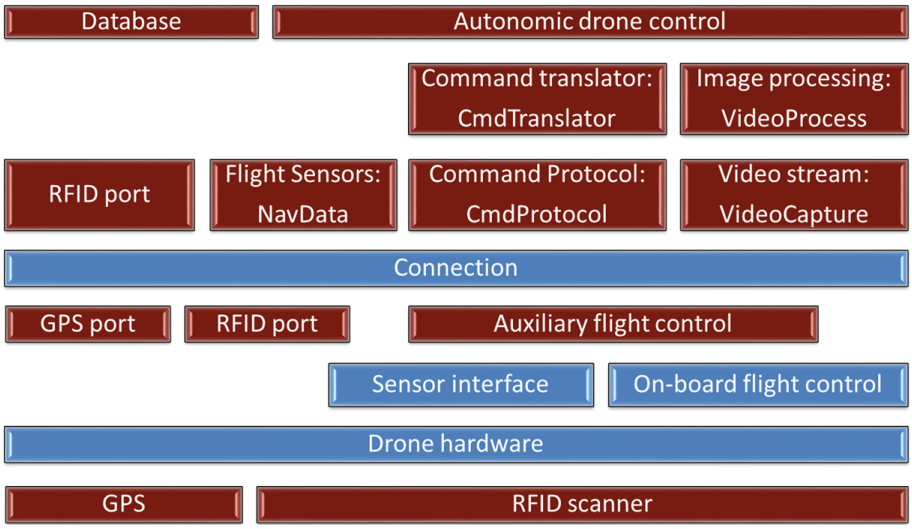

Figure 5: System hierarchy

proximity sensors) access is required to the control OS. Alternatively a single board computer can be fitted, to provide the extra functionality, however it will add extra weight load, which will reduce battery life.

Since the AR Drone has extra serial interface, and the os is open with a telnet daemon, it is possible to install our own RFID scanner on it, and run our own application to handle it.

The way how our system sits on top of the drone's on-board factory software is as follows: below the connection layer we need an application to provide GPS port handler to transmit localisation data, RFID port handler to transmit data from the scanner, and Auxiliary flight control for e.g. emergency proximity sensors. For this extra communication the on-board application opens its own ports.

Above the connection layer rest the services of the land station application. This layer is vendor dependent, and its aim is to cover the specialities of the drone towards the higher layers.

- RFID port is to receive data of the on-board scanner. It is basically a string of ASCII and some control characters presently.
- Flight sensors receive nav. data. The drone's own sensors supply a raw flow of its local position (height, orientation, etc.) which can be used by the auxiliary flight control. This layer has to translate the raw stream to our own specification.
- Command protocol translates and transmits the station's commands to the drone. It also handles encryption, error check, keep-alive mechanisms if there are any required by the drone's own protocol.

- Video stream receives the data stream of the on board camera. In case of AR Drone it is a non TI standard h264 container. This layer will decompress it and breaks it down to frames, stored in matrixes.

The layer above the vendor specific one should be entirely independent of the drone's implementation.

- Cmd Interpreter contains the command set of the system.

- Image processing provides shape recognition. The output of this module is position data (e.g. relative coordinates on the Cartesian system of the canvas) of the recognised visual markers of the RFID. As mentioned, these markers can be special graphic elements, e.g. coloured dots In a specific pattern, or it can also be represented by the shape of the box, e.g. geometric centre.

The highest level of functionality is the Autonomic flight control, and the database to serve it. This is the level, where highest level route planning and seek strategies run.

10.1. System blocks already completed

Work on some of the SDK is in its early stage of development. The following blocks have been completed to an alpha test status:

10.2. Test framework – Desktop

Written in Qt C++, with an easy to modify GUI, it is a flexible tool to assemble and test the building blocks.

10.3. RFID port – Desktop

It opens a tcp/ip socket to the on-board RFID port handler, and passes RFID scan data to – at present – the test framework.

10.4. RFID port – On board

Opens a serial (TTL polarity RS232) port and a tcp/ip port listening to the land station to initiate connection. In case land station (client) disconnects, the application will reopen the port.

10.5. Auxiliary flight control – On board

At present in lack of extra sensors it has not provided any functionality yet.

