معرفی پروژه های نرم افزاری مهم منبع-باز SDR (GNU Radio/OSSIE/DttSP)

گزارش پروژه ی درس رادیو نرم افزاری (spring 2009)

سید مهدی سجادی
۸۷۰۳۴۴۴
sajjadi@rocketmail.com

چکیده – یکی از مهم ترین قسمت های یک سیستم رادیوی مبتنی بر نرم افزار , بخش نرم افزاری آن است. در میان پروژه های نرم افزاری *SDR* , آن ها یی که منبع باز هستند , به دلیل قابلیت های مهمی که در اختیار کاربر عام قرار می دهند از محبوبیت خاصی بر خوردار اند. دو پروژه ی نرم افزاری مهم *SDR* که منبع باز بوده و برای استفاده در دسترس قرار دارند *GNU Radio* و *OSSIE* می باشند. *SDR* همچنین جای خود را در بین کاربران دنیای رادیوی آماتور باز نموده است بنحوی که سخت افزار ها و نرم افزارهای زیادی بدین منظور ارائه گردیده است. هسته ی بسیاری از این نرم افزار ها مبتنی بر کد منبع-باز *DttSP* است. در این گزارش ابتدا *GNU Radio* و پس از آن بطور مختصر *OSSIE* و *DttSP* معرفی می شوند.

۱- مقدمه

هدف سیستم های رادیو ی نرم افزاری (software radio) این است که تا حد امکان مرز دنیای دیجیتال و آنالوگ (مبدل آنالوگ به دیجیتال) به آنتن نزدیک تر شود , بنحوی که عمده ی کار پردازش سیگنال در دنیای دیجیتال انجام گردد. با این کار می توان مسائل سخت افزاری را به مسائل نرم افزاری تبدیل نمود. اما اگر فردی علاقه مند به فعالیت در این زمینه باشد با مشکلاتی روبرو است.

بعنوان نمونه یک فرد فعال در زمینه ی رادیو نرم افزاری بایستی اطلاعاتی نسبتا جامع درباره ی مجموعه ی وسیعی از مهارت ها داشته باشد. برنامه نویسی کامپیوتری , آشنایی با اصول مخابرات بدون سیم , پردازش سیگنال های دیجیتال و ... از جمله توانایی های مورد نیاز برای کار رادیوی نرم افزاری می باشند.

از طرفی دیگر , بسیاری از پروژه های رادیو نرم افزاری تعریف شده , ماهیتا غیر منبع-باز می باشند و دسترسی به کد اصلی مربوطه و بررسی آن و بهره برداری عمومی از آن ها امکان پذیر نمی باشد.

هدف اولیه ی پروژه GNU Radio فراهم آوردن زمینه هایی جهت تسهیل ورود به عرصه ی رادیو نرم افزاری است.

۲- معرفی GNU Radio

GNU Radio یک بسته ی نرم افزاری رایگان جهت آموزش, ساخت و بکار گیری سیستم های رادیو نرم افزاری است. ساختار کلی GNU Radio دو مفهوم اصلی را شامل می شود : ۱-کتابخانه ای از بلوک های پردازش سیگنال و۲-ابزاری جهت برقراری ارتباط مابین این بلوک ها. یک برنامه نویس بمنظور ایجاد یک سیستم رادیویی , گراف معادلی را ترسیم می نماید که در آن , گره ها بلوک های پردازش سیگنال و یال ها بیانگر جریان داده ی مابین آن ها می باشند.

تا قبل از GNU Radio , بسیاری از پروژه های مربوط به رادیو نرم افزاری , اختصاصی و دارای مالکیتی مشخص بودند. از اینرو Eric Blossom تصمیم گرفت تا فلسفه ی منبع باز را وارد دنیای رادیو نرم افزاری کند. این ایده مورد پذیرش Ricahard Stallman (بنیان گذار GNU) قرار گرفت و GNU Radio رسما بعنوان یک پروژه ی GNU در سال ۱۹۹۸ توسط Eric Blossom فعالیت خود را آغاز نمود. همانطور که اشاره شد فعالیت در زمینه ی رادیو نرم افزاری , مستلزم داشتن اطلاعاتی نسبتا جامع درباره ی موضوعات گوناگون است. از دیگر اهداف GNU Radio این است که ورود افراد با حوزه ی تخصصی مشخص به عرصه ی رادیو

نرم افزاری امکان پذیر گردد. مثلا یک مهندس مخابرات بدون اینکه وارد مباحث برنامه نویسی شود بتواند به طراحی سیستم های رادیو نرم افزاری بپردازد.

GNU Radio به عنوان شاخه ای از کد Pspectra شروع شد. (این کد در نتیجه ی پروژه ی SpectrumWare متعلق به دانشگاه MIT بوجود آمد و یکی از اولین ابزار های نرم افزاری ارائه شده جهت توسعه ی سیستم های رادیو نرم افزاری می باشد.) در سال ۲۰۰۴ بازنویسی کامل کد GNU Radio تکمیل شد بنحوی که امروز GNU Radio هیچ بخشی از کدهای اصلی Pspectra را شامل نمی شود. از طرف دیگر کد اصلی Pspectra بعنوان اساس پروژه ی SDR تجاری Vanu استفاده شده است. GNU Radio عمدتا مبتنی بر سیستم عامل linux است ولی امکان استفاده از آن در سیستم های عامل دیگر نیز وجود دارد.

