Obstacle avoidance and altitude control for autonomous UAV

Jakob Carlsson Stenström, Marcus Rodén

Degree Project in Electrical Engineering, 15 credits

Halmstad 2018-05-28
Abstract

Drones or UAVs are quickly becoming a bigger part of today’s society. Delivery services and transportation are fields were big development is being done. For the UAVs to be able to perform its given tasks safely more and more sensors are implemented.

This report covers the development and implementation of a sensor system to help an UAV to keep a fixed altitude and provide proximity measurements of the environment to avoid obstacles. The system is build around the ATmega328P microprocessor and uses I2C to communicate with the sensors. Measurements are filtered and published into ROS where the autopilot can reach the measurements and make decisions based on the readings. Additionally, simple algorithms to avoid obstacles have been implemented and simulated in the simulation software Gazebo. The altitude control system which has been the main focus of the project has been implemented with good results in both simulation and real flight tests. The system will be used in a competition held in Arizona, USA where the project team together with two other project will compete in the prestigious CPS-challenge.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Goal</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Delimitations</td>
<td>1</td>
</tr>
<tr>
<td>1.4 Requirements</td>
<td>2</td>
</tr>
<tr>
<td>1.5 Main Questions</td>
<td>2</td>
</tr>
<tr>
<td>2 Background and theory</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Related work</td>
<td>3</td>
</tr>
<tr>
<td>2.1.1 Proximity sensing</td>
<td>3</td>
</tr>
<tr>
<td>2.1.2 Obstacle avoidance</td>
<td>3</td>
</tr>
<tr>
<td>2.1.3 Altitude Control</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Theory</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1 Sensing</td>
<td>5</td>
</tr>
<tr>
<td>2.2.2 Filtering</td>
<td>6</td>
</tr>
<tr>
<td>2.2.3 Software</td>
<td>7</td>
</tr>
<tr>
<td>2.2.4 Hardware</td>
<td>8</td>
</tr>
<tr>
<td>3 Methodology</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Hardware</td>
<td>11</td>
</tr>
<tr>
<td>3.1.1 Sensors</td>
<td>11</td>
</tr>
<tr>
<td>3.1.2 Microcontroller</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Software</td>
<td>15</td>
</tr>
<tr>
<td>3.2.1 Modes</td>
<td>15</td>
</tr>
<tr>
<td>3.2.2 Motion Control</td>
<td>15</td>
</tr>
<tr>
<td>3.2.3 UAV Sensecontroller</td>
<td>16</td>
</tr>
<tr>
<td>3.2.4 Altitude</td>
<td>16</td>
</tr>
<tr>
<td>3.2.5 Obstacle avoidance</td>
<td>18</td>
</tr>
<tr>
<td>3.2.6 Contingency</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Result analysis</td>
<td>22</td>
</tr>
<tr>
<td>4 Implementation</td>
<td>25</td>
</tr>
<tr>
<td>4.1 ROS Action</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Vehicle platform</td>
<td>25</td>
</tr>
<tr>
<td>4.3 Mounts and placements</td>
<td>27</td>
</tr>
<tr>
<td>4.4 Reading the published data</td>
<td>28</td>
</tr>
<tr>
<td>4.5 From micro controller to flight commands</td>
<td>31</td>
</tr>
<tr>
<td>5 Result</td>
<td>33</td>
</tr>
<tr>
<td>5.1 Altitude</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Obstacle avoidance</td>
<td>38</td>
</tr>
<tr>
<td>5.3 Contingency plan</td>
<td>41</td>
</tr>
<tr>
<td>6 Discussion</td>
<td>43</td>
</tr>
<tr>
<td>6.1 Altitude</td>
<td>43</td>
</tr>
<tr>
<td>6.2 Obstacle avoidance</td>
<td>43</td>
</tr>
<tr>
<td>6.3 Contingency plan</td>
<td>44</td>
</tr>
<tr>
<td>6.4 Micro controller</td>
<td>44</td>
</tr>
<tr>
<td>6.5 Social Impact</td>
<td>45</td>
</tr>
<tr>
<td>7 Conclusion</td>
<td>47</td>
</tr>
<tr>
<td>4 References</td>
<td>48</td>
</tr>
<tr>
<td>Appendix</td>
<td>50</td>
</tr>
<tr>
<td>A Lidar Lite V3</td>
<td>51</td>
</tr>
<tr>
<td>B MB 1202 Sensor</td>
<td>52</td>
</tr>
<tr>
<td>C MB 1242 Sensor</td>
<td>53</td>
</tr>
<tr>
<td>D List of components</td>
<td>54</td>
</tr>
<tr>
<td>E Schematic och the UAV senscontroller</td>
<td>55</td>
</tr>
</tbody>
</table>
## List of acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMS</td>
<td>Battery Management System</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CPS-VO</td>
<td>Cyber-Physical Systems Virtual Organization</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-Intergrated Circuit</td>
</tr>
<tr>
<td>LiPo</td>
<td>Lithium Polymer</td>
</tr>
<tr>
<td>MAVLink</td>
<td>Micro Air Vehicle Link</td>
</tr>
<tr>
<td>OCV</td>
<td>Open Circuit Voltage</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional-Integral-Derivative</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>SCL</td>
<td>Serial Clock Line</td>
</tr>
<tr>
<td>SDL</td>
<td>Serial Data Line</td>
</tr>
<tr>
<td>SoC</td>
<td>State of Charge</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TOF</td>
<td>Time Of Flight</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned aerial vehicle</td>
</tr>
</tbody>
</table>
1 Introduction

Cyber-Physical Systems Virtual Organization (CPS-VO) is a community for CPS-researchers and developers including institutions from academic institutions to industries [1]. Halmstad University has been invited to participate in the CPS student challenge 2018. The goal is to develop an Unmanned aerial vehicle (UAV) with an intelligent autonomous navigation system for field rescuing of another UAV. The main focus of this project is not to make the UAV itself but instead examine how the rescuing and flying can be done autonomously and what kind of technology would be necessary. The challenge for the UAV contest is to take off and search in a specified area and retrieve a lost object back to base. The team participating from Halmstad University consist of three groups within the fields of Computer, Electrical and Mechatronics engineering as their bachelor theses.

UAVs plays a big role in today’s society and will in the future help people and industries accomplish tasks that would have been impossible before. This project will cover a sub-system to the CPS-Challenge 2018 team from Halmstad University. The subsystems main purpose is to provide the UAV with information about its environment, such as obstacles and ground distance.

1.1 Purpose

The purpose of this project is to examine how ground distance control, obstacles detection/avoidance and battery status system could be implemented on an UAV.

1.2 Goal

The goal of this project is to construct and implement a sensor system on an UAV with an onboard computer and an existing autopilot that support the PX4 flight stack. This will make it able to autonomously keep a fixed distance to the ground regardless of the grounds shape. An UAV with such a system shall also be able to detect and avoid obstacles and keep track of the battery status. In case of low battery, a contingency plan shall be activated for safety reasons.

1.3 Delimitations

The project scope is quite big considering both development and possible solutions. Some delimitation is required to keep the project to a reasonable level. They are listed below:

- The system will only be able to detect obstacles in front, left and right of the UAV.
- Obstacle avoidance system will only be able to handle big, static obstacles with a minimum size of $1\times1\text{m}$ and a height of maximum $20\text{m}$.
- The system will use sensors like infrared, ultrasonic and laser for detection.
- The system will not be developed for indoor or city navigation.
- Maximum flight speed and altitude will be restricted by the sensor system.
- Sensors will not be developed.
- Laws and regulations about usage of UAVs will not be taken into consideration.
- Weather conditions like strong wind, rain and snow will not be taken into consideration.
1.4 Requirements

Below there is a list of the main requirements that the project has to fulfill. The organizers of the CPS student challenge 2018 has given some of the requirements concerning both hardware and software, those requirements has to be met for participating in the challenge. Additional requirements have also been set to further increase the reliability and safety of the system.

Hardware
- The altitude control system shall have a precision of ± 10cm on an altitude less than 2 m.
- The altitude control system shall have a minimum altitude range of 20 m.
- When the contingency plan is activated, the UAV shall return back to its starting position.
- The obstacle detection system shall be able to detect obstacles within a range of minimum 2 m.
- The precision of the detection range shall be no more than ± 10cm.

Software
- All software shall be developed using ROS.
- A simulated scenario with the obstacle detection/avoidance and altitude control system shall be done in the simulation environment Gazebo.

1.5 Main Questions

The main questions for the dissertation are given below.

- How can the UAV detect and avoid obstacles?
- How can the UAV keep control of its altitude?
- How can the system be simulated and then integrated into the UAV?
2 Background and theory

The usage of UAVs has grown rapidly in the past years, both in the private and industrial sector. UAVs today are widely used for tasks like surveillance, geographic mapping, shipping and delivery, remote sensing as well as search and rescue operations [2]. Recent researches and developments are focused on autonomous usage of UAVs. Since autonomous means there is no actual pilot driving the vehicle, the UAV needs to be able to manoeuvre itself through unknown environment when performing its tasks. In order to operate safely it needs to be able to detect obstacles in its path and take decisions based on the information given by its built-in sensors. In the private sector an UAV can be a somewhat expensive investment. Because of today’s UAVs speed and range, it is quite easy to get carried away with the wind when flying or accidentally fly away too far or even collide with objects. This is also one of the main factors pushing the demand for UAVs with built in obstacle detection.

