CENTRALESUPÉLEC

2A

Smart Photonic Systems – EI

LIDAR

FINAL REPORT

Faculty Advisor

Delphine Wolfersberger, Nicolas Marsal, Marc Sciamanna

Peizhou ZHANG - 1900291

Yuqi SUN - 1900298

Jiale ZHANG - 1900289

Sacha BULATOVIC – 1900834

OUTLINE

1. State of the Art

LIDAR, which stands for Light Detection and Ranging, is a method for measuring distances by illuminating the target with laser light and measuring the reflection with a sensor.

There are three primary components of a LiDAR instrument: the transmitter, laser and receiver. A transmitter which illuminates a target with a laser beam, and a receiver capable of detecting the component of light which is essentially coaxial with the transmitted beam. Receiver sensors calculate a distance, based on the time needed for the light to reach the target and return. And LIDAR uses ultraviolet, visible, or near infrared light to image objects. It can target a wide range of materials, including non-metallic objects, rocks, rain, chemical compounds, aerosols, clouds and even single molecules. A narrow laser beam can map physical features with very high resolutions, for example, an aircraft can map terrain at 30-centimetre resolution or better.

LIDAR technology is commonly used to make high-resolution maps, with applications in surveying, geomatics, archaeology, geography, forestry, atmospheric physics, laser guidance, and laser altimetry. It allows scientists and mapping professionals to examine both natural and manmade environments with accuracy, precision, and flexibility. They use LIDAR to produce more accurate shoreline maps, make digital elevation models for use in geographic information systems, to assist in emergency response operations, and in many other applications. This technology is also used in control and navigation for some autonomous cars.

In this article, we will address the design and realization process of a relatively primary home-made LIDAR system based on Arduino in a limited budget, as well as the test results, and what the most important is, we will show you the problems we have encountered as well as our solutions, many of which are the common but hard-to-solve problems when using the Arduino IDE. The designed goal of this LIDAR is to measure an object from several meters and plot a 3D visual image.

2. General Solution

2.1 Choice of LASER Source

In order to let the LIDAR function in an eye-safe way, we need to choose the wavelength of the LASER carefully. Also, as mentioned above in *Chapter 1. State of the Art*, the realization of this LIDER does not involve measurement of long-distance, nor does it require a high resolution, and the interference of the sunlight can be omitted since the test can be arranged indoor, we do not seek LASER wavelength with strong energy level. Taking into account the budget of the whole project setup (as we will discuss in the next chapter), a wavelength of around 905nm emitted from InAsGa Semiconductor LASER Diode is chosen. This near-infrared light has a high relative spectral sensitivity, as shown in *Image 2.1-1*, and InAsGa Semiconductor LASER Diode is of low cost and easy to implement.

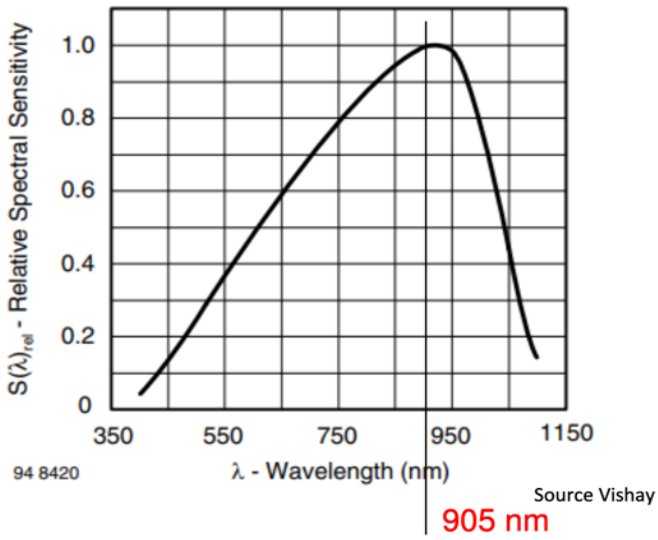

Image 2.1-1 Silicon Photodiode Relative Spectral Sensitivity (typ. 0.5A/W @max)

Also, choosing a near-infrared light source means the reflected light can be quite homogeneous when the LASER reached objects with different colors.

2.2 General Project Budget

The whole budget for this LIDAR is 100 Euros, apart from the Arduino MCU (*Arduino MEGA 2560* – 38.63€) and two servo motors (*TowerPro SG90 Mini Engrenage MicroServo 9g* - 2.21€), which are necessarily needed and cost 43.05€, the left 56.95€ does not give us a wide range of choices for

the LIDAR sensor, but enough to meet the design goal. The specific chosen model of LASER is thus *Module Micro LIDAR TFMINI Benewake UART (12 m)*, and it costs 37.96€ from *RobotShop*. The average power output is below 0.6W. The rest money will be addressed to other expenses, such as the breadboard and jumper wires.

Image 2.2-1 Module Micro LIDAR TFMINI Benewake UART (12 m)

More detailed information of the setup of the equipment can be found in *Chapter 2.3 General Design Solutions.*

2.3 General Design Solutions

The LIDAR will mainly consist of one LIDAR sensor for measurement function, two servos to form the position of the sensor in order to get a 3D plot, and of course the Arduino MCU alone with necessary wire rods for the control of servos and reading of the sensor.

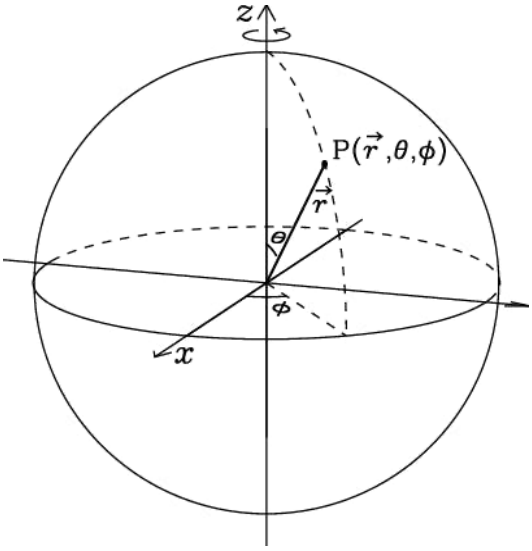

Image 2.3-1 Spherical Coordinate System

As shown in *Image 2.3-1*, by fixing an editing origin of a certain space, two servo motors form the angle θ and φ separately and the LIDAR sensor measure the distance r, with the transformation equations between spherical coordinate system and space rectangular coordinate system:

$$\begin{cases} x = r\,sin\theta\,cos\varphi \\ y = r\,sin\theta\,sin\varphi \\ z = r\,cos\theta \end{cases}$$

we can get so-called 'Point Cloud Data' (*Image 2.3-2*) to plot the 3D diagram.

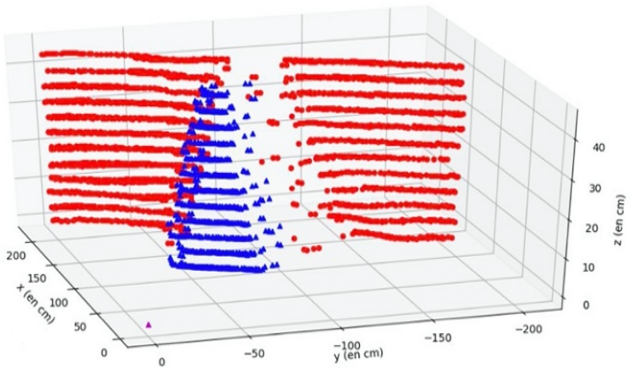

Image 2.3-2 An Example of Point Cloud Data

3. Equipment Setup

3.1 Main components

The main electronic components are:

- *Arduino Mega 2560*
- *TowerPro SG90 Mini Gear Micro Servo*
- *28BYJ-48 Stepper Motor*
- *Lidar TF-MINI-S Distance Sensor*

Notably here, one servo is changed to a stepper motor, as in practice, an Arduino driving two servos at the same can cause logic conflicts, so it's a compromise since a stepper motor is driven by the number of received pulses while a servo motor is driven by the width of a pulse and constantly adjusted by the integrated feedback loop.

3.2 Principles of Equipment Setup

3.2.1 TFMini-S LIDAR Sensor

TFmini-S is a miniaturized, single-point ranging product. Based on the TOF (time of flight) principle：

$$D = \frac{c}{2} \cdot \frac{1}{2\pi f} \cdot \Delta\varphi$$

It is designed with unique optics, electricity, and algorithms to achieve stability, precision, sensitivity and high-speed distance measurement function. This Lidar itself has high

distance measurement accuracy. It also has lower power consumption and more flexible detection frequency. It is super suitable to help construct a distance measure system.

