***Abstract***

This paper presents a detailed implementation of Goertzel Algorithm in C language for embedded system. Goertzel Algorithm is a signal processing technique used in target frequency detection. This implementation is designed for dual-tone multi-frequency (DTMF) signal detection and audio generation. The Algorithm is based on the mathematics methodology: Fast Fourier transform (FFT) and specific modules: Infinite Impulse Response (IIR) filter and Finite Impulse Response (FIR) filter. The code is tested and implemented on an embedded platform, and the results demonstrate a proposed system provides an accurate and reliable method for processing DTMF signals, finding the specific frequencies required in the decoding file.

~~Zver:~~

~~This report presents Goertzel Algorithm for dual-tone multi-frequency (DTMF) signal detection and generation. The Goertzel signal processing technique is used in c language and implemented to detect the target frequency in the signal. The results show that the proposed system provides an accurate and reliable method for processing DTMF signals, finding the specific frequencies required in the data bin decoding file.~~

***The introduction***

(Background, application, motivation)

The Goertzel Algorithm is a signal processing technique to detect specific frequencies in a signal. It is widely used in various applications, including dual-tone multi-frequency (DTMF) signal detection, which is predominately applied for push-button digital telephone sets. Our implementation focuses on DTMF signal detection and audio generation, which is critical for telecommunications. In this paper, we present a detailed implementation is present in C language for embedded system and a methodology, which includes equations and images to illustrate the signal processing steps. Our implementation based on Goertzel Algorithm loop includes feedback and feedforward, which refers to Infinite Impulse Response (IIR) filter and Finite Impulse Response (FIR) filter respectively, to ensure reliable and accurate frequency detection while reducing the computational complexity. We test our codes with specific encoding data bin file and our results show that the decoding audio is as same as the given audio file, which has proven the feasibility. Our implementation can serve as a basis for further developments in DTMF signal processing and other applications need specific frequency detection, such as telephone banking system. In discussion, we also discuss the advantages of Goertzel Algorithm in embedded system and elements to optimize the system, includes scale factor, error prevent functions and the memory efficiency.

***GOERTZEL ALGORITHM***

(Previous research, difference)

Various project contains DTMF signalling. In DTMF signalling system, two frequency tones represent a specific digit (1, 2, 3, 4, 5, 6, 7, 8, 9), character (A, B, C or D) or symbol (\* or #). To detect tones, this can be achieved by using filters or Discrete Fourier Transform (DFT) [1], [2]. For DFT, bakbfk.

(这里我打算等YZhang写完之后再做修改，以防我会写一些和discussion重复的内容。TMA)

把公式打进overleaf 参考课件及下图

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

***Implementation***

1000? 分成3部分：Goertzel算法部分，识别部分，音频wave generation部分

The method used in this article to detect a specific frequency is the Goertzel Algorithm. The Goertzel Algorithm implemented has two parts: feedback loop and feedforward:

The sample input signal must be detected in first order to calculate the delay buffers delay\_1 and delay\_2 at DTMF frequencies based on the relevant Goertzel coefficients (1) and (2). This completes the feedback loop of the Goertzel Algorithm. where N is the number of repetitions, k is the constant's value, ftone is the tone's frequency, and fs is the sample The Goertzel coefficients for the respective frequencies of delay1 and prod1 are combined in the code to form prod1. The outcome of the fixed-point method is 14 degrees to the right. Equation （3） processes the input signal, calculates the value of delay, and updates the values of delay, delay1, and delay2 to prevent overflow. Each input sample is iterated via the feedback loop. The Algorithm runs a feed-forward loop to determine the final Goertzel value after processing the necessary number of samples (206 in this case).