2.1 Related work

2.1.1 Proximity sensing

Proximity measurements are used for a variety of applications and have been implemented in more and more products. One application that many people have used is the parking assistance in modern cars, which gives the driver information about the cars environment through sound and visual feedback with a display in the car. Since the objects reflecting the signals in this case mainly consists of big solid objects like other cars, such sensing is often done with ultrasonic sensors [3]. The measurements can be done with different techniques depending on the application. In flight applications, laser is often used because of its long range to estimate height and distances. The Skydio R1 drone uses a fusion of visual and distance sensors to avoid obstacles in its flight path.

2.1.2 Obstacle avoidance

Conventional UAVs are typically controlled manually by an operator who navigates using a camera mounted on the UAV. Navigating the UAV from one point to a specified location or through way points autonomously requires detailed information about start location, current location, flight path and waypoints. One way of doing this is by reading sequences of GPS data and compute the UAVs future position and with onboard sensors and decide if that position is ”safe”. If the position is considered ”not safe” or if an obstacle is detected, an obstacle avoidance algorithm has to navigate the UAV safely around the obstacle and bring it back to its original path towards the goal.

One of the simplest way of doing this is with ”The Bug2 Algorithm” [4]. When an obstacle is encountered, the robot would simply circle the obstacle until the line segment representing the heading towards the goal is reached, then leave the obstacle continuing on route. However, since only the most recent data is taken into consideration, signal noise greatly affects the performance of the bug algorithm.

Another way of avoiding obstacles is by the ”Potential Field Algorithm” [5]. This algorithm calculates the path by viewing obstacles and goal location as forces on the robot. The goal position acts as an attracting force while obstacles repels the robot, the path is then

[https://www.skydio.com/2018/02/introducing-r1/](https://www.skydio.com/2018/02/introducing-r1/)
calculated as the resultant of these forces. The main drawback of this algorithm is that it performs poorly in tight spaces.

Lastly, the algorithm solving most of the above mentioned algorithms drawbacks is the "Vector Field Histogram Algorithm" or VFH \[6\]. The VFH overcomes the signal noise problem by using several recent sensor reading to create a polar histogram representing the probability that there is an obstacle to heading. The histogram is used to identify all the passages big enough for the robot to pass through. The path is then selected by the evaluation of a cost function, taking into account how much the robot has to deviate from its original heading. The path with the least "cost" is selected and considered the most efficient. This algorithm is robust and takes the robots actual kinematics into account. The complexity, however, makes this algorithm hard to implement on embedded systems due to the computation load.

### 2.1.3 Altitude Control

There is a lot of different altitude systems that can been used for UAVs, everything depends on what sort of accuracy is needed for that sort of operation the UAV is built for.

One way is to use a barometric pressure sensor that records the atmospheric pressure before the UAV take off. The differential barometric pressure data can then be used to calculate the distance to the ground. This, however, only works if the ground is completely flat since the result is always the difference between the UAVs starting point and current location, it does not take the grounds terrain into consideration \[7\]. Laser and ultrasonic technology is widely used for altitude control of UAVs during autonomous landings \[8\] and low altitude flights. This technology gives a good accuracy of the distance to the ground during landings, takeoff and low speed flight where the UAVs pitch and roll is less then the maximum working angle of the sensors. One of the most common ways of controlling the altitude is by using a Proportional-Integral-Derivative (PID) regulator where the differences in between the given altitude and the actual sensor data shall be as small as possible \[9\]. This is shown in figure \[1\].

![Figure 1: Altitude Example PID](image)

Figure 1: Altitude Example PID
2.2 Theory

2.2.1 Sensing

Distances can be calculated in many different ways. The most common way is done by the Time Of Flight (TOF) principle. A pulse of sound or light is emitted and the distance is measured by the time it takes for the signal to be reflected back to the transceiver. For optical sensors such as infrared or laser sensors the distance is calculated by

\[ D = \frac{c \times t}{2} \]

where, \( c \) is the speed of light and \( t \) is the measured time. The same principle can be used for sound based sensors, however since the speed of sound through air varies with temperature it has to be taken into account. The speed of sound at the earth ground level is typically calculated by:

\[ V = 331.45 \times \sqrt{1 + \frac{T}{273}} \]

where \( T \) is the temperature in Celsius. Once the velocity is calculated, the distance can be calculated. The most common ways of distance measurement will be described below.

**IR-Sensors:** The infrared sensor works in accordance with the reflection principle to detect obstacles and measure distance. The sensor mainly consists of an infrared transmitter, a receiver and some kind of calibration or adjustment device. The transmitter emits an infrared signal, if there is an object to reflect the signal the receiver will detect the object in front of the sensor. IR sensors are mainly used for short range applications, typically distances up to a couple of meters. To measure distance, the receiver is usually connected to a comparator and potentiometer. Since the amplitude of the signal decreases with distance, the potentiometer is used to set the threshold of which the sensor should react, in this case the distance. To increase precision, the emitted signal is set a specific frequency which the receiver is looking for, this help to reduce noise from other light sources. The advantages of the IR-sensor is mainly its low power consumption and ability to detect a wide variety of materials since it is only affected by the surface reflective capabilities like soft materials. The material can also in some cases be its drawback, for instance when trying to measure distance to a building covered with glass like a skyscraper. It is also highly affected by atmospheric conditions like dust, fog and rain. Temperature can also contribute to errors in readings.

**Ultrasonic Sensors:** The basic functionality of the ultrasonic sensor or sonar as it is called, is the same as of the infrared sensor. Instead of using light it uses high frequency sound pulses, typically at 40 kHz or above. The difference in time between the emitting and detection of the reflected pulse is measured, this time can then be converted to distance using the equation:

\[ D = \frac{t \times 331.45 \times \sqrt{1 + \frac{T}{273}}}{2} \]

Ultrasonic sensors can be used to measure distance up to a few meters in outdoor conditions. Good ultrasonic sensors are also equipped with thermal transducers to compensate for varying temperatures. Sonars can detect most materials including glass and liquids and is more affected by the materials density or sound-damping capabilities. It has a higher
tolerance to atmospheric conditions and provides very precise readings of big solid objects. It has a long range compared to other cheap proximity sensors but are highly affected by the angle of the object relative to the sensor. If the angle is great enough the sound pulse will simply bounce away and never reach the transceiver resulting in infinity or max readings.

**Laser Sensors:** Range measurement and detection with laser technology offers good accuracy in both small and long ranges. Some lasers can measure distances up to several kilometers away. Due to the focused beam it transmits, it is also an issue when detecting small object or when a specific point of measurement is desired. It has a higher tolerance to atmospheric conditions compared to infrared but can be quite expensive. Generally it comes with heavy hardware i.e. more weight and power consumption, which is an important factor for a battery driven flying operation.

### 2.2.2 Filtering

Although many store bought sensors have circuits built in for reducing noise, additional filtering may be needed. This way the filter can be calibrated to a desired behavior. To filter measurements of distance, averaging filters are used most commonly. They work by taking several recent readings into account and producing a result based on the average value. This filters smoothing-“gain” can be set by changing the number of readings used for each measurement. Another way is by using a exponential filter which calculates a new smoothed value based on a new measurement and its last calculated value like shown below:

$$Y_n = G \times x_n + (1 - G) \times Y_{n-1}$$

Here the amount of smoothing is controlled by the parameter G and it can vary between 0.0 and 1.0, values close to 0.0 resulting in a great smoothing and values close to 1.0 resulting in low smoothing.

Another common way of filtering is using the Kalman filter which instead uses the input data to estimate the value. It is a robust model that has even been called ”one of the greatest discoveries of the twentieth century”[12]. It is not really a filter that excludes certain signals but more a mathematical model that uses a set of equations and continuous data inputs to estimate the true value. It is also an iterative filter meaning it has to run continuously throughout the readings. The simplified estimation process can be described by three steps. Firstly the Kalman gain (G) is calculated based on the error in the last estimate ($E_{EST}$) and the error in the measured value ($E_{MEA}$) by:

$$G = \frac{E_{EST}}{E_{EST} + E_{MEA}}$$

The gain varies between 1 and 0 and determines how much ”trust” to put into the new measured value when estimating the current value($EST_t$) as seen in the equation below. If the measurement error($E_{MEA}$) is small, the gain will be close to 1 thus meaning more trust can be put into the new measurement.

$$EST_t = EST_{t-1} + G \times (MEA - EST_{t-1})$$

The last step is to calculate the error in the estimation($E_{EST}$) before the next iteration.

$$E_{EST} = \frac{E_{MEA} \times E_{EST_{t-1}}}{E_{MEA} + E_{EST_{t-1}}}$$
2.2.3 Software

ROS

Robot Operating System (ROS) is not a operating system in the traditional sense. Instead it is a framework of code and tools that allow you to run your desired function and make it run the way you intended it. It is open-source based and offers all the tools needed to program the behavior of your robot, including low level device control, implementation of commonly used functionality, message passing between processes and general package management. Processes in ROS are called nodes and every node is usually responsible one or more tasks. The nodes communicate with each other via logical channels called topics. Each and every node can obtain or send data to other nodes by subscribing or publishing information on different topics. An example is shown in figure 3 how the message passing could look for a robot detecting an obstacle.

![Diagram of message passing between nodes](image)

In this example the robot is fitted with a proximity sensor controlled by the process in node 1, the node is publishing its data to a topic which node 2 subscribes to. In this case the data is published in the form of a LaserScan message which is one the standard message types in ROS. The message consist of 1 or more distance measurements that has been done by the sensors. If the obstacle detection process (node 2) decides that the measurements...
represent an actual obstacle, it publishes a velocity command to the motor controller making
the robot stop or change its heading.

- **Service**: ROS Service is a closed task where no information about the current state can
  be read until the task is done. Neither can the robot be commanded to do something
  else while it is executing a service, for example when activating the actuators on a
  robot.

- **Action**: ROS Action is similar to a service but during its execution the robot can be
  given additional commands or feedback information about its current state.

