Part C

Sensors

C.1 Ultrasonic sensor

Ultrasonic sensors are transducer devices that can convert ultrasound wavefronts to electrical signals. To illustrate, a sound is emitted by the sensor, then it bounces on the surrounding environment and comes echoing back. This sound takes time to travel distances. The further the distance it needs to travel, the longer time it takes. After that, the sonar's microcontroller, which keeps track of the time that passes, can calculate the distance of the reflective objects detected and it produces an appropriate voltage to the host circuit.

C.1.1 Devantech SRF02 Ultrasonic Range Finder

SRF02 is a single transducer ultrasonic rangefinder in a small footprint PCB that supports both I2C and Serial interfaces (fig.C.1). According to the specifications, the minimum measurement range is approximately 15cm and the maximum 6m with 3-4cm resolution. Resolution is simply the smallest measurement increment that the device is capable of reporting. It's the same as the smallest increment of measurement on a tape measure or ruler. SRF02 can report the distance in cm, inches or uS.

Fig.C.1 Devantech SRF02 Ultrasonic range finder

The advantages of this device can be summarized in the followings points:

- Low-cost
- Light-weight
- Low-power consumption: 5V, 4mA Typ.
- The use of serial bus, which can be connected up to 16 devices to any uP or UART serial port, since individual addresses can be allocated.
- There is no need for calibration since it supports full automatic tuning.

In order to test the SRF02, experiments in various shape spaces were conducted to verify the behaviour related to the shape of the lobe. In the first stage of experimentation there is no noise caused by airflow (wind).

Table C.1

(All measurements taken with a surface perpendicular to the sonar)

Distance (cm)	**Measured Distance (cm)**	**Difference (cm)**
10	27	17 (dead zone)
30	29	3
80	78	2
130	128	2
170	169	1
210	207	3
310	306	4
400	396	4
500	495	5
600	595	5
> 600	0	NaN

By examining the table (tbl.C.1) the following conclusions are apparent:

- Reflective objects closer than the minimum are in the sensor's dead zone.
- For distances lower than 2m, it has an accuracy of 2cm. For distances longer than 2m, it has an error approximately 3 to 5cm.
- For distances greater than the maximum, it returns 0.

There is huge variation of the measurements according to the angle in which the reflective objects is hit by the sonar. The sensor can not accurately measure the distance of an object that has its reflective surface at a shallow angle, as a result the sound will not be reflected back towards the sensor. The other case is when the object is too small to reflect enough sound back to the sensor. The sensor's behaviour becomes clear by examining the beam pattern in (fig.C.2)

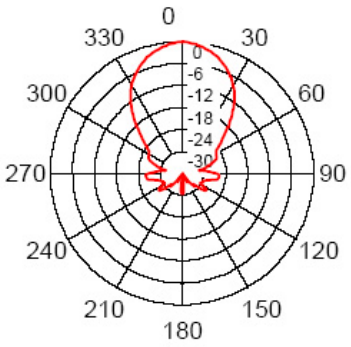

Fig.C.2 The SRF02's beam pattern, showing the sensitivity of the transducer in dB (image from robot-electronics.co.uk)

To experiment with the different behaviour, I performed a range of measurements in different angles. The set-up is illustrated in figure (fig.C.3).

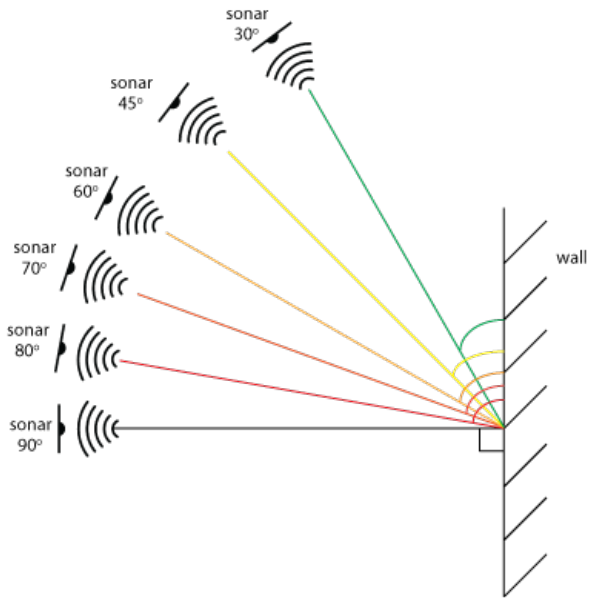

Fig.C.3 Measured distances in different angles to the wall

As it is illustrated by the beam pattern, in angles larger than 30 degrees the sensors sensitivity has already been reduced 8 times. The situation deteriorates for angles larger than 60 degrees where the performance of the sonar sensor is unacceptable. The table below (tbl.C.2), based on the experiments, verifies what I have described.

Table C.2

(Measurements taken in different angles towards the wall)

Distance (cm)	Angle (degrees)	Measured distance (cm)	Difference (cm)
100	90	98	2
100	80	97	3
100	70	94	6
100	60	90	10
100	45	145	45
100	30	149	49

An additional unwanted behaviour that the SRF02 displayed in the experiments was the wrong measurements due to the beam width. As an example, if the sensor is used in a 1m-width corridor, even though the distance in front will be clear and the wall perpendicular to the sensor, the measured distance will most probably be approximately 124cm. This is clearly illustrated in figure (fig.C.4).

I conducted two experiments that verified this behaviour. The set-ups are depicted in (fig.C.5). In both cases the measured distance were incorrect. The sonar detects the echo by listening for the returning wavefronts. In our case, points that were in periphery and not in the target responded with strong wavefronts and the sonar reported incorrect measurements. Another possible reason could be the phasing effect where the echo does not come from a point source. For example, in the case where a wall is in front of the sensor, not only the central reflection is encountered but also the reflections

from the surrounding area are slightly behind of the central. The sum of all reflections, which the sensor receives, can be either strengthened or weakened by phasing effects. Therefore, if the echo is fainted then it might be the following wavefront that is detected (Robot-electronics.co.uk, 2015).

