EITN90 Radar and Remote Sensing
Lab 2

February 8, 2018

1 Learning outcomes

This lab demonstrates the basic operation of a frequency modulated continuous wave (FMCW) radar, capable of range and velocity measurements. After completing this lab, you will

- see how an FMCW radar can be assembled from commercial off-the-shelf components, and integrated on a Raspberry Pi platform
- understand the use of I/Q-signals to identify direction of motion
- understand the basics of Doppler processing of radar signals using Fourier analysis
- understand the implications of bandwidth and discretization on resolution
- be able to identify the bottle neck(s) in the lab system

2 Equipment

The equipment used in this lab is depicted in Figure 1. It consists of a commercially available radar module, IVS-162 from Innosent GmbH, a control unit consisting of a Raspberry Pi computer with an AD/DA extension card ADC DAC Pi Zero from AB Electronics UK, and a non-inverting amplifier for the tuning voltage based on an LM358 operational amplifier. A basic schematic of the equipment is given in Figure 2.

If you would like to build the system your self, the total cost is around 1000 SEK, and assembly instructions can be found on the course web site.

2.1 Radar module

The core of the system is the radar module. This is described in detail in the datasheet and Application Note II, available at the course web site. According to the manufacturers web site, its recommended use is for door openers and industrial applications, and its features are

- VCO-Transceiver centered @ 24GHz
- FMCW/FSK capable; therefore measurement of distance as well as recognition of stationary objects possible (depending on modulation)

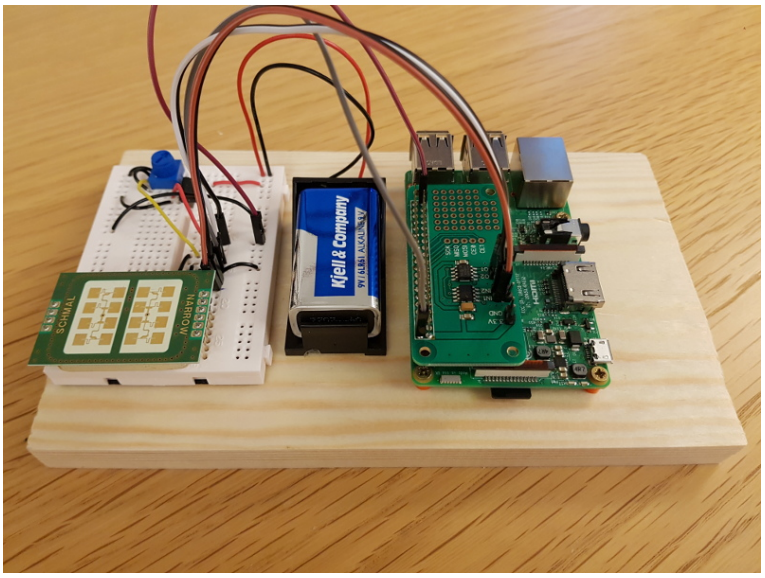

Figure 1: Equipment for an FMCW radar.