**Gazebo**
The software Gazebo is a simulation tool which is widely used in the robotic world. Gazebo
is a realistic simulator that can be used together with ROS to simulate different algorithms,
behaviors and additional hardware on a robot. Everything from a simple wheel based robot
to a complex multirotor aircraft [15].

**MAVLink**
Micro Air Vehicle Link (MAVLink) is a communication protocol that allows computers run-
ning ROS programs to communicate with autopilot systems such as the PX4. Like ROS it
uses the same architecture by publish-subscribe type data transmission on channels called
topics.

### 2.2.4 Hardware

**Autopilot PX4**
The autopilot PX4 is an open source autopilot system. It is a project aimed to provide a
system for small unmanned aerial vehicles in the academic and industrial communities. It
is built around the same architecture as ROS, where processes called nodes exchange infor-
mation by publishing and subscribing data to different topics which allows a direct interface
to other systems like ROS [15]. The hardware is very versatile, allowing interfacing to many
common protocols such as Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI),
Controller Area Network (CAN), Pulse Width Modulation (PWM) and compass heading.
This makes it a good platform when creating projects that require the use of third party
hardware. The PX4 comes with a variety or presets depending on what platform is used
such as, multicopters (drones), planes, rovers, boats and bipedal robots. Once implemented,
the robot can be controlled by publishing data into different ROS-topics like velocity com-
mand and position command. The velocity command is used by publishing a Twist-type
message and the position control is used with a Pose-type message. The two message types
are described on the next page.
The twist message basically consist of two vectors, the linear vector controls the movement velocity and the angular vector controls the pitch, roll and yaw. In a pose message however, the position vector represents a position in a 3-dimensional space, the orientation is set using quaternions which represent the rotation, in 3 axis, of a rigid body in respect with a coordinate frame.

**I2C bus**
The I2C bus is a dual leader multimaster bus consisting of a Serial Data Line (SDL) and Serial Clock Line (SCL). Both leaders are connected to positive supply voltage through pull up resistors, supply voltage is often 3.3V or 5V. Data on the I2C bus can be transferred with a speed of 100kbit/s, 400kbit/s or 3.4mbit/s depending of what mode the bus works in (Normal, Fast or High Speed). The meaning of multimaster bus is that it supports to have more then one master unit connected to the bus and transmit and receive data from slave devices such as sensor, display etc. The bus system is constructed in a way so the slaves has a unique 7 to 10 bit address, the maximum number of devices that can be connected to the same bus is restricted to bus capacitance limit of 400pF \[17\] and the address space. A slave can only transmit data when a master asks for it. Data from a slave can only be obtained by one of the masters at the same time.

**UART**
Universal Asynchronous Receiver Transmitter (UART) is an integrated circuit used for send and receive data through a serial port from a controller such as a computer or a micro controller. The UART contains a parallel-to-serial converter for sending data and a serial-to-parallel converter for receiving data \[18\].

The UART communicate trough asynchronous serial communication, this mean no clock is needed for synchronize the data transfer in between two units.
LiPo battery

Lithium Polymer batteries are a newer type of battery used in a variety of electronic devices. They offer a high capacity relative to their weight and have very high discharge rates meaning they can deliver a big current during continuous operation. The chemical composition makes it possible to produce LiPo batteries of pretty much any shape or size, but it also makes the batteries sensitive. Puncturing the battery can lead to fire and even explosions which means they need special care for general usage. A Lipo cell has a nominal voltage of 3.7V (4.2V at full charge) and batteries are often constructed with several cells connected in series. The complex architecture of the LiPo battery demands a good Battery Management System (BMS) to be safely charged. Knowing the State of Charge (SoC) in each cell is therefor crucial when charging to prevent damaging of the cells. Estimation of the SoC assuming the the initial state of charge is known SoC(0) can be done by integrating the battery current over time[19], as shown below:

\[ \text{SoC}(t) = \text{SoC}(0) - \frac{1}{C} \int_0^t I_{\text{cell}}(\tau) d\tau \]

Where \( C \) is the nominal cell voltage and \( I_{\text{cell}} \) represents the cell current. It is a simple way also known a Coulomb counting algorithm. As mentioned in [19] this method is very sensitive to measurement errors which can lead to large errors in the SoC estimation over time. Another way of estimating the charge level is by simply measuring the Open Circuit Voltage (OCV). This method requires detailed knowledge of the discharge characteristics of the battery which is far from linear.
3 Methodology

3.1 Hardware

3.1.1 Sensors

**Obstacle detection**

Regarding the detection of obstacles, all of the sensors mentioned in section 2.2.1 could be used. However, there is a big difference in what kind of obstacles they are able to detect. The laser, for example, would provide great accuracy in the distance measurement, but it would only be able to detect very large obstacles at certain places due to its narrow beam. This makes it suitable for measurements like height but not for the detection itself. The detection area with lasers could be described as a cross, with low or close to no ability to detect objects between the sensors pointing direction.

The infrared sensor, like the laser also uses light but with a wider beam, allowing for detection in a bigger area and with better ability to detect smaller obstacles. The range is much shorter but considering the application that the system is built for, only ranges up to 4-6 m is necessary. The drawbacks of the infrared sensor, mentioned in [20], mainly consist of environmental factors like weather. Things like fog, dust or rain can highly affect the sensors readings. Since the sensor is based on light reflection, rain for example would scatter the beam of light resulting in incorrect readings. Fog and dust can stick to the lens if the sensors is not protected well enough.

With all this information taken into consideration the conclusion was that a ultrasonic sensor would be best suited for the detection task. Depending on the model it provides a wider beam, it is not affected by surface reflectiveness or weather conditions like dust. Also, the CPS-challenge will be held in a desert environment. The MaxBotix I2CXL-MaxSonar-EZ is the ultrasonic sensor chosen for the project. It is a series of sensors with different noise tolerances and varying ability to detect smaller objects. All of the sensors provide the desired detection range as well as a good enough resolution to meet the set requirements. It only weights 6 grams and uses no more than 50 or 100mA during continues operation depending on the supplied voltage. The size is small, only 19x22x25 [mm] giving many options for placement. Since it was unclear at this stage how the noise tolerance and detection size would affect the end functionality two sensors were selected. The MB1202 with a lower tolerance but a higher ability to detect smaller objects as well as the MB1242 which offers a greater tolerance but less ability to detect small objects. Tests during the implementation will reveal which one of them is better suited for the application.
Altitude
There is a few different sensors that can be used for the altitude control system and the most common is to use laser or ultrasonic sensors [8]. To use the autopilot’s embedded barometer is one option but then the system is not able to detect level differences on the ground and the accuracy will not be in the same grade as a laser or ultrasonic sensor. The altitude control system for this project is going to use a laser sensor due to the fact that laser has better specification then ultrasonic for this sort of application. The laser is both better in accuracy, and maximum range compared to ultrasonic. The chosen sensor is the Lidar Lite V3 for this system due to fact that specifications match the project requirements, see appendix [A] for the data sheet of the sensor.

Battery
There is a few different sensors that works together with the PX4 autopilot, the accuracy is almost the same but not the maximum current that they can handle. The Pixhawk autopilot comes with a battery sensor that can be used on smaller UAVs where the maximum continuous current is 30amps. That sensor would not be enough since the UAV that is going to be used has 6 rotors that can consume up to 17amps each and other electrical components shall also be placed on it. The only choice for the sensors that was available on the European market was the ”Mauch 076” that could handle up to 200 amps and batteries up to 14cells (58.8volt).
3.1.2 Microcontroller

Any development board with support for I2C could be used for the project. Many of the off the shelf boards however has many additional features that are not necessary for this application and the size and weight is not appropriate for being placed on an UAV where more weight result in less flight time and a bigger size then necessary, which less space for other components. Connectivity is also bad when it comes to using multiple I2C components on the same bus. Arduino Nano is an of the shelf board that has a size of 18x45mm and a weight of 7grams according to the manufacture but it does not support multiple I2C components on same bus without using a breadboard or some sort of self developed shield. Therefore the choice for this project was to design a new micro controller with just the necessary components that is needed for all the chosen sensor mentioned in Section 3.1.1.

The developed microcontroller has been developed with the base from the open hardware board Arduino Nano using the PCB design program OrCAD. The board called ”UAV SenseController” has an ability to connect up to four I2C components and has a size of 28x38mm and a weight of 5gram.

The processor for the controller is a ATmega328p due to the fact that the project members has previous experience with that processor and the performance is more then enough for handle the I2C data. A UART circuit is chosen so the communication from and to the micro controller can go through serial. Other components such as capacitors, resistors has been chosen to SMD capsule of size 0603. The choice of oscillator for the controller is a 16Mhz, which is in range of recommendation for the ATmega328p and has the necessary parallel load capacitors in the same capsule. The reason for choosing an oscillator with embedded load capacitance is to not have to take calculations for a stable frequency info consideration. the total size of this oscillator is also smaller then placing two SMD capacitors next to an oscillator without embedded load capacitance. All chosen components for the controller can been seen in appendix [D].

