Appendix B

Sensors Circuits

In this chapter I present all the circuits I utilized to experiment with the sensors. It includes the schematics, the tables with the pins connections as well as comments about specific aspects and behaviours I experienced.

B.1 Ultrasonic sensor

Devantech SRF02 features both I2C and Serial Interfaces. For the experiments, the later was used. The serial interface is a standard TTL level UART format at 9600 baud and can be connected directly to the serial ports on any microcontroller. In my case the microcontroller was an Arduino Uno and the exact circuitry is shown in figure (fig.B.1).

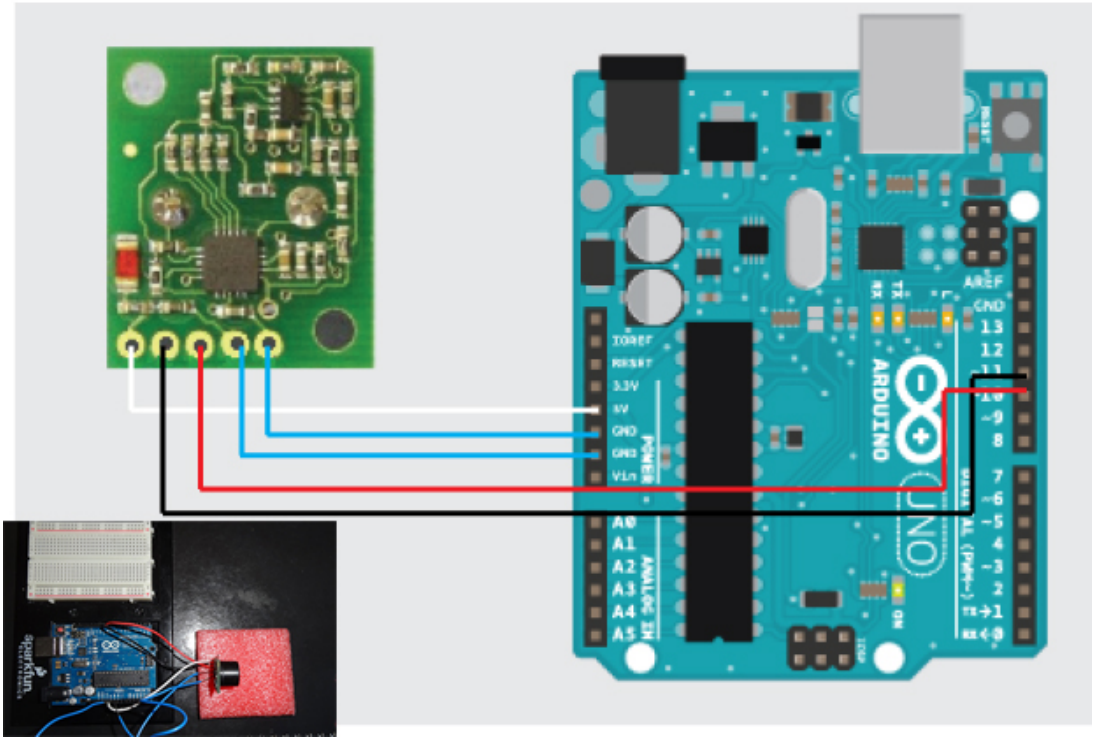

Fig.B.1 SRF02 interfaced with an Arduino Uno

The circuit connections are described in the following table (tbl.B.1):

Table B.1

(Connection between SRF02 and Arduino Uno)

SRF02	Arduino Uno
5V Vcc	5V
Rx	11 digital (Tx)
Tx	10 digital (Rx)
Mode	GND
Ground	GND

To select Serial Connection the mode pin should be connected to ground. The Arduino sketch that was used for the measurements is included in the appendices and the online repository (goo.gl/buq1hn).

B.2 Infrared sensor

Sharp GP2Y0A02YK0F infrared sensor has an analog output that varies from 2.8V at 15cm to 0.4V at 150cm with a supply voltage between 4.5V and 5.5V. In order to experiment with the IR sensor, the most convenient way was to use the analog port of an Arduino Uno microcontroller instead of using multi-meter and calculator. The interface between the sensor and the microcontroller is shown in figure (fig.B.2).

