readme.md

Assignment 2: Data Compression using Discrete Cosine Transform in TelosB Mote

Made by: Morten Sahlertz, Trung Thai, Andreas Arberg and Jacob Kjærager

Introduction

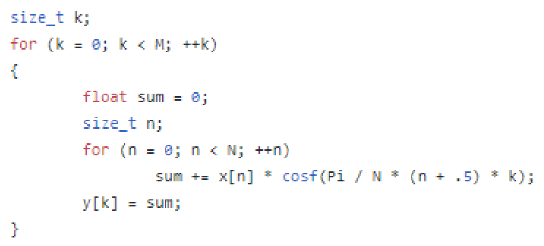
This project seeks to implement the Discrete Cosine Transformation on a TelosB Mote. The mote is running the Contiki OS. The project is successfull, and the M-sizes of the Cosine transformation component are regulated to observe a lower Mean Square Error.

Technologies

To this assignment two different technologies has been used. This is an DCT implementation in C and data visualization in Python

Discrete cosine transformation

The primary technology used for this task is the DCT. The technology is used in this context for compressing an ECG signal to be more energy effective to transmit from a constrained device as the TelosB mote. An example to how a DCT can be implemented in C can be seen at the Figure below.



```
size_t k;
for (k = 0; k < M; ++k)
{
        float sum = 0;
        size_t n;
        for (n = 0; n < N; ++n)
                sum += x[n] * cosf(Pi / N * (n + .5) * k);
        y[k] = sum;
}
```

In the code snippet a couple of parameters are given. This includes the following:

- N is the length of the original signal
- M is the amount of Cosine components that should be on the output
- x is the original signal
- y is the output buffer with the cosine components which should be on the size y[M]

Data visualization

To compare the two signals(original and the reconstruced from the DCT), a Python based data visualization implementation has been made. This will display the results in a generated .html-file with the help from the libary Plotly. The html-files can be seen in the Illustrations folder.

Experiment

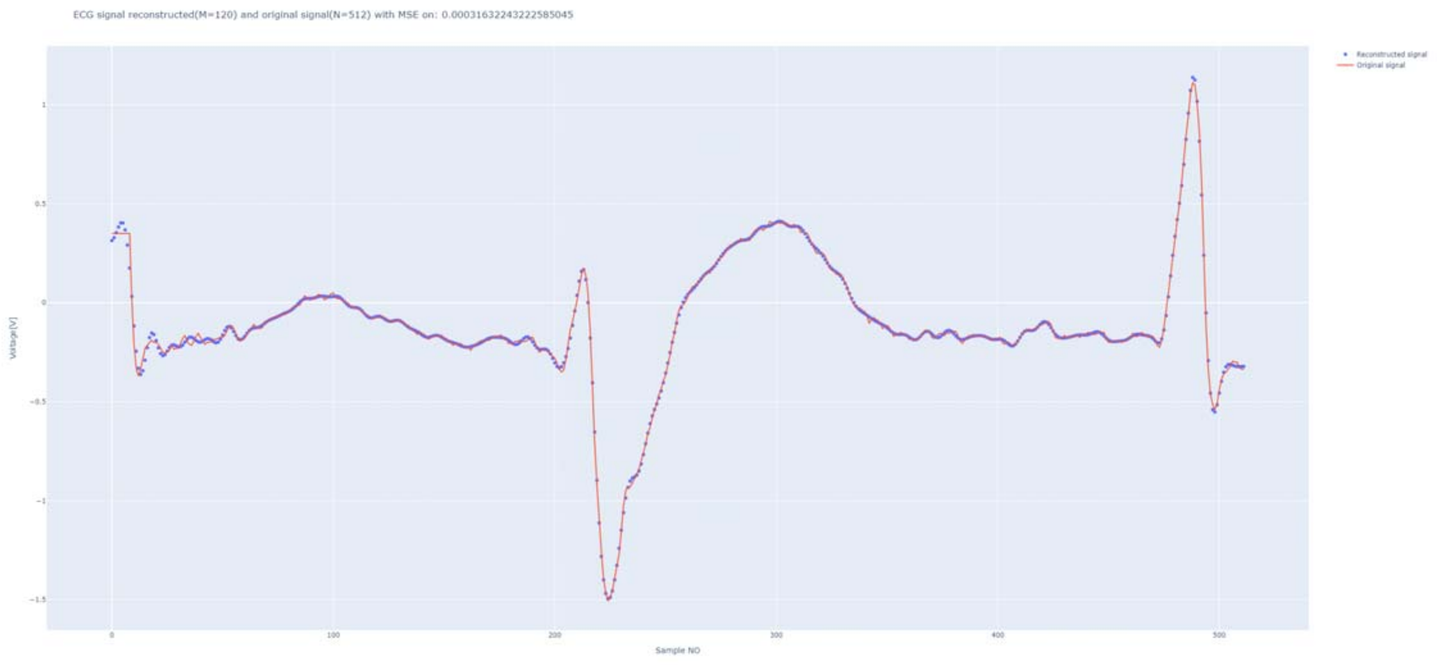
The 512 and 256 byte ECG signals are generated, and compressed with the function with different M-values as seen on the Figure above. When this is done the output and the original message will be written to two .txt files and transfered to a laptop. On the laptop the signals are visualized for comparison with the use of the Python based framework Plotly.

Results

The results section is divided into two sections with regards to the 512 and 256 tasks.