۲ – ۱ – کاربرد های عملی GNU Radio

افراد مختلفی از GNU Radio بهره می برند که گروه های تحقیقاتی دانشگاهی , صنعتی , دولتی , DARPA و ... , همچنین هکرها و کاربران رادیوی آماتور را شامل می شوند.

کاربرد های GNU Radio را می توان بصورت زیر دسته بندی کرد:

گیرنده/فرستنده ها , تحقیقات درباره ی شبکه های بدون سیم , شبکه های ad-hoc , رادیو هوشمند (cognitive radio) , رادار پسیو , تعیین مکان جغرافیایی , سیستم های رادیویی آماتور , SIGINT , گیرنده ی همزمان چند کانال , تحلیل طیف فرکانسی و نجوم رادیویی .

نمونه هایی از پروژه های اجرا شده یا درحال اجرا به شرح زیر است:

- یکی از مشکل ترین کار های انجام شده پیاده سازی سیستم دریافت و ارسال تلویزیون دیجیتال زمینی آمریکای شمالی است(HDTV) که از استاندارد ATSC استفاده می کند. البته این سیستم هم اکنون قابلیت عملکرد real-time را نداشته و نیاز به کار بیشتر دارد.
- مرکز تحقیقات BBN کد 802.11 (استاندارد Wifi) را با سرمایه گذاری DARPA نوشته است. البته با توجه به محدودیت سرعت USB 2.0 و half-duplex بودن آن , از لحاظ عملی این پروژه با مشکلاتی روبرو است. (خود GNU Radio ماهیتا full-duplex است.)
- آزمایشگاه مهندسی سیستم UW Quantum , کد سیستم تصویر برداری ملکولی MRFM (ریز بینی بوسیله ی نیروی رزونانس مغناطیسی) را با سرمایه گذاری دفتر تحقیقات ارتش نوشته است.
- یکی از پروژه های در حال اجرا , سیستم رادار پسیو است. در این سیستم ها از سیگنال های الکترومغناطیسی رادیویی و تلویزیونی بعنوان منبع سیگنال استفاده می شود.

تلاش هایی جهت دریافت سیگنال تلویزیون آنالوگ انجام شده که در مرحله ی DeInterlace کردن فریم ها با مشکل مواجه شده و نیاز به کار بیشتر دارد. همچنین پروژه هایی نظیر GPS نرم افزاری ,شبکه های سنسور توزیع شده , خواننده/آشکار ساز RFID , استاندارد GSM , Open BTS , zigbee و Bluetooth و... نیز با استقبال علاقه مندان مواجه شده و فعالیت های زیادی جهت تکامل آن ها انجام شده است. GNU Radio برای ارزیابی امنیت سیستم های مبتنی بر رادیو نیز استفاده می شود.

فهرستی از پروژه های انجام شده و یا در حال اجرا در (CGRAN) [۲] موجود می باشد.

در آخر بایستی اشاره نمود که هدف نهاییGNU Radio پیاده سازی تمام سیستم های رادیویی موجود و طراحی و ساخت رادیو های جدید است.

۲ – ۲ – ترکیب C++ و Python

زبان های برنامه نویسی استاندارد انتخاب شده برای GNU Radio زبان های Python و C++ می باشند که هر کدام نقش متفاوتی را ایفا می کنند. تمامی بلوک های پردازش سیگنال و عملیاتی که در آن ها سرعت اجرا اهمیت دارد توسط C++ نوشته می شوند. Python وظیفه ی ایجاد شبکه ها , گراف های جریان داده , مدیریت و نحوه ی ارتباط بلوک ها را بر عهده دارد. همچنین از Python

بمنظور ایجاد رابط گرافیکی کاربر (GUI) نیز استفاده می شود.

کتابخانه ی بلوک های پردازش سیگنال GNU Radio جامع بوده و بیش از ۲۰۰ بلوک را شامل می شود و همواره با عرضه ی نگارش های جدیدتر , بلوک های جدیدی به این کتابخانه اضافه می شود. با این حال GNU Radio بنحوی طراحی شده که می توان در صورت نیاز بلوک های پردازش سیگنال جدیدی را به آن اضافه نمود و کتابخانه ی آن را گسترش داد.

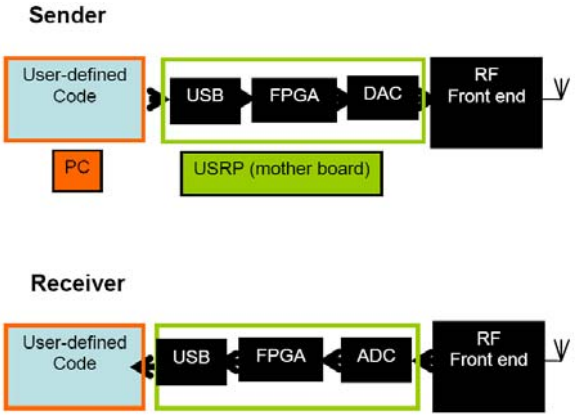
Python از لحاظ یادگیری زبانی نسبتا ساده بوده و قابلیت های زیادی را در اختیار برنامه نویس قرار می دهد. Python یک زبان برنامه نویسی اسکریپتی بوده و نیاز به کامپایل کردن ندارد , با این حال بسیاری از افراد به دلیل آشنایی قبلی با زبان C++ ترجیح می دهند که کل کد نویسی را در این زبان انجام داده و مجبور به فراگیری زبان جدیدی نباشند. از طرفی استفاده از Python در موارد مشخصی مطلوب نیست(مثلا برای سیستم های embedded) . از اینرو همواره این تمایل وجود داشته که بتوان کد نویسی را مستقل از Python انجام داد. در نگارش ۳.۲ از GNU Radio این امکان بوجود آمده که بتوان کل برنامه را فقط به زبان C نوشت.