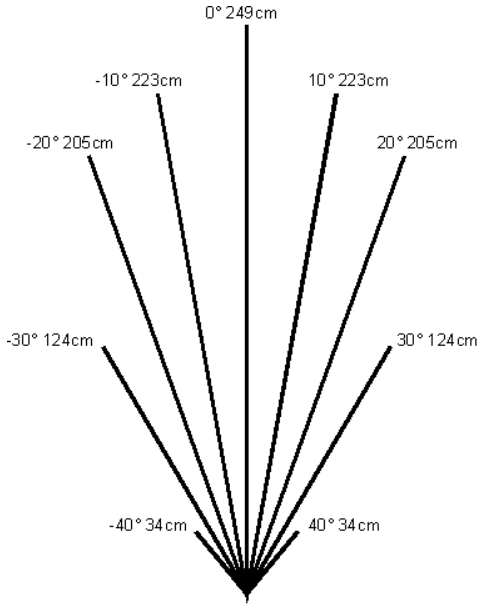

Fig.C.4 Measured beam pattern for the SRF02 (image from robot-electronics.co.uk)

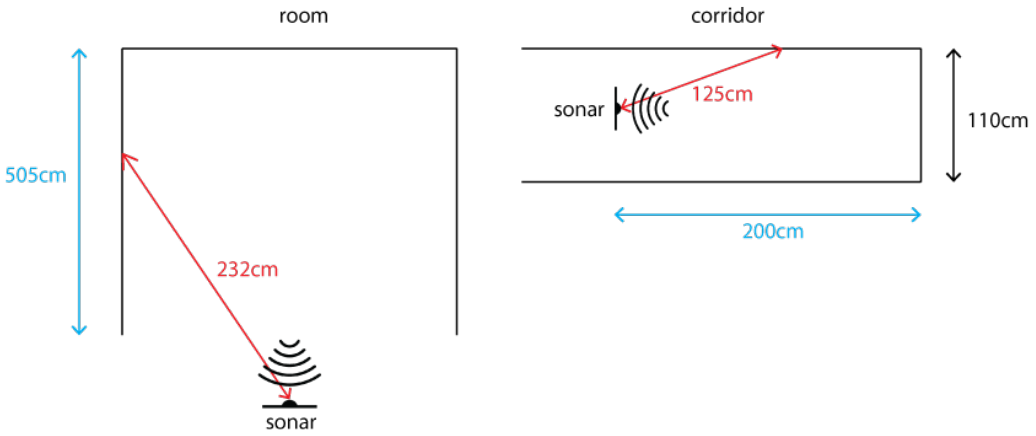

Fig.C.5 Side lobe points lead to incorrect distance measurements even though the space in front is clear and the reflective surface perpendicular aligned.

The second experiment using an ultrasonic sensor was to explore its behaviour related to the wind. It is an important factor since it will be used outdoors, mounted on a UAV and the acoustic frequencies might be affected by it. Inside a lab, a simple way to simulate wind was to use a pedestal fan.

Under certain conditions, it produces airflow, not too focused but in a certain directions. I tested the setup with the fan, in highest speed, in different positions related to the ultrasonic sensor. I was cautious enough not to place the fan in a point where it could be identified as an object and thus could alter the results because of the presence of an object and not due to the airflow (fig.C.6).

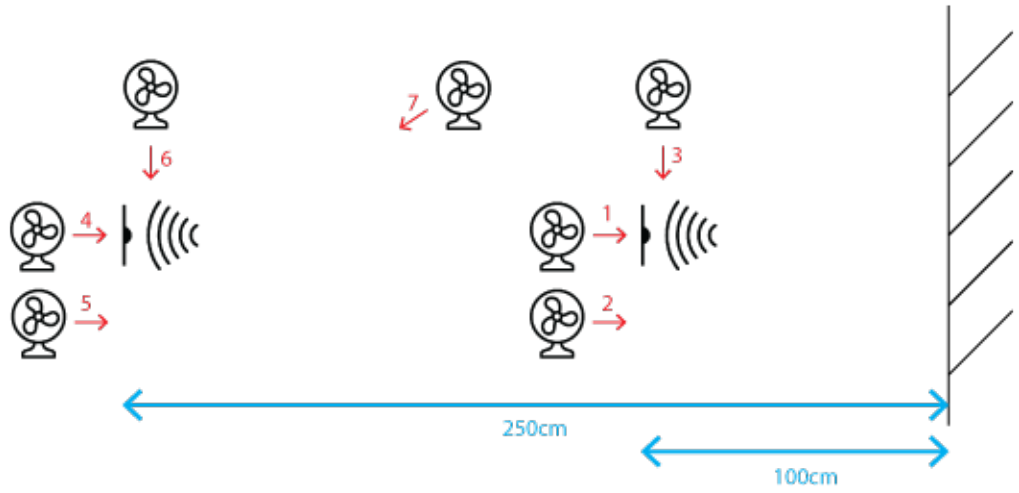

Fig.C.6 Experiments of ultrasonic sensor with the presence of airflow (the fan was operating at the highest speed)

The outcomes of the described experiment proved the stability of the sensor in most of the cases (tbl.C.3). More precisely, when the airflow was not directed towards the ultrasonic sensor, the influence was minor (positions 1-6). The only interesting behaviour was observed in position 7. In this position, when the fan was switched-off, the measurement was always 250cm. Whenever the fan was turned on, some of the measurements (approx. 1 every 5) reflected the distance between point 7 and the sensor, including an error. Generally speaking, it seems that the ultrasonic sensor is affected only when the airflow is directed towards the sensor and it is near or inside its sensitivity field. Unfortunately, I couldn't move the fan in many other positions since it was sensed as an object. Moving the sensor to greater distances was not attempted since the sensor itself is prone to side lobes and the measurements would not be stable.

Table C.3

(Measurements with a source of airflow in different positions and directions)

Distance (cm)	Position	Measured distance (cm)	Difference (cm)
100	1	98	2
100	2	100	0
100	3	103	3
250	4	248	2
250	5	250	0
250	6	247	3
250	7	249*	1*

C.2 Infrared sensor

Infrared sensors base their functionality on the emission of light. This sensors use Light-Emitting Diodes (LEDs) as a source of light. The sensor detects a specific light wavelength in the infrared spectrum. Any light with wavelengths longer than 700 nm is called infrared and is not normally visible. The LED produces light at the same wavelength as what the sensor is able to receive. Moreover, it can calculate the distance by measuring the intensity of the received light. For instance, when an object is close to the sensor, the LED light bounces off the object and returns into the light sensor.

C.2.1 Sharp GP2Y0A02YK0F

The Sharp's infrared red sensor (fig.C.7) consists of an integrated implementation of a PSD (Position Sensitive Detector), IRED (Infrared Emitting Diode) and signal processing circuit. It exploits the triangulation method to detect the distance. Because of that, it is not affected by the environmental temperature and the operating duration.