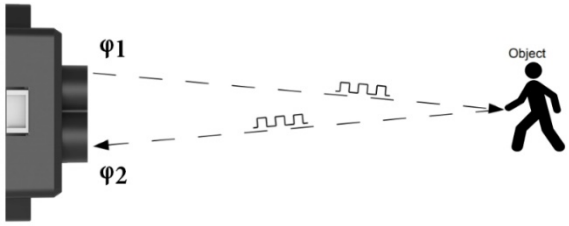

Image3.2.1-1 Schematics of TOF Principle

Key parameters of Lidar TF-MINI-S Distance Sensor:

- *Operating range: 0.1m~12m*
- *Accuracy: ±6cm (0.1-6m) ; ±1%@ (6m-12m)*
- *Range resolution: 1cm*
- *Field of View: 2°(theoretical)*
- *Frame rate: 1~1000Hz (adjustable)*
- *Supply voltage: 5V±0.1V*
- *Communication level: LVTTL (3.3V)*

Be very careful here, the LIDAR is likely to have a reading failure when the view of LIDAR is reaching an edge of the object, as demonstrated in *Image 3.2.1-2* below.

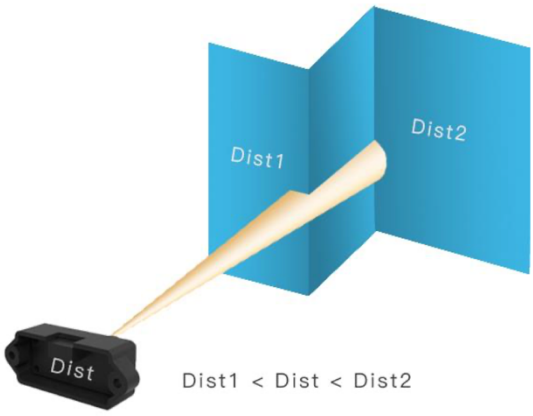

Image 3.2.1-2 Distortion at the Edge

Also, the operation range has a lower bound of 10cm, which means distortion is likely to show up in this range. An example is shown below:

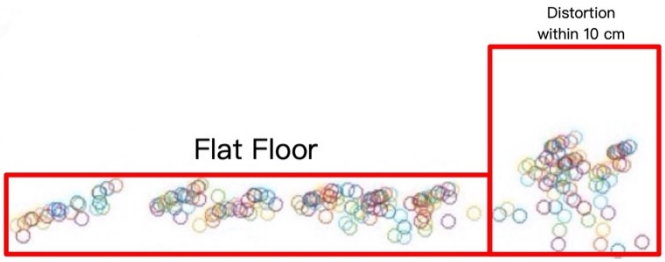

Image 3.2.1-3 Distortion within 10cm

It is very important to understand the communication protocol of this sensor, as most of the problems we encountered during the project were due to the unsuccessful communication between the sensor and the Arduino board.

Byte0 -1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x59 59	Dist_L	Dist_H	Strength_L	Strength_H	Temp_L	Temp_H	Checksum

Data code explanation	
Byte0	0x59, frame header, same for each frame
Byte1	0x59, frame header, same for each frame
Byte2	Dist_L distance value low 8 bits
Byte3	Dist_H distance value high 8 bits
Byte4	Strength_L low 8 bits
Byte5	Strength_H high 8 bits
Byte6	Temp_L low 8 bits
Byte7	Temp_H high 8 bits
Byte8	Checksum is the lower 8 bits of the cumulative sum of the numbers of the first 8 bytes.

Image 3.2.1-4 Data Format and Code Explanation

As shown above, the first two bytes are used for identifying the beginning of a measurement data, and from *(Byte2 + Byte3*256)* we get the distance, from *(Byte4 + Byte5*256)* we get the strength of the signal.

An example code can be found in the Product Manual of TFmini-S. A link of this manual is attached in the reference, but when not using a 3.3V TTL logic level converter (on which is the example code based on), some problems may occur during the communication with the Arduino board (the example is not necessarily correct!). Normally the Arduino board can read faster than the serial port can transmit, which cause no reading at all! Thus, a delay is necessary and of vital importance under the first Determine statements!

Another problem will also occur when implanting the example LIDAR reading code into the motor controlling code, since the code is not put in a repeating loop! The example code is in the *void loop(){}* so the serial port will constantly read the data from the sensor, but when merging the code with motor control code, it is very likely that *Serial.read()* does not get lucky enough to catch the header byte of the data transmitting from the LIDAR sensor, and the program will skip the data reading process and rotate the motor to the next position! A possible solution to this is creating a *while* loop for the LIDAR reading program using a *bool* indicator on whether the serial port has successfully made an output.

The last and final problem maybe the delay of the serial port of the Arduino, that is to say, the data output may not be real-time (Sometimes the data keeps outputting even the LIDAR sensor is unplugged!). The problem is with the Arduino serial port hardware, and there is no nice and neat code can solve that. We had to read the data from the serial port a several times and put it in a *String* and delete it before try to read the real data, which is quite nasty. And in order to get a precise reading, it is very important to wait until the reading of the LIDAR is stabilized rather than simply read several times and get an average. Even by doing this, sometimes there is still a slight possibility that the Arduino will read a distance way too large than realistic, so when drawing a 3D diagram of the result, these bad dots should be deleted, otherwise the plot will only shows two dots: one being the false reading which is far away and the other being the real dots packed together with a high density.

If you want to see the example of solving the above two problems, please refer to *Chapter 5 Arduino CODE.*

3.2.2 Servo (*SG90 Micro-Servo 9g*)

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a closed-loop servomechanism so that it uses position feedback to control its motion and final position. The input to its control is a pulse-width signal indicating a position commanded for the output shaft. Servomotors are generally used as a high-performance alternative to the stepper motor. This servo motor is used to control the vertical angle θ of the Lidar.

Driving a servo in a complex Arduino setup will commonly cause a conflict, and at this moment no once-for-all solution has been found. It is strongly recommended that a control function for the servo is written instead of simply using the *library* <*Servo.h*>. Like always, an example can be found in *Chapter 5 Arduino CODE.*

Also, the package of a stepper motor and a LIDAR sensor attached to the servo (shown in *Chapter 3.2.5 Product Packaging*) might be too much weight for the servo, so it is recommended that, after each measurement and the stepper motor moved to the next position, reorient the servo at the current angle.

3.2.3 Stepper Motor (28BYJ-48)

A stepper motor is a brushless DC electric motor that divides a full rotation into a number of equal steps. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate. The stepper motor is used control the horizontal angle φ of the Lidar.

Since it is not controlled by simply putting an angle, a converting process is required between the real angle and the steps. The step angle of this model is 1.8°, but taking into consideration the gearing system inside the motor, it takes 512 pulses for the stepper motor to complete a full circle. And since the number of pulses is of course an integer, so when converting the real angle φ to the steps n by the equation $n = 180\varphi/256$, it is a must to ensure that n is a precise number and an integer. Otherwise the data on horizontal angle φ would be biased, and after a complete scan of one plane, the stepper motor would be able to return to the original point (a bias in editing origin).

We also need to use a L9110S Driver Board to interface a 28BYJ-48 Stepper Motor to Arduino board. It is used to regulate current flowing through a circuit and to control other components, just like a stepper motor. Each channel of this driver board has a direct current output of 800mA, and the module input voltage is 2.5-12V. It behaves as a voltage boost module, which makes the stepper motor ideally connected to the circuit.

3.2.4 Element Assembling

This schema in *Image 3.2.4-1* is realized through *Fritzing,* which is an application specialized for Arduino developing. As for the LIDAR sensor, the green wire is connected to 19RX1, white to 18TX1, red to 5V, black to GND.