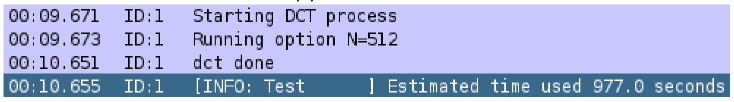
512 byte assignment

The 512 byte assignment is implemented with a M-value of 120. The result is visualized below.



MSE: 0.000207

The time taken for this to happen is found to be:



```
00:09.671  ID:1   Starting DCT process
00:09.673  ID:1   Running option N=512
00:10.651  ID:1   dct done
00:10.655  ID:1   [INFO: Test       ] Estimated time used 977.0 seconds
```

The time is found with 1 Cosine component and the overall time is therefore 120 times the found value (The image says seconds, however the found value is in milliseconds)

$120 * 977ms = 117240ms$

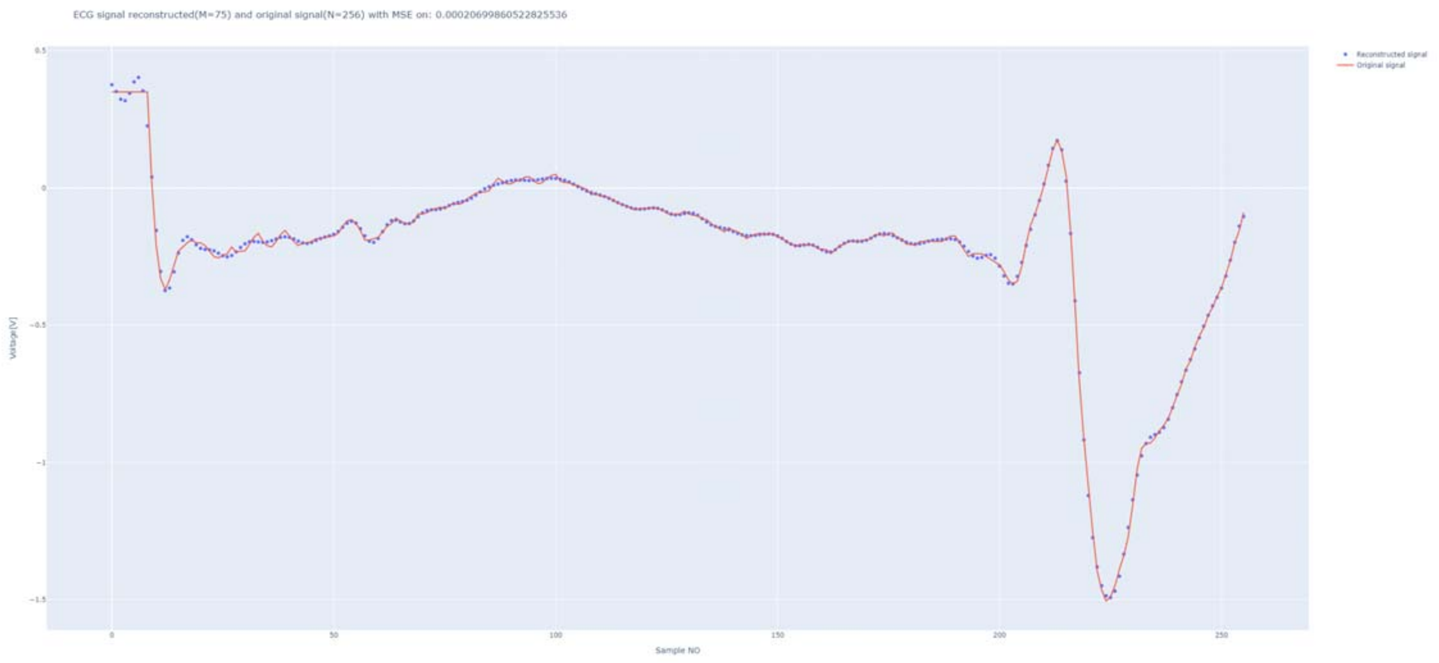
This will give an energy consumption of:

$117240ms * 1.8mA * 3V = 633mJ$

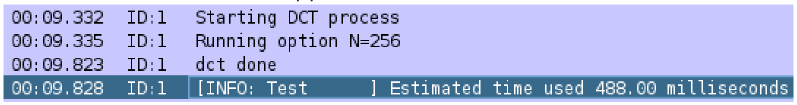
The mA and voltage is the average consumption taken from the TelosB mote's datasheet.

256 byte assignment

The 256 byte assignment is implemented with a M-value of 75. The result is visualized below.



MSE: 0.000316
The time taken for this to happen is found to be:



```
00:09.332  ID:1   Starting DCT process
00:09.335  ID:1   Running option N=256
00:09.823  ID:1   dct done
00:09.828  ID:1   [INFO: Test       ] Estimated time used 488.00 milliseconds
```

The time is found with 1 Cosine component and the overall time is therefore 75 times the found value.
$75 * 488ms = 36600ms$
This will give an energy consumption of:
$36600ms * 1.8mA * 3V = 198mJ$
The mA and voltage is the average consumption taken from the TelosB mote's datasheet.

Conclusion

The project is succesfull and the DCT is properly implemented on the TelosB mote. The reconstructed signal is pretty good and the MSE is pretty low. This is considered to be a good implementation and the tradeoff in accuracy for a 76% reduction in size for the 512 byte case and 71% reduction for the 256 byte signal. However the function seem to be time consuming, and thereby also energy consuming. A possible fix for this could be the implementation of fixed-point arithmetics.