EITN90 Radar and Remote Sensing
Lab 1

January 23, 2018

1 Learning outcomes

This lab demonstrates the basic operation of a simple scanning radar, capable of range measurements only. After completing this lab, you will

- see how a simple scanning radar/sonar system can be constructed using commercial off-the-shelf electronic components, and how bottle-necks can be identified
- understand how controlling the specular scattering can be used to reduce the signature of objects
- understand how a high pulse repetition frequency can introduce false echoes
- understand how a finite beam width of the sensor affects the radar image

2 Equipment

In this lab we will use a ranging sensor based on ultrasound waves. The principle is the same as for electromagnetic waves: the distance to an object is given by the time ΔT for an echo to return to the sensor,

$$R = \frac{v\Delta T}{2} \tag{1}$$

where $v = 340\,\mathrm{m/s}$ is the speed of sound in air. A photograph of the system can be seen in Figure 1. It consists of three parts:

- An ultrasound sensor HC-SR04, operating at 40 kHz.
- A stepper motor with a driver circuit board.
- A control unit based on the Arduino platform.

The parts have been assembled using connection cables without soldering. They are placed in a custom made 3D-printed frame. The sensor is mounted on the stepper motor using a 3D-printed holder. Data sheets for the different parts are available next to this document. Please don't disconnect or change any of the wiring during the lab.

The arduino control unit can be programmed in a language similar to C/C++. Connecting the unit to a regular computer using a USB cable, supplies the unit with its +5 V

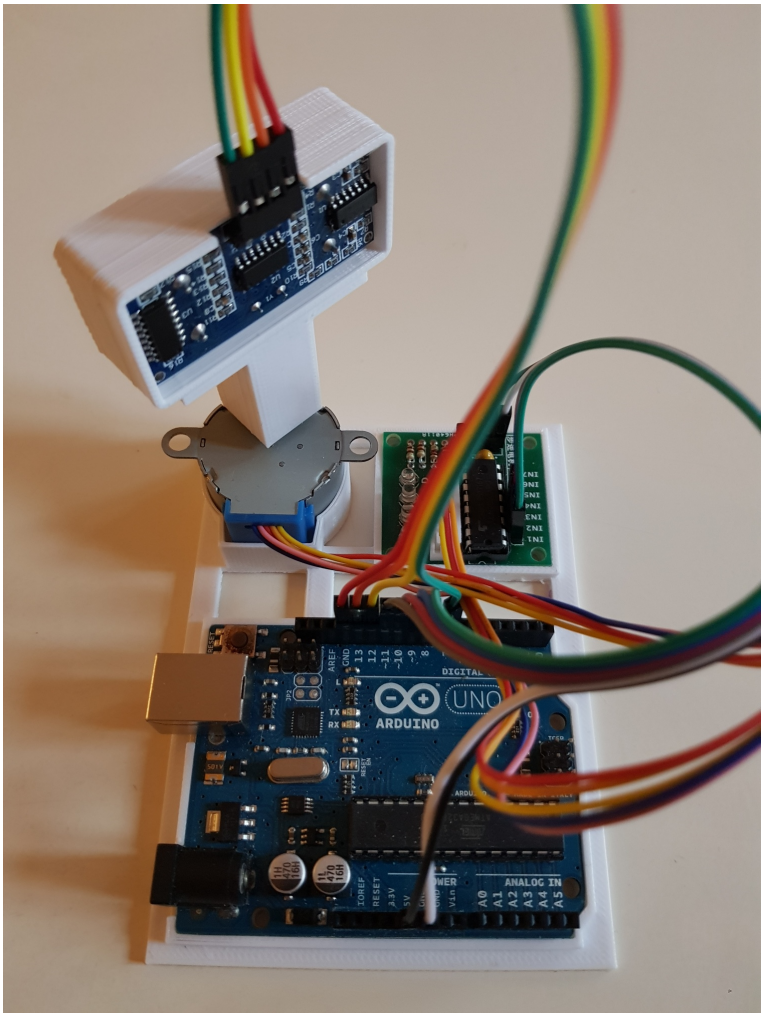

Figure 1: Equipment for a scanning radar (sonar).

supply voltage. USB is also used to transfer programs to the unit and to communicate with the unit over the serial interface. The easiest way to program the unit and use the serial interface is using the arduino software[1], but we will also use matlab to read the data from the serial interface later. In this lab, there are ready-made scripts to use, which are discussed under each respective experiment below. If something gets out of hand, you can simply unplug the USB cable to powerdown the control unit. There is also a reset button on the arduino board if the program on the unit freezes.