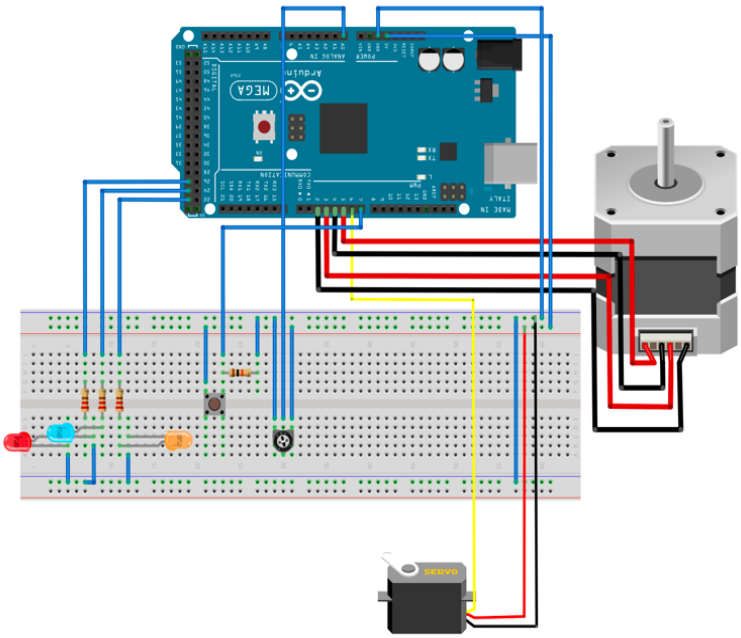

Image 3.2.4-1 Circuit for Motors

The three LEDs, one button switcher and one Sliding rheostat are for the realization of initialization of the whole system. BLUE light indicates the system is ready for the LIDAR to start working. When the button is pushed, Orange LED will first light up, meanwhile allowing the user to adjust the initial position of the stepper motor (initial horizontal angle φ_0) by the rheostat. After another push of the button, it is possible to adjust the initial position of the servo (initial vertical angle θ_0) when the RED LED is on. The range of scanning and the precision of angles ($\Delta\theta$, $\Delta\varphi$) can be adjust in the beginning of the Arduino Code.

Unfortunately, an integrated setup would cause conflicts of control, leading to a separation between initialization system and the rest (LIDAR position and LIDAR reading). However, it is included in *Chapter 5 Arduino CODE* anyway, and we wish it can serve as an inspiration.

3.2.5 Product Packaging

The packaging of the LIDAR is hand-made from an envelope of an Amazon delivery, holes are cut out for the cable, the motors and sensors, and also for the control of the initialization process. And in the meantime, it is easy to open and disassemble, as nothing is actually glued (except the servo is glued to the inner wall of the box in order to provide a steady position for the sensor) but stuck using the designed dimensions of the box itself. The height of the LIDAR sensor is 85cm.

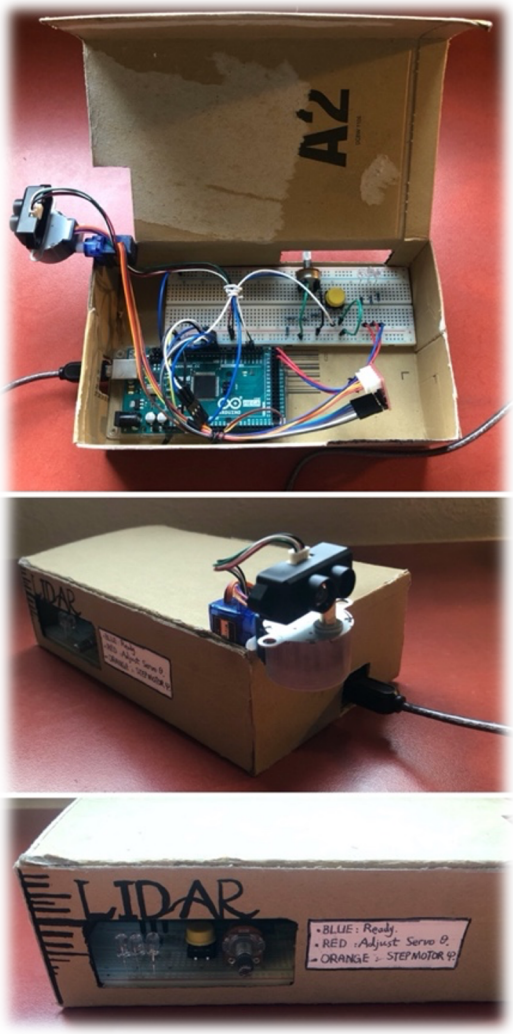

Image 3.2.5-1 Product Packaging

This kind of packaging could, likewise, cause distortions in a 3D scanning when the scanned object is too closed, as the point where LIDAR reading is conducted, the rotational point of the servo and the rotational point of the stepper motor are not the same. That is to say, the packaging way is not compact enough when measuring short-distance objects. This problem can be seen in the last and final test of the

LIDAR, demonstrated in the end of *Chapter 4.2 Three Dimensional Test.*

4. Test Results

4.1 One Dimensional Test

A test for one dimension was carried out to see if the LIDAR can function in a proper way. As shown in the *Image 4.1-1* below, the LIDAR was placed in a big box with a book and a toilet roll inside.

Image 4.1-1 Arrangement for One Dimensional Test

The reading from the LIDAR is converted into an *.xls* file which MATLAB can read directly, and afterwards, the MATLAB generated the result diagram as followed:

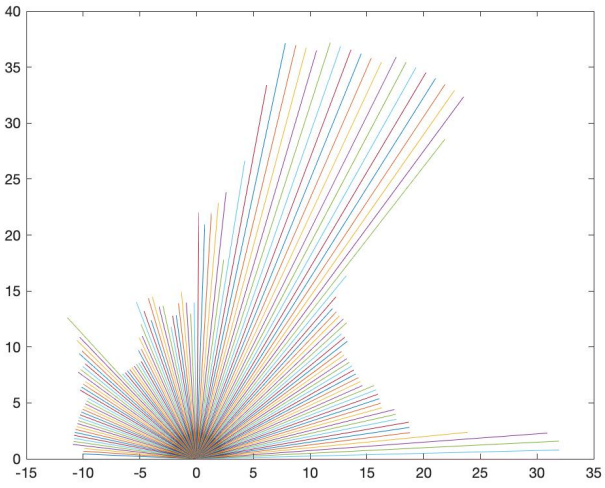

Image 4.1-2 1D diagram of the Results

We are seeing unstable read when the LIDAR is measuring the surface of the book, which might be because of the reflection on the surface material of the book is not ideal, for it is dark and glossy. But the major problem here is the bias in the angles. The stepper motor was initially placed carefully to let the LIDAR measure the distance to the nearest long edge of the box, and the full scanning range is 180°, so from the 1D result diagram, we are seeing the gap between the book and the toilet roll where it should be the toilet roll! This means at each time, the stepper motor is moving less steps than we instructed. And this is where we figure out the problem we've already talked about in *Chapter 3.2.3 Stepper Motor*: the stepper motor is omitting or increasing the steps while the real angle cannot be converted to an integer step number.

4.2 Three Dimensional Tests

We are looking forward to a mature 3D scan of an object after solving the problems mentioned before. So as for this ultimate test, we chose a can as the object, as its shape is complex enough and also, small enough so that the scanning time wouldn't be too long.

This time, the serial communication between the Arduino MEGA 2560 and MATLAB is deployed and the 3D result diagram should come point by point in the plotted figure. a pinot cloud data of 30*25 is used for the plotting, covering 70.32° in horizontal range and 60° in vertical range. The steps of angles ($\Delta\theta$, $\Delta\varphi$) is (2°, 2.82°). A MATLAB code is created to cater this specific experiment, which is attached in *Image 4.2-1* below.

```
s = serial('/dev/cu.usbmodem14201');
set (s,'BaudRate',9600);
fopen(s);

datanumber = 750;
t = 1;
data = [];
r = 0;
PointCloudData = [];
while (t < datanumber)
    data(t,1) = str2num(fgetl(s));
    data(t,2) = 60 + str2num(fgetl(s));
    data(t,3) = -35.16 + str2num(fgetl(s));
    r = data(t,1);
    PointCloudData(t,1) = r*sin(data(t,2)*pi/180)*cos(data(t,3)*pi/180)
    PointCloudData(t,2) = r*sin(data(t,2)*pi/180)*sin(data(t,3)*pi/180)
    PointCloudData(t,3) = r*cos(data(t,2)*pi/180)
    x = PointCloudData(t,1);
    y = PointCloudData(t,2);
    z = PointCloudData(t,3);
    hold on
    if (data(t,1)<100)
        plot3(x,y,z,'o')
        grid on
        view(3)
        drawnow
    end
    t = t+1;
end
fclose(s);
```

Image 4.2-1 MATLAB Code for 3D test 1

Careful, the MATLAB serial monitor and the serial monitor of Arduino cannot be open at the same time (collision)!