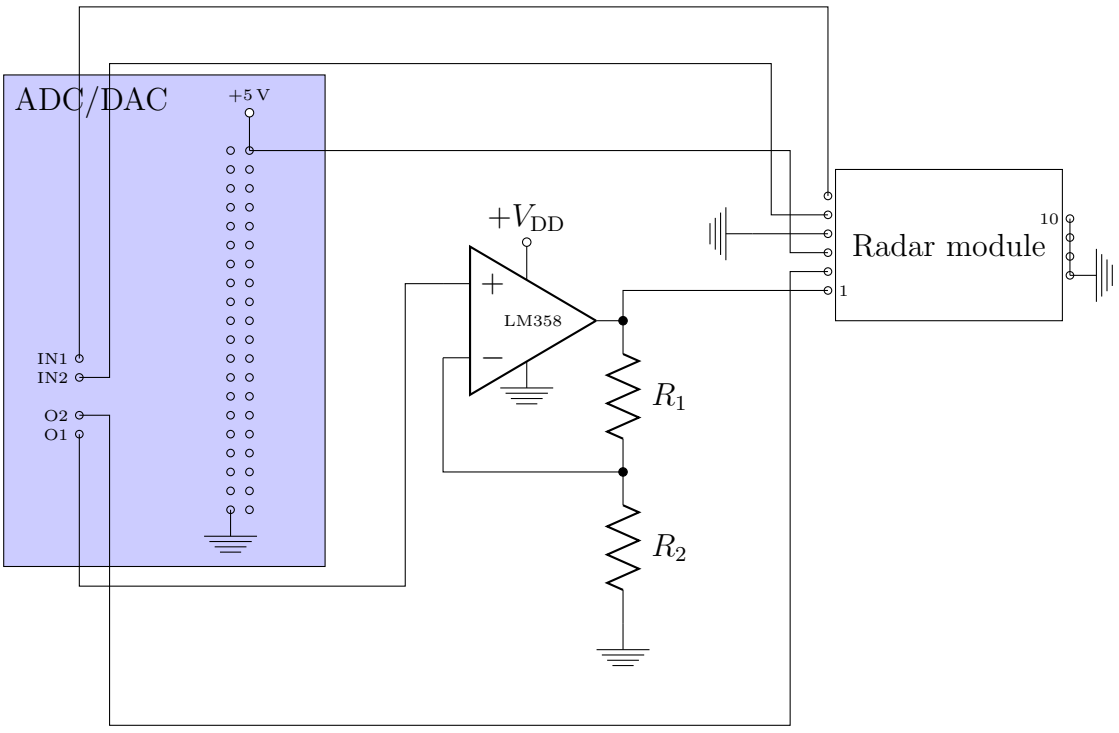

Figure 2: Schematic of the radar system.

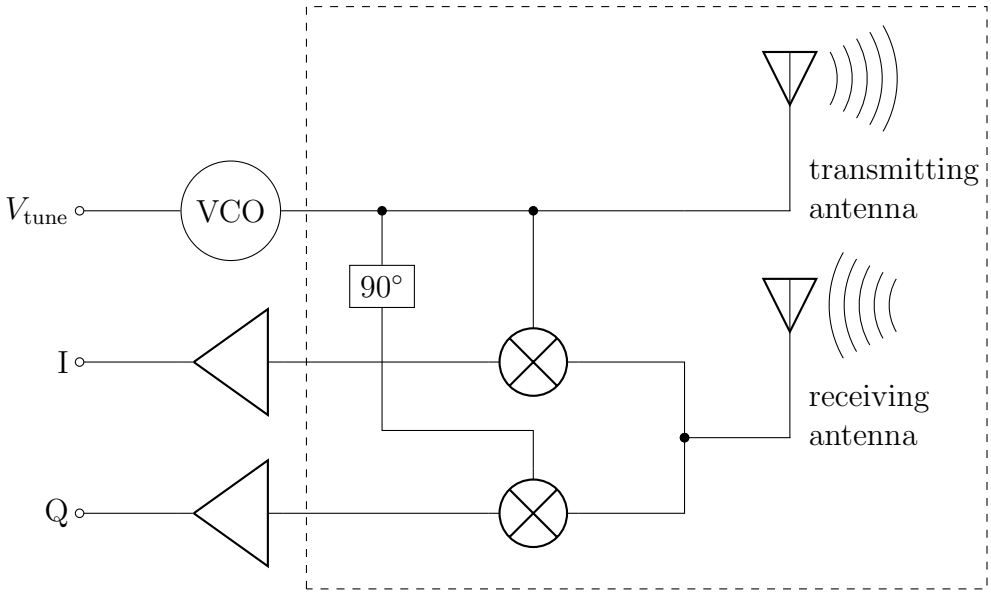

Figure 3: Principal operation of the radar sensor. The region inside the dashed line is operating on the carrier wave 24 GHz. The amplifiers include a low pass filter.

- split transmit and receive path for maximum gain
- stereo (dual channel) operation for direction of motion identification
- IF-pre-amplifier, bandwidth limited for lowest noise performance
- compact outline dimensions

A simplified diagram is given in Figure 3. The operational frequency of the sensor is determined by a tuning voltage, where $V_{\text{tune}} \in [0, 10\,\text{V}]$ corresponds to $f \in [24.0\,\text{GHz}, 24.6\,\text{GHz}]$, see Figure 7 in Application Note II. The received signal is mixed with the transmit signal, low passed and amplified, and appears at the I (inphase) output. The received signal is also mixed with a 90° delayed transmit signal, low passed and amplified, and appears at the Q (quadrature) output. Further data is available in the datasheet.