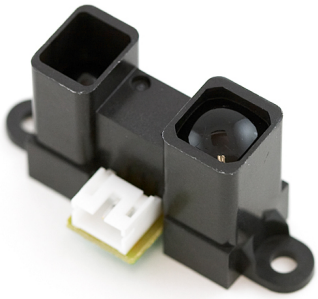

Fig.C.7 Infrared proximity sensor Sharp GP2Y0A02YK0F (image from sparkfun.com)

The key features of the sensor are the following:

- Distance range: 20 to 150 cm
- Analog output type
- Consumption: 5V, 33mA Typ

Various experiments were performed with the IR sensor to test its behaviour in different distances and situations. The conversion between the voltage and the distance is not a straightforward issue. According to the specifications (GP2Y0A02YK0F, 2006), the measurements' characteristics of the output follow a certain graph given in figure (fig.C.8)

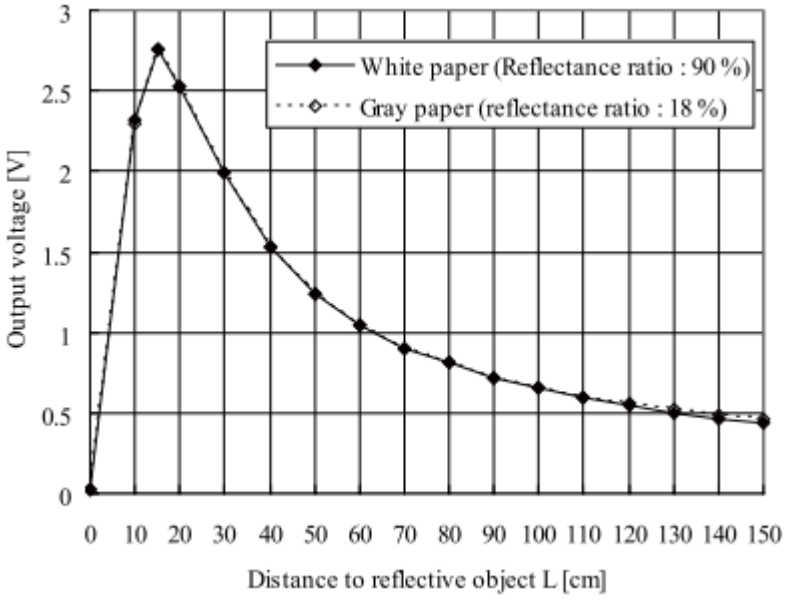

Fig.C.8 The relation of the output voltage with the distance for the Sharp IR sensor

(image from (GP2Y0A02YK0F, 2006))

The relation of the output voltage over the distance is not linear. One approach could be to use this graph and attempt to find the best fitting line equation. To facilitate the coding, the output voltage is converted to values from 0-1023, which is the analog reads that the Arduino measures (5v/1024 units). Matlab was used to draw the graph and evaluate the corresponding best-fit equation (fig.C.9). Using a two-term power series model, the estimated equation is the following:

$$V = 3965d^{-0.6257} - 86.46$$

Where V is the output voltage and d is the corresponding distance. An approximate solution of the above equation regarding the distance is the following:

$$d \approx \frac{2.7655 \cdot 10^8}{50(50v + 4323)^{0.593625} \cdot v + 4323(50v + 4323)^{0.593625}}$$

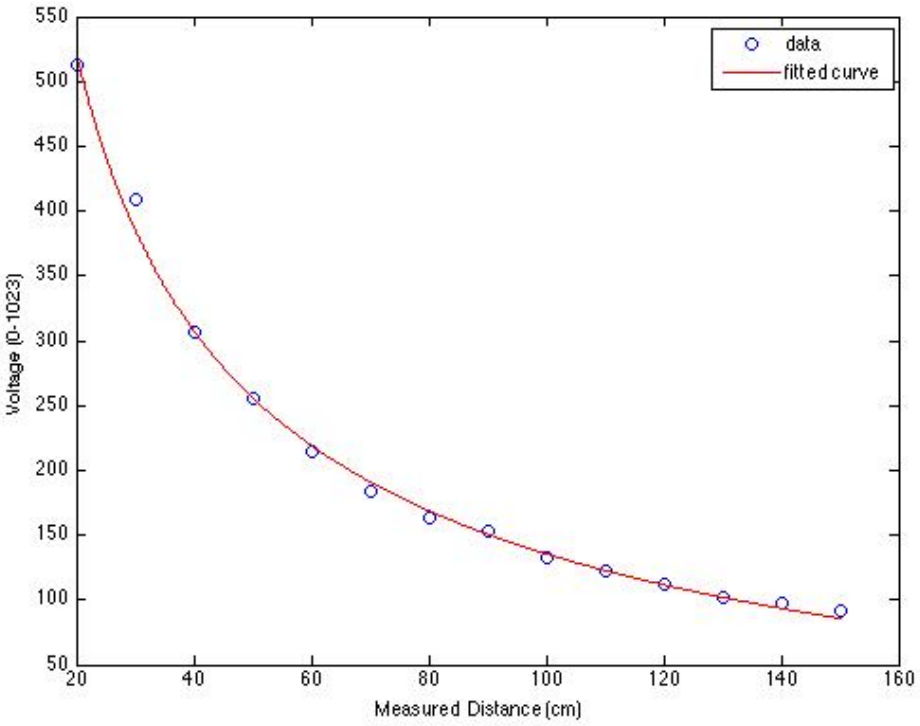

Fig.C.9 Best fit equation for the non-linear relation of voltage and measured distance

Using this equation, a range of measurements was performed as shown on table (tbl.C.4). The set up was the IR sensor pointing towards a yellow wall. The performance was not acceptable. The error in distances above 100cm is huge. This behaviour could be justified because of the following reasons:

- The best-fit approximation in the range from 80 to 150cm is not well adjusted. This happened because the Sharp's graph is not so detailed and minor changes in the voltage correspond to major changes in the distance. Consequently the resolution is poor. Moreover, an approximate solution was used in order to calculate the distance from the voltage since the exact was too complex.
- The specifications mention that it is recommended to stabilize the power supply line, by inserting a by-pass capacitor of 10μF or more between V_{cc} and GND. Unfortunately, it was not implemented.

Table C.4

(All measurements taken with a surface perpendicular to the sonar)

Distance (cm)	**Distance measured (cm)**	**Difference (cm)**
13	18	(Dead zone)
30	30	0
50	53	3
80	89	9
110	128	18
140	164	24

Similar behaviour with the sonar was observed in the cases where the objects were not perpendicular aligned with the IR sensor. In addition, an influencing factor was the reflectivity of the object. For example, white paper has a reflectance ratio of 90% while grey paper 18%. This is particularly important for distances above 110cm where the sensor's resolution is poor.

In conclusion, the performance of the sensor could be improved if a voltage to distance graph made from scratch and not based on the one shown in the specifications. This would facilitate accurate correspondences between the pairs instead of having to guess from the graph, especially in the low-resolution part. In addition, since as it is analog, the output is depended of the stability of the input voltage and the flatter is the input the less noise will be present in the output. Nevertheless, the range of the IR sensor is very restricted and since it is not suited to the UAV applications, it is not worth spending more time on it.

C.3 Optical flow sensor

Optical flow sensors are mainly utilized to provide precise measurements of ground speed and therefore the position of mobile robots. They are particularly useful in GPS denied environments allowing long-term dead reckoning navigation.