Image 4.2-2 Arrangement for 3D Test - 1

The result is satisfying since it indeed demonstrates the shape of the floor and the background wall, also the presence of the can (Specially shown in *Image 4.2-3* of the side view). However, the reading on the can is still not clear enough. On one hand, it's, without doubt, the result of insufficient point data, on the other hand, however, It is also the same problem as the reading on the surface of the book in *Chapter 4.1 One dimension* (Since the reading of a can is compulsory for the report, and I don't have a less glazed can…). We are also seeing a distortion within the range of 10cm (bottom right corner of *Image 4.2-3*), which we've discussed in *Chapter 3.2.1 TFMini-S LIDAR Sensor.*

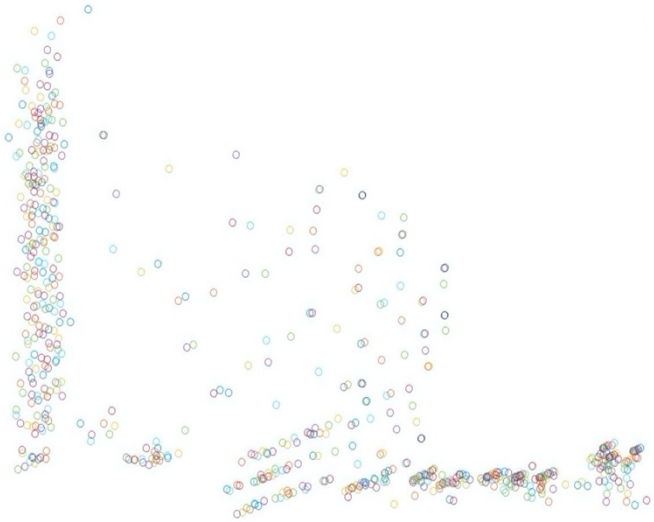

Image 4.2-3 Side View of the Can

From the top view (*Image 4.2-4*) we can see clearly the horizontal scanning range, and also, in the image of the front view (*Image 4.2-5*), the regular lattice of the background wall indicates the good performance of the LIDAR.

Image 4.2-4 Top View of the Can

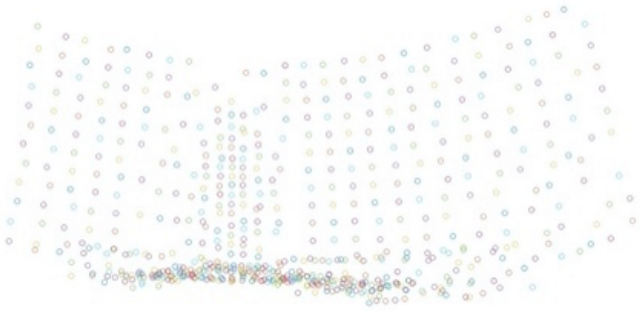

Image 4.2-5 Front View of the Can

A second 3D test is conducted using a less glazed object, which is a toilet roll, as it not only has a good performance at reflection, but also at diffuse reflection. In order to separate the object from the background more clearly, the object is now placed further away from the wall, and different colors are used to demonstrate the distance of points. Last but not least, since we have seen good result in the performance of the LIDAR (although not showing the object, but there are several indicators of good functionality), the resolution of the scanning is set to the maximum, with $\Delta\theta$ and $\Delta\varphi$ equal to 1° and 0.7°, and the scanning range is 90° horizontally and 30° vertically. You can also find a video of this resulting generating in the files attached with this report.

Image 4.2-6 Arrangement for 3D Test - 2

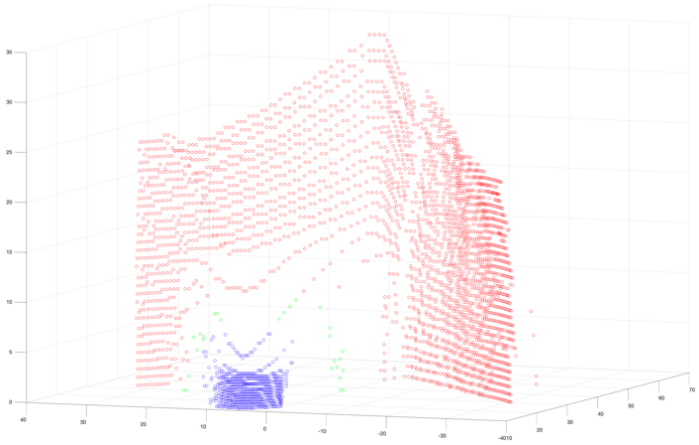

Image 4.2-7 3D View of the Roll

The result in a 3D view shows the shape of the roll and the shape of two background walls along with the vertical angle they form. The green dots are largely corresponding to the data measured for the two planes where the view of LADAR reached the two edges of the roll. These two planes are also where distortions on the background walls can be found. This problem is specially addressed in the official manual of the LIDAR sensor, and it is discussed in *Chapter 3.2.1 TFMini-S LIDAR Sensor*. A clearer view of this kind of system error can be seen from the top view (*Image 4.2-8*).

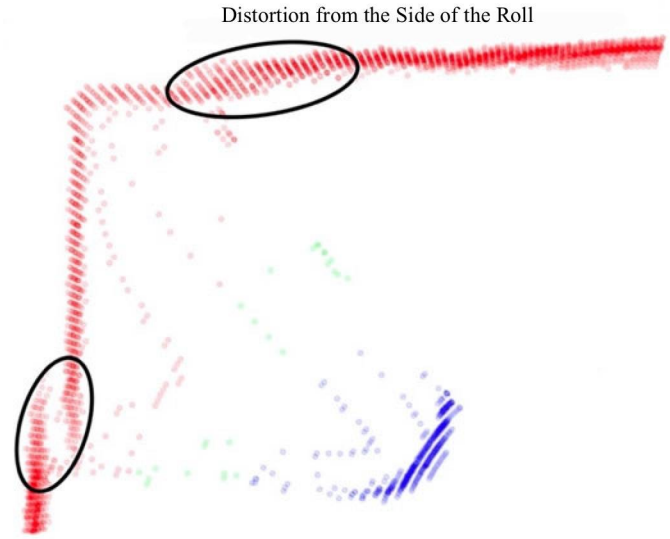

Image 4.2-8 Top View of the Roll

Since 30° in vertical range of the scanning is more than needed, the height of the roll seems small compared to the background wall. A clearer image is shown in *Image 4.2-9* on the right.

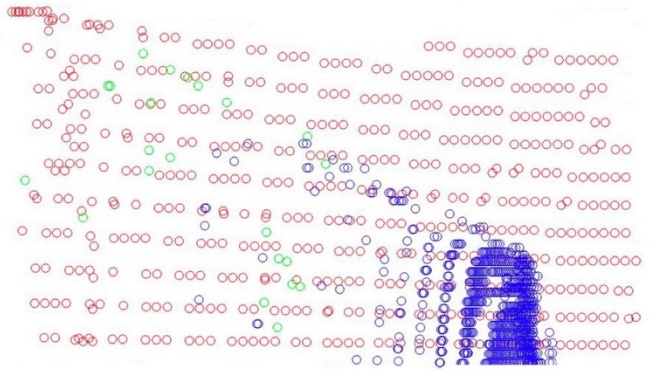

Image 4.2-9 Side View of the Roll (Partial)

The last and final test is to scan a relatively complex setup, as shown in *Image 4.2-10*, the three boxes and the horizontal floor are covered by tissue in order to get a better diffuse reflection, and dimensions of the experimental arrangement is marked on it. Since the discontinuity of the background showed in *Image 4.2-8* during the last test is not ideal, the whole setup is placed against the wall to reduce the distortion at the edge of the object, as the longer distance between the scanned object and the background, the server this problem could be. However, at the same time, more different colors are used to represent the depth of the scanning. The color range are chosen by pre-calculating the distance range of the scan, using black for 0~20cm, blue for 20~30cm, green for 30~35cm, cyan for 35~45cm and yellow for distance greater than 45cm. the colors are arranged from dark to light as the distance increases.