The explanation of the circuit connections is depicted in the table (tbl.B.2). The required Matlab file for the conversion and the Arduino sketch are included in the appendices and the online repository (goo.gl/oHyf0l).

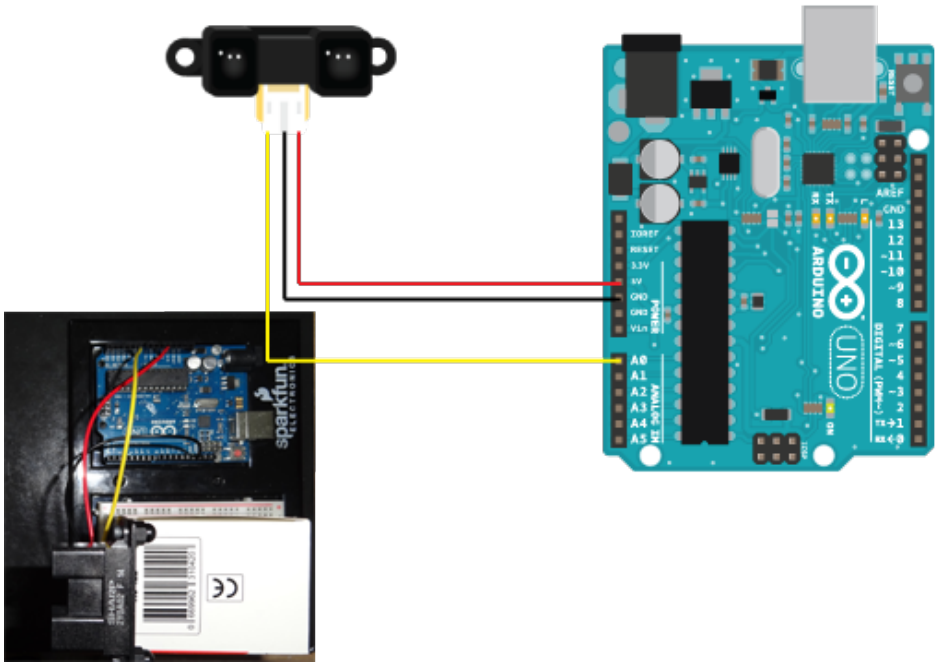

Fig.B.2 GP2Y0A02YK0F interfaced with an Arduino Uno

Table B.2

(Connection between GP2Y0A02YK0F and Arduino Uno)

GP2Y0A02YK0F	Arduino Uno
5V V_{cc}	5V
V_o	A0 analog
Ground	GND

B.3 Optical sensor

Px4flow smart camera supports various interfaces. The available connections include I2C, micro USB and two USART serial communication ports. For the project's experiments I utilized the first two inputs. For the I2C interface, there was no provided cable on the delivered package. To overcome this lack, I assembled a custom cable using the required Hirose DF13 4 positions plug.

Before starting any use of the px4flow, there are two required procedures that have to be performed. Both tasks were performed using the QGroundControl program (goo.gl/71fQ2M).

- The device should be updated to the latest firmware.
- When using the Video Downlink widget from QGroundControl, the focus should be adjusted by loosening the locking screw and turning the lens at an object at 3m distance.