C.3.1 Px4flow smart camera

One of the first attempts to use this type of sensors in the UAVs was based on mouse sensors. The mouse sensor returned the average movement of the visible surface's features. In order to convert these values into real distances, which correspond to the movement, the altitude should be known. By combining the measurement of a barometer or an ultrasonic sensor, this distance can be calculated. Other factors that affect the optic flow are the vehicle's roll and pitch changes. If these rotations have known values, reported from the IMU for example, it can be introduced to the calculations and provide the correct distances covered.

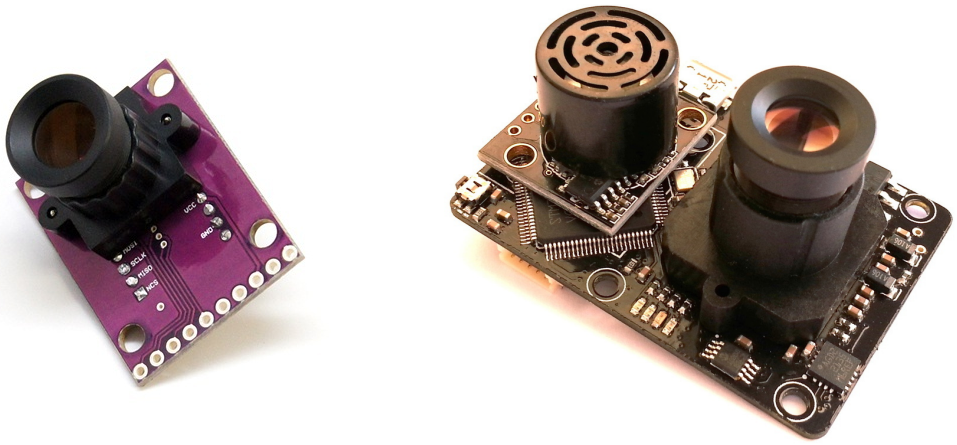

Fig.C.10 Mouse-based optical sensor and p4flow smart camera (images from copter.ardupilot.com)

The main disadvantages of using mouse-based optical flow sensors are the following:

- They operate satisfactorily only in well-lit environments. In typical indoor and outdoor low-light conditions, their practical use is limited.
- They need additional devices to be present such as sonar and IMU to have all the required data for the calculations.
- They require a fast MCU (microcontrollers) for the calculations. The flight controller is already used and loaded with the coordination operation of all the other connected devices.

A highly improved optical flow sensor that can support more advanced operation scenarios is the px4flow (fig.C.10). The main features that improve its functionality can be summarized as follows:

- It has a CMOS image sensor with a 752×480 resolution having a very high light sensitivity.
- It is equipped with an ARM Cortex M4 CPU customized with a special imager bus peripheral for real-time image processing reaching a maximum rate of 250Hz in optical flow calculations.
- It has a high precision on-board gyroscope to compensate angular velocity.

- It is equipped with an on-board ultrasonic range sensor to measure the distance towards the surface and to scale optical flow values to metric velocity values.

Up to this point I have explained that the optical flow sensors can measure the ground speed or translational velocity. This is based on the following equation:

$$\upsilon_{trans} = \upsilon_{optic} \frac{Z}{f}$$

where Z is the current distance to the scene, f the focal length and υ the optical flow. It is assumed that the rotational velocity is ether zero or known, measured by the gyroscope. An interesting aspect of this equation is that it can be used to calculate, with high accuracy, the altitude instead of the translation speed if the latter is known. The ground speed can be known either using the GPS or the airspeed measurements.

Before moving to the experiments regarding the px4flow smart camera it is important to cite some restricting factors of the optical flow sensors in general. The optical flow equation (OFE) is defined as:

$$I_x \upsilon_x + I_y \upsilon_y + I_t = 0$$

where I_x, I_y and I_t are the spatial and temporal derivatives, respectively, of the image sequence I at a given point, and (υ_x, υ_y) is the 2D velocity at that point. The underlying model on which OFE is based on is that the luminance remains constant between frames along the motion trajectory, so that the total derivative of the image function is zero, hence the OFE. The OFE is one equation with two unknowns, υ_x and υ_y. Hence further constraints are needed to obtain a solution. The Lucas and Kanade method (LS) makes the assumption that the motion field within a small window is constant and this constraint is used with the OFE to construct a least squares problem and

hence estimates of the vector (υ_x, υ_y). All points in the region are assumed to have the same velocity, hence giving multiple equations in 2 unknowns, enabling a LS solution to be obtained. All in all, the method should work well within textured regions with anisotropic intensity variation, which means changing in all directions, and in which the motion is locally constant and can be approximated as such.

There are various tests that can be carried out using this sensor. They can be divided to two categories, tests related to the ground speed estimates and tests related to the altitude. Unfortunately, for the first category the testing configuration is rather complicated compared to the other sensors. The best way to assess the ground speed is to compare them with data computed using a VICON motion tracking system. Alternatively, the ground speed can also be compared with the corresponding values obtained using an on board GPS unit. Considering that the calculated ground speed will not be exploited from the radiation mapping UAV and that the testing is time consuming, it was decided not to proceed with this procedure. However, the device will be part of the configuration for the UAV's altitude hold mode and as such is going to be tested for the purposes of my project.

The px4flow device is equipped with a high quality ultrasonic range finder, the Matbotix HRLV Maxsonar EZ4. The specifications cite that it supports 1 millimetre resolution having a maximum range of 5000mm and a dead zone at 300mm. The special property of EZ4, which makes the difference from the competitors, is the narrow beam width providing the least sensitive to side objects. Consequently, the HRLV-MaxSonar-EZ4 is a very good choice when only larger objects need to be detected since it may ignore some small and medium targets. Another advantage of using the EZ4 is that it provides the most noise tolerant acoustic sensitivity compared to the other ultrasonic sensors of the same category. The Matbotix EZ4 range finder supports three different interfaces, analog Voltage, RS232 Serial and Pulse Width. However, since the sensor is part of the px4flow, the commination has to be accomplished through this. As it is described in the special section "Sensors

Cirtuits", I utilize two individual ways to access the data, Adruino microcontroller and ROS packages.

Similarly with the Devantech SRF02 sensor I conducted various experiments to verify the accuracy of the sensor (tbl.C.5). The setup of the experiment was the sensor facing a wall in the perpendicular direction. The measured distances were very precise. Above a dead zone of the 30cm, the performance was excellent. A very interesting aspect in the behaviour of the sensor was the narrow beam width. I repeated the same corridor test like with the SRF02. It is very impressive that the EZ4 sensor was measuring the correct distance till the value of 350cm (fig.C.11).