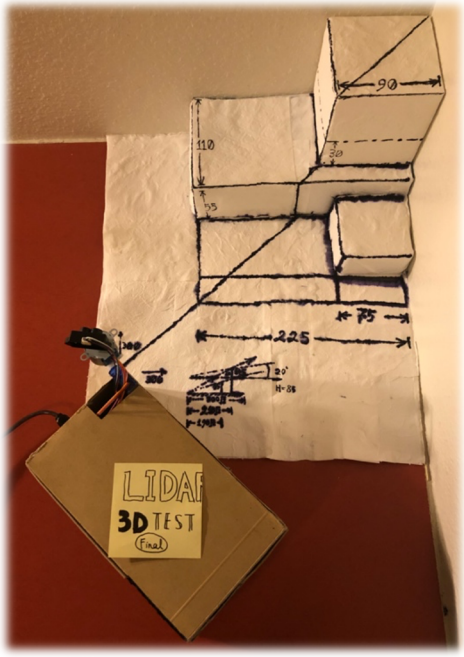

Image 4.2-10 Arrangement for 3D Test - 3

The scanning range of this test is (-20°, 20°) in vertical off the horizontal plane, and 90° in horizontal plane. The resolution of the scanning is also set to the maximum, with $\Delta\theta$ and $\Delta\varphi$ equal to 1° and 0.7°. The whole scanning process took 3.3 hours and a video can be found in the files attached with this report.

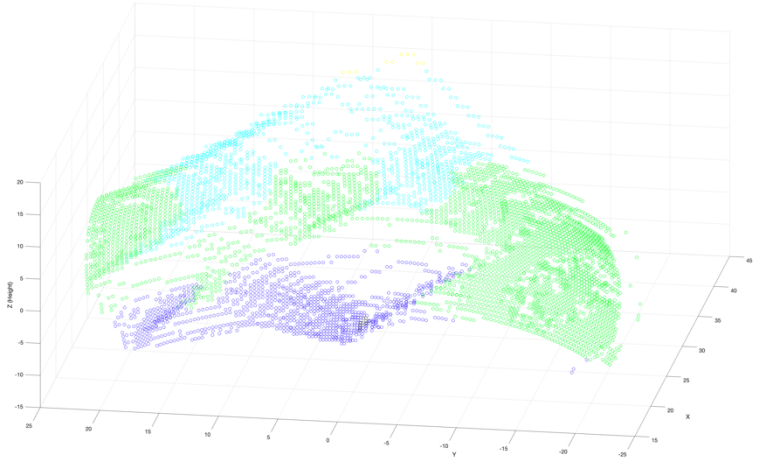

Image 4.2-11 3D View of the BOXES

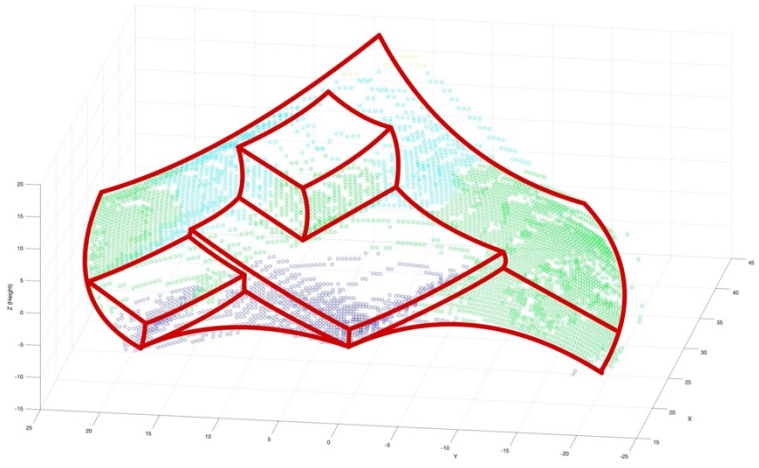

Image 4.2-11' 3D View of the BOXES (with hand-drawn auxiliary lines)

The red auxiliary lines are carefully hand-drawn after taking thorough view of the scanning diagram. All the *.flg* pictures in the file attached to the report can be opened with MATLAB and the view perspective can be easily changed by dragging the mouse pointer.

The curve of these lines is the result of the no-compact-enough joints of the motors and the LIDAR, as discussed in *Chapter 3.2.5 Product Packaging*. It can be seen more clearly in the side view of the scanning. At this stage, especially during a quarantine of the pandemic, we don't see any solution to better arrange the equipment to reduce this kind of system error. However, this problem can be dealt with simply by locating the LIDAR sufficiently far away from the object (horizontal angle range near 0°), at the cost of lowing the resolution rate of the scanning. What we suggest is this: always locate the height LIDAR near the middle of the object, and scan the object within a vertical angle close to 0° off the horizontal plane. By our experiments, this can significantly reduce this distortion of the object, but hardly eliminate it.

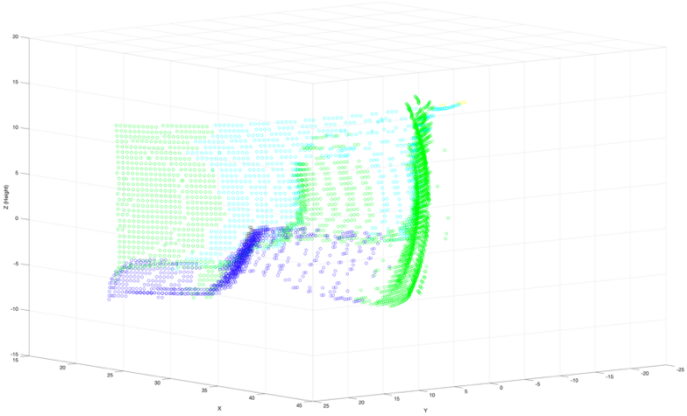

Image 4.2-12 Side View of the BOXES - 1

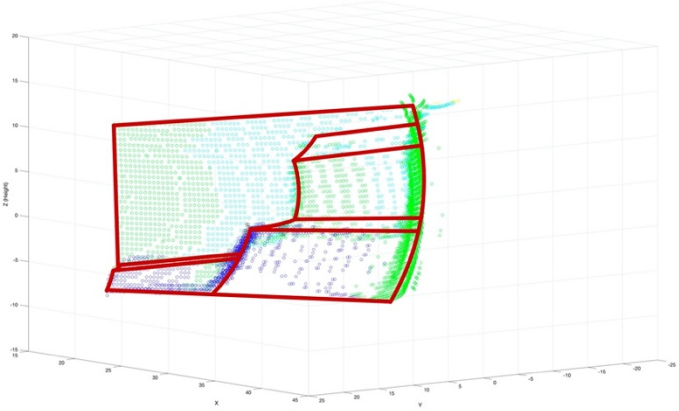

Image 4.2-12' Side View of the BOXES - 1 (with hand-drawn auxiliary lines)

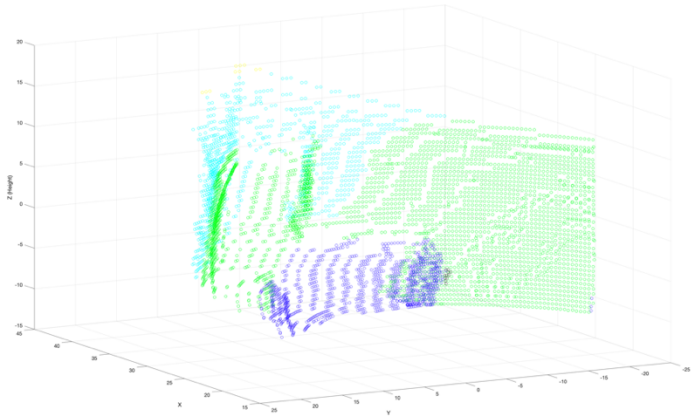

Image 4.2-13 Side View of the BOXES – 2

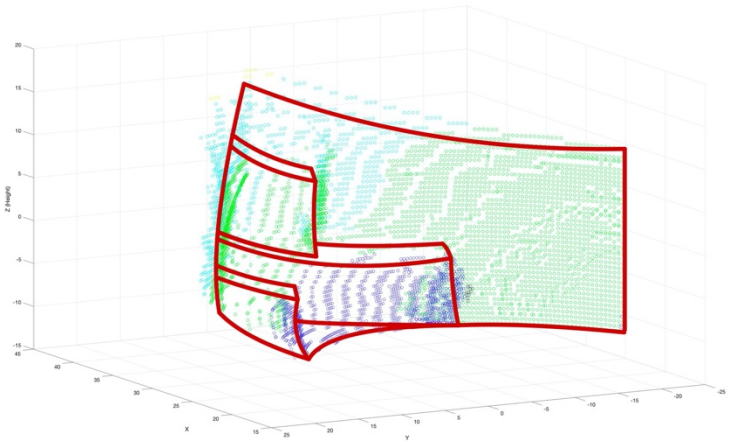

Image 4.2-13' Side View of the BOXES - 2 (with hand-drawn auxiliary lines)

From the top view of the result diagram, we try to actually read some length of the scanned objects, as there is not so much distortion from this view.

