گزارش‌کار پروژه مخابرات دیجیتال. بخش ۲

««««یک افسانه دیجیتال»»»»

قسمت دوم: سلاح‌های جدید فولادی و یک جنگ پر از خون

طرح ریزی برای قسمت ۲

یک شروع قدرتمند

حال که دانش کافی برای شروع به دست آورده‌ام می‌توانم پروژه را از ابتدا شروع کنم.

هرچند هنوز در خیلی از بخش‌ها مشکل دارم. خصوصا در بخش‌های کدینگ. شروع داده‌ها را شاید با یک الگوریتم شناخته‌شده مشخص کنم. برای تشخیص یا تصحیح خطا ممکن است از یک روش خاص که به آن کدینگ لگاریتمی می‌گویم استفاده کنم. (واقعا اسم این روش را نمی‌دانم)

هم‌چنین برای مساله همسان نبودن ساعت‌های فرستنده و گیرنده چاره‌ای نیاندیشیده‌ام و اصلا کدینگ ندارم.

در بخش قبلی فهمیدم که برای رمزنگاری، کدینگ کانال و کدینگ منبع، شناخت اندیس اعداد مهم است.

پس در نهایت باید یک کدینگ برای پیداکردن ابتدای هر بلوک (پیدا کردن اندیس داده دریافت شده بین قطاری از داده‌ها) راه‌کاری بیاندیشم.

آزمایش‌های واقعی‌تر

سیستم مخابراتی من با این که شروع به کار کرده‌است و می‌تواند خروجی بدهد، اما باید بتواند در شرایط واقعی نیز کار کند. پس باید آزمایش‌هایی متناسب با شرایط واقعی را طراحی کنم.

اولین آزمایش فیلتر کردن سیگنال در کانال است. پهنای باند کانال ما نامحدود نیست و همین باعث می‌شود دقت سیستم ما کمتر بشود.

آزمایش دیگر مساله دریافت داده‌ها از وسط توسط گیرنده است. کدینگ خاصی برای این کار لازم است.

پیاده‌سازی الگوریتم‌ها در حالت کلی

برای یک مساله الگوریتم‌های مختلفی وجود دارند.

معمولا ما الگوریتمی را طراحی می‌کنیم که دقیقا همان مساله را حل کند. اما گاهی می‌توان الگوریتم‌ها را چنان طراحی کرد که در حالت های کلی تر مساله نیز جواب دهد.

اما این چه فایده‌ای دارد؟

علاوه بر این که امکان گسترش‌پذیری را به ما می‌دهد، گاهی الگوریتم‌های کلی ساده‌تر نیز هستند!

این مقایسه مانند مقایسه استقرای عادی با استقرای قوی است. باید کار بیشتری انجام دهیم اما ممکن است حتی کار ما را ساده‌تر نیز بکند.

خوبی دیگر این در طراحی ماژولار سیستم است. یعنی هر بلوک کار خودش را به خوبی انجام دهد و مستقل از سایر بلوک‌ها باشد. وقتی الگوریتم‌ها در حالت کلی طراحی شده باشند، بلوک‌ها قابلیت جابجایی بین برنامه‌ها را پیدا می‌کنند.

من این‌بار سعی کردم مدولاسیون را به M تایی تغییر دهم و بلوک‌ها را در حالت کلی طراحی کنم.

سیستم نهایی هم‌چنان برای BPSK به همان کیفیت کار می‌کند. برخی الگوریتم‌ها ساده‌تر شده‌اند و الان سیستم در مدولاسیون‌های بالاتر نیز می‌تواند کار کند.

کارهایی که کردم

کدم را بیشتر شبیه به یک برنامه کردم

کدی که نوشته بودم در ابتدا مهندسی شده نبود. همه کد در یک فایل قرار داشت و اصلا فانکشن تعریف نکرده بودم. البته برنامه را از همان ابتدا طوری نوشتم که تا حدی گسترش‌پذیر باشد.

پس در یک بازبینی کل برنامه را تصحیح کردم. فانکشن‌ها را تعریف کردم و حجم کد را کاهش دادم. تعریف توابع در متلب یک چالش بود!

مدولاسیون M تایی PSK

با توجه به این که در پیاده‌سازی مدولاسیون 2PSK موفق بودم، سعی کردم حالت M تایی آن را نیز پیاده سازی کنم.

در مدوله کردن داده‌ها از M سیگنال با فازهای مختلف استفاده کردم تا نتیجه مطلوب حاصل شود.