Construction

The schematic with all the chosen components has been sketched up in OrCAD where each specific component has been initiated with a footprint that has been made in the software "Library Export" according to the data sheets. See schematic in Appendix [E]. The PCB design has been made after the netlist of the schematic in OrCad PCB editor and components has been placed on both side of a dual layer PCB with copper thickness of 1 oz(0.034). The tracks in between components is made with a width of 6 mils (0.1525mm) due to the fact of that shall be enough for handling the total current that is restricted to 500mA by the fuse component when powered by USB which is the only available power source for the controller. The theoretical total current with all sensors connected is supposed to be the half of this according to the data sheet of the sensors mentioned in section 3.1.1 and other components on the PCB. Figure 6, 7 shows both layers of the microcontroller.
The main reason for the construction of this controller was to get the opportunity to connect up to four I2C components at same time on same board with the necessary connections. Components that is needed for the chosen sensors shall works according to the data sheets. I2C is a open drain bus and pull up resistors is needed on both the Data(SDA) and Clock(SCL) line if not some of the connected slaves on the bus has that itself, the LiDar Lite sensors that is used for this project is one of a kind there no resistors is needed. The maxbotic ultrasonic sensors is needed pull up resistors on the bus if they are alone on it. For this project the maxbotic sensors are gonna be connected to the same bus as the Laser so no one is needed, but the controller is prepared to connect resistors if it is gonna be used in some other application or without the Laser.
3.2 Software

3.2.1 Modes

The chosen autopilot is the Pixhawk 2.1 that supports the PX4 flight stack. The pixhawk 2.1 is widely used for this type of application and a lot of spare parts and additional equipment is available on the market which is the reason for choosing the pixhawk 2.1.

The PX4 autopilot has different autonomous flight modes that can be used by the UAV. The Autopilot can be used either with a software such as "Qgroundcontrol" or "Misson planer", or through a companion computer that is running ROS. Regardless of which type of usage the flight modes can be reached. Some of the different flight modes are described below.

**Auto-TakeOff**
This mode is supposed to be used for taking off. The take off altitude is being set in a parameter with an unit of meters. The take off altitude will be reached with feedback given from the barometer sensor if no other altitude sensors are connected.

**Auto-Land** This mode is almost the same as the takeoff except instead of taking off it trigger a landing. The only parameter that can be set for this mode is the landing speed \([\text{m/s}]\). the feedback during this mode is the barometric and the IMU sensor if no other altitude sensor is connected.

**Auto-RTL** During start up and initialization of the pilot a home position will be set. The Return to Launch (RTL) mode can be used when the UAV should go back and land on its starting position.

**Offboard** Offboard mode is used for when a companion computer is connected to the pilot. During this mode ROS scripts can be used and commands in both velocity, position, set points and targets can be set via serial cable and MAVLINK from the companion computer.

3.2.2 Motion Control

As mentioned in Section [2.2.4] the PX4 pilot can be commanded by different types of commands such as velocity and position. Due to circumstances resulting in that the maximum velocity could not be set during simulation in Gazebo when position command was sent to the pilot, the choice for this project was to control the UAV with velocity command instead.

A ground base for controlling an UAV with a PX4 pilot with velocity control was found on a github repository\(^2\). The github code is used as a base for the project and the main reason for that is that the code had the possibility to send position commands and with help of a defined control system convert it to a velocity of ROS message of the type Twist.

The autopilot can either be controlled by sending global positions with message data of type latitude, longitude and altitude or by local position in a coordinate system that is defined in meters and a world frame of type ENU (East North Up). The ENU frame means that positive X-axis is east, Y-axis is north and Z is up. When using local pose \(x,y,z = 0,0,0\) is always the starting position.

3.2.3 UAV Sensecontroller

The sensecontroller’s job is to read data from the sensors and publish the readings into ROS. The software running on the controller is a merge of open-source code made for the Lidar Lite V3 and Maxbotix ultrasonic sensor. It includes functions like changing the addresses of the I2C units, scanning for connected hardware and reading a measurement from a specific address. The filtering is also done on the sensecontroller using a single variable Kalman filter. The sensors also need to be read in a specific sequence in order to make sure the sonars does not pick up each others signals and report false data, as stated in data sheet a maximum sampling frequency of 10 Hz is recommended, in other words a minimum of 100ms delay between readings.

3.2.4 Altitude

The system shall keep control of the altitude during those operation where the actual speed and altitude of the UAV is restricted so it does not affect the chosen laser sensors maximum distance and working angles before the laser beam bounce. Therefore the UAV has to be commanded using velocity control so that the speed can be restricted. The actual pitch, roll and placement of the sensors make sense of the actual altitude to the ground, all those parameters has to be taken into consideration when calculating this. The system overview is shown in Figure 8.

The ROS topic ”/mavros/Localpose/pose” is a topic that gives the actual position of the UAV represented in X,Y,Z and the orientations of the UAV in quaternions of X,Y,Z and W. For the altitude system a new topic will be made that is a copy of the /mavros/Localpose/pose topic but the Z-position (altitude) is change to the calculated altitude from the sensor reading instead of the barometric altitude. That topic can later on be used for controlling the UAV during operations which need good precision and keeping same altitude to ground regardless the ground structure. The raw data from the altitude sensor can be reach through

\[3\text{Lidar Lite V3 arduino library https://github.com/garmin/LIDARLite_v3_Arduino.Library} \]
\[4\text{MaxBotix I2C sensor arduino library https://github.com/simondlevy/MB1242} \]
\[5\text{A arduino implementation of a Kalman filter is available at https://gist.github.com/Zymotico/836c5d82d5b52a2a3695} \]
the /Laser height topic that the micro controller is publishing as mentioned in Section 3.2.3.
The actual altitude will be calculated according to the math operation below.

\[
\text{Actual Altitude} = (\text{Pitch/Roll}_{\text{Calc}} \times \text{MeasuredAltitude}) - \text{Placement}_{\text{comp}}
\]

Calculating actual altitude according to Pitch/Roll
The pitch and roll angles can be calculated from the given data from the IMU by using the eulertransformation functions that is defined in python from the library "tf". The orientation data can be reached from the IMU topic /mavros/imu/data and has to be transformed for getting the pitch and roll of the UAV.
If the IMU and the Laser is placed at same place on the UAV the offset placement calculations does not have to be taken into consideration, it just depends on the pitch and roll angle. The actual altitude according to the illustration Figure 9 is being calculated by using trigonometrical functions as shown below.
Actual Altitude = (cos(Pitch) * cos(Roll)) * Measured Altitude

![Diagram](image)

Figure 9: *How to calculate the actual distance to ground depending on the pitch and roll of the UAV*

Placement compensation
Depending where the altitude sensors are placed, an offset in X,Y direction from the IMU sensor will occur, this has to be taken into consideration when calculating the actual altitude of the UAV. Even the Z direction has to be taken in consideration, otherwise the measured altitude is not taken from the lowest point of the UAV. The Calculation of the compensation value for where the sensor is placed on the UAV is being done with the compensation formula below according to Figure 10

The compensation of Z can either be changed in the micro controller from the raw data or when the actual altitude is being set into the new ROS topic, this is for adjust from which point on the UAV the altitude is being measured from.
Placement\(_{\text{Comp}} = - (X \cdot \cos(\text{pitch}) + Y \cdot \cos(\text{pitch}))\)

![Figure 10: How to calculate the actual distance to ground depending on the pitch and roll of the UAV](image)

### 3.2.5 Obstacle avoidance

The system has three sensors as mentioned in section [3.1.1], one pointing forward in the heading direction and the other two on each side of the UAV (left, right). When using this system the forward sensor has to always point into the heading direction of the UAV, therefore the yaw orientation is important. The system is commanded using two different velocity controls so the speed could be restricted and the coordinate system changed according to next flight command. One of the velocity controls being used is the one that was already defined in the base as mentioned in section [2.2.4] called "velctr" and a other one that is developed is called "velpose". The difference between the two velocity modes is that "velctr" is adding the incoming position into its current positions and "velpose" is used as a position control but with velocity commands so the speed can be set after users choice. Example if a command with position \(x = 1\) is sent to the "velctr" the UAV will change in x direction 1 m from its current position in x, if same command will be sent when "velpose" is active the UAV will go to position \(x = 1\) in its own coordinate system based of the GPS.

#### Heading direction

The heading direction of the UAV is being set by sending commands in form of quaternions to the pilot, this because the heading direction is based on the orientation data of the IMU sensor represented in quaternions. The IMU compass on the UAV is represented in a value between \(-180^\circ\) to \(180^\circ\). The heading direction is calculated using arctan function of \(X, Y\) based on current position and destination position, this is showed in Figure [11]. When the heading direction is calculated it is being transformed to quaternions and then sent into the velocity controller that is using its control system to go in the sent direction.
θ = \begin{cases} 
\arctan \left( \frac{X}{Y} \right) & X \geq 0 \\
\arctan \left( \frac{X}{Y} \right) + \pi & X < 0 
\end{cases}

Figure 11: Heading calculation illustration

### Detect obstacles
The detection phase keeps control of the incoming data from the three sensors and compares their distance with a specified threshold. If the distance of some of the sensors is less than the threshold, a true or false Boolean will be published in a ROS topic.

### Avoid obstacles
Once the obstacle is detected, the system has to identify where the obstacle is and make a decision not to fly into it. When the obstacle is identified, the UAV changes flight mode to "velcttr" so positions can be set according to its current position. If the side sensors find the obstacle, the UAV will continue toward its current heading and do side changes either right or left direction with a 45° angle from its current heading. If the obstacle is detected by the front sensors, the UAV will incline. Once the obstacle is passed or the detection Boolean is false, the UAV changes back to the flight mode "velpose" and the goal destination is sent again with a new heading based on that. *Figure 12* illustrate how the system works.
**Left/right changes**

As mentioned above, the UAV will either turn left or right with an angle of 45° out from its heading direction when avoiding obstacles. The choice for going 45° out instead of straight to left or right is because the UAV shall still move forward to the goal position when avoiding obstacle and don’t just move left or right before it continues to the goal position.