The cost for the different parts is around 70 kr for the sensor, 80 kr for the stepper motor, and 100 kr for the control unit. Add to this some cost for connection cables and 3D-printing.

3 Experiment 1 — Ranging and unambiguous range

In our first experiment we are going to focus only on the ranging capability of the sensor. We will also go through how the control unit can be programmed.

3.1 The script

The script below can be downloaded from the course web site and is used to control the sensor:

- `range.ino`

Download the script and open it in the `arduino` application on your computer. In the beginning of the script there are a number of parameters, setting the behavior of the system. Make sure you read through the code and understand what it does. However, you do not need to understand the full syntax of the programming language, we will only do minor changes to the code, i.e. modify parameters.

When the script is first uploaded to the control unit, it goes through an initialization phase where first the function `setup()` is executed, and it then goes into an eternal loop function `loop()`. In the initialization, we typically need to explain to the control unit how the different pins are to be used. In this experiment, we will use the distance sensor sensor without rotating it. The initialization of the motor is therefore omitted here.

Before moving on, please go through the tasks below:

- Compute the wavelength λ of the ultrasound wave.
- Compute the duration in time of one full ultrasound burst (8 cycles). Compute the corresponding distance.
- One measurement cycle is controlled by the function `Measure()`. Go through this code in detail, and compare with the description of the sensor in the datasheet.
- In the function `Measure()`, the number 58.2 is used to convert the time measured to a distance. Do your own calculations and show how to find this number.
- Test the program for compiling errors by clicking the icon with a check mark to the farthest left. You should see a progress bar, and some text output in the window below the code.

[1] available at: https://www.arduino.cc/en/Main/Software

3.2 Measurements

Connect the control unit to your computer with a USB cable. When you have controlled that the program compiles without errors (check mark icon), you can upload the script using the icon with a right-pointing arrow next to the check icon.

The script is programmed to transmit the calculated distance reading to the computer using the serial interface. To observe the data sent by the system, you can use the Serial Monitor of the arduino software. To open it press the rightmost icon with a magnifing glas (or press Ctrl-Shift-M). This will open a window where you can see the data sent on the serial port. With the system powered up and script uploaded, you should see a sequence of numbers being output.

- Place an object at close range (e.g. 10 cm) in front of the sensor. Verify that the range presented corresponds to the physical distance between the object and the sensor. Can you verify the resolution claimed in the datasheet of 3 mm?
- Draw a diagram detailing the events in `Measure()`, including the typical times spent on each stage (cf. the timing diagram in the datasheet). You can measure the time using the function `micros()` and then send the measurements to the computer using the serial interface. An example can be seen where the echo time is measured inside the code. Do the timings change when the distance to the object changes?
- Change the position of the object. What is the maximum distance where you get reliable distance readings? Can you relate the readings to the timing parameters in your diagram from task 2?
- How do you need to modify the code to obtain accurate distance readings beyond 1 metre?
- Experiment with placing an object close to the sensor again. Are there conditions where you get false distance readings? Think about how waves are scattered.
- Set up two objects at different distances, both of which are illuminated by the sensor. Which one corresponds to the number displayed on the serial port?

Feel free to do your own experiments with the setup. You can investigate the ambiguity of the measurement, or the interference of multiple sensors. Or you can try to modify the code to also measure velocity.

4 Experiment 2 — Images and beam pattern

We will now start using the motor in the system. By rotating the sensor, we can take measurements at different angles, and build up a classical radar range image.

4.1 The script

A different script will be used, which includes control of the motor:

- `scanningradar.ino`

A stepped motor has several inputs. These inputs are given digital levels, which need to be cycled through in order to make the motor turn, according to the schedule below:

SWITCHING SEQUENCE

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

One full forward and backward cycle of this is implemented in the two functions `StepForward()` and `StepBackward()`. The `loop()` function is modified so as to scan back and forth within an angular sector of 90° (controlled by the variable `ScanAngle`). The `Measure()` function is modified to take an argument `n` corresponding to the current angle of measurement. It now also outputs two numbers to the serial port: the angle and the range.

- Go through the code and make sure you understand the changes.
- Test the program for compiling errors.
- Turn the sensor manually to a good starting position (this will be very difficult to adjust when the program is running). It will start to scan clockwise.

In order to make a radar image, we need to read the angle and range information written by the control unit to the serial port, store these in a data structure, plot the range as function of angle, and update the plot as the sweep goes on. This is done by the matlab script `radar.m`.

4.2 Measurements

Connect the control unit to your computer with a USB cable. When you have controlled that the program compiles without errors (see above), you can upload the script using the icon with a right-pointing arrow next to the check icon.