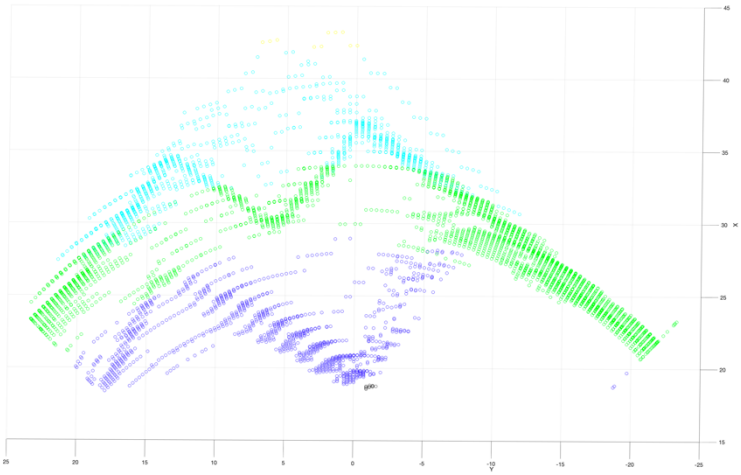

Image 4.2-14 Top View of the BOXES

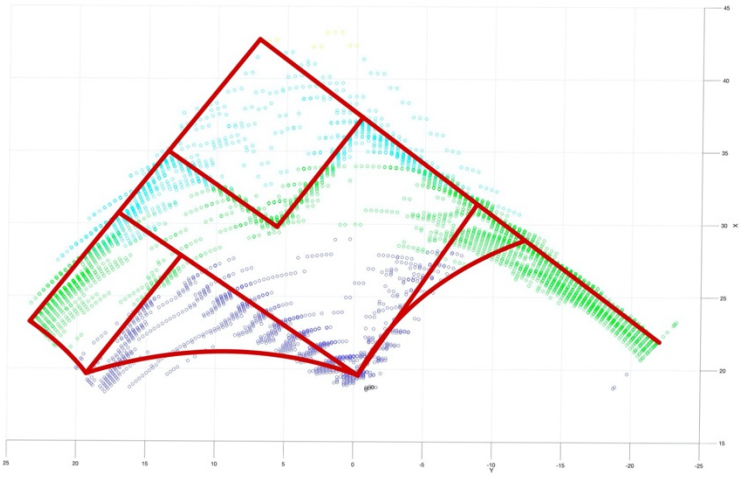

Image 4.2-14' Top View of the BOXES (with hand-drawn auxiliary lines)

From the plot above (*Image 4.2-14*), we can read some dimensions as shown in *Image 4.2-15* on the right.

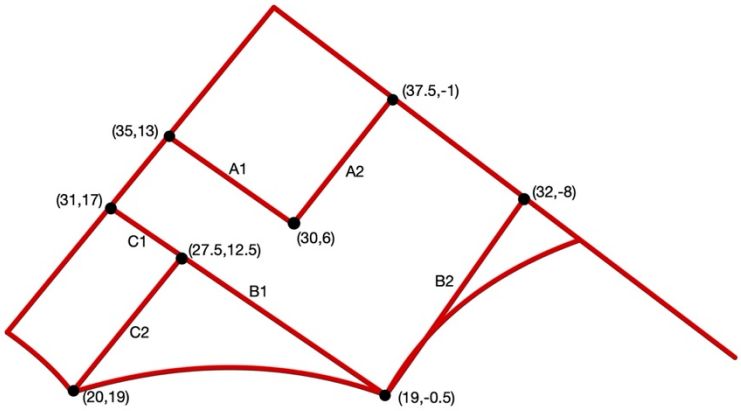

Image 4.2-15 Coordinates of Top View of the BOXES

By comparing them with the real values, we get the table:

	Measured Val. (cm)	Real Val. (cm)	ERROR Rate
A1	8.6	9.0	4.4%
A2	10.2	9.0	13.3%
B1	21.2	22.5	5.7%
B2	15.0	11.0	36.3%
C1	5.7	7.5	24.0%
C2	9.9	7.5	32.0%

From the tables we can conclude that the readings on the edges of A1, B1, C1 are relatively more accurate than the readings on edges of A2, B2, C2. The inaccuracy lies in the distortion which comes from the insufficient compact joints of motors and LIDAR sensor, and it's explainable systematically since for the LIDAR sensor, readings deviating further away from 0° in horizontal plane should be affected more than those towards the it, in this case, the right side of the result diagram is distorted more severely than the left side as it biases more from 0° horizontally, this is why the readings on A1 and A2 is way more precise than others in general. And, it is because, as said in the suggestions we made, the vertical distortions near the horizontal plane are significantly reduced.

If manual setup of the location of the LIDAR is included, we can reduce the distortions addressed in the final test. The solution is to set the horizontal scanning range close to 0° to get a small partial image of the whole object, and afterwards, manually move the LIDAR to another location to get another partial image. By repeating doing this, and by repeating this sufficient more, we can hopefully get a result diagram with no distortion theoretically, leaving the only possible improvement that could be done to our LIDAR: the resolution.

The most stirring thing, however, about this new scenario of scanning, is that instead of have a depth view from only one perspective, we can get a real 3D image of the scanned object! However, the precision of relocating the LIDAR could pose problems.

Another solution is to change the way we use the stepper motor. It is possible us the stepper motor to drive a gearing system, which make the LIDAR sensor rotate around the object, and the servo forms the vertical angle and the LIDAR reads a distance. As shown in *Image 4.2-16* below. The object to be measured is placed in the middle of the big gear, and the LIDAR is attached to the servo.