Table C.5

(All measurements taken with a surface perpendicular to the sonar)

Distance (cm)	Distance measured (cm)	Difference (cm)
25	30	(Dead zone)
40	40	0
60	60	0
100	101	1
150	151	1
200	201	1
300	301	1

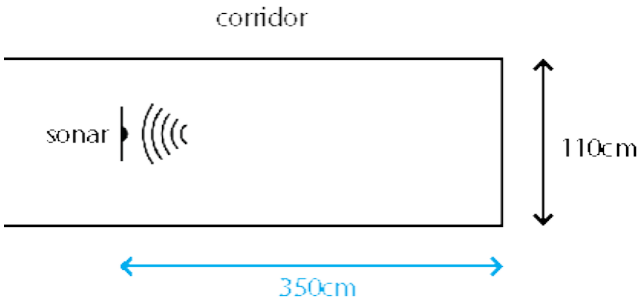

Fig.C.11 Matbotix EZ4 tested in a corridor experimenting with the beam width.

C.4 Laser sensor

Laser sensors belong to the high-end devices of the proximity sensors. They are more expensive than the others but they offer higher ranges with enhanced accuracy. Unlike infrared LED sensors, the semiconductor laser emits light from a very tiny area in only one direction and at one wavelength or a few closely spaced together.

C.4.1 LIDAR-Lite

PulsedLight's laser is an innovative device. It offers competitive features in low price, compared to other laser sensors, and with the ease of I2C or pulse width modulation (PWM) connectivity. According to the specifications, it supports up-to 40-meter range capability with 2.5cm accuracy, 1cm resolution and fast acquisition rate equals to 50Hz. In addition, it has a small size, low power consumption and it is lightweight (fig.C.12).

In the manufacturer's site it is stated that the success of the LIDAR-Lite is based on the specific implementation. They use a customised version of the Time-of-Flight distance measurements approach, originating in military applications. These systems transmit relatively high power infrared pulses using a laser diode. The distance estimation is accomplished by measuring the time delay between the transmitted pulse and the detected return pulse. In general, it is a long-range operation with low accuracy. In order to improve the accuracy of these systems they had to use signal processing. The challenge was to avoid complex and expensive hardware to achieve this as other existing systems implement it. PulsedLight manage to develop a signal processing technique using a simple approach without the complexity and cost of direct analog–to–digital conversions (Technology Brief, 2015).

Fig.C.12 LIDAR-Lite Laser Rangefinder (image from sparkfun.com)

The experiments of the sensor was conducted in various distances and angles. In the first table (tbl.C.6), where the reflective objects were aligned perpendicular, the performance followed the specifications, showing accuracy close to 1-2cm. In greater distances like the 10m, I should admit that even my verification method, using a tape measure, was error-prone so the accuracy is negotiable but in favour of the device. Especially for the 10m measurement, it should be mentioned that it was conducted in a corridor (1.1 meters wide). Consequently, in contrast with the sonar where the wide acoustic lobe is problematic, laser sensors have a concentrated beam and perform uninfluenced of the location's shape.

Table C.6

(Reflective objects are perpendicular aligned)

Distance (cm)	Distance measured (cm)	Difference (cm)
30	34	4 (close to dead zone)
60	60	0
100	100	0
250	250	0
400	402	2
500	501	1
1050	1040	10
> 4000	0	NaN

Another useful aspect that emerged from the tests was the behaviour regarding the material of the object and the hit angle. The following table (tbl.C.7) shows that glass degraded the performance only when the beam was hit in angle. On the other hand, when the material was concrete the hit angle was not an influencing factor.

Table C.7

(Measurements in different angles and materials)

Distance (cm)	Angle (degrees)	Material	Distance measured (cm)	Difference (cm)
100	90	Concrete	100	0
100	45	Concrete	101	1
100	90	Glass	101	1
100	45	Glass	135	35

To conclude with, LIDAR-lite showed an impressive performance compared with the previous tested sensors. It offers a long range, good accuracy and its behaviour is not restricted to the surrounding environment.

C.5 Applications

An interesting and creative task is to use the sensors to build more complicated and useful devices, which can be used in localization and navigation applications in robotics. The most versatile and promising sensor to use from the previous presentations is the LIDAR-Lite. The px4flow was delivered as an autonomous operating package ready to use and does not allow any space for modifications.

The LIDAR-Lite is a reliable laser sensor having high reading rate and accuracy. It can be used as a simplified version of a lidar that essentially provides laser scans. To leverage the full features of the LIDAR-lite, I needed

to combine it with a microcontroller and a servomotor. The most popular choice was to utilize an Arduino Uno with a small yet classic servo in inexpensive robotic applications Hitec HS-422 Deluxe (fig.C.13).

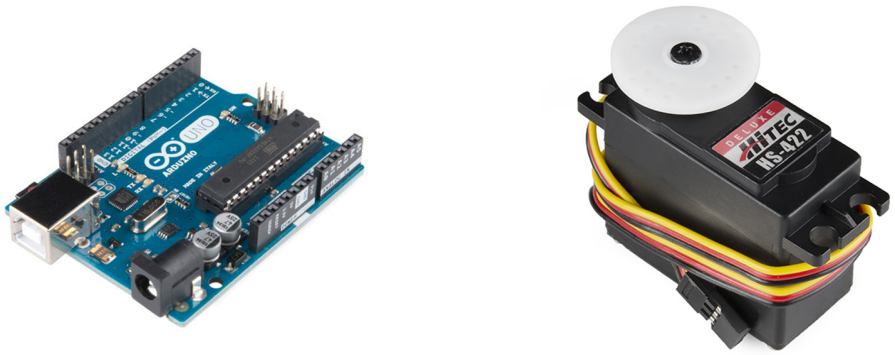

Fig.C.13 Arduino Uno and Hitec's HS-422 Deluxe (image from sparkfun.com)

C.5.1 Mapping application

One of the most common applications of a laser scanner is the mapping. To map the surroundings, I need a servo to rotate the laser in continuous angles and measure the related distances. In order to accomplish this task the laser was mounted on the servo using a specific bracket (fig.C.14). This mechanical construction was pretty stable although the integration of 3D printed case could offer a more compact design for easy use in robotics applications.