To observe the data sent by the system, go to the menu Tools — Serial Monitor (or press Ctrl-Shift-M). This will open a window where you can see the data sent on the serial port. With the system powered up and script uploaded, you should see a sequence of pairs of numbers being output, separated by a comma. This is the angle and range, being output by the `Measure()` function.

In order to read the data into matlab, you need to first close the Serial Monitor (otherwise you will have two programs trying to read (and discard) the same data on the

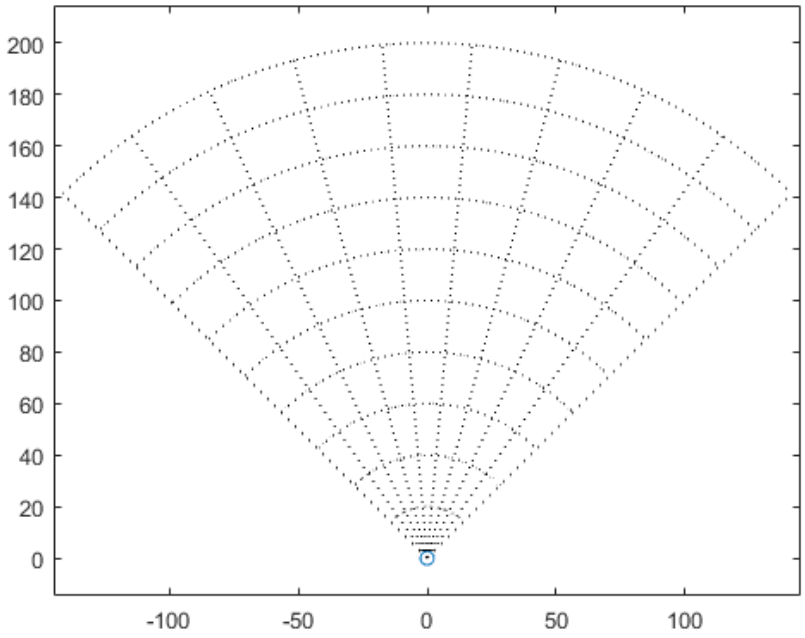

Figure 2: Grid for drawing the beampattern

serial port). Start up matlab and the script `radar.m`. You should now see a stream of numbers being output in the command window, and a figure with the radar data being updated as the radar scan goes on.

- Insert a large, flat object facing the sensor (for instance a book cover or a wall), so there is a strong echo present for a large part of the sweep. What differences can you see between the image of this object and the real physical object?
- Place two or more objects at different distances and angles. Determine the minimum separation distance.
- As can be seen in the datasheet for the sensor, it detects objects within a certain angle, which distorts the image. Determine this angle by inserting an object that can be thought of as a point scatterer (or rather line scatterer, we only need it to be thin in the plane of the scan). The angular width of this object on the radar screen will correspond to the full beamwidth of the sensor. Try your measurements at different distances and draw the measurements in a grid like produced by the matlab script (cf. Fig. 2).

5 Implications for a real radar surveillance system

The system investigated in this lab has similarities with airport surveillance radar systems. Since the speed of sound is 340 m/s and the speed of light is $3 \cdot 10^8$ m/s, the scale difference between the systems is roughly a factor of 10^6. Airport surveillance radar typically make continuous 360° rotations of the radar antenna, which has a narrow beamwidth

Figure 3: Example of an airport surveillance radar antenna, the ASR-9 (source: Wikipedia).

in azimuth and wider in elevation. This can be seen from the shape of the reflector in Figure 3, which has larger aperture in azimuth than in elevation.

The following questions are based on the design of the ASR-9 system in Figure 3.

- For a desired surveillance range of 100 km, what is the highest PRF that allows unambiguous range?
- What bandwidth (or pulse length) is required for a resolution of 150 m?
- What is the two-way flight time of a pulse to the edge of the range?
- The radar is designed to transmit 18 pulses at 256 different angular positions. Ignoring the mechanics, what is the shortest time necessary to complete one full 360° sweep? How does this compare to the actual speed of rotation used (12.5 RPM)?
- With a transmit peak power of 1.2 MW at 2.8 GHz and antenna gain of 33 dB, what is the expected peak SNR for an aircraft with radar cross section $\sigma = 10\,\mathrm{m}^2$ at the edge of the surveillance range?
- Compare the parameters of the HC-SR04 Ultrasound sensor (cf. previous experiments) to the ASR-9 surveillance radar system. Complete Table 5.

Table 1: Comparison with a real radar system

	HC-SR04	ASR-9
Principle	Ultrasound	Radar
Propagation Speed		
Frequency		
Bandwidth		
Rotation Speed		

Once you completed all tasks, discuss your results with the lab leader to be approved for the lab.