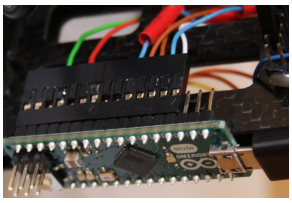
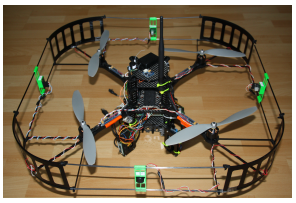
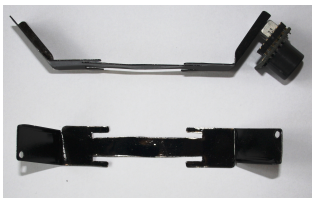
Excel@FIT 2015
http://excel.fit.vutbr.cz

Multicopter collision avoidance system

Radim Hrazdil*

Abstract
There are a lot of people who like making models of planes, helicopters or multicopters, but not all of them can actually fly them. This article presents (HW/SW) system that prevents user from crashing into obstacles and provides convenient way to control any UAV (Unmanned Aerial Vehicle).
The main target are vehicles equipped with Robotic operating system (ROS), however the device is designed to be easy to use for another platforms as well and because it is intended to be used on aerial vehicles, it is also designed to be light, extensible and energy-efficient. To achieve this I used ultrasonic sonars mounted on the UAV and connected to the Arduino.
This project results in sonar-based distance measurement device with collision avoidance system ins ROS which is a starting point for further development of safety mechanisms and anti-crash user interface, for example using fuzzy logic to avoid collisions.
Main contribution of this work is design and realization of a measuring device using Arduino for controlling ultrasonic sensors. Thanks to Arduino, the device can be used independently on platform. ROS packages can be used by other developers as a starting point for their projects.

Keywords: Multicopter — Quadcopter — UAV — ROS — Ultrasonic Sonar — Collision avoidance

Supplementary Material: Demonstration Video — Downloadable Code

*xhrazd13@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Multicopter market is growing quickly as multicopters are becoming cheaper, smaller and more popular. Of course, multirotor aircrafts would not be so popular if there were no applications for them. Multicopters proved themselves to be of a good use in many tasks – aerial filmmaking, aerial photography, industrial oversight, package delivering, military surveillance and many others.

But flying a multicopter is not that easy and it takes hours and hours of training. If I could make multicopters easier to fly, my solution could save a significant amount of money for companies starting in multicopter business who need to pay employees their training hours.

During flight operation, pilots are normally limited in determining accurate position of their UAV and objects around it. Cameras field-of-view may not be sufficient to see obstacles from sides of the vehicle, so unexpected gust of wind may result in crash. There is also possibility of losing camera signal, when flying out of reach. Flying without camera can be confusing as well, because in most cases it is difficult to distinguish front from rear part. Another thing to consider is flying in a space limited places like buildings, halls or tunnels, which is not trivial task even for an experienced pilot. With our system installed, the pilot can fully focus on the mission itself, rather than precise flying.

Similar project was developed at the University of Würzburg in Germany [1]. In this project 12 SRF02 ultrasonic sonars were installed around the quadcopter to cover 360 degree circle. As a result, overlapping beams of adjoining sensors allowed detecting obstacles with higher resolution. One of the major issues of the chosen sensor distribution was that ultrasonic sensors disturb each other [2]. This had to be solved by dividing sonars into three groups. However, disadvantage of this solution is that not all the space around the quadcopter is covered. Under certain circumstances, these blind spots could cause problems. For reading sonar measurements was used a separate microcontroller.

Collision avoidance module divides the space around a quadcopter into three zones in each direction and its behavior can be described as a finale state machine. Each direction has a state that defines how fast the quadcopter can move in that particular direction.

- State 1 – safe zone.
- State 2 – close zone.
- State 3 – dangerous zone.

In close zone, speed toward the obstacle is limited depending on the current distance. After reaching the dangerous zone, PID controller is activated. PID controller ensures no further approach towards the obstacle.

Experiments have shown that under good conditions, which means straight surfaces around the quadcopter, the system was capable of reproducing surrounding environment with a deviation of just about few centimeters. The system was also almost unaffected by smoke, which would cause optical sensors to fail. However, the system lacks on detecting soft or foamed material.

In conclusion, to ensure maximum possible reliability, multiple types of sensors must be used.