The coordinate system for the UAV is ENU as mentioned in section 3.2.2. For getting the X and Y coordinates so the UAV can go either right or left, it has to be calculated using the GPS heading of the UAV with the math operations below.

\[
\begin{align*}
Right &= \begin{cases} 
X = -\cos(\text{Heading}^\circ + 45^\circ) \\
Y = \sin(\text{Heading}^\circ + 45^\circ)
\end{cases} \\
Left &= \begin{cases} 
X = \cos(\text{Heading}^\circ - 45^\circ) \\
Y = -\sin(\text{Heading}^\circ - 45^\circ)
\end{cases}
\end{align*}
\]

### 3.2.6 Contingency

The chosen battery sensors mentioned in section 3.1.1 are supposed to be the main part of the contingency plan. To get a good status of the battery, the sensor has to be calibrated correctly. The software for the PX4 autopilot called “QGroundcontrol” is made for calibration and setup of the actual hardware but also provide full flight control for the UAV and the ability to monitor the status of it such as position, battery, errors, flight modes, etc. The battery on the UAV that the project is using is three 5400mA 6 cell (6s) Lithium Polymer (LiPo) batteries connected in parallel resulting in a capacity of 17400mAh.

**Calibration**

Calibration of the battery sensors has been done with the PX4 setup program together with multimeter according to the calibration guide. The first phase of the calibration was to fill in the “voltage divider”, here the battery sensor measure the voltage and the user double checks the sensor value with a multimeter. The multimeter voltage shall then be written in to the software and the “voltage divider” being set automatically in consideration of the multimeter and sensor result. The second phase is to set the “current divider”, this process is exactly the same as the voltage divider but comparing the current instead of voltage and the “current divider” parameter is being set. Figure 13 shows the calibration setup.
The calibration also asks for how many cells the battery has and what the maximum/minimum cell voltage is for the battery. A fully charged Lipo battery has a cell voltage of 4.2V as mentioned in section 2.2.4 and an empty cell voltage depends on each battery. Due to the fact that the project member did not find any discharge characteristic for the battery in the UAV, a small discharge test and approximation of this has been done. The discharge test has been done from a fully charged battery (100%) down to 10% of the capacity and the mean cell voltage has been taken for each % of the discharge. For safety reasons of the battery health the discharge just went down to 10% of the battery, approximation in matlab has been done to get the empty cell voltage of the battery that is 3.59V. Figure 14 shows the measured discharge characteristic for the battery and the approximation of it.
Contingency plan
The autopilot has different flight modes that is described in section 3.2.1. During setup of the autopilot thresholds can be set for the battery status. The threshold for the UAV has been set so it will be on the ground when the battery is at 20%, when the battery drops under this threshold the UAV change flight mode to “LAND”. Another threshold for activating the ”Auto RTL” mode is set to 30%. The ”Auto RTL” threshold is just set so the UAV shall have some chance to come back to its starting position, this can not be changed during operation based on how far the UAV is from its starting position. If not UAV reach its starting position during the ”AUTO RTL” mode it goes down for a landing at the current position when the battery pass the 20% threshold. The threshold of 20% has been chosen in consultation with an employee a Halmstad University due to safety reason of the batteries.

3.3 Result analysis
The UAV will first be tested in Gazebo where parameters can be trimmed and different behaviors can be tested. When the desired behavior is achieved the tests will move on to real tests. The UAV will go through several tests where its detection and avoidance capabilities will be measured. To be able to test all the different functions the UAV will have to be put into different scenarios where each and every function can be tested. All scenarios will be constructed in a way so that results can also answer if the requirements have been met. In this section the different testing scenarios will be presented as well as what factors will be measured.

Altitude control system (CASE 1)
The altitude control systems task is to keep a fixed distance to the ground and potential obstacles. To test this the UAV will be commanded to fly along a path where obstacles of different heights have been placed. It will be set to keep a specific height during the flight and the actual height data will be recorded to a rosbag file. Since the obstacles height is known, the expected resulting height when it detects the obstacle is also known. By comparing these two, the deviation can be calculated, thus the altitude control systems capability. This scenario will be created to test the UAVs precision altitude control system. To do this the UAV will be commanded to fly along a straight path with a fixed height setting where objects of different heights will be placed.

![Figure 15: Testing scenario (CASE 1) for the altitude control system, the offset between the ideal and expected flight path is only for illustrational purposes.](image-url)
Detection and avoidance system
To test the detection and avoidance system different scenarios will have to be created. Since the sensors detecting objects beside the UAV work similarly to the height control, the scenario will be the same but this time with obstacles placed on both sides of the UAVs flight path. The front sensor will be tested by placing obstacles of varying height in front of the UAV.

Left and right sensors (CASE 2)
The side mounted sensors job is to provide the UAV with proximity readings on the left and right. In case a set threshold is reached the UAV will try to compensate by strafing left or right until the threshold is no longer reached. In other words, the UAV will always try to keep a minimum distance to its surroundings. In case there is obstacles on both sides, the UAV should try to center its position between the two obstacles. To evaluate the results the distance measurements of both sensors will be sampled to examine how good the UAV is able to compensate for the detected obstacle. The complete flight path will also be recorded for general judgment. An example of this scenario is described in figure 17.

Figure 16: Testing scenario for the side mounted sensor (CASE 2), the offset between the ideal and expected flight paths is only for illustrational purposes.
Front sensor (CASE 3)
The front sensor is the most crucial sensor because a failure to avoid an obstacle placed in front of the UAV most likely would result in a crash and possibly damage to the hardware. To test this, obstacles of varying height will be placed in front of the UAV. Once an obstacle is detected in front, the UAV is expected to increase its altitude until the obstacle is no longer present. Since the altitude system is always present, the UAV then makes a final climb, increasing its altitude by the given flight altitude and then moves onto the obstacle. Once on top the goal position is published again and the UAV continues towards the goal. The flight path will be recorded during these test for assessment as well as different proximity settings on which the UAV should react. To evaluate, the relation between different proximity settings to distance traveled will be examined.

Figure 17: Testing scenario for the front sensor (CASE 3), the offset between the ideal and expected flight paths is only for illustrational purposes.

Contingency plan
To test the contingency plan the UAV will be commanded to takeoff and fly 10 m from its starting position (home position). Depending on the current charge level of the batteries, the first threshold which activates the "AUTO RTL" function will be set to 10% below the current level and the second threshold which activates "LAND" will be set 15% below. Once in the air, the state and battery levels will be recorded. Since the batteries cannot be simulated this test has to be done on the real UAV. The result of this test will verify if the contingency plan works as planned and confirm if the battery voltage approximations are correct.
4 Implementation

4.1 ROS Action

The altitude control and obstacle system has been included in a ROS action that is described below. These action will be active until the given input is reached ± a threshold. The altitude control and obstacle system has been included in a ROS action that is described below. These action will be active until the given input is reached ± a threshold. The Altitude system can be reached from every other actions when using velocity modes to command the UAV because it is implemented in the base code that has been used under development when commanding in velocity control. If the given altitude to the system is more then the restricted one it will automatically set the altitude to the restricted one and the user will be informed.

- **GotopositionSensor.action**
  Go to position action is supposed to take a few input argument such as Latitude, Longitude and Altitude or X,Y,Z in meters. This action is supposed to go to given position and keep same altitude the whole way and detect obstacles.

- **Gotopositionvelpose.action**
  This action is just a test action where the heading direction can be tested together with the implemented "velposeCtr" and the altitude system. the input arguments shall be the same as the other action above and continue until its goal destination is reached.

4.2 Vehicle platform

The main platform for the project is an UAV build on a F650 frame with a companion computer on. The frame has six rotors and enough power to keep it stable and to lift when external hardware such as sensors, micro controllers etc has been added. Due to the size of the frame there is a lot of room to place the external hardware that is necessary for this project. A picture of the UAV that has been used for the project is shown in Figure [18].

![](image)

Figure 18: The UAV using for this project

The autopilot Pixhawk 2.1 has been equipped with GPS, battery sensors, telemetry and a receiver to a manual remote control as additional hardware. The telemetry is being used to communicate with the pilot through the software Qgroundcontrol and get live data from it during autonomous operations. Qgroundcontrol is also the program where all external and
embedded hardware on the pilot has been initialized, all from the remote control, calibration of the sensors, frame firmware, parameters for contingency etc. In section [3.2.1] autonomous modes for the pilot was mentioned. The pilot also has a few manual modes and safety switches that is necessary for this project during start up, tests and for taking over in case of fault, those is described below. Figure [19] shows how the switch buttons on the manual control has been programmed with different modes and safety switches.

**Manual mode**
This mode is a full manual mode, where the user of the remote controlling the UAV without any help from the autopilot.

**Altitude mode**
Altitude mode gives the pilot easier control of the UAV, instead of manual mode. The altitude can be set using the throttle stick, half throttle indicates that the UAV keeps same altitude, up and down with the throttle changes altitude either up or down. This mode has been set to the manual controller for easy taking over from offboard mode to manual remote without uncontrolled changes in altitude occurs.

**Position mode**
when this mode is active the UAV stays in the current position even if wind gust comes. This mode is set to the remote control for extra safety reason in case of when taking over from offboard and the pilot lose control of the UAV, then it is possible to switch over to position mode and think about what has to do before switching over to either altitude och manual mode again.

**Arm and killswitches**
When arming the UAV the rotors is starting and the UAV is prepared for operation. This can either be done by the manual control or in offboard mode from the companion computer. The UAV can not be armed before the kill switch on the manual remote is in the correct position and the kill switch button on the UAV is pressed.

Figure 19: *How to remote controller is programmed for the project.*
4.3 Mounts and placements

The sensors needed to be mounted on the UAV so that they wouldn’t pick up or detect any of the UAV’s parts. A clear line of sight is needed for them to work. Easy replacement is also desired in case of a crash or if they somehow got damaged. Different solutions were tested and discussed but for the ultrasonic sensors the conclusion was that the arms of the UAV would be the optimal placement, at least for the left and right sensors. To mount them a "snap-in" mount was created using SolidWorks. Once the mount is attached to the sensor it can easily be placed on the UAV-arms in a matter of seconds.

