EMBEDDED SIGNAL PROCESSING SYSTEMS (course 1TE682) Uppsala University – Spring 2015 Report for Project 3

Tim Josefsson
Dept. of Information Technology
Uppsala University
Uppsala Sweden
Tim.Josefsson.9673@student.uu.se

Philip Åkerfeldt
Dept. of Information Technology
Uppsala University
Uppsala Sweden
Philip.Akerfeldt.4987@student.uu.se

30th March 2015

Contents

1 Introduction **3**

2 Observations **3**

3 Method **4**

4 Implementation **4**

5 Performance evaluation **5**

6 Results & Discussion **5**

7 Conclusion **6**

1 Introduction

In 1842 Austrian physicist Christian Doppler made an interesting discovery. He found that the frequency of sound from a moving source changed as the source was moving relative to a observer. What he found was that the frequency of the incoming signal would be higher as the source moved towards the observer and the frequency would be lower as the source was moving away from the receiver, as shown in figure 1. This discovery was thus named the Doppler Effect.

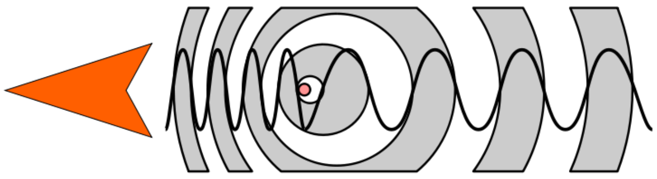


Figure 1: The Doppler Effect illustrated.[1]

The Doppler effect has been adopted in a myriad of different ways over the last century and one very common and prominent application of the Doppler effect is to measure the speed of moving objects. Today this is commonly used in traffic cameras and the like and the way this works is by placing a transmitter and receiver along the road and having the transmitter send out a continuous high frequency signal, this signal will then bounce on oncoming vehicles and produce an echo which will be caught by the receiver. The echo will then have a different frequency as opposed to the original signal and by using the findings of Christian Doppler one can calculate the speed and direction of the object which produced the echo. An illustration of this can be seen in figure 2.

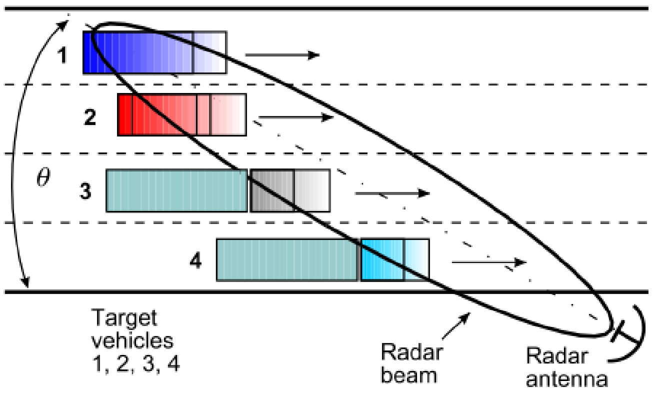


Figure 2: Doppler effect used for speed measuring in traffic.[2]

2 Observations

When looking at the signals from the movement of the fan it is of interest to study the Fourier transform of the signal echo, studying the signal on a digital oscilloscope will show something resembling figure 3 where we can clearly see the original signal (in this case 40kHz) and the frequency shifted signals produced by a object moving away from the sensor or towards the sensor.

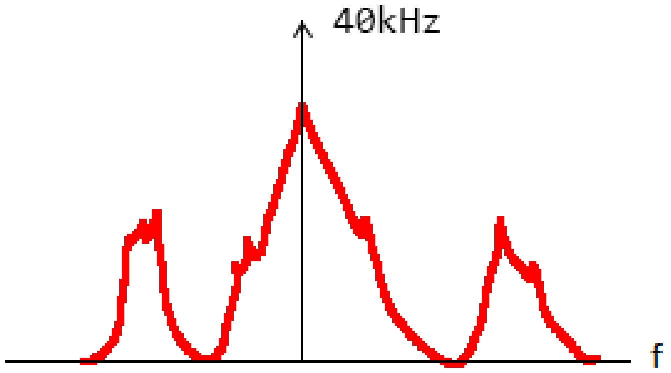


Figure 3: A very rough sketch of the transform of the signal received at the receiver.