2.2 Operational amplifier

The Raspberry Pi extended with the ADC DAC Pi Zero card, can supply a variable voltage but only up to 2.048 V. In order to utilize a wider bandwidth of the radar module, this voltage is ramped up by a non-inverting amplifier circuit based on the LM358 operational amplifier. This amplifier is driven by a 9 V battery, allowing the tuning voltage to change between 0 and 7 V (the maximum output of the OP is always a bit lower than the driving voltage). With this setting, the operating frequency of the radar module changes from $f_1 = 24.0\,\text{GHz}$ to $f_2 = 24.425\,\text{GHz}$, as the output voltage O1 from the AD/DA card changes from 0 to 2.048 V.

2.3 AD/DA card

The ADC DAC Pi Zero card has two outputs and two inputs. Output O1 is used to control the tuning voltage of the radar module, and output O2 is used as an enable signal

(when low, the radar module is enabled). The inputs IN1 and IN2 are used to sample the I and Q outputs of the radar module.

2.4 Raspberry Pi

The final part of the system is a Raspberry Pi unit, which is a full fledged computer with a linux operating system installed. It can be programmed in any language, however we will stick to the language `python`, which is a script language pretty similar to matlab. As a demonstration of the speedup possible with a compiled language, we also have a version of the control program written in C, but no coding is necessary from you.

The unit is powered by a USB cable inserted in the mini-USB connector

2.4.1 Connecting the Raspberry Pi

There are dedicated keyboards and mouses with USB connectors available in the lab. There are also HDMI cables for connecting to a monitor, and a USB cable for powering the Raspberry Pi through its micro-USB port.

- Connect keyboard, mouse, and monitor to the Raspberry.
- Only after doing the above, connect the Raspberry micro-USB to a regular USB port on one of the stationary computers (for power only). Starting with the monitor plugged in makes the Raspberry choose the correct screen resolution.
- You should see a startup sequence on the screen. It ends with a warning message about standard password not being changed, click away this window.
- Open a command window (ctrl-alt-T or the icon on the top row). This is where we will be operating.
- If you need to save some data from the Raspberry, like a saved figure or such, you can use a USB stick.

2.4.2 Booting the Raspberry Pi

Once the Raspberry Pi is booted, you will see the warning message in Figure 4. The warning message can be ignored as it is a warning to change the default password of the Raspberry Pi.

Changing screen resolution

The Raspberry Pi usually detects the optimum resolution of the connected screen. But, in some cases, it does not work, and the screen resolution should be set manually. To set the resolution manually open the *Terminal* and type:

```
sudo raspi-config
```

Once the configuration menu is open go to *Advanced Option* > *Resolution* and set the screen resolution. The device will then require a reboot.

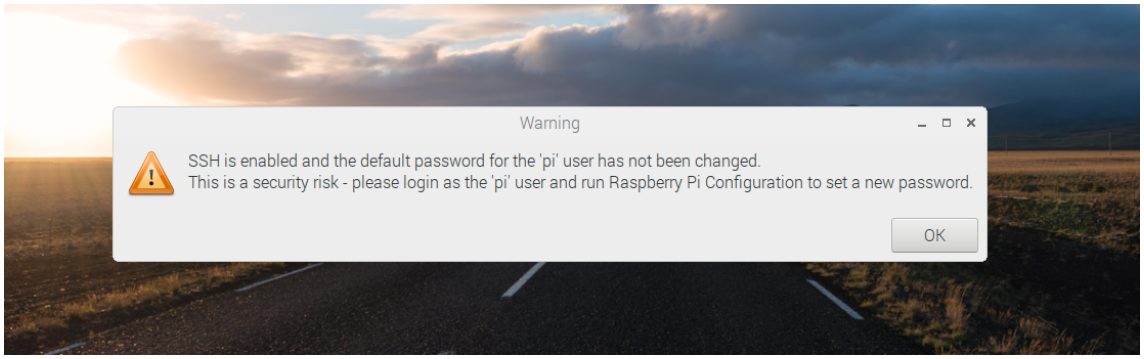

Figure 4: Startup warning message.

2.4.3 Basics on command line handling

You do not need to understand much of linux to use the Raspberry Pi. Some basics are collected here, feel free to use google vigorously during the lab.

`ls`: Lists the files in the current folder.

`pwd`: Lists the name of the current folder.

`cd`: Changes folder: `cd <name of folder>` moves to the new folder, `cd ..` moves to the parent folder. Folder names are separated by a slash, `/home/pi/myfiles`.

`nano`: A simple text editor. Runs in the command window, with a menu of commands displayed at the bottom.

`leafpad`: A text editor with a simple GUI.

`python3`: (Version 3 of) the python language. Write `python3 doppler.py` to run the script `doppler.py`. Python is a popular language in the open source community, and has computational abilities similar to matlab. One important difference to matlab is that a trailing semicolon is not necessary.