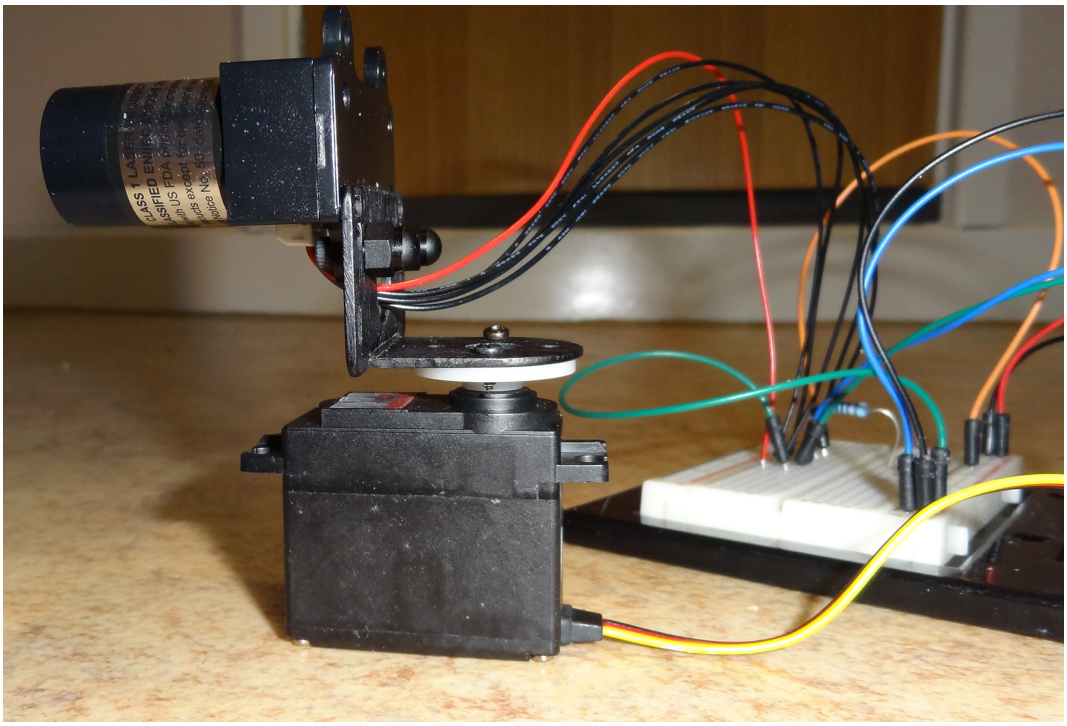

Fig.C.14 Mapping device consists of a servo, a mounting bracket and a laser sensor

The structure of this task consists of two working parts (fig.C.15). For the first part an Arduino sketch has been developed which is responsible for the following functions:

- To read the angle of the servo
- To measure the related distance. More precisely, it gets two measurements and provides the mean value of them so as to avoid a fair amount of noise.
- To send both data to serial port with a rate of 9600bps

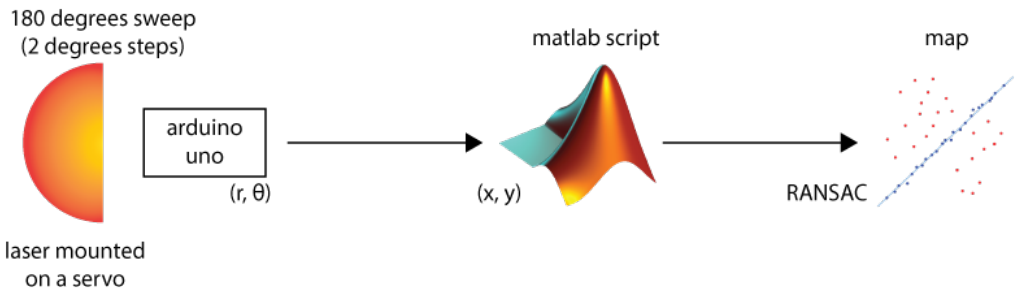

Fig.C.15 The logic flow of the proposed mapping algorithm

This part is attached to the hardware and utilizes the microcontroller to gather the useful information and transmit it to the other end for processing. The servo rotates the laser forward and backward by 180 degrees. Every 2 degrees it stops and performs the tasks I have previously described. Consequently in a loop, the total number of readings is equal to 180, the first 90 taken during the anticlockwise transition from 0 to 180 degrees and the remaining 90 during the clockwise rotation.

On the other side, a Matlab script has been developed as a client. This script takes input from the serial port using the USB, which is connected to the Arduino Uno. Particularly, the script accomplishes the following tasks:

- Reads the angle and the related distance
- Filters values outside the LIDAR-Lite range
- Converts from polar coordinates to Cartesian, x and y
- Feeds the points to a RANSAC method to detect potential lines in the data.

This script exploits the measurements to yield useful patterns in the space that could be the base of a map. It makes use of the random sample consensus (RANSAC) algorithm (Fischler and Bolles, 1981), which is an iterative method to estimate parameters of a mathematical model from a set of observed data, which contains outliers. The assumption is that the data consists of inliers and outliers. By referring inliers I mean data whose distribution can be explained by some set of model parameters. However, inliers are expected to be noisy. On the contrary, outliers are data that do not fit the model.

For the estimated lines the following balance should be kept:

- Enough lines to describe details of the environment. Consequently, the total points should be divided into subgroups and feed the algorithm individually.
- Eliminatation of repeated lines of the individual subgroups of points that give the same equation. In order to avoid that, the implemented approach

discards the lines, which have less than 5 degrees difference in the direction.

A sample of the mapping output is illustrated in (fig.C.16). In this case the mapping device has been placed in the middle of the room. Since the device covers 180 degrees, the output should look like a π (pi). The outcome of the mapping approach is similar to the expected one for this simple case.