![Image](488x247.png)

Figure 20: Sonar arm mount

For the front detection and height sensor another mount had to be made due to the fact that there is no arm pointing straight forward on the UAV. Instead the mount uses the frame. To make sure the Lidar Lite V3 didn’t hit or detect the UAV itself an extended arm was created.

![Image](488x247.png)

Figure 21: Combined sonar and lidar frame mount

The arm screws onto the frame with regular M3 screws. The additional support at the base is to keep the arm from rotating. The length of the arm also determines how much to compensate the measured height when UAV pitches or rolls.
4.4 Reading the published data

Once the programming of the sensecontroller was done and the sensors implemented on the UAV a couple of tests were done to further examine weather the MB1202 or MB1242 was better suited for the detection task. The tests were done with the sensors mounted on the UAV, the actuators turned off and by reading the data the sensecontroller published into ROS. Firstly an accuracy test was done. This was done by varying the distance to a drywall with 5cm intervals from 20cm which is the minimum reading of the sensor to 200cm. The measurements were done using a measuring tape fixed to the wall.

The test did not reveal much difference between the sensors, only that the accuracy of both sensors could be considered good. The small offset in the readings is due to the fact the distance was measured from the wall to the plastic cap of the sensor. The distance calculations are actually done relative to the center of the sensor and not the cap.

The second test was to examine the maximum angles of detection. The sensors were both placed on a fixed distance to a drywall. The angle was then increased by steps of 5 degrees at a time, the result is shown below.

![Accuracy test result](image)

![Angular test result](image)

Figure 22: Accuracy test result

Figure 23: Angular test
In this test the differences became clearer. The MB1242 which according to the data sheet \[C\] has the highest noise tolerance was not able to detect and read any distance above 35 degrees. Instead the sensor returned its maximum result which is 765 cm. The MB1202 \[B\] on the other hand was able to detect the distance at higher angles but with an error that increased with the angle.

Next a disturbance test was done. Since the sonars are placed close to motors it was necessary to examine if the motors rotation and noise would compromise the sonars readings. To do this test the UAV was placed in the middle of a corridor with one MB1202 mounted on the left arm and a MB1242 mounted on the right arm. It was placed so that both sensors returned the same value, in this case 216 cm. The motors were turned on and the throttle was set to approximately 30%, since the UAV needed to stay in place to do the readings no more throttle was possible because it would result in actual take off. With the motors spinning, 250 samples were taken to see if the readings were affected by the motors noise or possibly their EMF (electro-magnetic field) which is created to drive the motors. The sensecontroller publishes new readings at 10Hz resulting in a total sample time of 25s. Since the sensors are sound based, the most likely cause of disturbance is probably the motors noise. The The result is shown below:

![Figure 24: Disturbance test](image)

The motors did not seem affect the readings, at least not in this test. The result however does not exclude that the sensors could be affected by EMF since no further tests were done to stress test its tolerance. Neither does the data sheet provide information about the sensors bandwidth, only that the signal frequency is 42KHz, since acoustic measurements of the motors were not possible, no conclusion could be drawn from the result regarding the acoustic tolerance. The MB1202 deviated 1 cm from its original readings with the motors turned on. It is possible to the vibrations in the chassis caused this, but most likely the deviation is due to human error. It is possible that the UAV was placed so that the actual distance to wall was 216.5cm, since the resolution of the sensor is 1cm, this could cause the readings to vary between 216 and 217cm. The MB1242 was did not seem affected at all, it continued to report its original reading all throughout the 250 samples.
Lastly a test was made while flying. The UAV was commanded to take off and hover at an altitude of 3 m. While in the air, samples on the sonar readings was taken with no obstacle close to the UAV. This should result in its max readings of 765 cm. This time the throttle was set to approximately 60% resulting in much more acoustic noise from the motors. This test showed a big difference in the readings.

The first 100 samples is taken during the take off meaning that the sensors could have picked up the ground itself, but from there no obstacle was present. The MB1202 reported quite many misreadings while the MB1242 reported less than 10 wrong readings. Errors like these have been experienced while testing the sensors but hard to measure. This last test however revealed that the MB1242 definitely was better suited for the detection task.
4.5 From micro controller to flight commands

The UAV has an onboard system consisting of a companion computer running all the programmed ROS nodes as well as the autopilot. The ROS nodes does all the decisions and calculates new flight commands to the autopilot based on the incoming data from sensors, GPS and remotecontrol.

The micro controller publishes the sensors data at a frequency of 10Hz through serial communication. The package that allows for this communication is called ROS serial\cite{21}. The PX4 autopilot however is able to override the given commands from both remotecontrol and ROS in case the contingency plan is activated.

![Flowchart of the entire system](image)

Figure 26: Flowchart of the entire system
5 Result

In this section the results of the implemented system and programmed behavior will be presented. Many of the results are simulated due to the fact that UAVs are not allowed to be flown anywhere but only in allocated places. Since the place allocated for the project consisted of an open terrain like environment it was not possible to test the avoidance behavior. Instead more time was spent on getting the altitude system functioning correctly since it was more important for the competition.

5.1 Altitude

Simulation (CASE 1)

To test the altitude control a scenario was created in gazebo with different obstacles placed along the flight path, three boxes of varying heights as well as a ramp as shown in Figure 27. The UAV was commanded to fly straight towards the ramp, and to keep an altitude of 4 m and stop 1 m from the wall on top of the ramp. The three boxes height measures 1, 2 and 3 m.

![Simulation scenario (CASE 1)](image)

Figure 27: Simulation scenario (CASE 1)

Figure 28 and 29 shows the result of the test where Figure 28 shows the UAVs actual height in the simulation environment. The position is measured along the flight track with a maximum distance of 70m representing the goal which is located on top of the ramp. Figure 29 shows the measurement done by the simulated altitude sensor. This shows how fast the UAV can compensate for varying distance to the ground.

![Simulation result, actual height (CASE 1)](image)

Figure 28: Simulation result, actual height (CASE 1)
Angular compensation

The compensation calculations done in ROS was tested indoor with the UAV suspended at a fixed distance from the floor. The goal here was firstly to examine how big angles the system could handle, as well as the sensor itself. For the result to be considered good the distance to the ground should not vary more than ±10cm at angles up to 30 degrees. The maximum working angle is based on the flight speed of the UAV, with the current velocity settings the UAV never exceeds these angles.

The UAV was then tilted back, forward and sideways while sampling the raw data measured by the sensor, the calculated distance as well as pitch and roll of the UAV from its IMU sensor. The result is shown in the graph below.

Figure 29: Simulation result, laser measurements of the UAVs height. (CASE 1)

Figure 30: Test setup
Figure 31: Result graph of the angular compensation test

**Precision**

Range and precision test of the altitude system has been done in an indoor environment with ranges of 1 to 45.5 m, the sensors have not been mounted to the UAV during this test. The goal of this test is to examine the maximum altitude that can be flown with the UAV with the altitude system. The result from the test is shown in Figure 32.

Figure 32: Precision test of altitude system
**Px4 altitude compare to Altitude system**

Real flight test using velocity control and flying on an altitude of 2 m before it incline to 5 m. The goal of the test was to see how steady the UAV could keep the same altitude with the system and see which altitude the embedded altitude sensor in the autopilot shows during this test.

![Figure 33: Barometer compared to altitude system](image)

Figure 33: Barometer compared to altitude system
**Flight test**

Real test flight using velocity control with a set waypoint of 7m east based from the UAVs current position with a altitude of 3m and then land on that point. During the test the a table with a height of 85cm was placed in the middle of the route. The purpose of this test was to see how good the UAV could keep a fixed distance to the ground regardless of the ground shape and also how good it could land with help of the system. A video of the test can be seen here [video link]. Figure 34, 35 shows the result of the, where Figure 34 shows the table and landing part from the whole test shown in figure 34.

![Graph of flight test results](https://example.com/graph.png)

**Figure 34:** Flight test with a landing and a table of height 85cm placed in the path
5.2 Obstacle avoidance

Simulation (CASE 2)

In this test the UAV is once again commanded to fly along the flight track shown in Figure 36. This time with obstacles placed on both sides of the flight path. The UAV is also set to keep a minimum distance of 4 m to its surroundings (left and right). The result of the test shows that the UAV with the system could detect and also avoid big obstacles. The test was just done with a minimum distance range of 4 m to the obstacles, to evaluate the result probably more test had to be done with different obstacles and other thresholds for detecting the obstacles and see how good it keeps the distance to these
Simulation (CASE 3)

In this test three obstacles are placed in front of the UAV. Two walls with a height of 10 and 20 m as well as the ramp, figure 37 shows the simulation environment. The test is focused on examining how the behavior changes with different proximity settings on the avoidance maneuver. The flight altitude was set to 4 m. Figure 38 39 shows the test results.
1m was obviously a too short distance for the UAV to be able stop before it hit the obstacle. However the result shows that all proximity settings above 1m resulted in a successful avoidance.
**Heading Direction**

In this test the "Velpose" flight mode that is developed for the UAV was tested in the action `gotopositionvelpose.action`. The main purpose for this test was to test so the UAV change heading direction after the given destination. This is a part of the obstacle avoidance system so the front sensor always pointing forward. The Figure 40 show the result where both the flight path and the heading direction of the UAV taken from the IMU sensor. The IMU compass is defined as followed $X = 0^\circ$, $-X = -180/180^\circ$, $Y = 90^\circ$, $-Y = -90^\circ$.