In this paper, I propose array of ultrasonic sonars to cover as much space as possible with sufficient accuracy of measuring surrounding obstacles. I focused on achieving low hardware requirements, low weight of the whole system (including wires, sonar holders, etc) and fast response time.

The target is to mount up to 14 ultrasonic sonars to create something like a protective shield around the quadcopter. For reading sonars measurements, I chose Arduino Micro, which is very small and light, yet capable of powering up to 5 sonars at once.

Contribution of this project is hardware and software design of a device, capable of measuring distance to obstacles with frequency about $5Hz$. Along with the device, I developed modules for integration into ROS framework, measuring algorithm for Arduino and simple GUI application for sonar functionality check. I assembled prototype of designed device (Its detailed documentation is in progress) with weight less than $100g$. As a power supply, USB is fully sufficient with $500mA$ limit. Thanks to Arduino, designed solution is platform-independent and can be used for example on light, small and relatively cheap Raspberry Pi.

This project has laid the groundwork for future improvements and development – implementing fuzzy logic controller or fusing sonar and visual odometry data for better positioning.

2. Technologies

Robot Operating System

ROS is a framework for writing robot software [3]. It is a collection of libraries, tools, utilities and conventions with one aim – simplify development of complex robot behavior across platforms. Software in ROS is designed and developed as a collection of small, mostly independent programs called nodes. All nodes run at the same time and all nodes can communicate with each other. The concept is very similar to object oriented programming. Consider a group of nodes, each working on very simple task, but they all together solve some complex problem.

Ultrasonic sensor

Sonar measures distance by measuring time of a $42kHz$ acoustic sound wave transmitted to and reflected from an obstacle. Therefore, as long as an object is capable of reflecting sound, neither its color, transparency or any other visual characteristics matters. The minimum distance sonars can measure is about $30mm$ and maximum about $10m$, but parameters of each model are different.

For the developed application, I chose Maxbotix MB1220 XL-MaxSonar®[1]. It offers the best balance between wide or narrow beams and provides a good balance between small and large target detection. It also offers high noise tolerance, which is important for UAV applications. Maximum range is $765cm$ and minimum range is $20cm$. Closer or farther distances are reported as maximal, respectively minimal so there is virtually no dead zone.

Arduino Micro

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino Micro board is based on the ATmega32u4, developed

[1] http://www.maxbotix.com/documents/XL-MaxSonar-EZ_Datasheet.pdf

[1] http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf

in conjunction with Adafruit. It offers 20 digital and 6 analog input/output pins, which is enough to connect required number of sensors. Micro can be powered via micro-USB, which is also very easy to use for communication with computer thanks to built-in serial library.

3. Hardware and Software Design

Sensor wiring

Because I want to be able to control up to 14 ultrasonic sensors, I have to design wiring that wouldn't require more pins than are available on the Arduino Micro. At the same time, it has to allow as high refresh rate as possible.

Two methods were combined to save pins and achieve highest frequency rate – chaining and starting multiple sensors at once. Chaining feature of Maxbotix sonars allows to start only the first sonar. The following sonar is started by the previous one after it finishes ranging procedure. As front, rear and side sensors shouldn't disturb each other, I decided to start them all at once and read their results from analog output, which can be done very quickly.

Sensors are started by 20*us* pulse to the pin 4 (blue and brown wires in the figure 1). Analog voltage output from sensor is on pin 3 and pulse width representation is on pin 2 (green wires). Chaining is accomplished by connecting pin 5 of first sensor to pin 4 of following sensor.

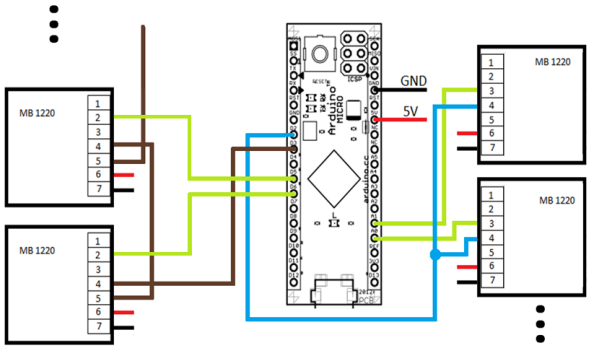


Figure 1. Sensor wiring diagram.

Sensor distribution affects two important parameters – refresh rate (because of potential disruptions) and weight. I created a sensor beam model (figure 2) in Autodesk Autocad to visualize space covered by sensors and based on the model, I designed sensor distribution covering about 90% space around the quadcopter. Author of the quadcopter model is Daniel Lucena, who published it on GrabCad.com for free use[2]. Model of quadcopter with mounted sensors is available in supplementary material.