K=N\*（ftone/fs） （1）

Coeff=2cos(2\*pi\*K/N) (2)

Qn = x(n) - Qn-2 + coeff\*Qn-1 ; 0≤n<N (3)

The feedforward portion is likewise finished, and the final Goertzel value is computed, to detect a given frequency using the Goertzel Algorithm. According to equation 4, prod1 and prod2 must equal the squares of the two delays to apply this strategy, which is the second-order IIR filter's final states after sample processing. The result of Coeff, delays1, and delays2 is prod3. To scale the outcome, the Goertzel value is right-shifted by 15 bits. then multiply the Goertzel value by 16 after left shifting it by 4 bits to boost its magnitude.

|Yk (N) |^2 = Q^2 (N) + Q^2 (N-1) - Coeff\*Q(N)\*Q(N-1) (4)

每部分记录内部变量

表格

描述已自动生成

画 flow chart\*2 :Goertzel算法部分，识别部分（参考下图）

图示

描述已自动生成

音频生成介绍下原理

The DTMF digital is identified during the audio creation process based on the Goertzel output. The two frequencies are then merged to form an array, and the associated DTMF number is identified based on the location, i.e. two distinct frequencies. After creating a two-tone multi-frequency digital audio signal using the DTMF two frequency combinations for (row = 0; row <4; row++) and for (col = 0; col<  4; col++), one for the row find the target value and assign it to tone1 and tone2, if a non-zero output is detected, it is stored in DigitPosition[count++] = i checking two frequencies, which is count=2. for (i = 0; i< 5000; i++) iterates to produce 5000 audio signal samples. The audio signal is produced using sine waves, and tone1 and tone2 are used to determine the total of the two sine waves. In order to generate audio signals based on digits, the generated audio signal is finally stored in the buffer array.

图示

描述已自动生成

***Results***

300?

***Conclusion***

Result conclusion可以合并成performance?

***Discussion***

In this paper, we have presented a detailed methodology and implementation of Goertzel Algorithm in C language for embedded system, focusing on DTMF signal detection and audio generation. By implementing high-order Goertzel Algorithm, our proposed system offered a reliable and accurate method for frequency detection with focusing on preventing overflow and minimizing memory usage, resulting in impressive performance at the source.

<subtitle: Overflow Prevention Techniques in Goertzel Algorithm>

Overflow is a common problem appears in the embedded system and this might cause data lost and memory efficiency low. To prevent overflow, scale factors are added to ensure the intermediate values (prod1[8], prod2[8], prod3[8], delay[8], delay\_1[8], delay\_2[8], Goertzel\_Value) in Algorithm fit within the variable range. This is achieved by determining the magnitude of intermediate values.

In this paper, the intermediate values, including Goertzel value, are calculated as a signed 16-bit integer, which have the range of -16384 to 16384. However, in the duration of iteration, the intermediate values could exceed the range between -16384 to 16384, by multiplication with Goertzel coefficients. To prevent this, Goertzel value is scaled down 15 bits before the feedforward loop. After implementing feedforward loop, Goertzel value is scaled up by 4 bits to increase sensitivity.

It should be noted that the choice of scale factor should be in an appropriate range. A larger scale factor might result in more precise calculation result while leading more rounding error. In terms of smaller scale factor, it might result in faster calculation but leads to loss of data.

In our implementation, for Goertzel value, a scale down factor of 15 bits and a scale up factor of 1 and 5, which refers in mian\_gtz\_OneFreq.c and mian\_gtz\_allfreq.c respectively, have been chosen since it provides a good balance between data precision and memory efficiency. These scale factors are derived by both codes test and calculation. In codes test of mian\_gtz\_OneFreq.c, test standard is to keep Goertzel value a stable value except zero. As a result, Goertzel value equals to zero could result in small input and unstable value could result in overflow. By setting value to one, the Goertzel value is keeping eight, which is a stable and non-zero value. In the calculation of main\_gtz\_allfreq.c,

<subtitle: Error Prevention Functions in Memory Allocation>

Memory allocation errors could cause unexpected program termination and damage to files and hardware components. In this implementation, to prevent the memory allocation errors, error checking functions for ‘malloc’ function was added in code optimization. Function ‘malloc’ is used to allocate the blocks of memory with size of 2810000 bytes and return a pointer to the first byte of the block. The error checking function first check if the ‘malloc’ function allocates the memory successfully, then return a pointer to null if memory allocation failed. In the case of null pointer, the program exits and displays the message to users about the memory allocation error without crashes, which might result in data loss. This error prevention function ensures stability of the program.

