redundant information; typically presented in the Project Plan

Hardware requirements

Hardware requirements are all functionalities related to the decision made about hardware and how they're programmed.

Description	Must have	Should have	Could have	Won't have
A maximum horizontal scan range of 180°	X			
Detect obstacles at a minimal range of 100m	X			
Send motor position and laser data to the user application	X			
Receive commands from the application	X			
At 90° the motor can move from left to right in 1 second	X			
Has fail-safes in place when going out of range or when connection to the user app is lost	X			
The minimal & maximum speed can be modified		X		
Reports the intensity of a measurement		X		
The horizontal scan range can be modified		X		
The data sent by the device has a checksum			X	
Automatically centers the laser at startup			X	
Can also change its vertical angle				X
The default middle of the scan range (straight forward) can be modified				X

Constraints

With any project one must know the limitations of the development, these are called the constraints. It is important in any project to have clearly defined constraints to avoid spending time on adding unnecessary features that end up costing too much time. With as a result, important features not being completed in time.

Time

The original design of the assignment wanted to have the prototype move both vertically and horizontally. This was to account for the change in angle as the ship moves closer towards the lock. After a few weeks of development, it was found that this would take too much time to incorporate, so instead it was decided that the prototype would only scan horizontally instead.

Hardware

Sensor maritime has developed their own custom development board which they refer to as the IO box. This is a modified Infineon board and on top of that it has custom shield to even further expand its possibilities. This board was designed to be easily modified in order to suit a great many possible projects. While still maintaining uniformity across developers, making it easier for them to cooperate.

This does however add a constraint to the project. in order to program for this board a special IDE has to be used called DAVE™. DAVE is a modified version of Eclipse, it has been expanded with a number of functionalities specifically for this Infineon board.

This IDE is not widely used making finding information on it very difficult when part of it is not functioning correctly.

Sprint breakdown

As mentioned earlier this project is broken down into smaller sprints with repetitive phases. The way this document has structured the description of these by categorizing them per phase. And then describing what happened during this phase in each sprint. In order to avoid confusion regarding what is being done in each sprint, this chapter will describe in a generalized manner what each sprint focused on.

Sprint 1

This sprint was about getting orientated in the project. all actions were focused on setting up the basic requirements needed to fulfill the following sprints. This entails gaining basic knowledge about the tools and work environment, as well as creating a broad/basic design of the product to serve as a guideline until the respective aspects can be more specifically refined.

Sprint 2

This sprint was focused on the development of the hardware section of the product. This means deciding which hardware is to be used, and how the system will be set up. As well as the code controlling it.

Sprint 3

This sprint was geared towards the user application. This entails deciding on the right platform, the code structure behind it. But also regarding user interacting and implementing it into the overall design.

approach is missing; research approach missing;
overall jurnal like presentation layer

the report should contain at most pseudocode, presented source code does not add any value to the writing

Communication

This class is responsible for establishing & maintaining a connection with the Lockscout. As well as handling incoming and outgoing data.

```
//Client for UDP communication.
public UdpClient client;
//Endpoint of device.
public IPEndPoint endpoint;
//Check if the device is current connected.
public bool connected;
//Current position of the motor.
public int currentPosition;
//Current speed the motor is moving at.
public int currentSpeed;
//Intensity of the measurement of the laser.
public char[] intensityData;
//Data returned from the laser. by default this is distance and intensity but can also be an OK when sending commands.
public char[] laserReply;
//Checks if the endpoint has been bound to the socket.
public bool endPointBound = false;

//Endpoint on this end.
IPEndPoint communicationEndpoint;
//UDP socket to communicate with the device.
Socket socket;
```

Description of variables

For a detailed explanation about how this class' functions operate please consult appendix 4.1.