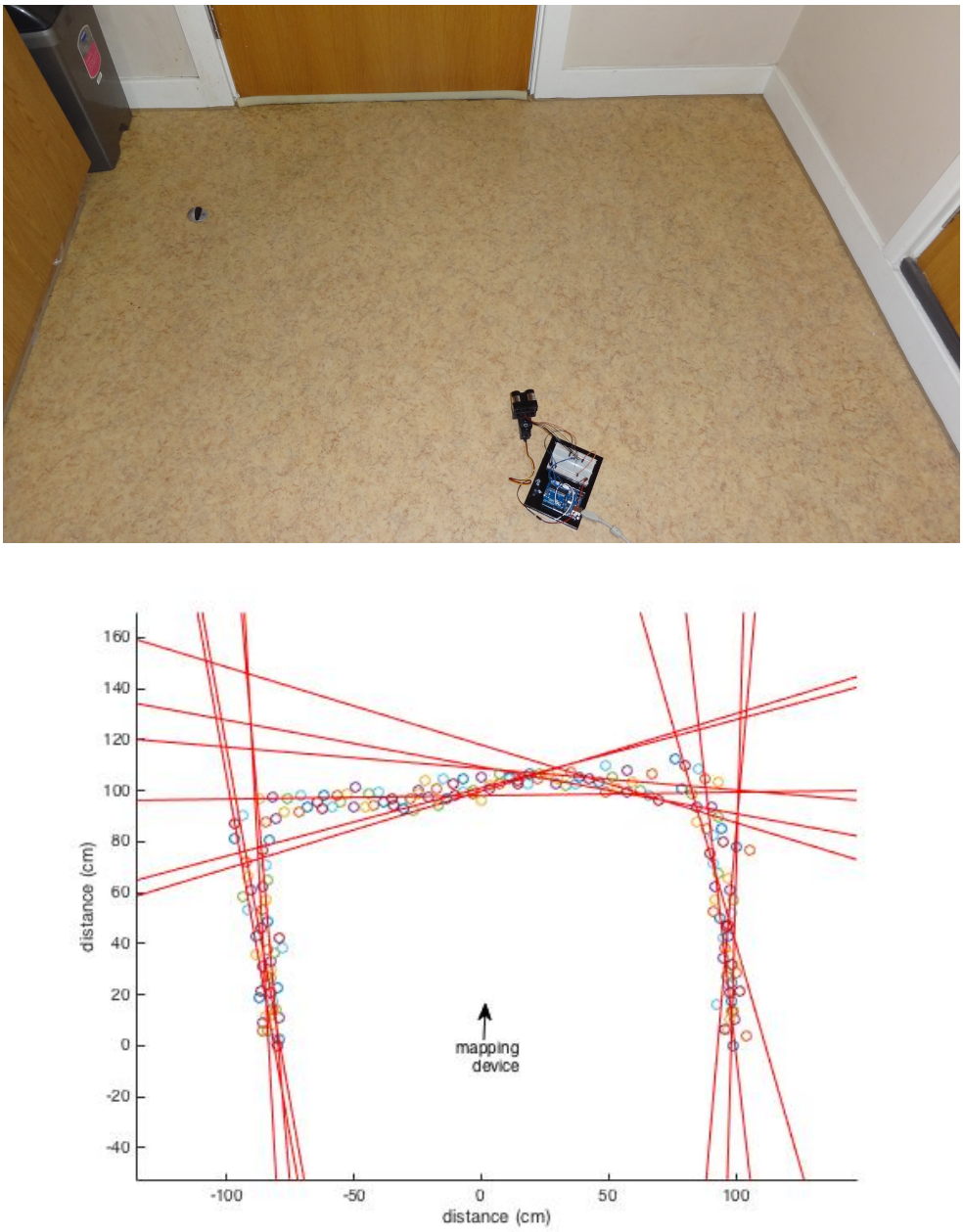

Fig.C.16 The setup of the experiment and the outcome of the mapping algorithm

The required code, both for Arduino IDE and Matlab, as well as the circuitry is available in the "Sensors Circuits" section.

C.5.2 Proximity application

For this section, and after having implemented a simple mapping algorithm, I wanted to develop a ROS enabled application ready to by employed in any compatible system. My initial plan was to deploy a LIDAR-like application where the laser sensor would continuously rotated and transmitted, via a topic, an appropriate sensor message of type *sensor_msg/laserScan*. Although I put a lot of effort into it, this was not feasible, mainly due to the instability of the system. More precisely I encountered the following problems:

- The Arduino memory was 80% occupied by loading only the required libraries (*ros.h* and *i2c.h*), leaving limited space for the working sketch.
- The power source was overused since both laser and servo were fed from the 5V output of the Arduino Uno.
- The servomotor under complex and tedious continuous translation was drifting sometimes and in addition it couldn't provide an absolutely stable cycle to compose the laserScan ROS message.

The combination of the laser sensor and the servo driven from an Arduino Uno controller was not adequate to support such a complex and precise task and provide the expected reliability. As an alternative, since I wanted to develop a working ros-enabled sensor, I utilized the existing hardware configuration to deploy a proximity sensor. This sensor provides three distances, the front and the sides (0^o, -90^o and 90^o). To improve the accuracy, in every position the laser sensor measures the distance twice and returns the average. In my attempt to implement more complex scenarios, like sweep ranges of 20 degrees for every main direction, I faced drifts and synchronisation problems. Consequently, I kept the design simple and reliable for the given hardware.

One of the great features that ROS provides is the ability to utilize in robotics systems simple and cheap sensors using the Arduino microcontroller. It exploits the fact that the Arduino is connected to the computer using a serial connection and data is transmitted using this port. Providing a specific interface package called *rosserial,* it allows transmition and reception of ROS messages by coding Arduino IDE sketches with just including the ROS library. The entire flow of the information from the laser sensor to the published topic is summarized in the following figure (fig.C.17)

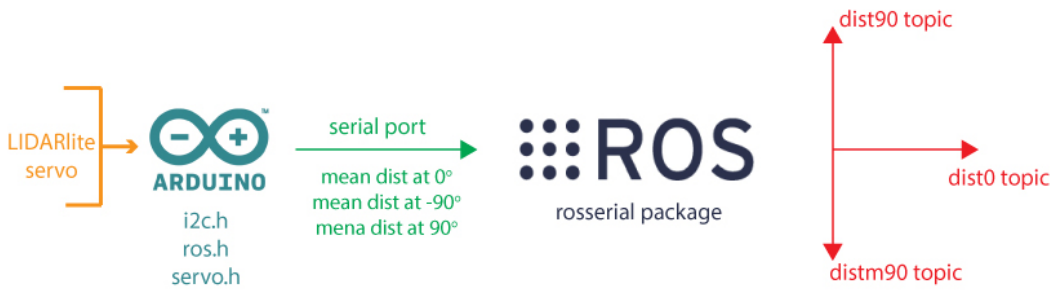

Fig.C.17 The flow of the information for the proximity application algorithm

Since I had already found the pathway that I had to follow, the implementation was straightforward. On the Arduino side the sketch included specific ROS commands. More precisely, aside from including the servo and the I2C libraries, this time the ROS library was added in complementarily. By doing this, I could declare in the Arduino IDE objects like ROS nodes and publishers. Consequently, I declared three publishers, one for every measured direction. For every publisher, an appropriate message was transmitted with a rate between 0.5 and 1Hz, as explicitly measured with convenient time function. This unstable period was one reason why I couldn't compose a laserScan message, since for this type of messages, it is a requirement.

It is also worth citing the different approach that I followed for the movement of the sensor. Unlike the mapping application, this time I did not use the convenient servo methods *write* and *read*. For this specific task I wanted total control and accuracy of the movement. With the given controller this

could be feasible with a simple scenario. To move the sensor accurately, in the three directions, as fast as it was possible without drifts and lost of synchronisation I had to use the *writeMicroseconds* method. The signal consists of positive going pulses ranging from 544 to 2400uS long. The servo positions its output shaft in proportion to the width of the pulse. At the beginning, I discovered the exact values for the 0^o, 90^o and 180^o or in the -90^o, 0^o and 90^o according to the measurement device frame (heading). After the calibration, I investigated the synchronization issues related to the time that it allowed the servo to move before issuing the next move command. I optimized to the shorter interval without having instabilities. As soon as the servo had moved to the desired position and the laser ha estimated the average distance, the microcontroller emitted the distance by publishing a message to the specific topic.

As I have described so far, the ROS topics are composed by the Arduino and are transmitted. However, the last stage -in order to have a working system- is to employ the *rosserial* protocol. There is a need for a specific protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over a character device such as a serial port or a network socket. In my case, I needed to execute the serial node python script, having as input the USB that the Arduino was connected to. After that, the proximity topics were visible in the entire ROS ecosystem.