هنگام دیمدوله کردن، هر قطعه سیگنال را در سیگنال‌های سینوسی و کسینوسی ضرب کردم و ۲ عدد حاصل را به تابع PSKdemod متلب دادم.

خروجی قابل قبول بود.

هم‌چنین همانطور که انتظار می‌رفت، با افزایش مرتبه مدولاسیون، کیفیت ارسال کاهش پیدا می‌کرد.

در کد قبلی من از دستور پیش ساخته PSKdemod متلب استفاده می‌کردم، در ادامه اما سعی کردم از تابع PSKdemod متلب استفاده نکنم. پس تابع خودم را نوشتم.

مستقل شدنِ بیشتر از متلب با پیاده‌سازی مجدد آشکارساز

وقتی پالس‌ها با مولفه‌های سینوسی و کسینوسی که در آشکارساز ضرب می‌کنیم، با خود سیگنال اختلاف فاز داشته باشند، فیزورها خواهند چرخید. این چرخش می‌تواند در حدی باشد که همه داده‌ها اشتباه دریافت شود. راه حل باز استفاده از کدینگ دیفرانسیلی است.

من این بار به جای استفاده از توابع خود متلب، سعی کردم فیزورها را به تابع خودم بدهم که آشکارساز دیفرانسیلی داده است.

در آشکارساز من، فاز اهمیتی ندارد و تنها اختلاف فاز مهم است.

هم‌چنین وقتی از این روش استفاده می‌کنم نیازی به دیکدینگ دیفرانسیلی نیز نیست.

در نهایت من توانستم الگوریتم‌هایم را در حالت کلی تر بازطراحی و پیاده‌سازی کنم.

هم‌چنین از تابع PSKdemod متلب نیز استقلال یافتم.

یک مزیت تابع من نسبت به PSKdemod متلب این است که وقتی اختلاف فاز وجود داشته باشد، اتفاقا تابع من بهتر کار می‌کند. بهتر از دیمدولاسیون عادی و سپس دیکدینگ دیفرانسیلی.

کدینگ دیفرانسیلی

قبلا در کدینگ من، یکسان بودن یا یکسان نبودن ۲ رقم متوالی مهم بود. اما برای حالت M تایی اختلاف ۲ رقم مجاور را استفاده کردم.

در بخش دیکدینگ نیز همینطور. البته با دیمدولاسیون جدید می‌شود دیکدینگ را حذف کرد.

جایگزین بیت‌های زوجیت در سیگنال‌های M تایی.

در حالت باینری از بیت‌های زوجیت برای تشخیص خطا استفاده می‌کردم. در حالت M تایی باید روش معادلی وجود داشته باشد.

کاری که من انجام دادم این بود که به جای xor کردن چند عدد، آن‌ها را باهم جمع می‌کردم و باقیمانده حاصل را بر M می‌گرفتم. این عمل مشابه xor در مبنای ۲ عمل می‌کند.

حل مشکل پیداکردن ابتدای داده با اضافه کردن flag (پرچم راهنما) به داده

فرض کنید گیرنده دقیقا از ابتدای داده‌ها شروع به دریافت نکرده‌است. مثلا قدری دیرتر شروع به دریافت داده کرده و بخشی از داده وجود ندارد.

یا فرض کنید یک بیت در میانه راه گم شده و گیرنده به جای دریافت ۱۰۰۰ بیت، ۹۹۹ بیت دریافت کرده‌است.

مشکلی که به وجود می‌آید این است که دیگر نمی‌توانیم یک کدینگ درست داشته باشیم. چه کدینگ کانال و چه کدینگ سورس.

راه‌کار اضافه کردن یک‌سری پرچم راهنما به ابتدای هر بلوک داده است.

اسم کدینگ استفاده‌شده را نمی‌دانم اما گویا در مبحث شبکه کاربرد دارد.

پرچم راهنمای استفاده شده به شکل دنباله‌ای از ۱ هاست که با یک صفر از هر دو طرف احاطه شده است.

مثل این : 0 1 1 1 … 1 1 0

مشکلی که پیش می‌آید این است که ممکن است در خود داده چنین ترکیبی وجود داشته باشد. برای حل این مشکل هرکجا که شباهت وجود داشت، یک عدد به داده‌ها اضافه می‌کنیم تا شباهت از بین برود. این عمل البته باید برگشت پذیر باشد.

پرچم راهنما: بخش گیرنده

گیرنده در هنگام خواندن داده‌ها به دنبال راهنما می‌گردد. وقتی آن‌را پیدا کرد از آنجا شروع به دیکد کردن داده‌ها می‌کند. ابتدا خود راهنما را از داده حذف می‌کند و سپس بیت‌هایی را که برای از بین بردن شباهت بین داده اصلی و پرچم اضافه شده بودند را حذف میکند.