2.4.4 Saving Python Figures

Throughout the lab and for the report you will need to save measurement figures. Python figures such as the one shown in Figure 5 can be saved by clicking the save button and then specifying the file path. External USB memory storage can be found under the path

/media/pi/USB_NAME

where USB_NAME is the name of the USB stick.

3 Experiment 1: CW Doppler Radar

In our first experiment we will use a fixed frequency and study Doppler shift only. This is the most basic operation of the radar module, aimed for motion detection. In order to detect the direction of motion, we use the I and Q signals.

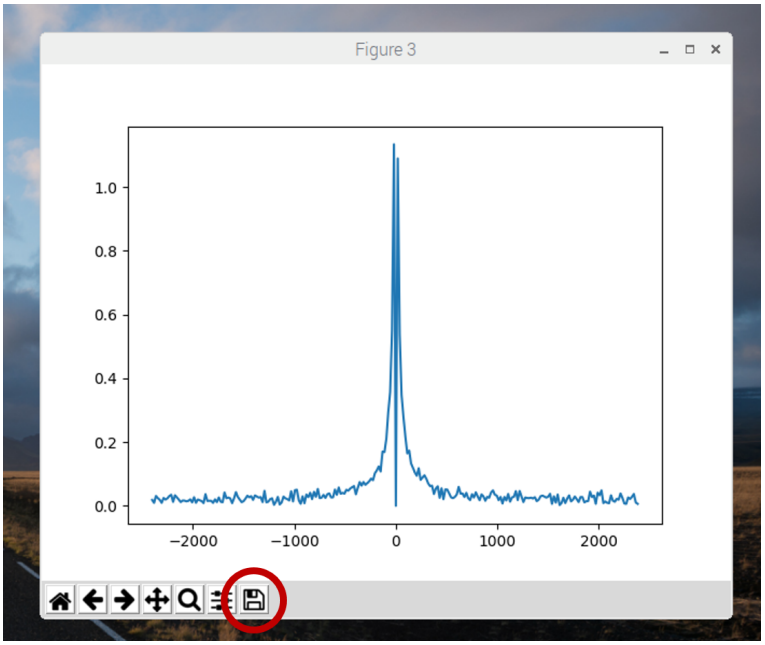

Figure 5: Python Figure, Save button highlighted with red circle.

3.1 The script

The script to control the radar module is

- `doppler.py` (located in /home/pi/myfiles/radar/)

This script consists of an infinite loop, which reads the analog voltage levels on the I and Q pins of the radar module, and combines them to a complex digital value. The script takes N consecutive measurements, and constructs the analytical signal

$$a_n = I_n + \mathrm{j}Q_n, \quad n = 1, \ldots, N \tag{1}$$

Since the sampled signal is overlayed on a DC level of about 2 V, the mean value is first subtracted and then the Fourier transformation is applied (using the Fast Fourier Transform, FFT)

$$\boldsymbol{A} = \mathrm{FFT}(\boldsymbol{a} - \langle \boldsymbol{a} \rangle), \quad \text{where} \quad \langle \boldsymbol{a} \rangle = \frac{1}{N} \sum_{n=1}^{N} a_n \tag{2}$$

The locations of the peaks in $\boldsymbol{A}$ correspond to the main Doppler frequencies. Positive frequencies mean the object is coming towards the radar, negative frequencies that it is going away.

The script outputs the dominating Doppler frequency in plain text, and updates a graph of the entire spectrum. In order to get results from the suggested measurements below, you need to open the script in a text editor (either `nano` or `leafpad`), and make suitable changes. For instance, with respect to outputting different results, look for the variable `PlotMode` inside the script.

Make sure you keep a copy of the original script in case you accidentally change something you cannot correct.

3.2 Measurements

In order to have a well-defined speed to observe, hold a small fan in front of the radar module. This should generate a number of spikes, and the separation between the spikes is proportional to the fan rotation frequency.

- Use the radar range equation to estimate the SNR of the radar module as function of range based on parameters in the data sheet. Use a target RCS of $\sigma = 1\,\mathrm{m}^2$. What is the maximum range at which you can expect SNR $> 0\,\mathrm{dB}$? (Hint: the data sheet expresses output power in Equivalent Isotropic Radiated Power EIRP $= G_\mathrm{t} P_\mathrm{t}$. The receive gain G_r can be estimated from the beam widths in azimuth and elevation.)
- Capture and save a graph of the raw time domain data, including the DC level.
- Capture and save a graph of the time domain data where the DC level has been stripped off.
- Capture and save a graph of the frequency domain data. Estimate the SNR. Calculate the rotation frequency of the fan.
- Insert timing controls in the code, and measure the time for
  - The measurement loop.
  - The signal processing part using the FFT.
  - The output part, including writing text to the screen and plotting. Try with graphical plotting, and only text output.