ارتباط مابین زبان Python و بلوک های پردازش سیگنال که به زبان C++ نوشته شده اند توسط رابط SWIG برقرار می گردد.(Simplified Wrapper and Interface Generator) بلوک های پردازش سیگنال ماهیتا کلاس های تعریف شده در زبان C++ هستند. با استفاده از رابط SWIG می توان این کلاس ها را بعنوان توابعی در Python فراخوانی نمود. بنابراین از دید Python , این بلوک ها کلاس هایی هستند که در ماجول های اصلی پایتون تعریف شده اند.

USRP – ۳ – ۲

GNU Radio تقریبا یک سیستم مستقل از سخت افزار می باشد. به این معنی که یک کاربر ملزم به استفاده از سخت افزار معینی نیست. توسعه دهندگان GNU Radio در ابتدا از یک تیونر مودم تلویزیون کابلی به عنوان RF FE و از یک برد تجاری نسبتا گران (۱۲۰۰ دلار) با سرعت ۲۰ msample/sec بعنوان مبدل آنالوگ به دیجیتال استفاده می کردند. بمنظور تسهیل و ارزان تر شدن دسترسی به سخت افزار مورد نیاز , سخت افزار USRP (universal software radio peripheral) توسط Matt Ettus طراحی گردید. Matt Ettus یکی از توسعه دهندگان GNU Radio و نویسنده ی تعدادی از بلوک های مهم کتابخانه ی بلوک پردازش سیگنال می باشد.

USRP شامل ۴ عدد مبدل آنالوگ به دیجیتال ۱۲ بیتی با سرعت ۶۴ میلیون نمونه در ثانیه و ۴ عدد مبدل دیجیتال به آنالوگ ۱۴ بیتی با سرعت ۱۲۸میلیون نمونه بر ثانیه می باشد. برای عملیات پیش پردازش از یک FPGA مدل EP1C12 از شرکت Altera استفاده نموده و برای ارتباط با کامپیوتر از رابط USB 2.0 استفاده می کند. این سخت افزار قابلیت نصب ۴ برد جانبی (۲ عدد گیرنده و۲ عدد فرستنده) را داراست. با ترکیب برد های جانبی مختلف می توان در محدوده فرکانسی DC تا ۲.۹ گیگاهرتز به ارسال و دریافت سیگنال رادیویی پرداخت.



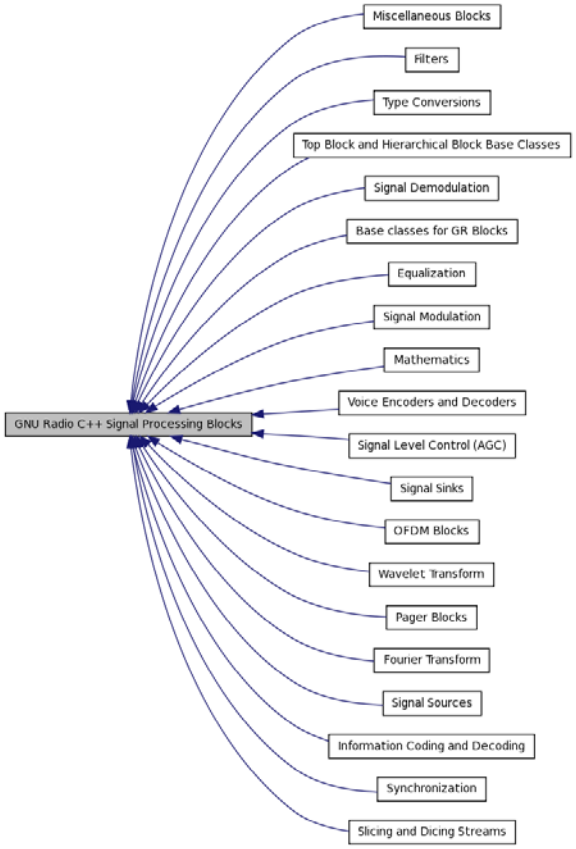
شکل۱: ساختار USRP

همانطور که اشاره شد کاربر ملزم به استفاده از USRP نمی باشد و می تواند از سخت افزار های مورد علاقه ی خویش استفاده نماید. از طرف دیگر سخت افزار USRP را نیز می توان همراه با نرم افزار های دیگری به غیر ازGNU Radio استفاده نمود. به عنوان نمونه پروژه ای به نام Simulink-USRP در دانشگاه کارلسروهه انجام گرفته که با استفاده از آن می توان بسیاری از ویژگی های USRP را در Simulink بکار گرفت. در واقع یک BlockSet به نام USRP در Simulink تعریف می شود که امکان رد و بدل کردن داده با USRP را بصورت Real-time بوجود می آورد. در سپتامبر سال ۲۰۰۸ کمپانی Ettus نسخه ی دوم USRP را عرضه

کرد که در آن از یک FPGA قوی تر (Xilinx Spartan 3-2000) , مبدل های آنالوگ به دیجیتال و دیجیتال به آنالوگ سریع تر و با رزولوشن بالاتر و رابط شبکه ی Gigabit Ethernet (علاوه بر USB 2.0) استفاده شده است.

