Remote Signal Processor

Real-Time Signal Monitoring and Mathematical Analysis

Overview

- Client side signal processing
- Capture Microcontroller Signal
- voltage samples converted to digital
- transmitted as a 128 byte packet
- Validate with MAC filter in Wireshark
- Captured using Libpcap on a Linux Client
- Enabling Various Complex Processing

The Client in Action

- Brief glimpse without detailed explanation
- This is the main focus of the project: processing and analyzing incoming signal data
- Live data is received from the microcontroller

Signal Data Transmission via ADC

- Microcontroller captures analog voltage data through ADC (Analog-to-Digital Converter)
- Translates the analog data into 128 bytes of floating point digital data
- Transmits the 128-byte packet via Ethernet
- Board, Ethernet Interface

Ethernet Packet Visualization

- Real-time capture of the 128-byte packet in Wireshark
- Show how the packet is transmitted from the microcontroller
- Filter on MAC address
- Briefly explain the contents of the packet: floating point data representing voltage signals

Client-Side Processing of Voltage Data

- Real-time capture of the 128-byte packet in Wireshark
- Show how the packet is transmitted from the microcontroller
- Filter on MAC address
- Briefly explain the contents of the packet: floating point data representing voltage signals

Conversion and Initialization for FFT

- On the microcontroller: 128 bytes of floating point digital data are converted to a 128-byte packet.
- On the client:
- Packet is converted back to floating point.
- Data is used to populate the real field of std::complex, with 0.0f as placeholders for the imaginary part.
- This complex data is then fed into the FFT for frequency

Signal Normalization and Magnitude Extraction

- Normalization: Calculate the mean value of the real field of std::complex.
- Magnitude: Extract absolute values from the complex FFT results.
- The magnitudes are then fed into a priority queue for further processing.

Graph Representation from FFT Data

- Revisit FFT data and use it to initialize a graph.
- Graph is based on FFT magnitudes before conversion to absolute values.
- Edges added based on signal peaks and valleys.
- Graph is represented as an adjacency matrix for easy visualization of relationships between frequency components.

Matrix Operations Using Packet Data

- Demonstration of matrix operations using the packet data:
- Matrix Addition
- Matrix Subtraction
- Multithreaded Matrix Multiplication
- Packet data serves as the input for all matrix operations.

Advanced Analysis and Visualization

- FFT results are visualized using a 2D wave chart for frequency representation.
- FFT output is multiplied as polynomials for further mathematical analysis.

Bibliography

- Introduction to Algorithms (CLRS), Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.
- Algorithms in C++, Sedgewick, R. (1992). Algorithms in C++. Addison-Wesley.
- Unix Network Programming, Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004).
- Advanced Programming in the Unix Environment, Stevens, W. R., & Rago, S. A. (2013).
- The Linux Programming Interface, Kerrisk, M. (2010). No Starch Press.
- The Art of Electronics, Horowitz, P., & Hill, W. (2015). Cambridge University Press.
- The Design and Implementation of the FreeBSD Operating System
- McKusick, M. K., & Neville-Neil, G. V. (2014). The Design and Implementation of the FreeBSD