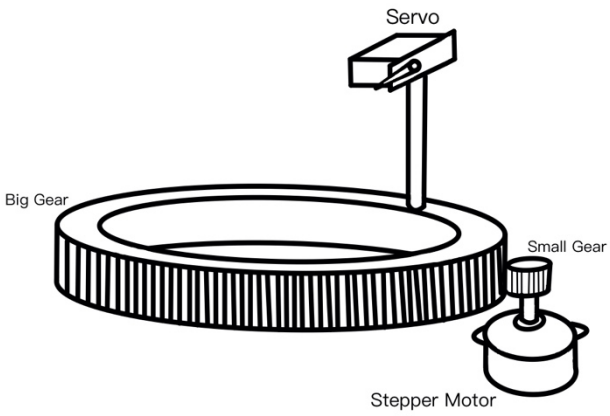

Image 4.2-16 Another Way to Arrange the Motors

5. Reference

5.1 Benewake - Product Manual of TFMini-S

https://www.gotronic.fr/pj2-sj-pm-tfmini-s-a00-product-mannual-en-2155.pdf

5.2 DFRobot - Product Manual of TFMini-S

https://wiki.dfrobot.com/TF_Mini_LiDAR_ToF__Laser_Range_Sensor_SKU__SEN0259

5.3 EXAMPLE – LIDAR (GitHub)

https://github.com/TravisLedo/LidarScanRender

5.4 EXAMPLE – LIDAR Reading on Arduino

https://www.gotronic.fr/pj2-35741-arduino-1710.pdf

5.5 3D Indoor Mapping System Using 2D LiDAR Sensor for Drones

International Journal of Engineering & Technology,7 (4.11)(2018) 179-183

by **M.R. Shahrin, F.H. Hashim, W.M.D.W. Zaki1, A. Hussain, T. Raj**

6. Arduino CODE

6.1 Initialization(to initialize the positions of the LIDAR, setting the initial angles of the servo and the stepper motor)

```
/*declaring-----------------------------------------------------------------------------------------*/
//control of the motors
const int servoPin = 10;      //pin of the servo
int posTheta_initial = 0;     //the initial position of servo
//working status indicator: whether initializing or working
int led1 = 23;                //orange: adjusting stepper motor
int led2 = 25;                //blue: ready for LIDAR
int led3 = 27;                //red: adjusting servo
int t = 0;
/*-------------------------------------------------------------------------------declaring*/

/*control functions of motors-----------------------------------------------------------------------*/
void servo(int angle)
{  for(int i=0;i<50;i++)
   {  int pulsewidth = (angle * 11) + 500;
      digitalWrite(servoPin, HIGH);
      delayMicroseconds(pulsewidth);
      digitalWrite(servoPin, LOW);
      delayMicroseconds(20000 - pulsewidth);
   }
   delay(100);
}//homemade function to replace <servo.h> to solve the colision with the LIDAR
void anticlockwise_test(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 2; i < 6; i++)
      {  digitalWrite(i, HIGH);
         delay(20);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(anti-clockwise) when initialing
void clockwise_test(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 5; i > 1; i--)
      {  digitalWrite(i, HIGH);
         delay(20);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(clockwise) when initialing
/*-----------------------------------------------------------control functions of motors*/


/*functions run once--------------------------------------------------------------------------------*/
void setup()
{
   pinMode(7,INPUT);          //switcher
   pinMode(led1,OUTPUT);      //orange
   pinMode(led2,OUTPUT);      //blue
   pinMode(led3,OUTPUT);      //red
   pinMode(servoPin, OUTPUT); //servoPIN
   for (int i=2;i<6;i++)      //stepper motor
   {pinMode(i,OUTPUT);}
}
/*-------------------------------------------------------------------functions run once*/

/*main loop-----------------------------------------------------------------------------------------*/
void loop()
{
   int n = digitalRead(7); //read of the switcher
   if (n == HIGH) //begin initialization
   {delay(500);t = t+1;n = 0;}
//orange light on: initializing the stepper motor
   while (t==1)
   {
     digitalWrite(led3,LOW);
     digitalWrite(led1,HIGH);n =digitalRead(7);
     if (n == HIGH)
      {delay(500);t = t+1;digitalWrite(led1,LOW);} //exit
     if (analogRead(0) < 500)
      {anticlockwise_test((500-analogRead(0))*5/1023);}
     if (analogRead(0) > 700 )
      {clockwise_test(analogRead(0)*5/1023);}
   }
//red light on:  initializing the servo
    while (t == 2)
    {
      digitalWrite(led3,LOW);
      digitalWrite(led2,HIGH);n =digitalRead(7);
      if (n == HIGH)
      {delay(500);t = 0;digitalWrite(led2,LOW);} //exit
      posTheta_initial = map(analogRead(0),0,1023,0,179);
      servo(posTheta_initial); //set the servo to exit of initialization  a long press
    }
//blue light on: ready for LIDAR
    digitalWrite(led3,HIGH);
}
/*-----------------------------------------------------------------------------------------main loop*/
```

6.2 **LIDAR 1D** (one-dimensional scan, realized only by the stepper motor)

```
int dist;       //LiDAR actually measured distance value
int strength;  //LiDAR signal strength
int check;     //check numerical value storage
int i;
int uart[9];   //store data measured by LiDAR
String clearReceived; //used to clear old data
const int HEADER=0x59;  //data package frame header
float posPhiGain = 5.625;    //the rotation of stepper motor
                             //! make sure it's an int after *256/180 !
float PhiRANGE = 112.5;      //the detecting range of Phi HORIZONTAL(stepper mootor)
float posPhi = 0;
bool ojbk = 1;

void setup()
{
   for (int i=2;i<6;i++)      //stepper motor
   {pinMode(i,OUTPUT);}
   Serial.begin(9600);//set the Baud rate of Arduino and computer serial port
   Serial1.begin(115200);//set the Baud rate of LiDAR and Arduino serial port
}

void anticlockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 2; i < 6; i++)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}
void clockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 5; i > 1; i--)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}

void LIDAR_read(float posPhi)
{
   while(ojbk)
   {
      if (Serial1.available())//check whether the serial port has data input
      {
         for (int k=1; k<=10; k++)
         {
          clearReceived = String(clearReceived) + String(char(Serial1.read()));
          delay(1);
         }
         clearReceived = "";
         if(Serial1.read()==HEADER)//determine data package frame header 0x59
         {
            delay(10);
            uart[0]=HEADER;
            if(Serial1.read()==HEADER)//determine datapackage frame header 0x59
            {
               uart[1]=HEADER;
               for(i=2;i<9;i++)//store data to array
               {
                  uart[i]=Serial1.read();
               }
               check=uart[0]+uart[1]+uart[2]+uart[3]+uart[4]+uart[5]+uart[6]+uart[7];
               if(uart[8]==(check&0xff))//check the received data as per protocols
               {
                  dist=uart[2]+uart[3]*256;//calculate distance value
                  strength=uart[4]+uart[5]*256;// calculate signal strength value
                  Serial.print("dist = ");
                  Serial.print(dist);//output LiDAR testsdistance value
                  Serial.print('\t');
                  Serial.print("strength = ");
                  Serial.print(strength);//output signal strength value
                  Serial.print("\t");
                  Serial.print("phi = ");
                  Serial.print(posPhi*180/256); //real Phi angle
                  Serial.print('\n');
                  delay(10);
                  ojbk = 0;
               }
            }
         }
      }
   }
}

void loop()
{
   PhiRANGE = PhiRANGE*256/180; //convert the Phi angle to stepper motor STEPs
   posPhiGain = posPhiGain*256/180;
   for (posPhi = 0; posPhi < PhiRANGE; posPhi += posPhiGain)
   {
      LIDAR_read(posPhi);
      delay(200);
      clockwise(posPhiGain);
      delay(2000);
      ojbk = 1;
   }
   LIDAR_read(posPhi);
   ojbk = 1;
   delay(200);
   anticlockwise(PhiRANGE);
   delay (200);
   PhiRANGE = PhiRANGE*180/256; //convert back
   posPhiGain = posPhiGain*180/256; //convert back
}
```

6.3 **LIDAR 3D** (3-dimensional scan)

```
int posThetaGain = 20;       //the rotation of servo motor
float posPhiGain = 11.25;    //the rotation of stepper motor
                             //! make sure it's an int after *256/180 !
int ThetaRANGE = 100;        //the detecting range of Theta VERTICAL (servo motor)
float PhiRANGE = 112.5;      //the detecting range of Phi HORIZONTAL(stepper mootor)
/* FOR ADJUSTMENT*/          //! make sure it's an int after /posPhiGain !

//control of the motors
const int servoPin = 10;      //pin of the servo
float posTheta = 0;           //the position of servo motor
float posPhi = 0;             //the position of stepper motor
int posTheta_initial = 0;     //the initial position of servo
//LIDAR reading
int dist;
int strength;
int check;
int uart[9];
int i;
const int HEADER=0x59;
bool ojbk = 1;                //whether the LIDAR has successfully read a data
String clearReceived;         //used to clear old data
int led3 = 27;                //red: LIDAR working

void setup()
{
   pinMode(servoPin, OUTPUT); //servo is at PIN6
   for (int i=2;i<6;i++) //stepper motor
   {pinMode(i,OUTPUT);}
   Serial.begin(9600); //set the Baud rate of Arduino and computer serial port
   Serial1.begin(115200); //set the Baud rate of LiDAR and Arduino serial port
}

void anticlockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 2; i < 6; i++)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}
void clockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 5; i > 1; i--)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}
void servo(int angle)
{  for(int i=0;i<50;i++)
   {  int pulsewidth = (angle * 11) + 500;
      digitalWrite(servoPin, HIGH);
      delayMicroseconds(pulsewidth);
      digitalWrite(servoPin, LOW);
      delayMicroseconds(20000 - pulsewidth);
   }
   delay(100);
}

void LIDAR_read(float posTheta, float posPhi)
{
   while(ojbk)
   {
      if (Serial1.available())  //check whether the serial port has data input
      {
         for (int k=1; k<=10; k++)
         {
          clearReceived = String(clearReceived) + String(char(Serial1.read()));
          delay(1);
         }
         clearReceived = "";
         if(Serial1.read()==HEADER)  //determine data package frame header 0x59
         {
            delay(10);
            uart[0]=HEADER;
            if(Serial1.read()==HEADER)  //determine datapackage frame header 0x59
            {
               uart[1]=HEADER;
               for(i=2;i<9;i++)  //store data to array
               {
                  uart[i]=Serial1.read();
               }
               check=uart[0]+uart[1]+uart[2]+uart[3]+uart[4]+uart[5]+uart[6]+uart[7];
               if(uart[8]==(check&0xff))  //check the received data as per protocols
               {
                  dist=uart[2]+uart[3]*256;  //calculate distance value
                  strength=uart[4]+uart[5]*256;  //calculate signal strength value
                  Serial.print("dist = ");
                  Serial.print(dist);  //output LiDAR testsdistance value
                  Serial.print('\t');
                  Serial.print("strength = ");
                  Serial.print(strength);  //output signal strength value
                  Serial.print("\t");
                  Serial.print("theta = ");
                  Serial.print(posTheta);
                  Serial.print("\t");
                  Serial.print("phi = ");
                  Serial.print(posPhi*180/256); //real Phi angle
                  Serial.print('\n');
                  delay(10);
                  ojbk = 0;
               }
            }
         }
      }
   }
}

void loop()
{
   digitalWrite(led3,HIGH);
   PhiRANGE = PhiRANGE*256/180; //convert the Phi angle to stepper motor STEPs
   posPhiGain = posPhiGain*256/180;
   for (posTheta = 0; posTheta <= ThetaRANGE; posTheta += posThetaGain)
   {
      posTheta_initial = map(analogRead(0),0,1023,0,179);
      servo(posTheta+posTheta_initial);
      delay(2000);
      for (posPhi = 0; posPhi < PhiRANGE; posPhi += posPhiGain)
      {
         LIDAR_read(posTheta, posPhi);
         delay(200);
         clockwise(posPhiGain);
         delay(2000);
         ojbk = 1;
      }
      LIDAR_read(posTheta, posPhi);
      ojbk = 1;
      delay(200);
      anticlockwise(PhiRANGE);
      delay (200);
   }
   PhiRANGE = PhiRANGE*180/256; //convert back
   posPhiGain = posPhiGain*180/256; //convert back
}
```