۲ – ۴ – کتابخانه ی بلوک های پردازش سیگنال

یکی از ویژگی های ساختاری GNU Radio , ارائه ی یک کتابخانه ی وسیع از بلوک های از پیش تعریف شده و آزمایش شده است. این بلوک ها به زبان C++ نوشته شده و وظیفه ی پردازش سیگنال را انجام می دهند.



شکل۲: ساختار بلوک های پردازش سیگنال

عمده ی بلوک های موجود در این کتابخانه , توابع پردازش سیگنال پر استفاده می باشند. از اینرو برای انجام بسیاری از پروژه های رادیویی , نیازی به ایجاد بلوک های دیگر نمی باشد. در این موارد کد نویسی به زبان Python کفایت نموده و نیازی به کدنویسی به زبان C++ نیست.

هر بلوک دارای ویژگی های مشخصی مانند تعداد پورت های ورودی و خروجی و نوع داده های جاری در پورت های آن است. در مورد بعضی از بلوک ها (مانند فیلتر fir) نوع داده های میانی نیز مشخص است.

یکی از مشکلات بزرگ GNU Radio عدم وجود راهنمایی کامل و جامع است که بلوک ها را کاملا توصیف نموده و نحوه ی استفاده و آرگومان ها را بخوبی تفسیر کند. در GNU Radio بمنظور تولید مستندات راهنما از doxygen استفاده می شود. Doxygen یک سیستم تولید راهنمای خودکار است که با توجه به کد اصلی و توضیحات (comment) موجود در آن به تولید راهنما می پردازد. با اینکه متن حاصله اطلاعات خوبی در اختیار کاربر قرار می دهد ولی در بعضی از موارد کاربر مجبور می شود که به کد اصلی رجوع کند.

حال بطور خلاصه به معرفی بعضی از مهم ترین ماجول های کتابخانه ی پردازش سیگنال می پردازیم:

- منابع تولید سیگنال: این ماجول شامل بلوک هایی جهت تولید انواع سیگنال ها با شکل موج مشخص (سینوسی , مربعی , مثلثی و...) , سیگنال های شبه تصادفی و منابع نویز می باشد. همچنین این ماجول شامل کلاس هایی به منظور خواندن اطلاعات از USRP , بردار و فایل نیز می باشد.
- فیلتر ها : این ماجول شامل فیلتر های گوناگون از جمله فیلتر های FIR معمولی و وفقی , فیلتر های IIR , تبدیل هیلبرت , فیلتر moving average , ترکیب فیلتر FIR با down conversion (کاهش فرکانس)و decimation (کاهش نمونه ها) , فیلتر interpolating (اضافه کننده ی نمونه ها) و... می باشد.
- عملیات ریاضی: این ماجول شامل عملیات ریاضی متداول از قبیل چهار عمل اصلی بر روی داده های صحیح , اعشاری , مختلط و همچنین اپراتور های منطقی , میانگین rms , مزدوج مختلط و... می باشد.
- مدولاسیون سیگنال: این ماجول شامل کلاس های مدولاسیون CPFSK , مدولاسیون فرکانس , نگارنده ی OFDM (مورد استفاده در مدولاتور OFDM) و مدولاتور فاز می شود.

- کد گذاری و کدبرداری اطلاعات : این ماجول شامل کدگذار و کدبردار تفاضلی , کدگذار کانولوشنی , الگوریتم viterbi و ... می شود.
- تبدیل فوریه : این ماجول شامل کلاس هایی به منظور محاسبه تبدیل goertzel , تبدیل fft و عکس آن است.
- تبدیل موجک : این ماجول تبدیل موجک را با استفاده از روتین های gsl محاسبه می کند.
- بلوک های OFDM : مجموعه ی بلوک های این ماجول بمنظور پیاده سازی OFDM (مدولاسیون , دمدولاسیون , همزمان سازی و ...) بکار می روند.
- کدگذار و کدبردار صوت : کلاس های این ماجول کدگذاری و کدبرداری مدولاسیون cvsd و gsm با نرخ کامل (full-rate) را بر روی صدا انجام می دهند.

انواع سیستم های مدولاسیون مانند FM , AM (باند باریک و باند پهن) , SSB , PSK , DBPSK , DQPSK , D8PSK , QAM, (۸ , ۱۶ , ۶۴ , ۲۵۶ تایی) , CPM , CPFSK , GMSK , FSK و ... پیاده سازی شده است. اخیرا سیستم مدولاسیون OFDM نیز پیاده سازی گردیده و به کتابخانه ی بلوک های پردازش سیگنال اضافه شده است. کدهای تصحیح خطا (Viterbi , Reed-Solomon و کدهای Turbo) پیاده سازی شده و در کتابخانه موجود می باشند.