  Include the data in the lab report for three different values of N.
- Remove the fan, and try to detect the movement of your hand as it moves towards or away from the radar. Do you see the different sign of the Doppler shifts?
- Can you suggest ways of improving the system, for instance leading to increased SNR or better Doppler shift resolution?

Transfer the saved graphs to a USB stick, to be used in the lab report. Ask the lab leader if you are uncertain on how to do the file transfer.

4 Experiment 2: FMCW Radar

In this experiment we use a tuning voltage to control the frequency of operation for the radar module. Due to the limited speed of the AD/DA, the result will be a frequency stepped modulation, where each step is similar to the situation in Experiment 1. The added information in this case will be the variation of phase between different frequencies of operation. This enables us to determine not only velocity, but also range.

4.1 The script

The script enabling the frequency modulation is

- `fmcw.py` (located in /home/pi/myfiles/radar/)

It is similar to the doppler script in the previous experiment, but instead of having the same frequency for all measurements, it is stepped from 24.00 GHz to about 24.425 GHz. The frequency can be stepped from lower to higher (up-chirp) or from higher to lower (down-chirp). The script does one up-chirp and one down-chirp, and saves the corresponding analytical signals (complex combination of I and Q)

$$a_n^{\text{up}} = I_n^{\text{up}} + \text{j}Q_n^{\text{up}}, \qquad n = 1, \ldots, N \tag{3}$$

$$a_n^{\text{down}} = I_n^{\text{down}} + \text{j}Q_n^{\text{down}}, \qquad n = 1, \ldots, N \tag{4}$$

Each of the signals is overlayed on a DC level and a ramp function corresponding to the tuning voltage, which needs to be subtracted. This is implemented in the function `EstimateAmplitudeOffset()`, the details of which are not explained here.

Once the DC level and ramp have been stripped off, the script then computes the Fourier transform of both signals and extracts the dominant frequencies Δf_1 and Δf_2 in each signal. The range and velocity can then be calculated according to

$$R = \frac{cT}{4B}(\Delta f_2 - \Delta f_1) \tag{5}$$

$$v = \frac{c}{4f_0}(\Delta f_2 + \Delta f_1) \tag{6}$$

where $B = 425\,\text{MHz}$ is the bandwidth of the system. See the Application Note II from the module manufacturer for more details.

4.2 Measurements

Point the radar module towards the ceiling. This should give a stable signal when there is no movement in the path of the radar.

- Capture and save a graph of the time domain raw data, including DC level and ramp.
- Capture and save a graph of the time domain data where the DC and ramp have been stripped off.
- Capture and save a graph of the frequency domain data. Try to estimate the SNR. What is the distance to the ceiling?
- Insert timing controls in the code, and measure the time for
  - The up-chirp and down-chirp sweeps.
  - The signal processing with the ramp extraction and FFT.
  - The output part, including writing text to the screen and plotting. Try with graphical plotting, and only text output.

  Include the data in the lab report for three different values of N.

- Can you introduce some asymmetry between positive and negative frequencies by introducing velocity components, like waving your hand or use the fan?
- Can you estimate the range resolution? Can it be improved using zero padding of the data? Produce and save suitable illustrations to be used in the lab report.

5 Experiment 3: porting to C

In order to improve on the pulse repetition frequency in the python script (which is an interpreted language), the corresponding code has been ported to C.

- `fmcw.c` (located in /home/pi/myfiles/radar/fmcw_c/)

How much faster is this? Can you identify the bottle-neck of the system?

6 Written report

You get approved on the lab only after a written lab report has been approved. One purpose of this is to give you feedback on your writing before the final project is due.

The report should be self-contained and structured as follows:

- **Introduction**
  Give a brief overview of the working principles of CW Doppler radar and FMCW radar. Make use of references to literature.
- **Method**
  Describe how you set up your experiments and achieved your results.
- **Results**
  Present your results. These should include results corresponding to the bullet lists under experiments 1 and 2, and an estimate of the speedup of the system using the C implementation in experiment 3.
- **Analysis and discussion**
  Analyze your results. How accurate are they? Can you suggest an application where a corresponding radar system could be of use?