3 Method

In order to utilize the Doppler effect to calculate the speed of an incoming or departing object we are first interested in finding the frequency shift, f_{dk}. The frequency shift can also be expressed as a function of the transmitted signal and the received echo giving us the following function:

$$f_{dk} = f_{rk} - f_0 \tag{1}$$

where f_{rk} is the frequency of the received echo and f_0 the frequency of the transmitted signal. Now we also know that f_{dk} can be expressed as a function of the velocity of the object v_k, the speed of sound c, the transmitted frequency f_0, and the angle of the sensor (see θ in figure 2). This gives us the following formula:

$$f_{dk} = 2\frac{v_k}{c} f_0 cos(\theta) \tag{2}$$

Inserting equation (1) into equation (2) and solving for v_k gives us:

$$v_k = \frac{c}{2cos(\theta)}(\frac{f_{rk}}{f_0} - 1) \tag{3}$$

In our implementation we know that the sensor will be at a angle $\theta = 45°$ so we can further simplify equation (3) into the following:

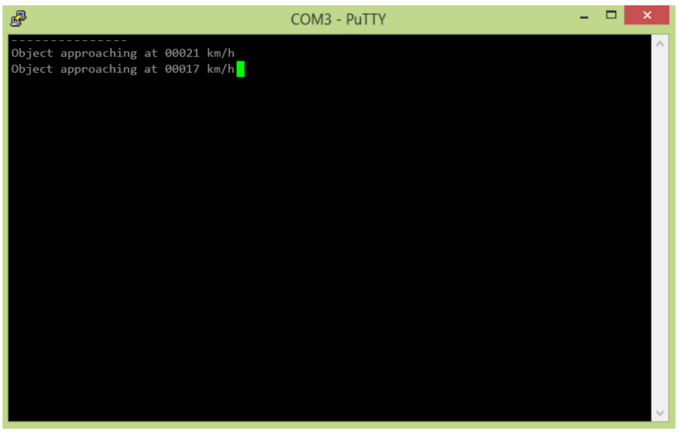
$$v_k = \frac{c}{\sqrt{2}}(\frac{f_{rk}}{f_0} - 1) \tag{4}$$

Now in order to find the direction of the object we are measuring we simply need to look at the ratio $\frac{f_{rk}}{f_0}$ since we know that the frequency of a object approaching the sensor will be higher and the frequency of a object departing from the sensor will be lower. Thus if the ratio $\frac{f_{rk}}{f_0} < 1$ we know that the object is departing from the sensor and if $\frac{f_{rk}}{f_0} > 1$ we know that the object is approaching the sensor.

4 Implementation

In order to implement the method above on the embedded system we make use of the Fourier transform of the incoming signal (the echo). We first find the index of the transmitted signal,

at 40kHz. After this we use this index to partition the buffer into two partitions, one containing frequencies higher than 40kHz and one containing frequencies lower than 40kHz. We then find the index of the spike in each partition which constitutes the lower and higher frequency of objects moving towards or departing from the sensor. Finally we apply equation (4) on the lower and the higher frequency, convert that result to km/h and print it to PuTTy producing the following result:
The interested reader will find the entire c-code implementation in the appendix.



```
---------------
Object approaching at 00021 km/h
Object approaching at 00017 km/h
```

Figure 4: Example of PuTTy when running the embedded system.

5 Performance evaluation

The implementation that has been done is fairly robust in case of accuracy, resolution and detectability. This does not, however, mean that the system is a finished product. There are still some things that can be optimized even though the result are really good. The system determines the accuracy (or resolution) by dividing the sampling frequency in a specific amount of samples which in this case is 1024. If this sampling amount would be increased a better accuracy (or resolution) would be possible to acquire.

There are numerous factors that can affect the detectability of the system. Whereas our system can differentiate between a lorry and a small car is hard to say without empiric tests and observations. One thing is certain about the system that was implemented and that is that it didn't really have that much of an issue with distinguishing between the different fans. Something that was a concern when doing this was however how to position the fans correctly in order to get a good reading. How to improve this is easily done with hardware in form of a platform in which the measuring equipment and fans reside. If the solution were meant to be on the software side of the system then that is something that will have to be developed further.