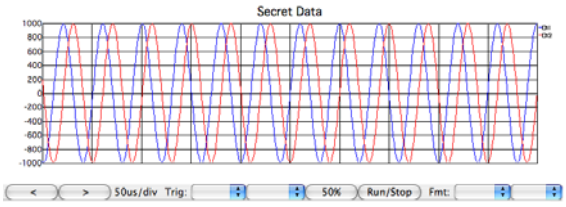
مدل های متنوعی از کانال و خرابی های آن را با استفاده از بلوک های GNU Radio می توان شبیه سازی نمود. این شبیه سازی ها هنگام پیاده سازی بلوک های مدولاتور و دمدولاتور بسیار مفید است چرا که پیش از آزمایش دستگاه بصورت عملی می توان در شرایط کنترل شده از عملکرد بلوک اطمینان حاصل کرد. همچنین این کار به یاد گیری پردازش سیگنال RF بدون نیاز به سرمایه گذاری سخت افزاری کمک می کند.

در صورت نیاز علاوه بر کتابخانه ی پردازش سیگنال GNU Radio می توان از کتابخانه های پردازش سیگنال FFTW (کتابخانه ای منبع-باز به منظور محاسبه ی تبدیل فوریه ی گسسته در ابعاد گوناگون , با سایز ورودی دلخواه , برروی داده های حقیقی و مختلط , تبدیل DCT/DST و...), کتابخانه ی علمی GNU (gsl) , کتابخانه ی منبع-باز Python برای مهندسی , علوم و ریاضیات (SciPy) , کتابخانه ی ++IT و کتابخانه ی SharpDSP استفاده نمود. همانگونه که قبلا نیز اشاره شد کاربر می تواند در صورت نیاز , بلوک پردازش سیگنال مورد نیازش را خودش به زبان C++ بنویسد. همچنین یکپارچه سازی کتابخانه ی موجود با کتابخانه ها و یا بلوک های پردازش سیگنال متفرقه که به زبان C++ نوشته شده اند نیز امکان پذیر است.

۲ – ۵ – رابط های گرافیکی کاربر GUI

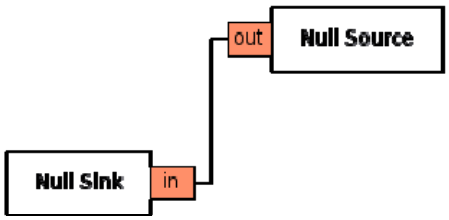
رابط های گرافیکی کاربر بمنظور نمایش گرافیکی خروجی های سیگنال (signal sink) و هچنین تعامل گرافیکی با کاربر طراح رادیو استفاده می شوند. بطور عمده ۳ رابط گرافیکی کاربر برای GNU Radio طراحی شده است:

- رابط WXGUI یک رابط گرافیکی مبتنی بر بسته ی نرم افزاری wxPython است. wxPython به برنامه نویسان زبان Python اجازه می دهد تا بسادگی برای برنامه هایی خود رابط های گرافیکی با امکانات فراوان ایجاد کنند. این مجوعه شامل یک نمایشگر هیستوگرام , یک اسیلسکوپ (به منظور نمایش زمانی سیگنال خروجی) , یک تحلیلگر طیف بر مبنای FFT , نمایشگر water-fall و یک نمایشگر constellation است.



شکل۳ – اسکوپ wxgui

- رابط QtGUI برمبنای چارچوب توسعه ی نرم افزار Qt است (Qt بیشتر بمنظور توسعه ی رابط گرافیکی کاربر استفاده می شود). QtGUI امکاناتی همانند WXGUI را با ساختار گرافیکی متفاوت ارائه می دهد.
- GRC (GNU Radio Companion) یک محیط گرافیکی شبیه simulink است که به منظور ایجاد گراف های جریان سیگنال و تولید کد اصلی مربوط گراف جریان سیگنال طراحی شده است.



شکل۴- شمای ساده از GRC

طراح با استفاده از این ابزار می تواند از کد نویسی Python اجتناب کند. این ابزار خصوصا برای یک مهندس مخابرات که اطلاعات کمی از برنامه نویسی دارد و یا نمی خواهد وارد کد نویسی Python شود بسیار مفید است. همچنین اگر کتابخانه ی پردازش سیگنال GNU Radio کافی باشد , طراح از کد نویسی C++ نیز بی نیاز بوده و کلا نیازی به کد نویسی ندارد.

۲ – ۶ – ویژگی های نگارش ۳.۲ و نگارش های بعدی

زیرساختار هسته , چند بخشی شده تا با معماری پردازنده های چند هسته ای هماهنگ تر باشد. استفاده از Python اختیاری شده و می توان کل برنامه را به زبان C++ نوشت. مجموعه ی وسیعی از بلوک های پردازش سیگنال اضافه گردیده است. این نسخه , نسل دوم سخت افزار USRP را پشتیبانی می کند. در این نگارش بطور رسمی از پردازنده های cell شرکت IBM پشتیبانی می شود. (توسعه دهندگان GNU Radio بر این باورند که با استفاده از پردازنده های cell می توان گیرنده/فرستنده هایی real-time ساخت که پهنای باندی از مرتبه ی ۲۰ مگاهرتز را پشتیبانی می کنند.)