نتیجه یک سری داده‌است که ابتدای آن مشخص است.

حال این داده را می‌توان کدگشایی کرد.

نوشتن کد این بخش برای من خیلی سخت بود.

فیلتر کردن سیگنال در کانال

سیگنالی که در کانال ارسال می‌شود باید فیلتر شود. درواقع پهنای باند ما محدود است.

یک فیلتر باید در ابتدای کانال اضافه شود. این فیلتر محدودیت ارسال سیگنال ما را نمایش می‌دهد. هم‌چنین برای جلوگیری از نشت فرکانس، فرستنده ممکن است خودخواسته سیگنال را فیلتر کند.

یک فیلتر دیگر در انتهای کانال باید اضافه شود. این معادل فیلتر گیرنده است. پهنای باند دریافت سیگنال در گیرنده محدود است. هم‌چنین برای جلوگیری از تداخل با سیگنال‌های دیگر یا کاهش توان نویز، گیرنده ممکن است خودخواسته سیگنال را فیلتر کند.

هم‌چنین این ۲ فیلتر می‌توانند مشابه هم باشند.

من از دستور `filter` متلب برای فیلتر کردن استفاده کردم و طراحی فیلتر را به کمک دستور `butter` متلب انجام دادم.

با اضافه کردن فیلترها کیفیت ارسال من قدری کاهش پیدا کرد.

افزایش پرفورمنس برنامه با استفاده‌کردن از توابع ریاضی متلب

از این پس در بخش‌های پردازشی و ریاضی، به جای نوشتن توابع خودم سعی می‌کنم از توابع آماده متلب استفاده کنم. آن‌ها سازگاری بیشتری با مفسر دارند و سریع‌تر اجرا می‌شوند.

البته این کار را تنها برای بخش های ریاضی انجام می‌دهم. مثل میانگین گرفتن از هر ردیف یک ماتریس.

برای بخش‌های مخابراتی اما رویه من کاملا برعکس است و تلاش می‌کنم از توابع خود متلب کمتر استفاده کنم. چون کنترل کامل را به سرعت ترجیح می‌دهم.

نتیجه‌گیری

- متلب اصلا زیبا نیست. جامعه کاربران اینترنتی خوبی هم ندارد. ولی قابل تحمل است!
- از بدی‌های نرم‌افزار گنو اوکتاو هم بنویسم! Gnu Octave

پرفورمنس متلب و اوکتاو را مقایسه کردم. با این که عوامل زیادی در این موثر هستند اما در یک قطعه کد مشابه، متلب حدود ۱۰ بار از اوکتاو سریع تر بود. یعنی تست من برای متلب حدود ۰.۲ ثانیه و برای اوکتاو حدود ۲ ثانیه طول کشید.

هم‌چنین بسیاری از توابع متلب هنوز در اوکتاو وجود ندارند. برای این که پروژه من هم با متلب سازگار باشد و هم با وجود محدودیت‌های اوکتاو هم‌چنان کار کند، من گاهی مجبور می‌شوم سراغ توابع جایگزین بروم.

هرچند هنوز ترجیح می‌دهم توسعه را در اوکتاو پیش ببرم و برای تست‌های نهایی هر قسمت از متلب استفاده کنم.

- ریاضیات! مقداری ریاضیات یاد گرفتم!

  عملیات شبیه XOR را برای حالت کلی و مبناهای غیر از ۲ طراحی کردم.

  در تلاش برای تعمیم مدولاسیون PSK، ابداع‌های جدیدی داشتم.

  کدینگ‌های خودم را طراحی کردم.

- تعمیم الگوریتم‌ها برای حالت کلی باعث می‌شود سیستم قوی‌تر و قابل اتکا تر به نظر برسد.

- خیلی وقت پیش تلاش کرده بودم یک پروژه مخابراتی انجام دهم و داده‌هایی را در کانالی مخصوص مدوله و ارسال کنم. اما چندان موفق نبودم و سیستمی که ساخته بودم بسیار نویز پذیر بود.

  بعد از این پروژه حس می‌کنم میتوانم پروژه قبلی خودم را پیش ببرم و کیفیت سیستم را به میزان قابل توجهی افزایش دهم.

  پس بعد از این سراغ پروژه قدیمی خودم خواهم رفت!

- این پروژه مخابرات واقعا لذت بخش و satisfying است.

  و درنهایت من از پیشرفت پروژه در این بازه زمانی راضی بودم.