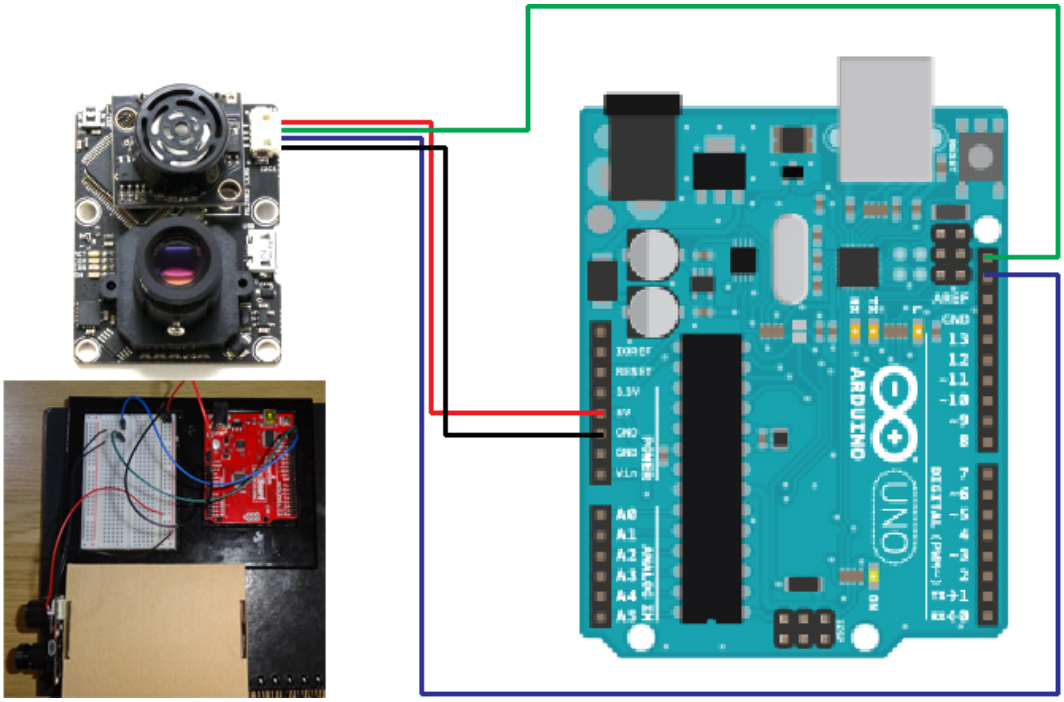

Fig.B.3 Px4flow interfaced with an Arduino Uno

The diagram of the circuit is shown in figure (fig.B.3). Moreover, the following table (tbl.B.3) illustrates the required connections to use the sensor in the I2C mode.

Table B.3

(Connection between px4flow and Arduino Uno)

Px4flow	Arduino Uno
5V V_{cc}	5V
SCL	SCL
SDA	SDA
Ground	GND

The required sketch is included in the appendices and the online repository (goo.gl/MGtnhO).

In addition to previous setup, it is possible to utilize px4flow using ROS drivers. For this case, the device is connected via a USB cable to a computer running ROS (fig.B.4). In order to obtain data from the device, it requires installing the *px-ros-pkg* stack (goo.gl/bvo1cf) in a working catkin workspace. After that, following the guide from the px4flow node documentation (goo.gl/n8IMX9) it is very easy to have access to the video from the camera as well as to the *opt_flow* topic. The *OpticalFlow.msg* type messages that are published in this topic include information such as the ground distance, the flow in x and y direction, the velocity in x and y direction and the quality of the optical flow estimate. The experiment was performed using ROS Hydro installed on Ubuntu 12.04 LTS. In order to access the device using the proposed launch file, the proper access permissions should be set for the device, for instance (*sudo chmod a+rw /dev/ttyACM0*). In addition, the exact device, for example *ttyACM0*, should be declared in the *px4flow_parameters.yaml* file that stores the global settings of the corresponding launch file.

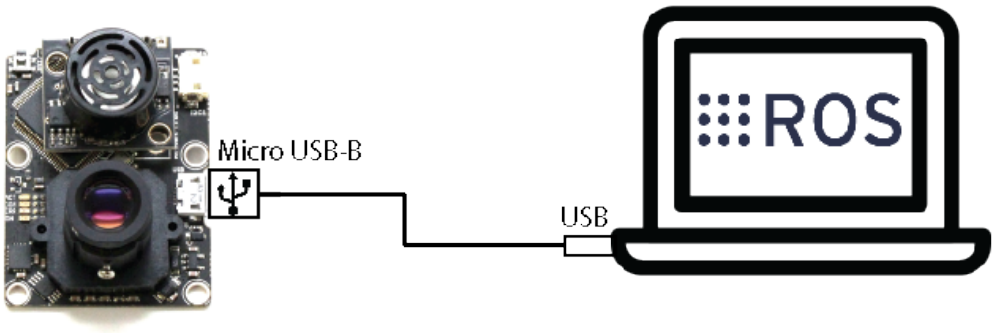

Fig.B.4 Px4flow interfaced ROS

B.4 Laser sensor

LIDAR-lite laser sensor supports two communication methods, I2C interface and PWM (Pulsed Width Modulation). For the experiments, the former was

employed. As with the previous sensors, an Arduino Uno was used to interface with the sensor. More precisely, the sketch utilizes the "Arduino I2C Master Library" from the DSS Circuits (Truchsess, 2015), which is a more sophisticated I2C library than what the default Wire library Arduino IDE offers.