To validate the implementation, I tried out the device at the end of a corridor (fig.C.18). As can be seen in the table (tbl.C.8) the measured distances lie inside sensible margins. It is clear that the deviation from the case where I tested the laser sensor in a static setup was greater but still usable in proximity check approach. One additional issue that I noticed was the degradation of the accuracy when I increased the servo speed. This had occurred not because of the laser sensor, since it had a high operating rate (~50Hz), but mainly because of the instability and drift of the servo.

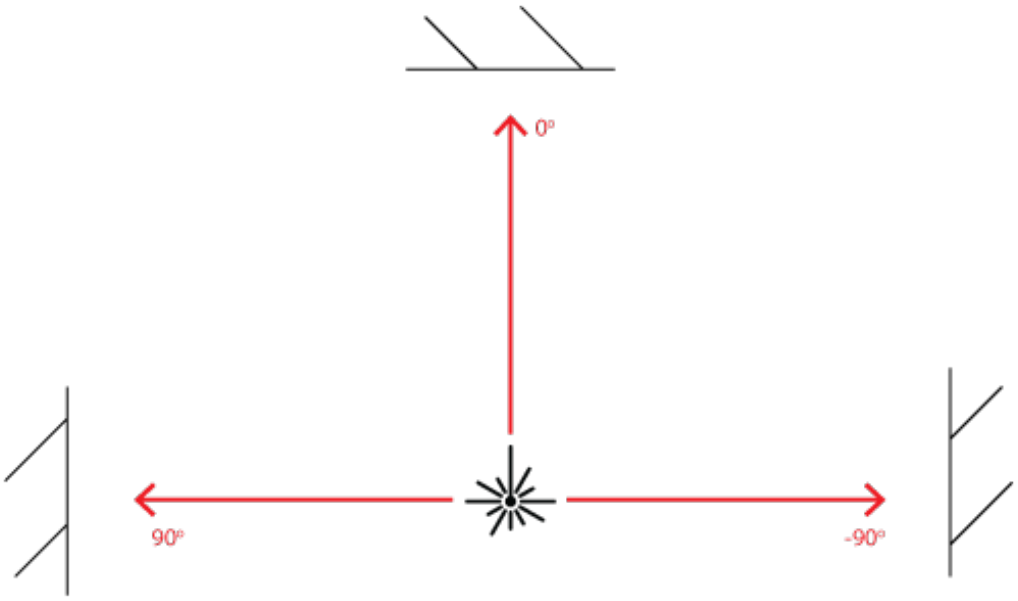

Fig.C.18 The set up of the experiment for testing the proximity application

Table C.8

(Measurements in the three individual directions)

	Distance Measured (cm)	**Distance (cm)**
dist0	52, 57, 55, 54, …	55
dist90	77, 79, 78, 81, …	79
distm90	84, 86, 87, 88, …	86

By following this implementation I created a ROS enabled device that could act as a laser proximity sensor. It is a feasible system since the housing of the device can be easily designed and composed in a 3D printer. In terms of the microcontroller, there is no need for an Arduino Uno since an Arduino Nano v.3.0 (45mm length and 18mm width) can effectively do the work. This is because both of them are based on the ATmega328 chip, which provides 2 KB of SRAM and 1 KB of EEPROM. The circuit with the Nano board was implemented so as to test the behaviour and the stability in reality (fig.C.19). In essence there was no difference in the behaviour compared to the Arduino Uno. The main difference was in the available power to drive the servo and the LIDAR-Lite and a compatibility issue. More details about the implementation can be found in the "Sensors Circuits" part. To conclude, it is

possible to have a compact and simple design that can report three accurate distances in a reasonable rate.

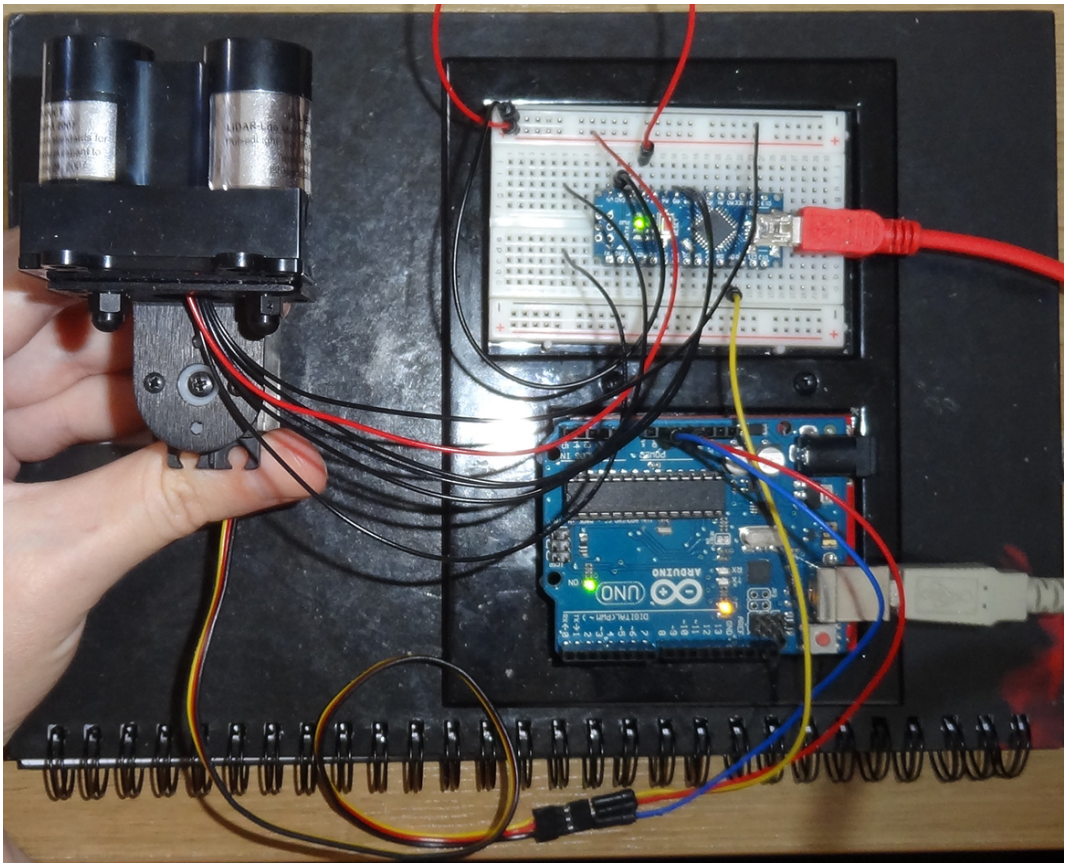

Fig.C.19 The proximity application implemented using an Arduino Nano board. In this picture Arduino Uno feeds only the V_{dc} and GND pins of the servo