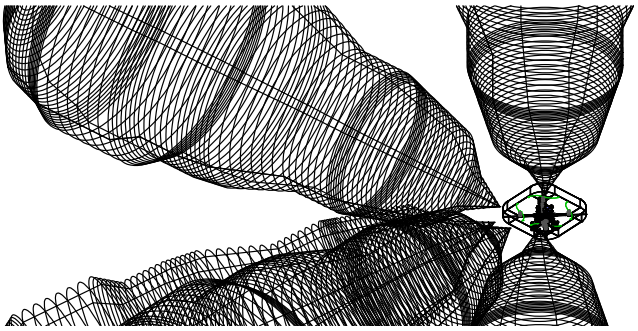


Figure 2. Quadcopter and sonar beam model for inspecting coverage and finding best sonar distribution.

Application Architecture

Application package in ROS could composed of three main nodes, as shown in the figure 3. One node for reading data from Arduino, parsing it and publishing parsed data to a sonar_data topic. The second node for controlling the quadcopter, allowing to publish control commands to a special topic where they can be modified instead of directly sending them to fcu/control topic. For that purpose there is the last node, which is subscribed to both topics and fuses sensor data with user commands. Depending on sensor readings it modifies user commands and these modified commands sends to fcu/control, which is a topic for publishing control messages.

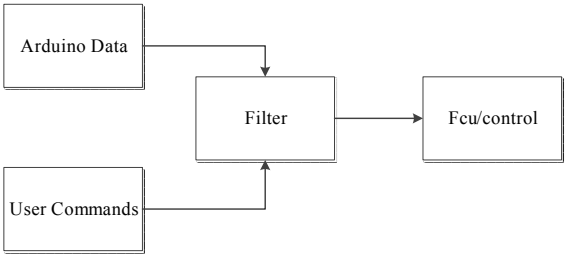


Figure 3. Application nodes.

4. Implementation

Reading sensor data

Arduino Wiring language[3] is quite simple and similar to C. There is a function analogRead(pin) for reading voltage on one of six analog pins. This function implicitly maps input voltage between 0 and 5 volts and returns an integer value, which in our case represents centimeters. For reading pulse width on digital pins, there is a function pulseIn(pin, value), where value is the type of the pulse to read, either HIGH or LOW. The function returns length of the pulse in

[2] https://grabcad.com/library/asctec-pelican-quadrotor-1

[3] http://arduino.cc/en/reference/homePage

Measured distance	Digital result	Analog result
20	25	19
50	62	44
80	100	71
110	139	98
140	178	126
170	217	156
190	243	174

Table 1. Table of analog and digital measurements before any calibration.

microseconds, so I need to divide the returned value by 58[4] to convert it to centimeters.

Calibration

After conducting several experiments, it turned out results obtained by analog and digital readings differ very significantly and neither of them are accurate. A set of 20 measurements was taken by analog and digital reading and entered into a table. From the table 1 is obvious that without any calibration, this system would be hardly usable.

Linear error is possible to model by simple linear equation 1, where y is desired correct distance, x is measured distance and a, b are real numbers.

$$y = ax + b \tag{1}$$

From graph in the figure 4 is obvious that the error is linearly dependent on measured distance and therefore with measured values I can calculate coefficients a, b for both digital and analog function. For calculating mentioned parameters I used linear regression to calculate slope of the analog and digital function.

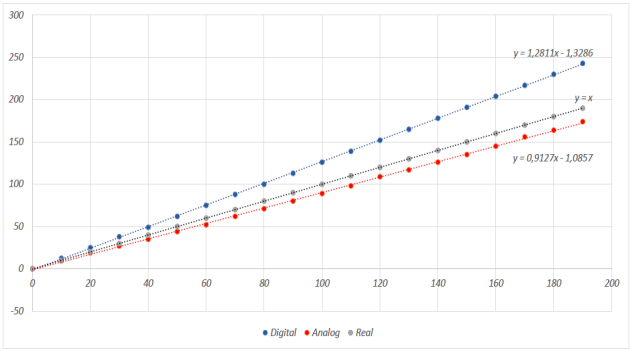


Figure 4. Graph representing error of measured values by digital and analog reading.

With known slope, I get coefficients a, b and I can derive inverse functions, to get the final expression for calculating the calibrated range.