The diagram of the circuit is shown in the figure (fig.B.5). Furthermore, the following table (tbl.B.4) illustrates the required connections in order to use the sensor in the I2C mode.

Table B.4

(Connection between LIDAR-Lite and Arduino Uno)

LIDAR-Lite	Arduino Uno
5V V_{cc}	5V
PWR EN	(unused)
MODE	(unused)
SCL	SCL
SDA	SDA
Ground	GND

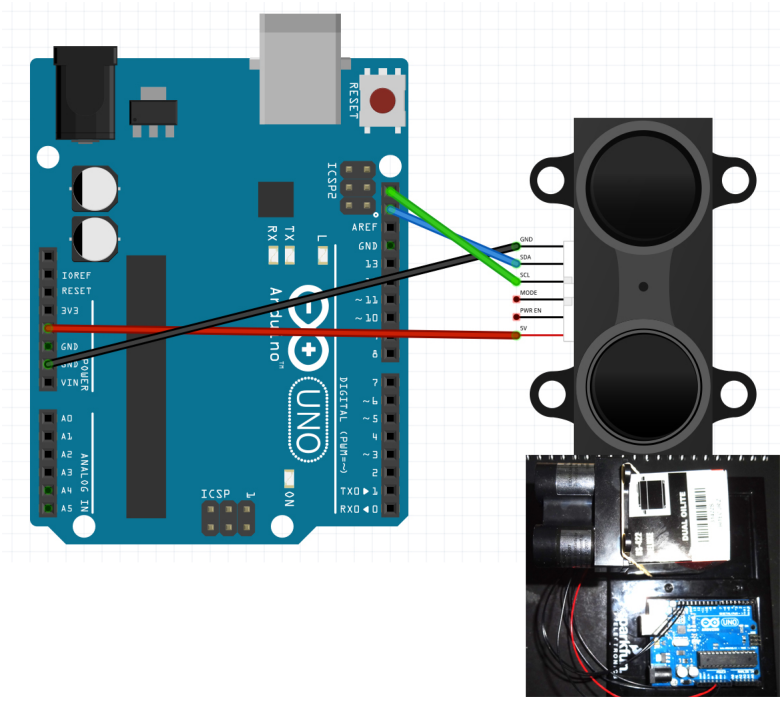

Fig.B.5 LIDAR-Lite interfaced with an Arduino Uno

The required sketch is included in the appendices and the online repository (goo.gl/DSToIp).

B.5 Laser applications

For the mapping and proximity application the same setup was used. Particularly, both applications required the combination of the laser sensor with an Arduino Uno microcontroller and a servomotor (fig.B.6). For the interface between the laser sensor and the Arduino Uno, the I2C library, which was developed by the DSS circuit, was utilised. At the same time, I experimented with taking a distance reading using Pulse-Width Modulation (PWM), which does not require I2C communication. Although it is a simpler approach, it lacks of precision. Using this approach there is no available method to calibrate the sensor. On the other hand, using the I2C approach there is an easy zeroing procedure where the user should only set the value of a specific register (*CalibrationOffsetVlue*) with the required offset.

The motion of the servo was implemented using the Servo library. For the mapping task I used the convenient methods *read* and *write.* The first command reads the current angle of the servo while the second writes a value to the servo, controlling the shaft accordingly. Using the provided servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. For the proximity application, which was more demanding in terms of accuracy, speed and synchronisation of the movement, the *writeMicroseconds* command was used. It writes a value in microseconds (uS) to the servo, controlling the shaft accordingly. For the provided servo, this will set the angle of the shaft. Further on that, using that method the servo was calibrated exactly to the proper (uS) so as to correspond to the required angles. Thus, the 555uS corresponds to 0 degrees, the 1500uS to 90^{o} and the 2390us to 180^{o} for this specific servo.

