Research Proposal - Using Modern Micro-Controllers for Audio Frequency Shift Keying in Software

Joshua Jerred

August 20, 2023

Introduction

Historically, data modulation for use in radio networks has been implemented in hardware. For each different type of modulation, you needed different hardware, both on the transmitting and receiving end. Over the years there has been a shift towards agnostic hardware that is given it's power through Software Defined Radio (SDR). Micro-controllers (MCUs), just like any other hardware, have continually become more computationally capable while also becoming cheaper and consuming less power.

A paper published in 1998 by John Hansen, of State University of New York, talked about modulation of simple AX.25 packets on top of the AFSK modulation scheme, using PIC MCUs [1]. **something here about the PIC he used** Since this time, ST has released the STM32 series of MCUs. These are 32-bit ARM Cortex-M MCUs which are just as affordable as the PIC MCUs were. A standard performance STM32, such as the STM32F4 series, has up to 2 MB of flash storage, 384 KB of RAM, and runs at up to 180 MHz[2].

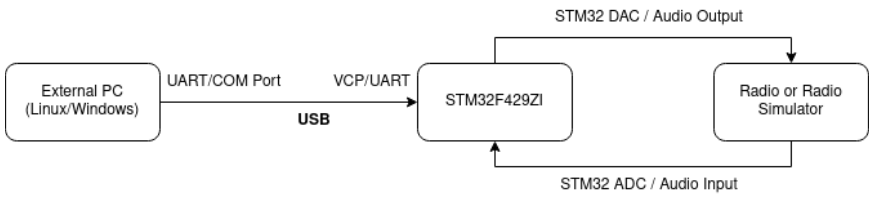
Audio Frequency Shift Keying (AFSK) is a method of modulating the frequency of a carrier signal to encode data. This modulation mode is commonly used in amateur radio applications, most commonly used alongside the AX.25 packet structure. Traditionally modulating data at a rate of 1200 bits-per-second; it's not especially fast, but it can provide reliable data communication over long distances. This modulation method, along with AX.25, is used during amateur radio events hosted on the International Space Station. Radio operators can send packets to the ISS using low-cost equipment and

Often times a specialized bit of hardware, called a Terminal Node Controller (TNC), is used for AX.25 and AFSK implementation. These devices can range in price, normally starting out over $150. It seems as if the STM32 should be more than powerful enough to modulate and demodulate AFSK signals in software without support from any external hardware, creating common hardware that can be used for many different modes of audio modulation.

Hardware Considerations

The only hardware requirement would be an STM32 that has at least one internal Digital-to-Analog converter (DAC) and at least one Analog-to-Digital converter (ADC). The easiest way to do this would be with an STM32 Nucleo board, a bit of hardware used for prototyping hardware design, similar to an Arduino. Availability is a concern, so a good choice would be the STM32F429ZI. ST sells a Nucleo prototyping board using this MCU. The F429 has the DACs and ADCs required for this task. Running at 180 MHz with 2 MB of flash, 256 KB of SRAM, it should be powerful enough to do so.

Using a Nucleo board will allow for a no hardware past a USB cable and 3.5mm audio jack. The F429 has a UART that can be accessed through the onboard ST-Link programming hardware, allowing a single USB connection to provide power and data communication with an external host. The external host/controller will be able to send and receive packets through a Virtual Comm Port on the UART interface.



Software Considerations

Seeing as the time to complete this research is fairly short, the software stack should be kept simple. The following is the list of libraries that would be used in development.

STM32 HAL The STM32 Hardware Abstraction Library is a MCU specific library for interacting with the hardware inside the MCU. This would be used for core clock configuration, I/O pin configuration, ADC and DAC interfacing, and any other code that requires use of memory mapped hardware. Although it's not strictly required to use an STM32, it drastically reduces development time.

STM32 USB/VCP The STM32 USB and Virtual Comm Port library is used for interfacing USB hardware with the MCU. In order to be able to communicate with the MCU externally, the STM32 USB/Virtual Comm Port library would be used so the host computer, running either Linux or Windows, would see the device as a generic serial device. This would be a simple method of communication that is still flexible. One thing to pay attention to using this method is the fact that there may be hardware interrupts driving the communication which could be a problem with real time audio generation and sampling.

ETL ETL (Embedded Template Library) is a library that provides common functionality from the C++ standard library for embedded platforms. Although the standard library can be used, the ETL is mindful of memory management practices on embedded devices. The ETL avoids heap usage whenever possible as heap fragmentation is a concern when you are already working with so little RAM. One feature of the ETL is a hierarchical finite state machine structure (HFSM) that may be helpful in this use case.

GCC Arm None ABI GCC Arm None ABI is a bare metal GCC compiler that supports ARM-Cortex platforms. Along with the compiler, there are basic templates that include bare minimum assembly entry point code for configuring the interrupt vector table and setting up an environment for C/C++ code prior to the call to main. Additionally there are template GCC linker files for each STM32 MCU that set up the address space and required sections in order to compile and run C/C++ code.

Sources

References

[1] John Hansen A. Pic-et radio: How to send ax.25 ui frames using inexpensive pic microprocessors. *ARRL and TAPR Digital Communications Conference*, 27th:29–37, 1998.

[2] STMicroelectronics. *DS9484 - STM32F429xx Data sheet.*

https://ieeexplore-ieee-org.libproxy.utahtech.edu/document/8475994
https://ieeexplore-ieee-org.libproxy.utahtech.edu/document/4502213
https://www.ti.com/lit/an/slaa618/slaa618.pdf?ts=1691858157091
https://www.tapr.org/pdf/DCC1998-PICet-W2FS.pdf
https://ieeexplore-ieee-org.libproxy.utahtech.edu/document/8475994