For an example, lets assume measuring from digitally connected sensor. The sensor reports distance

[4] http://www.maxbotix.com/articles/033.htm

	Parameter a	Parameter b
Digital	1.2811	1.3286
Analog	0.9127	1.0857

Table 2. Table of calculated coefficients for reading correction.

$100cm$.

$$y = \frac{100 + 1.3286}{1.2811} \doteq 79.1 \tag{2}$$

The result of equation 2 corresponds with value 80 from the table 1, which is the distance we actually measured. As sensors have resolution of $1cm$, I consider this result to be accurate.

During experiments I observed certain instability between individual measurements. Arduino was reading accurate values but in one of few cases, it returned number bigger than the operational range of the sensor. Most probable cause of this issue were voltage drops, because in my design, four sensors are initiated at the same time. Each sonar has a current peak $100mA$. To avoid voltage drops, I need low-pass filter [4], that would eliminate high frequency noise. One simple low-pass filter consists of a resistor in series with the load and a capacitor in parallel with the load. Parameters recommended by Maxbotix[5] are 10Ω resistor and $100uF$ capacitor.

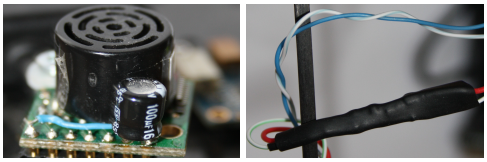


Figure 5. Soldered capacitor on sensor and in heat shrink tube.

ROS Nodes

ArduinoIO, a node for reading Arduino messages, reads data from USB port (/dev/ttyACM0) in a cycle, while roscore is up and running. When the node successfully reads a string of data from Arduino, it parses it and for each individual measurement publishes message containing two integers – sensor number and the measured distance. Messages are published on sonar_data topic.

For controlling quadcopter with keyboard, I used node written by Adam Crha [6], but redirected messages to be published on keyboard_control.

The most important node is the Filter. It is subscribed to both topics, sonar_data and keyboard_control.

[5] http://www.maxbotix.com/images/XL-Filter.jpg

[6] http://merlin.fit.vutbr.cz/wiki/index.php/Beran-Crha

When a new message comes from sonar_data, it updates the distance in direction defined by the sonar number. Also updates the state of this direction depending on the reported distance. When a new message comes from keyboard_control, speed is lowered based on the state of the desired direction.

5. Conclusions

This paper described hardware and software design of a measuring device and implementation of an anti-crash package for Robotic operating system.

Designed device can control up to 14 sensors with high frequency of measuring. Under good conditions, sensors can measure with deviation 1*cm*. Achieved coverage of space around the UAV is about 90%.

The main contribution of this work is design and realization of a measuring device using Arduino for controlling sensors. Thanks to Arduino, the device can be used independently on platform. I implemented of an anti-crash package for the most widespread robot programming framework nowadays – ROS.

The anti-crash package works independently on the type of used sensors, so if any other team wants to start developing with optical type of sensors, they can use this package as a starting point to speed up the development.

In the future, I am going to implement advanced controller for positioning the UAV utilizing PID controller or a fuzzy regulator, which would allow to set parameters like weight or speed for specific model.

Acknowledgements

The author thanks Ing. Vítězslav Beran, Ph.D. for his supervision and guidance.

References

[1] Sergio Montenegro Nils Gageik, Thilo Müller. Obstacle detection and collision avoidance using ultrasonic distance sensors foranautonomous quadrocopter. Technical report, University of Würzburg, Aerospace Information Technology(Germany), September 2012. [Online; Accessed 03-April-2015]. Available at http://www8.informatik.uni-wuerzburg.de/fileadmin/10030800/user_upload/quadcopter/Paper/Gageik_Mueller_Montenegro_2012_OBSTACLE_DETECTION_AND_COLLISION_AVOIDANCE_USING_ULTRASONIC_DISTANCE_SENSORS_FOR_AN_AUTONOMOUS_QUADROCOPTER.pdf.

[2] Bob Gross. Maxsonar operation on a multicopter. online, February 2013. [Online; Accessed 03-April-2015]. Available at http://www.maxbotix.com/articles/067.htm.

[3] Jason M. O'Kane. *A Gentle Introduction to ROS*. Independently published, October 2013. Available at http://www.cse.sc.edu/~jokane/agitr/.

[4] Wikipedia. Low-pass filter — Wikipedia, the free encyclopedia, 2013. [Online; accessed 01-April-2015]. Available at http://http://en.wikipedia.org/wiki/Low-pass_filter.