تا بحال بیشتر تمرکز GNU Radio بر روی پایین ترین لایه ی سیستم رادیویی (معادل با لایه ی فیزیکی در زبان شبکه) قرار داشته است. این لایه , حوزه ی پردازش سیگنال داده های دنباله ای , پیوسته و نا محدود است و معماری فعلی GNU Radio بخوبی این موضوع را نشان می دهد. تمایل زیادی برای استفاده از GNU Radio در لایه های بالاتر سیستم رادیویی وجود دارد. جایی که داده ها بفرم بسته بندی شده (packetized) مورد بررسی قرار می گیرند. پروتکل های لایه ی MAC که چگونگی دسترسی به کانال مخابراتی را تعیین می کنند و سیستم TDMA که نیازمند زمان بندی دقیق و ردیف کردن شکل موج های ارسالی و دریافتی هستند نمونه هایی از این دست هستند.

علاوه بر این سیستم های رادیویی غالبا نیازمند پردازش داده های توصیف کننده ی (metadata) سیگنال های در حال گذر از سیستم می باشند. نگارش ۳.۳ مدل جریان داده ی دنباله ای موجود را با معماری داده های بسته بندی (پیام با طول محدود) تکمیل خواهد کرد. این امر به توسعه دهندگان امکان نوشتن بلوک هایی با قابلیت عملکرد در هر یک از این دو حوزه را می دهد. سازمان استاندارد VITA , یک استاندارد انتقال IF دیجیتال ارائه نموده است VITA) (49. نگارش ۳.۳ یک پیاده سازی از این استاندارد را در بر خواهد داشت.

در آخر اشاره می شود که GNU Radio جایگاه خود را در پروژه های تحقیقاتی و برنامه های آموزشی رادیو نرم افزاری پیدا کرده است. سخت افزار USRP بعنوان RF FE بسیاری از پروژه های رادیوی نرم افزاری (مانند OSSIE) مورد استفاده قرار می گیرد. با این وجود GNU Radio برای تجاری شدن راه درازی در پیش دارد. چرا که نرم افزار آن در مرحله ی تکامل و توسعه قرار دارد و هنوز به پختگی کامل نرسیده است. با این حال علاقه مندان بسیاری در سرتاسر دنیا بطور فعال در حال توسعه و اشکال زدایی کد های GNU Radio می باشند و مطمئنا در آینده از GNU Radio بیشتر خواهیم شنید.

۳- معرفی پروژه ی OSSIE

پروژه ی Open Source SCA Implementation Embedded (OSSIE) توسط گروه Wireless (یک گروه تحقیقاتی در دانشگاه Virginia Tech) در سال ۲۰۰۴ آغاز شد. هدف این پروژه فراهم آوردن یک بستر رادیو نرم افزاری ساده , قابل گسترش و منبع باز برای توسعه ی شکل موج ها ی مبتنی بر مشخصات معماری مخابرات نرم افزاری (SCA) می باشد. (عملکرد یک سیستم رادیو نرم افزاری توسط نرم افزار هایی تعیین می شود که به آن ها شکل موج گفته می شود.)

SCA توسط گروه JTRS تالیف گردیده است. این معماری بمنظور بهینه کردن روند توسعه ی سیستم های مخابراتی رادیو نرم افزاری ارائه گردیده است. ساختار SCA بمنظور نیل به اهداف زیر طراحی شده است:

- قابلیت جابجایی نرم افزار مابین سیستم های مبتنی بر SCA .

- بکارگیری استاندارد های تجاری بمنظور کاهش هزینه های توسعه .
- کاهش زمان مورد نیاز برای توسعه ی نرم افزار از طریق استفاده ی مجدد از ماجول های طراحی .
- تکمیل و اضافه کردن معماری ها و چارچوب های تجاری .

OSSIE عمدتا بمنظور امور تحقیقاتی و آموزشی در زمینه ی SDR و مخابرات بدون سیم ارائه گردیده است. مجموعه ی نرم افزاری شامل بدنه ی اصلی (core framework) رادیو نرم افزاری(منطبق بر استاندارد های SCA) , ابزاری جهت توسعه سریع اجزای SDR و شکل موج ها و همچنین یک کتابخانه ی در حال تکامل از اجزا و شکل موج های پیش ساخته است. همچنین تمرینات آزمایشگاهی رایگان بمنظور آموزش و تمرین در SDR ارائه شده و در حال تکامل است.

پروژه ی OSSIE به زبان C++ و با بکار گیری omniORB CORBA ORB (که به صورت آزاد موجود است) نوشته شده است. (CORBA جزو مشخصات استاندارد SCA است)

واسطه ی درخواست شیئ (object request broker (ORB نوعی نرم افزار میانی (میان افزار = middleware) است که به برنامه نویس اجازه می دهد تا برنامه ای موجود در یک رایانه را از رایانه ای دیگر فراخوانی کند (از طریق شبکه).

ORB ها قابلیت تعامل سیستم های شیئ توزیع شده (distributed object systems) را ارتقا می بخشند چرا که با استفاده از آن ها کاربران می توانند با کنار هم قرار دادن اشیایی از تولید کنندگان مختلف (که با یکدیگر از طریق ORB ارتباط برقرار می کنند) سیستم مورد نظر خود را تولید کنند. ORB ها وظیفه ی تبدیل ساختار های داده ی در حال پردازش به دنباله ای از بایت ها (که از طریق شبکه ارسال می گردند) را بر عهده دارند.