6.4 **LIDAR Merged** (the program merging the 3D LIDAR and the initialization, but sees logic conflicts)

```
/*declaring-----------------------------------------------------------------------------------------*/

/* FOR ADJUSTMENT*/
int posThetaGain = 20;       //the rotation of servo motor
float posPhiGain = 11.25;    //the rotation of stepper motor
                             //! make sure it's an int after *256/180 !
int ThetaRANGE = 100;        //the detecting range of Theta VERTICAL (servo motor)
float PhiRANGE = 112.5;      //the detecting range of Phi HORIZONTAL(stepper mootor)
/* FOR ADJUSTMENT*/          //! make sure it's an int after /posPhiGain !

//control of the motors
const int servoPin = 10;      //pin of the servo
float posTheta = 0;           //the position of servo motor
float posPhi = 0;             //the position of stepper motor
int posTheta_initial = 0;     //the initial position of servo
//LIDAR reading
int dist;
int strength;
int check;
int uart[9];
int i;
const int HEADER=0x59;
bool ojbk = 1;                //whether the LIDAR has successfully read a data
String clearReceived;         //used to clear old data
//working status indicator: whether initializing or working
int led1 = 23;                //orange: adjusting stepper motor
int led2 = 25;                //blue: adjusting servo
int led3 = 27;                //red: LIDAR working
int t = 0;
/*-------------------------------------------------------------------------------------------declaring*/

/*control functions of motors-----------------------------------------------------------------------*/
void servo(int angle)
{  for(int i=0;i<50;i++)
   {  int pulsewidth = (angle * 11) + 500;
      digitalWrite(servoPin, HIGH);
      delayMicroseconds(pulsewidth);
      digitalWrite(servoPin, LOW);
      delayMicroseconds(20000 - pulsewidth);
   }
   delay(100);
}//homemade function to replace <servo.h> to solve the colision with the LIDAR


void anticlockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 2; i < 6; i++)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(anti-clockwise)
void clockwise(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 5; i > 1; i--)
      {  digitalWrite(i, HIGH);
         delay(3);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(clockwise)
void anticlockwise_test(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 2; i < 6; i++)
      {  digitalWrite(i, HIGH);
         delay(20);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(anti-clockwise) when initialing
void clockwise_test(int num)
{  for (int count = 0; count < num; count++)
   {  for (int i = 5; i > 1; i--)
      {  digitalWrite(i, HIGH);
         delay(20);
         digitalWrite(i, LOW);}}
}//the control of the stepper motor(clockwise) when initialing
/*-------------------------------------------------------------------control functions of motors*/
```

```
/*functions for LIDAR-------------------------------------------------------------------------------*/
void LIDAR_read(float posTheta, float posPhi) //the LIDAR takes 3 mearsurements and get an average
{
   while (ojbk)
   {
      if (Serial1.available())//check whether the serial port has data input
      {
         for (int k=1; k<=10; k++)
         {
          clearReceived = String(clearReceived) + String(char(Serial1.read()));
          delay(1);
         }
         clearReceived = "";
         if(Serial1.read()==HEADER)//determine data package frame header 0x59
         {
            delay(10);
            uart[0]=HEADER;
            if(Serial1.read()==HEADER)//determine datapackage frame header 0x59
            {
               uart[1]=HEADER;
               for(i=2;i<9;i++)//store data to array
               {
                  uart[i]=Serial1.read();
               }
               check=uart[0]+uart[1]+uart[2]+uart[3]+uart[4]+uart[5]+uart[6]+uart[7];
               if(uart[8]==(check&0xff))//check the received data as per protocols
               {
                  dist=uart[2]+uart[3]*256;//calculate distance value
                  strength=uart[4]+uart[5]*256;// calculate signal strength value
                  Serial.print("dist = ");
                  Serial.print(dist);
                  Serial.print("\t");
                  Serial.print("Stre = ");
                  Serial.print(strength);
                  Serial.print("\t");
                  Serial.print("theta = ");
                  Serial.print(posTheta);
                  Serial.print("\t");
                  Serial.print("phi = ");
                  Serial.print(posPhi*180/256); //real Phi angle
                  Serial.print('\n');
                  delay(10);
                  ojbk = 0;
               }
            }
         }
      }
   }
}
/*-------------------------------------------------------------------------functions for LIDAR*/

/*functions run once--------------------------------------------------------------------------------*/
void setup()
{
   pinMode(7,INPUT);          //switcher
   pinMode(led1,OUTPUT);      //orange
   pinMode(led2,OUTPUT);      //blue
   pinMode(led3,OUTPUT);      //red
   pinMode(servoPin, OUTPUT); //servo is at PIN6
   for (int i=2;i<6;i++)      //stepper motor
   {pinMode(i,OUTPUT);}
   Serial.begin(9600);        // the arduino system at 9600 bps
   Serial1.begin(11500);   //open the serial port for measurement at 115200 bps
}
/*-----------------------------------------------------------------------------functions run once*/
```

```
/*main loop-----------------------------------------------------------------------------------------*/
void loop()
{
   int n = digitalRead(7); //read of the switcher
   if (n == HIGH) //begin initialization
   {delay(500);t = t+1;n = 0;}
//orange light on: initializing the stepper motor
   while (t==1)
   {
     digitalWrite(led1,HIGH);n =digitalRead(7);
     if (n == HIGH)
      {delay(500);t = t+1;digitalWrite(led1,LOW);} //exit
     if (analogRead(0) < 500)
      {anticlockwise_test((500-analogRead(0))*5/1023);}
     if (analogRead(0) > 700 )
      {clockwise_test(analogRead(0)*5/1023);}
   }
//blue light on:  initializing the servo
    while (t == 2)
    {
     digitalWrite(led2,HIGH);n =digitalRead(7);
     if (n == HIGH)
     {delay(500);t = t+1;digitalWrite(led2,LOW);} //exit
     posTheta_initial = map(analogRead(0),0,1023,0,179);
     servo(posTheta_initial); //set the servo to initial position
     delay(100); //make the beginning of the LIDAR a long press
    }


//LIDAR is functionning
    while (t == 3)
    { digitalWrite(led3,HIGH);
      delay(1000);
      digitalWrite(led3,LOW);
      PhiRANGE = PhiRANGE*256/180; //convert the Phi angle to stepper motor STEPs
      posPhiGain = posPhiGain*256/180; //convert the Phi angle to stepper motor STEPs
      for (posTheta = 0; posTheta <= ThetaRANGE; posTheta += posThetaGain)
      {
         servo(posTheta+posTheta_initial);
         delay(2000);
         for (posPhi = 0; posPhi < PhiRANGE; posPhi += posPhiGain)
         {
            LIDAR_read(posTheta, posPhi);
            delay (200);
            ojbk = 1;
            clockwise(posPhiGain);
         }
         LIDAR_read(posTheta, posPhi);
         delay (200);
         ojbk = 1;
         anticlockwise(PhiRANGE); //put the stepper motor to initial position
       }
       t = 0;//exit
       digitalWrite(led3,LOW);
       PhiRANGE = PhiRANGE*180/256; //convert back
       posPhiGain = posPhiGain*180/256; //convert back
    }
}
/*-----------------------------------------------------------------------------main loop*/
```