6 Results & Discussion

The implementation showed good results when showing measuring speed of both the departing fan and the approaching fan. It was easy to see whenever the speed of any fan decreased or increased. However as with all things the implementation was not perfect and even though the embedded managed to tell the speed fairly good there were also a lot of moments where it failed to produced reliable results, for instance showing the object as stand-still while it was in fact moving. This could be caused by the nature of signals since the echo from one fan could

easily cause problems with the measurements of the other fans. We also notice that the exact placement of the sensor relative to the fans caused measuring problems where once you had good values you could get much worse values if you just shifted the fan slightly.
Another problem could be that the sound of the transmitted signal interferes with the receiver seeing as they are located close to each other.

7 Conclusion

In general the ultrasonic sensor serves as a good choice for measuring speed and even with the basic technology used to implement the system presented in this report we were still able to achieve somewhat reliable results. However as discussed in the report the implementation is not without fault and definitely not perfect. There are several problems that appeared during the implementation process but these problem can all be remedied by using more sophisticated equipment in an real-life situation as opposed to measuring fans in a lab to simulate moving objects.
All in all the project is deemed to be successful in producing a good, working, application of speed measurement.

Appendix

The c-code for the embedded system:

```c
dsp16_trans_realcomplexfft(&DSP_COMP_BUFFER,&DSP_REAL_BUFFER, N_of_bits);
dsp16_vect_complex_abs(&DSP_REAL_BUFFER,&DSP_COMP_BUFFER, DATA_SIZE);

DSP_REAL_BUFFER[0]=0;
DSP_REAL_BUFFER[1]=0;
uint16_t speed = 330; //Speed of sound in air (m/s)

volatile uint16_t max = dsp16_vect_max(&DSP_REAL_BUFFER,(DATA_SIZE/2));
//find f0
uint16_t f0_index;
for(int i=0; i<=(DATA_SIZE/2); i++){
  if (max == (DSP_REAL_BUFFER[i])){
    f0_index = i;
    break;
  }
}
//find the indexes for the frequency shift
volatile uint16_t lower_max = dsp16_vect_max(&DSP_REAL_BUFFER,(f0_index-2));

uint16_t lower_index;
for(int i=0; i<=(f0_index-2); i++){
  if (lower_max == (DSP_REAL_BUFFER[i])){
    lower_index = i;
    break;
  }
}
volatile uint16_t upper_max = dsp16_vect_max(&DSP_REAL_BUFFER[(f0_index+2)],((
    DATA_SIZE/2)-f0_index-2));

uint16_t upper_index;
for(int ind=(f0_index+2); ind<=(DATA_SIZE/2); ind++){
  if (upper_max == (DSP_REAL_BUFFER[ind])){
    upper_index = ind;
    break;
  }
}

//calculate delta frequency
uint16_t delta_lower = dsp16_op_abs((lower_index-f0_index))*1000;
uint16_t delta_upper = ((upper_index-f0_index)*1000);

//calculate velocity
  uint16_t sqrt2 = 1.414;
uint16_t out_velocity = (speed/sqrt2)*(delta_upper/f0_index);
uint16_t inc_velocity = (speed/sqrt2)*(delta_lower/f0_index);
inc_velocity = ((inc_velocity*18)/5)/1000;
out_velocity = ((out_velocity*18)/5)/1000;
//print results
usart_write_line(USART, "\n\r----------------");
usart_write_line(USART, "\n\rObject approaching at ");
User_Usart_int2str(inc_velocity);
usart_write_line(USART, " km/h");
usart_write_line(USART, "\n\rObject departing at ");
User_Usart_int2str(out_velocity);
usart_write_line(USART, " km/h");
```

References

[1] Wikipedia, *Doppler Effect.* http://en.wikipedia.org/wiki/Doppler_effect

[2] Ping Wu, *Lab session 6: Correlation and Spectral Analysis.* Uppsala University, 2015.