Visualization

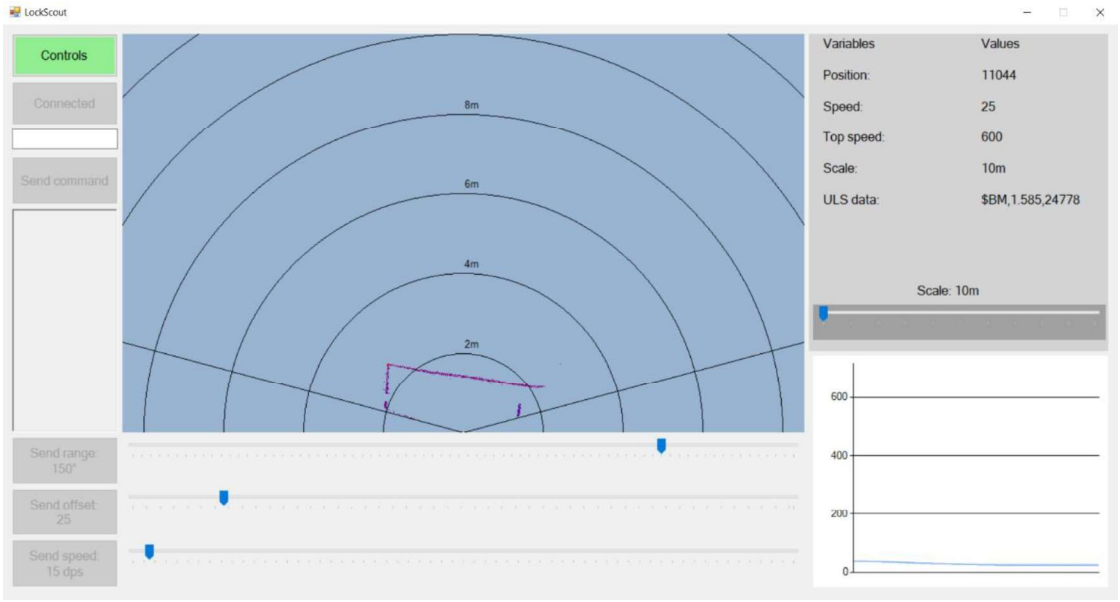
This class exclusively take care of everything regarding the radar on the application. It does all the math on how the data is projected, handles all the controls of the visualization and updates the buffer of values.

```
//Grid on the radar.
double[] circleGrid;
//Array of points on the radar.
public float[] dotMap;
//Array of the intensity of each point.
public int[] intensityMap;
//Laser data
public char[] basicMeasurement;
//Short array to compare current vs previous position to determine direction, an array is used instead of a single into to account for fluctuations.
public int[] directionValues;

//Width of the radar.
int width;
//Height of the radar.
int height;
//Vertical center of the radar.
int radarOriginX;
//Horizontal center of the radar.
int radarOriginY;
//Range represented by the height of the radar,
public float maxRange;
//Colour intensity of a dot.
public int intensity;
//Maximum mount of positions the motor can have.
public int positions;
//Current position the motor has.
public int currentPosition;
//Previous position the motor had.
public int lastPosition;
//Highest position the motor is allowed to take. (on the radar this is the right border)
public int maxPosition;
//Lowest position the motor is allowed to take. (on the radar this is the left border)
public int minPosition;
//Length of the range borders in the radar.
float boundaryLength;

//Bitmap where the visuals will be printed on.
Bitmap radarBitMap;
Graphics G;
Brush B;
//Pen to draw the grid.
Pen gridPen;
//Pen for the radar boundary lines.
public Pen linePen;
```

Description of variables



And here you see the final version of the user application, although calling it a user application is actually not accurate. This application is meant for future developers to use for the configuration and testing of the hardware.
on the left side you can see a text field, this is where commands be set to configure the laser. The reason the laser control does not have separate buttons like can be seen on the rest of the application is because the laser has a very high degree of settings that can be changed. It would have taken up more far too much space. Given even the provided serial application had 4 separate tabs of settings.

Besides that, the motor can be fully controlled with the buttons and slider seen in the image above. A user can control the range the motor can move, how fast its top speed is and how low the bottom speed is. The reason there is a top and bottom speed is because the speed is determined by a sine wave in order to slow it down just before changing direction. This is done for safety reasons so the hardware cannot damage itself.
The highest top speed available is based on the range of the motor, it is split into 5 sections.
the speed is measured in dots per second (dps).

1. 0-30 degrees = top speed of 50 dps.
2. 0-60 degrees = top speed of 150 dps.
3. 60-90 degrees = top speed of 225 dps.
4. 90-120 degrees = top speed of 350 dps.
5. 120-150 degrees = top speed of 475 dps.
6. 150-180 degrees = top speed of 600 dps.

If you look closely, you'll notice that the color of the visualization changes at different spots, this colour is an indicator of the intensity of the scan at that point, with red being the strongest and blue being the weakest.

overall presented quite informally