![Heading Direction test result](image)

**5.3 Contingency plan**

To verify that that the contingency worked as planned the UAV was flown approximately 10 m from its starting position (home). Once in place the autopilots navigational state and battery level was recorded. The PX4 has more than 20 different states but to simplify the result only the states used in the project will be described. The state is represented by a simple integer in the log files as shown in Figure 12.

- 0 - Manual Mode
- 1 - Altitude Flight Mode
- 2 - Position Flight Mode
- 5 - Auto RTL Mode
- 18 - Land Mode
Due to hard wind the UAV had to be controlled manually to avoid crashing. During the contingency test the current charge level of the battery was 60% so the first threshold (Auto RTL) was set to 50% and the second (Land) was set to 47%. For some reason the usual rosbag files were corrupted during this test, so instead the flight log from QGroundControl had to be used to examine the result.

![Figure 41: Battery level during the flight](image)

The results show that at approx $T=300$ the autopilot overrides the manual control and switched to mode 5 and the UAV head back to its home position. Soon there after at approx $T=350$ the mode is switched again to mode 18 and the UAV landed on its starting position. The battery was also measured after the flight and the battery meter showed 45% which lead to the conclusion that the voltage approximations were reliable.

![Figure 42: Flight Modes during test](image)
6 Discussion

Different behaviors have been tested and the UAV is now able to fly at a precise altitude as well as avoid obstacles which was the goal of the project. More time for testing the implemented system would have been necessary to be able to compare it better with other existing avoidance systems. Real flight tests of the avoidance system was not possible due to weather conditions and lack of time and equipment. Wind and rain was not expected to be such a big obstacle for testing the system but it really narrowed down the testing time. In spite this all the requirements for the system have been met and the timetable for the project schedule have been followed with exception for the testing of the avoidance system.

6.1 Altitude

The result shows that the altitude control system helps the UAV to keep a fix altitude to the ground regardless of the ground shape and it is able to compensate pitch and roll of the UAV. The overshoot when flying over an obstacle that can be seen in Figure 28 did not happen in the real flight test. The altitude system works good compared to the altitude sensor in the auto pilot (barometer) to keep a fixed and correct distance and also for smooth landing with good precision to ground. The data sheet of the sensor in the system did not had any information about the maximum working angle before the beam bounce. surprisingly the system was able to measure the distance even at angles bigger then 45° and compensate for that for getting the actual altitude according to the test result shown in figure 31. Due to a lot of wind during tests the maximum altitude and precision couldn’t be done with the system mounted on the UAV with a altitude more then 5m, in indoor environment the maximum range of the sensor was measured to a bit over 40 m with a precision of 10cm as written in the data sheet [A]. The result of the tests also indicates that the altitude system has to be restricted to a maximum altitude of 20 m due to the fact if the UAV is going over something such as a house with a height of almost the same as the current altitude of the UAV, in that case the UAV will incline until it reaches the altitude that was set from the user. When the house is passed the system will measure almost 40 m or even more if there is a pit in the ground. This could give problem for the altitude system according to the test result , where faults in the measurement has occur with in distance over 40 m shown in figure 14. If the maximum altitude of the system is restricted it should hopefully not measure an altitude over the maximum range of the sensor, if not the UAV is going over a cliff or something like that. Initially the plan was to restrict the velocity of the UAV to ensure that the sensor readings were correct. The tests however showed that it was not necessary since the system could compensate for bigger angles that ever will be with this UAV.

6.2 Obstacle avoidance

The result show that the system is able to avoid obstacles, at least if they are big and solid. Since the results have not been verified by real tests they can only be considered as experimental. The decision to use proximity sensors was made early in the project and since they do not provide any information about the shape of the obstacle only simple algorithms for avoidance could be used. Still, many hours were needed before an acceptable behavior was achieved. The task of actually flying around the obstacle proved to be much more complex then anticipated. Many of today’s UAVs that uses avoidance systems similar to the ones created in this project don’t actually fly around the obstacle, they just stop or
prevent the pilot from flying in a certain direction like DJI’s Phantom\(^7\). To be able to navigate around an obstacle and plan the flight path based on the shape of the obstacle is a very complex task that would require vision based sensors like the ones used on the Skydio R1\(^8\). Still, the implanted system shows that it is possible to navigate and avoid obstacles only with proximity sensors. In the simulation part of the project the system was just tested in a few different path along the positive X and Y. According to the test result of the heading direction\(^9\) the system had a problem that wasn’t found earlier. The problem occur when the UAV is going towards negative x directions due to the IMU compass in this direction has both 180/-180° defined. It was an unexpected surprise but the reason for the problem is found and also knowledge about why it happens.

### 6.3 Contingency plan

The first plan for the contingency plan was to subscribe on the battery topic /mavros/battery in each action that has been developed. When the battery drops under 30% and 20% of the full capacity the flight mode was supposed to be changed to ”AUTO RTL” and ”LAND” by commanding to a topic ”/mavros/state”. During the project the knowledge level of the autopilot was increased and parameters of how to set up different thresholds for the battery was found, so it did not have to be done in that way that was planned. The main part of the contingency plan became instead how to setup up the calibration parameters of the the sensor, there the empty cell voltage for each cell in the battery had to be included. No data sheet of the battery was found and no answer from the manufacture about this. According to this result of the contingency plan is very impressive it changes mode after the set threshold and the discharge characteristic that was done there an approximation of the empty cell voltage was more then good enough of 3.59V. The capacity of the battery in % given from the battery sensor of the UAV compared to the battery charger was almost the same, just a few % in difference. The result of the contingency plan\(^10\)\(^11\)\(^12\) indicates that is fulfills the project requirement, it keeps control of the battery status and change flight modes when the battery drops under the defined thresholds.

### 6.4 Micro controller

The developed micro controller has been included in all tests except the simulations and the contingency plan. The communication between all components and to the UAV seem to work without any problem in the traffic, no problem or error occured. If a similar construction would be developed again, an IMU sensor would be good to be embedded on the controller itself. This should make the system more stand alone and the pitch and roll data from the pilot would not be necessary to compensate for getting the actual altitude of the UAV before publishing into ROS.

\(^7\)https://www.dji.com/phantom-4-pro?site=brandsite&from=landing_page
\(^8\)https://www.skydio.com/2018/02/introducing-r1/
6.5 Social Impact

Extensive work is done these days to examine how UAVs can help in surveillance, search and rescue missions and explorations. They are effective in the way that they need no pilot, or they can be piloted from a distance meaning the pilot does not have to put himself in any danger if entering dangerous areas. They can search for missing persons or victims in a vast area in a short time. The fact that the UAV is somewhat expandable compared to human lives means they can enter areas that would not have been reached otherwise. In case the UAV needs to enter urban environments, an obstacle avoidance system could actually improve the UAVs’ capability of locating the victims or deliver medical supplies more safely if done autonomously. In future, areas that are most likely to be struck by natural events like earthquakes or volcano eruptions can have UAVs prepered to quickly fly out to an affected area to search for people and deliver medical supplies.
7 Conclusion

The goal of the project was to construct and implement an sensor system on an UAV with an already existing autopilot through ROS to improve it capability to keep an fix distance to ground and monitor the battery status of the UAV. An UAV with this system should also be able to detect and avoid obstacle with help of the sensors that has been chosen for the project. The results show that all the requirements set on the system has been achieved and in some cases exceeded. The altitude control system is able to handle varying ground and has greatly improved the autopilots ability to land safely and fly on a precisely given altitude. To achieve all this the project group started out by examining how this can be done by studying similar work done in the same field. Sonar sensors were selected for the obstacle detection and laser for the altitude measurements. When the properties of the sensors that were gonna be used was known the simulations and programming could start. Many hours were spend simulating different behaviors and setups to examine how sensors should be fitted on the UAV and how the sensor data could be used to command it. When the programmed behavior reached a acceptable stage the implementation part could begin. Instead of using a purchased development board a PCB was created from scratch that had all the features and communications that was desired for the chosen sensors. The PCB was assembled and programmed to filter the sensor data and then publish the data into ROS. The finished hardware could potentially be used by any similar project where proximity measurements are desired. Different types of mounts have also been constructed to examine where the sensors are best placed on an UAV for this type of application.

Many simulation scenarios has been tested for both the altitude and avoidance system. To verify and evaluate the result better, more time would have been desired for the obstacle avoidance part. More knowledge about how to evaluate the flight results would also have been desirable. The altitude system was given more time since it was more important. Without it, the UAV would not be able to even land safely with precision.

Overall the system performs as desired and the requirements have been met. The project has been very educational for the members who now have a much greater knowledge when it comes to constructing and testing robotic applications. The finished hardware is generic and has great potential to be used in future research.
References


Appendices
### Physical Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (LxWxH)</td>
<td>20 × 48 × 40 mm (0.8 × 1.9 × 1.6 in.)</td>
</tr>
<tr>
<td>Weight</td>
<td>22 g (0.78 oz.)</td>
</tr>
<tr>
<td>Operating temperature</td>
<td>-20 to 60°C (-4 to 140°F)</td>
</tr>
</tbody>
</table>

### Electrical Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>5 Vdc nominal</td>
</tr>
<tr>
<td></td>
<td>4.5 Vdc min., 5.5 Vdc max.</td>
</tr>
<tr>
<td>Current consumption</td>
<td>105 mA idle</td>
</tr>
<tr>
<td></td>
<td>135 mA continuous operation</td>
</tr>
</tbody>
</table>

### Performance Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range (70% reflective target)</td>
<td>40 m (131 ft)</td>
</tr>
<tr>
<td>Resolution</td>
<td>±0.1 cm (0.4 in.)</td>
</tr>
<tr>
<td>Accuracy &lt; 5 m</td>
<td>±2.5 cm (1 in.) typical*</td>
</tr>
<tr>
<td>Accuracy ≥ 5 m</td>
<td>±10 cm (3.9 in.) typical</td>
</tr>
<tr>
<td></td>
<td>Mean ±1% of distance maximum</td>
</tr>
<tr>
<td></td>
<td>Ripple ±1% of distance maximum</td>
</tr>
<tr>
<td>Update rate (70% Reflective Target)</td>
<td>270 Hz typical</td>
</tr>
<tr>
<td></td>
<td>600 Hz fast mode**</td>
</tr>
<tr>
<td></td>
<td>&gt;1000 Hz short range only</td>
</tr>
<tr>
<td>Repetition rate</td>
<td>~50 Hz default</td>
</tr>
<tr>
<td></td>
<td>500 Hz max</td>
</tr>
</tbody>
</table>

*Nonlinearity present below 1 m (39.4 in.)