<subtitle: High Memory Efficiency Implementation for DTMF>

Memory efficiency is a critical element in embedded system, since it is highly relevant to limited memory resources while be processing large input and implementing iterative program. To achieve high memory efficiency, Goertzel Algorithm has been chosen for its low memory requirements and efficient calculations. Advantages of Goertzel Algorithm is to exploits the periodicity of the phase factor, thus reducing the computational complexity associated with DFT [1].

In the comparison of DFT and Goertzel Algorithm, C. S. Burrus shows that while the first-order Goertzel Algorithm is not particular efficiency, the two-at-a-time second-order Goertzel Algorithm is significantly faster than a direct DFT [4]. This article has a detailed discussion in mathematics theory and the advanced knowledge of System and Signals, focusing on the optimization of Goertzel Algorithm. However, as have not learnt more advanced knowledge of theory, we may not fully understand the advanced optimization progress in the article. This part could be further study for enhanced the Goertzel Algorithm theory, with detailed equation derivation and explanation.

（discussion里是我关于overflow，error preventing function和memory efficiency的讨论，请大家简单看一下，如果有什么需要加减的，或者是需要更改的，请和我说。TMA）

***Reference***

[1] Professor Dahnoun, ‘’C for Embedded System’’, Lecture 13, EENG20004 course materials, University of Bristol, 2023 (for powerpoint)

[3] Professor Dahnoun, ‘’Goertzel Algorithm Implementation in C for Embedded System’’, EENG20004 course materials, University of Bristol, 2023 (for codes)

[2] A. Vitali, ‘’The Goertzel algorithm to compute individual terms of the discrete Fourier transform (DFT),’’www.st.com, pp. 1-2, Dec.2017.

Accessed on: Mar. 20, 2023. [Online]. Available:

[The Goertzel algorithm to comspute individual terms of the discrete Fourier transform (DFT) (st.com)](https://www.st.com/resource/en/design_tip/dm00446805-the-goertzel-algorithm-to-compute-individual-terms-of-the-discrete-fourier-transform-dft-stmicroelectronics.pdf#:~:text=In%20the%20Goertzel%20algorithm%2C%20a%20set%20of%20N,factor%20W%28t%29%20is%20defined%20as%20exp%28-j%202pi%2FN%20t%29.)

[4] C. S. Burrus, ‘’Fast Fourier Transform (Burrus,’’ LibreTexts ENGINEERING, vol. 4.4, May. 2022.

Accessed on: Apr. 11, 2023. [Online]. Available:

[4.4: Goertzel's Algorithm or A Better DFT Algorithm - Engineering LibreTexts](https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Fast_Fourier_Transforms_(Burrus)/04%3A_The_DFT_as_Convolution_or_Filtering/4.04%3A_Goertzel's_Algorithm_or_A_Better_DFT_Algorithm)

***­­­***(why goertzel best)

（reference部分的格式不对，请不要参照这个文档里的格式填写overleaf。我手里有一份draft原文档，请参照那里的格式。TMA）

**Implementation部分的reference 改格式**

https://www.st.com/resource/en/design\_tip/dm00446805-the-Goertzel-algorithm-to-compute-individual-terms-of-the-discrete-fourier-transform-dft-stmicroelectronics.pdf

<https://www.ti.com/lit/an/spra066/spra066.pdf?ts=1680647535812&ref_url=https%253A%252F%252Fwww.google.com%252F>