معماری ORB مشترک (CORBA) استانداردی است که توسط گروه مدیریت شیئ Object Management Group (OMG) تعریف شده است. این استاندارد موجب می شود تا اجزای نرم افزاری نوشته شده به زبان های برنامه نویسی مختلف و در حال اجرا بر روی رایانه های مختلف بتوانند با یکدیگر کار کنند. در این استاندارد از یک زبان توصیف رابط Interface Description Language (IDL) برای توصیف داده های در حال انتقال استفاده می شود.

omniORB یک ORB سازگار با CORBA است که تحت مجوز عمومی GNU (GPL) عرضه شده است. در حقیقت omniORB یک پیاده سازی منبع-باز از CORBA برای C++ و Python است.

پروژه ی OSSIE توسط دکتر Max Robert و دانشجویان کلاس درس SDR دکتر Jeff Reed و گروه تحقیقاتی در دانشگاه Virginia Tech آغاز شد.

OSSIE در حال حاضر تحت سیستم عامل لینوکس توسعه داده می شود. با این حال تلاش هایی جهت پیاده سازی بر روی سیستم های عامل دیگر نیز انجام می شود.

OSSIE عناصر کلیدی خصوصیات SCA را پیاده سازی می کند. سخت افزار USRP که در بالا به آن اشاره شد به عنوان RF FE در نظر گرفته شده اما قابلیت تطبیق با سخت افزار های دیگر نیز وجود دارد.

بر خلاف GNU Radio که از نبود راهنمای مرجع مناسب در بخش های مختلف رنج می برد , در اینجا با توجه به ماهیت و هدف کلی پروژه OSSIE که آموزش و پژوهش است , مستندات , فایل های راهنما , برنامه ها , ویدیو ها و تمرین های زیادی جهت مراجعه ارائه شده است.

OSSIE بر روی اکثر کامپیوتر ها با پردازنده های چند منظوره که از یک نگارش به روز لینوکس استفاده می کنند اجرا می شود. ارائه ی یک نگارش که شامل پشتیبانی بهینه از پردازنده های embedded است برای پاییز ۲۰۰۹ برنامه ریزی شده است. برنامه های شکل موج زیر در OSSIE پیاده سازی شده اند :

- گیرنده ی AM
- گیرنده و فرستنده ی FM باند باریک
- گیرنده ی FM باند پهن
- فرستنده / گیرنده ی صدای BPSK/CVSD
- گیرنده و فرستنده ی رادیوی هوشمند (cognitive) CIREN با داده های بسته بندی (packetized) شده و مدولاسیون BPSK/QPSK/16-QAM

برخلاف GNU Radio برقراری ارتباط , تعامل و پیکربندی ویژگی های بلوک ها توسط فایل های XML صورت می گیرد. (تعیین شده توسط استاندارد SCA) بعبارت دیگر ارتباط اجزا توسط میان افزار CORBA بر قرار می شود.

در مجموع پروژه ی OSSIE نیز یکی از پروژه های بسیار فعال در زمینه ی رادیو نرم افزاری است.

۴- معرفی پروژه ی Dttsp

در سال ۲۰۰۳ کمپانی Flex-Radio اولین گیرنده/فرستنده ی منبع باز SDR به نام SDR-1000 را برای رادیو آماتور به بازار عرضه کرد. همراه با این دستگاه یک بسته ی نرم افزاری کاملا منبع باز به نام SDRConsole که به زبان visual basic 6 نوشته شده بود عرضه گردید. در ابتدای سال ۲۰۰۴ این نرم افزار با PowerSDR که باز نویسی همان نرم افزار به زبان C# .NET بود جایگزین شد. این نرم افزار برای محصولات بعدی کمپانی Flex-Radio نیز مورد استفاده قرار می گیرد.

پروژه ی HPSDR توسط Phil Harman , Phil Covington و Bill Tracey آغاز شد. ایده اصلی این پروژه تقسیم کردن یک سیستم رادیویی به چندین بخش مجزا (modular کردن) بود. در این پروژه از نرم افزار PowerSDR در بخش PC استفاده می شود.

Quick Silver QS1R یک گیرنده و DDC (Digital Down Converter) منبع باز است که در پروژه های SDR مورد استفاده قرار می گیرد. به همراه این سخت افزار , نرم افزار منبع-باز SDRMAX ارائه شده است.

پروژه ی µWSDR یک پروژه ی SDR است که هدف از آن تولید خانواده ای از رادیو های مبتنی بر نرم افزار برای باند های مایکروویو و یا فرکانس های پایین تر می باشد.

در تمامی این پروژه ها (که به عنوان نمونه آورده شده اند) و تعداد کثیری از پروژه های SDR مربوط به دنیای رادیوی آماتور که عملا پیاده سازی شده اند از کد DttSP استفاده می شود.

DttSP یک پروژه ی منبع باز است که بمنظور فراهم آوردن کد مورد نیاز برای استفاده در پروژه های DSP گوناگون علی الخصوص رادیو های نرم افزاری و هوشمند آغاز گردید.

این پروژه توسط دکتر Frank Brickle و دکتر Robert McGwier از اعضای انجمن مایکروویو DTTS شروع شد.

DttSP فرایند های اولیه نظیر مدولاسیون , دمدولاسیون , همزمان سازی و تغییرات مناسب بر روی سیگنال را بنحوی پیاده سازی می کند که گیرنده/فرستنده ای که مراحل آشکار سازی و سنتز را بصورت پردازش دیجیتال سیگنال انجام می دهد عملکرد مطلوبی داشته باشد.