10.6. Video stream

Desktop application. To capture video streams of multiple formats, a flexible and versatile video stream converter library is needed. The obvious choice was avlib, who's history has been parallel with the development of video formats. After some early tests with some demo applications built around avlib (like ffplay or ffmpeg) it became apparent, that it would be compatible with the non-standard video format of AR Drone too. Avlib delivers the video-frame conversion in 7 sequential steps:

- register codec library
- open file and retrieve header
- read streams information in video
- select requested codec
- allocate buffer
- load frames
- convert frames to RGB matrices.

10.7. Image processing

As for the visual markers we chose coloured disks for the test of the SDK. The Qt desktop framework gets QImage format matrices from the image processing block. The operation is based on OpenCV, an open-source, cross platform library, developed by Intel. Based on a very simple matric based data structure, OpenCV provides real time, frame by frame video processing. It was written in C, later converted to C++ resulting an even faster and more flexible implementation. In complexity it covers the full range of image processing from image filtering to face recognition.

In the test application we used some basic filters, like greyscale, canny edge detection, and hough line transform. The compatibility with the Qt framework is more than satisfactory.

11. Conclusion

The basic building blocks of the development environment have been laid down. The IDE and the SDK represents a solid base to go on with work on more sophisticated areas. The long term strategy is presented, but on top of the project road map, already we have met some urgent tasks requesting immediate evaluation: At the moment the reading range of the ID2 scanner is about 50-70 mm with an outside antenna. The specification claims it about 200mm, and although workbench tests has not confirmed it in full extend yet, we measured 130 mm.

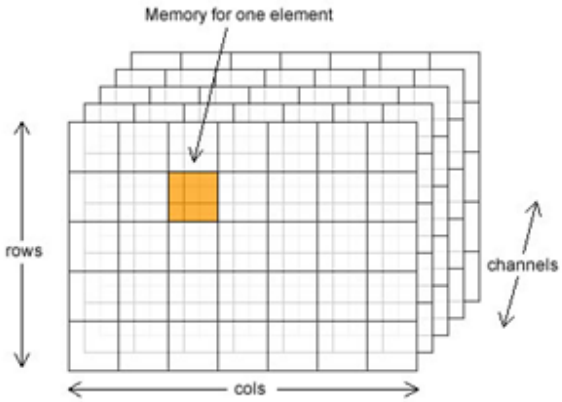

Figure 6: OpenCV data structure

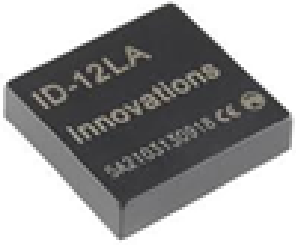

Figure 7: ID2 scanner

Fitted under the drone's belly however the antenna gets right under the PCB of the drone electronic, which consumes some of the electromagnetic energy. It is to be checked, how could the antenna be fitted in a more convenient position.

Figure 8: AR Drone with RFID antenna fitted

Emergency sensors. The Drone cannot detect horizontal proximity of objects, therefore flight often ends with an emergency landing. Another problem may occur when in-door operation the drone hits the ceiling and gets stuck to it due to the vacuum-cleaner effect. With additional infrared and ultrasonic sensors and

auxiliary flight control these accidents could be avoided.

Image capture and process on board the drone. Latency of the complex image conversion and analysis procedure can grew as long as seconds. It means flight control is well behind the events. Therefore it is necessary to analyse how could the image processing be made faster (there is a well known delay in avlib, which could be decreased). Alternatively it should be examined, how this process could be moved into the on-board system component.

References

[1] Piskorski, S. Brulez, N. Eline, P. D'Haeyer, F. AR.Drone Developer Guide (SDK 2.0), *Parrot* (2012)

[2] ID SERIES DATASHEET (ID2/12/20/2WR/12WR) *ID Innovations*, (2007) `http://www.id-innovations.com/httpdocs/EM%20moudule%20SERIES%202007-10-9_wfinal%20v22.pdf`

[3] Inkyu, H. Inkyu, V. Technical documentation for AR.Drone project, *Cyphy Lab. Queensland University of Technology*, (2013)

[4] Daugaard, M. Thyregod, T.V. Navigation for Robots with WIFI and CV, (2014) `http://taghof.github.io/Navigation-for-Robots-with-WIFI-and-CV`

[5] Varga, P. Qt strandkönyv, *E-Közigazgatási Szabad Szoftver Kompetencia Központ*, (2013)

[6] Rischpater, R. Application Development with Qt Creator *Packt Publishing*, (2013)

[7] Qt support, *Digia Plc*, (2014) `http://doc.qt.io/qt-5/index.html`