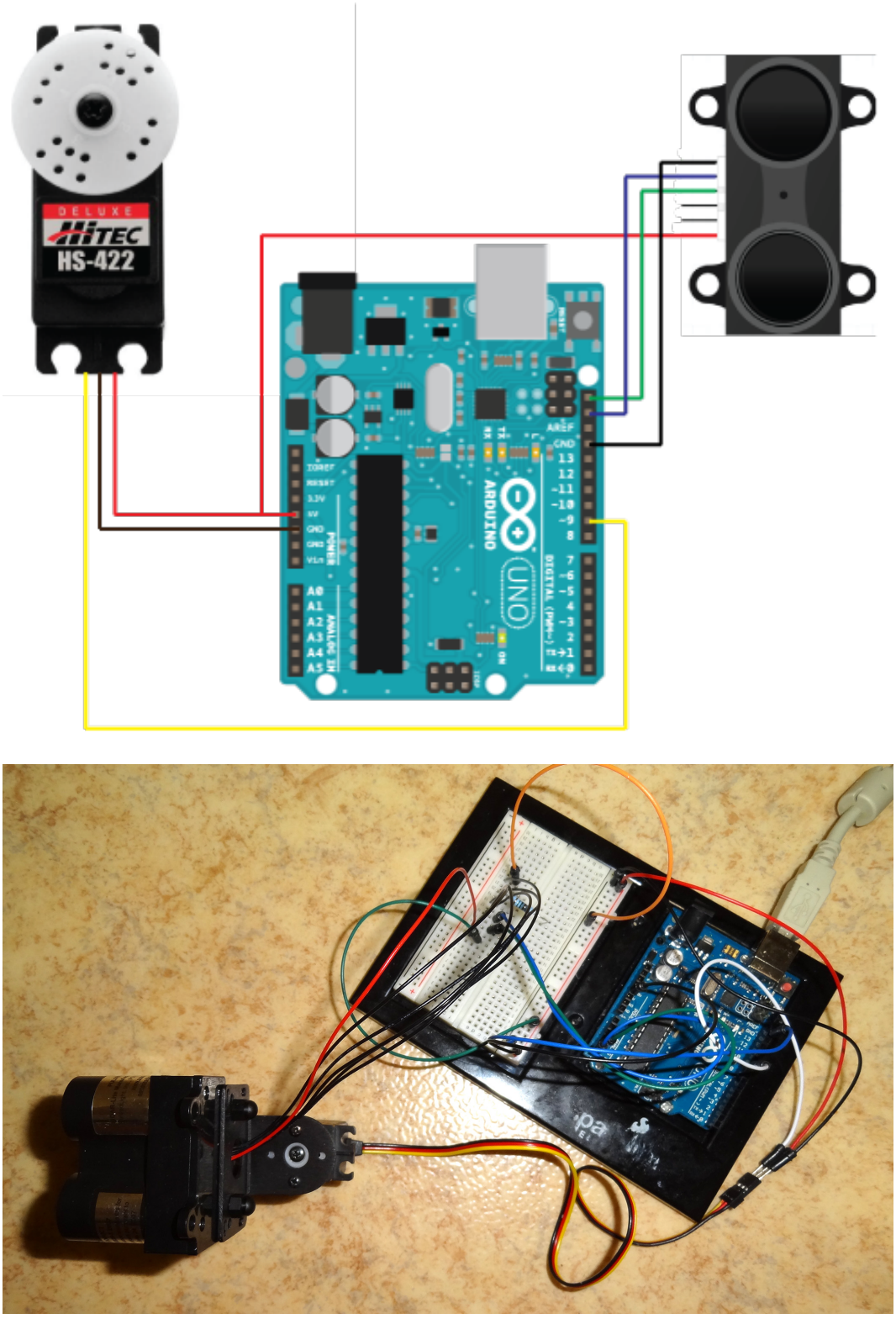

Fig.B.6 LIDAR-Lite, servo HS-422 deluxe connected to an Arduino Uno. The schematic of the circuit and a photo of the real implementation.

The connections required between the individual components for the proposed circuit are clarified in the following table (tbl.B.5)

Table B.5

(Connection between LIDAR-Lite, servo HS-422 and Arduino Uno)

LIDAR-lite	Arduino Uno
Pwr en	(unused)
Mode	(unused)
SCL	SCL
SDA	SDA
GND	GND

HS-422 Deluxe	Arduino Uno
Signal	9 Digital PWM
5V V_{cc}	5V
GND	GND

The code, which has been developed for the presented applications, is included in the appendices section and the online repository (goo.gl/99Nxqj).