با اینکه عمده ی مراحل توسعه بر روی لینوکس انجام شده است کد DttSP بمنظور استفاده در Visual Studio 6 یا Visual Studio 2003 برای windows موجود می باشد. DttSP به زبان ANSI-C نوشته شده و بطور گسترده از کتابخانه ی FFTW استفاده می کند.

یکی از هدف های اصلی از طراحی DttSP فراهم آوردن یک هسته ی مرکزی SDR موثر و با قابلیت هماهنگی با محیط های مختلف بود. به عنوان نمونه کلیه ی امور مربوط به SDR را می توان بطور محلی و از طریق یک کنسول گرافیکی و یا از راه دور و از طریق یک شبکه , کنترل و اجرا نمود بدون اینکه نیازی به هر گونه تغییر و یا باز پیکربندی باشد. از آنجایی که هدف اصلی DttSP بکارگیری آن در پروژه های عملی است توجه زیادی به امنیت و قابلیت اطمینان آن شده است .

نوآوری های زیادی در طراحی انجام شده تا هزینه های محاساتی پایین نگه داشته شود یک نمونه از آن استفاده ی وسیع از بافر های حلقوی (ring buffer) است که در فایل های memory-mapped واقع شده اند. این کار یک ارتباط سریع یک طرفه مابین process های کاربر بوجود می آورد بدون اینکه نیازی به handshaking و یا دخالت سیستم عامل باشد.

تاکنون رابط های گرافیکی کاربر زیادی برای هسته ی اصلی رادیو نرم افزاری DttSP (sdr-core) بوجود آمده است که بعنوان نمونه می توان از java-SDR و SDR-Shell نام برد. SDR-Shell یک رابط گرافیکی کاربر ساده است که برای استفاده در گیرنده های ساده ی SDR مثل SoftRock (یک گیرنده ی ارزان قیمت SDR) و SDRZero پیاده سازی شده است. Java-sdr نیز پروژه ای بمنظور ایجاد کنسولی به زبان java برای رادیو ی مبتنی بر نرم افزار SDR-1000 است که سازگار با سیستم های عامل لینوکس و MAC OS X باشد.

نتیجه گیری:

در این گزارش نقش نرم افزار های منبع-باز در توسعه ی بخش نرم افزار پروژه های رادیو نرم افزاری مورد بررسی قرار

گرفت. GNU Radio بعنوان یکی از فعال ترین و مهم ترین پروژه ها ی منبع-باز معرفی شد. همچنین OSSIE بعنوان یکی از بهترین ابزار های آموزشی و پژوهشی در زمینه های SCA و SDR مورد بررسی قرار گرفت. در انتها کد DttSP بعنوان یکی از کد های پر استفاده در کاربرد های عملی و مصارف رادیوی آماتور به اختصار معرفی شد. با وجود پروژه های منبع-باز , علاقه مندان به توسعه ی سیستم های رادیو نرم افزاری بدون هیچ گونه هزینه ای می توانند وارد این عرصه شده و کار دیگران را مرور کنند و به توسعه و رفع اشکالات موجود بپردازند. این ویژگی ها موجب شده تا تعداد علاقه مندان و فعالان در این حوزه روز بروز بیشتر گردد.

مراجع:

[1] GNU Radio Wiki Development site:

http://gnuradio.org/trac

[2] The Comprehensive GNU Radio Archive Network:

https://www.cgran.org

[3] http://www.ettus.com/

[4] http://www.swig.org/

[5] http://www.wxpython.org/

[6] Updated Unofficial Gnuradio Documentation (Simple User Manual):

http://lwn.net/Articles/259832/

[7] http://www.joshknows.com/

[8] https://radioware.nd.edu/

[9] http://staff.washington.edu/jon/gr-mrfm/

[10] http://sca.jpeojtrs.mil/

[11] http://omniorb.sourceforge.net

[12] http://www.flex-radio.com/About.aspx?topic=history

[13] http://dttsp.sourceforge.net/

[14] http://uwsdr.berlios.de/

[15] http://ewpereira.info/sdr-shell/

[16] http://www.sds.lcs.mit.edu/SpectrumWare

[17] http://ossie.wireless.vt.edu/

[18] http://www.fftw.org/

[19] http://openhpsdr.org/

[20] http://www.philcovington.com/SDR.html

[21] http://f4dan.free.fr/sdr_eng.html

[22] S. Cowling, Hands-On Software Defined Radio:

http://www.nonstopsystems.com/radio/article-sdr-hands-on.pdf

[23] Eric Blossom , “GNU Radio: Tools for Exploring the Radio Frequency Spectrum”

[24] Álvaro Palomo Navarro, Rudi Villing and Ronan Farrell, “SOFTWARE DEFINED RADIO ARCHITECTURES EVALUATION”

[25] Interview with the GNU Radio project's Johnathan Corgan:

http://lwn.net/Articles/336676/

[26] Naveen Manicka, “GNU RADIO TESTBED”

[27] BBN Technologies Corp, “GNU Radio Architectural Changes”

[28] OSSIE 0.7.4 Installation and User Guide

[29] “Enabling open-source cognitively-controlled collaboration among software-defined radio nodes” , Elsevier

[30] Nick McCarthy, Eric Blossom, Nate Goergen, Tim OShea, Charles Clancy, “High-Performance SDR: GNU Radio and the IBM Cell Broadband Engine”

[31] GNU Radio Source code