**Reduced sensitivity

### Interface Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface</td>
<td>I2C, PWM, External trigger</td>
</tr>
<tr>
<td>I2C interface</td>
<td>Fast-mode (400 kbit/s), Default 7-bit address 0x62, internal register access &amp; control</td>
</tr>
<tr>
<td>PWM interface</td>
<td>External trigger input, PWM output proportional to distance at 10 μs/cm</td>
</tr>
</tbody>
</table>

### Laser Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavelength</td>
<td>905 nm (nominal)</td>
</tr>
<tr>
<td>Total laser power (peak)</td>
<td>1.3 W</td>
</tr>
<tr>
<td>Mode of operation</td>
<td>Pulsed (256 pulse max. pulse train)</td>
</tr>
<tr>
<td>Pulse width</td>
<td>0.5 μs (50% duty Cycle)</td>
</tr>
<tr>
<td>Pulse train repetition frequency</td>
<td>10-20 KHz nominal</td>
</tr>
<tr>
<td>Energy per pulse</td>
<td>&lt;280 nJ</td>
</tr>
<tr>
<td>Beam diameter at laser aperture</td>
<td>12 × 2 mm (0.47 × 0.08 in.)</td>
</tr>
<tr>
<td>Divergence</td>
<td>8 mRadian</td>
</tr>
</tbody>
</table>

### Wiring Harness

<table>
<thead>
<tr>
<th>Wire Color</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>5 Vdc (+)</td>
</tr>
<tr>
<td>Orange</td>
<td>Power enable (internal pull-up)</td>
</tr>
<tr>
<td>Yellow</td>
<td>Mode control</td>
</tr>
<tr>
<td>Green</td>
<td>I2C SCL</td>
</tr>
<tr>
<td>Blue</td>
<td>I2C SDA</td>
</tr>
<tr>
<td>Black</td>
<td>Ground (-)</td>
</tr>
</tbody>
</table>

There are two basic configurations for this device:

- **I2C (Inter-Integrated Circuit)**—a serial computer bus used to communicate between this device and a microcontroller, such as an Arduino board (“I2C Interface”, page 4).
- **PWM (Pulse Width Modulation)**—a bi-directional signal transfer method that triggers acquisitions and returns distance measurements using the mode-control pin (“Mode Control Pin”, page 4).

### Connector

You can create your own wiring harness if needed for your project or application. The needed components are readily available from many suppliers.

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
<th>Manufacturer</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector housing</td>
<td>6-position, rectangular housing, latch-lock connector receptacle with a 1.25 mm (0.049 in.) pitch.</td>
<td>JST</td>
<td>GHR-06V-S</td>
</tr>
<tr>
<td>Connector terminal</td>
<td>26-30 AWG crimp socket connector terminal (up to 6)</td>
<td>JST</td>
<td>SSHL-002T-P0.2</td>
</tr>
<tr>
<td>Wire</td>
<td>UL 1061 26 AWG stranded copper</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Connector Port Identification

1. 5 Vdc (+)
2. Power enable (internal pull-up)
3. Mode control
4. I2C SCL
5. I2C SDA
6. Ground (-)
MB 1202 Sensor

MB1202: I2CXL-MaxSonar-EZ0
The I2CXL-MaxSonar-EZ0 is the highest sensitivity and widest beam sensor of the I2CXL-MaxSonar-EZ sensor series. The wide beam makes this sensor ideal for a variety of applications including people detection, autonomous navigation, and wide beam applications.

MB1202
I2CXL-MaxSonar®-EZ0™ Beam Pattern
Sample results for measured beam pattern are shown on a 30-cm grid. The detection pattern is shown for dowels of varying diameters that are placed in front of the sensor.

A 6.1-mm (0.25-inch) diameter dowel
B 2.54-cm (1-inch) diameter dowel
C 8.89-cm (3.5-inch) diameter dowel
D 11-inch wide board moved left to right with the board parallel to the front sensor face.

This shows the sensor’s range capability.

Note: For people detection the pattern typically falls between charts A and B.

MB1202 Features and Benefits
- Widest and most sensitive beam pattern in I2CXL-MaxSonar-EZ line
- Low power consumption
- Easy to use interface
- Will pick up the most noise clutter of any of the sensors in I2CXL-MaxSonar-EZ line
- Detects smaller objects
- Best sensor to detect soft object in I2CXL-MaxSonar-EZ line
- Can be powered by many different types of power sources
- Can detect people up to approximately 14 feet

MB1202 Applications and Uses
- Great for people detection
- Security
- Motion detection
- Landing flying objects
- Useable with battery power
- Autonomous navigation
- Educational and hobby robotics
- Collision avoidance
C  MB 1242 Sensor

MB1242: I2CXL-MaxSonar-EZ4
The I2CXL-MaxSonar-EZ4 has the highest noise tolerance and the narrowest beam of any of our indoor sensors. The sensor is calibrated and tested to provide stable range readings to large targets even in electrically and acoustically noisy environments.

MB1242
I2CXL-MaxSonar®-EZ4™ Beam Pattern
Sample results for measured beam pattern are shown on a 30-cm grid. The detection pattern is shown for dowels of varying diameters that are placed in front of the sensor.
A 6.1-mm (0.25-inch) diameter dowel
B 2.54-cm (1-inch) diameter dowel
C 8.89-cm (3.5-inch) diameter dowel

Note: The maximum detected distance for 3.3V is ~500cm and for 5V is ~640cm

A 5.0 V
B 3.3 V

Beam Patterns drawn to a 1:95 scale for easy comparison to our other products.

MB1242 Features and Benefits
• Narrowest beam sensor in I2CXL-MaxSonar-EZ line
• Low power consumption
• Easy to use interface
• Rejects small clutter
• Can be powered by many different types of power sources

MB1242 Applications and Uses
• Security
• Motion detection
• Landing flying objects
• Useable with battery power
• Autonomous navigation
• Educational and hobby robotics
• Collision avoidance
## List of components

<table>
<thead>
<tr>
<th>Component</th>
<th>Capsule</th>
<th>Price per unit (sek)</th>
<th>Quantity</th>
<th>Price (sek)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmega 328p TQFP-32</td>
<td></td>
<td>20,50 kr</td>
<td>1,00</td>
<td>20,50 kr</td>
</tr>
<tr>
<td>USB/UART SSop-28</td>
<td></td>
<td>41,50 kr</td>
<td>1,00</td>
<td>41,50 kr</td>
</tr>
<tr>
<td>Micro USB SMD conn SMD</td>
<td></td>
<td>18,56 kr</td>
<td>1,00</td>
<td>18,55 kr</td>
</tr>
<tr>
<td>Capacitor 100nF 0603</td>
<td></td>
<td>0,15 kr</td>
<td>5,00</td>
<td>0,75 kr</td>
</tr>
<tr>
<td>Capacitor 680uF SMD</td>
<td></td>
<td>7,77 kr</td>
<td>1,00</td>
<td>7,77 kr</td>
</tr>
<tr>
<td>Oscillator 16Mhz with load capacitors SMD</td>
<td></td>
<td>3,89 kr</td>
<td>1,00</td>
<td>3,89 kr</td>
</tr>
<tr>
<td>Green Led 0603</td>
<td></td>
<td>6,00 kr</td>
<td>1,00</td>
<td>6,00 kr</td>
</tr>
<tr>
<td>Reed Led 0603</td>
<td></td>
<td>6,00 kr</td>
<td>1,00</td>
<td>6,00 kr</td>
</tr>
<tr>
<td>Resistor 1k 0603</td>
<td></td>
<td>0,15 kr</td>
<td>2,00</td>
<td>0,30 kr</td>
</tr>
<tr>
<td>Resistor 4.7k (optional) 0603</td>
<td></td>
<td>0,15 kr</td>
<td>2,00</td>
<td>0,30 kr</td>
</tr>
<tr>
<td>Resistor 330 0603</td>
<td></td>
<td>0,15 kr</td>
<td>2,00</td>
<td>0,30 kr</td>
</tr>
<tr>
<td>Push Button SMD</td>
<td></td>
<td>4,90 kr</td>
<td>1,00</td>
<td>4,90 kr</td>
</tr>
<tr>
<td>schottky diod 0.5A SOD-123</td>
<td></td>
<td>2,79 kr</td>
<td>1,00</td>
<td>2,79 kr</td>
</tr>
<tr>
<td>Molex 53047 4pin THD</td>
<td></td>
<td>3,69 kr</td>
<td>4,00</td>
<td>14,76 kr</td>
</tr>
<tr>
<td>Resetable Fuse 500mA SMD</td>
<td></td>
<td>5,07 kr</td>
<td>1,00</td>
<td>5,07 kr</td>
</tr>
</tbody>
</table>
E  Schematic och the UAV senscontroller