For the mapping application the user should first execute the Arduino sketch and afterwards the related Matlab file. There is also the requirement to add in the Matlab's path the *ransac_homography* library (goo.gl/5yR8IX).

For the proximity application the ROS and Ubuntu version depends on which board is used. Therefore, Arduino Nano is compatible with ROS Hydro and Ubuntu 12.04 LTS while the Arduino Uno is compatible with ROS Indigo and Ubuntu 14.04. It is observed that in the case of Arduino Nano (fig.B.7), it is better to provide external power supply at least to the servo since the system, if based only on the Nano feed, is unstable. In order to upload the code in the board the appropriate access permission should be set for this device. For instance, in Ubuntu the user need to type: *sudo chmod a+rw /dev/ttyUSB0*, if the board is connected at *ttyUSB0*. After that, and assuming that the *roscore* is run, the use exploy the *rosserial* package by giving a command like: *rosrun rosserial_python serial_node.py /dev/ttyUSB0*.

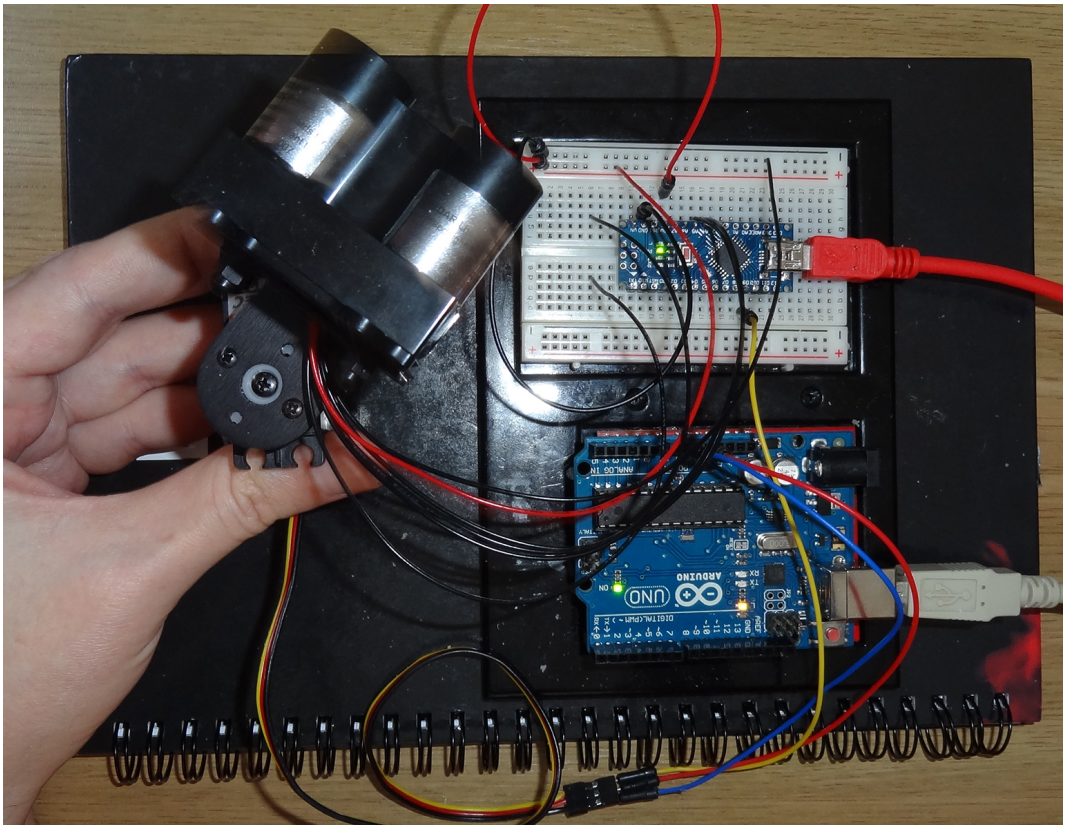

Fig.B.7 LIDAR-Lite, servo HS-422 deluxe connected to an Arduino Nano. Arduino Uno provides only power supply to servomotor.