دانشکده مهندسی برق
دانشگاه صنعتی شریف

فاز ۲ پروژه درس
برنامه نویسی شیءگرا

نیمسال دوم ۰۴-۱۴۰۳

دکتر آراسته، دکتر شیرعلی، دکتر هاشمی

**نمایه‌ای بر نرم افزار های
شبیه سازی مدار های الکتریکی**

دانشگاه صنعتی شریف
دانشکده مهندسی برق

فاز دوم پروژه ی درس برنامه سازی شی گرا

بخش های مشخص شده در کادر قرمز برای گروه های ۲ نفره شامل نمره ی امتیازی شده و برای گروه های ۳ نفره اجباری می باشد.

بخش های مشخص شده در کادر سبز برای گروه های ۳ نفره شامل نمره ی امتیازی می شود.

بخش های مشخص شده در کادر های زرد ربطی به محتوای پروژه نداشته و صرفا برای علاقه مندان قرار داده شده است.

فهرست مطالب

۱ مقدمه

در این فاز قرار است شما برای ابزار تحلیل مدار های الکتریکی ای که در فاز قبل به صورت CLI (Command Line Interface) طراحی کرده بودید یک محیط گرافیکی با استفاده از SDL (Simple Directmedia Layer) طراحی کنید.

منظور از طراحی محیط گرافیکی برای این ابزار تحلیل مداراتی است که به صورت گرافیکی توصیف می شوند؛ نه توصیف با استفاده از شماره گزاری Node ها و یا هر روش غیر گرافیکی ای.

توصیف مدار ها به روش گرافیکی این امکان را به ما می دهد که مدار های پیچیده را بدون نیاز به هیچ گونه ساده سازی و یا شماره گزاری Node ها توصیف کنیم و تحلیل های مورد نظر را به صورت عددی روی آن انجام دهیم.

همچنین از اهداف اصلی این فاز نمایش سیگنال های مربوط به Node های حاضر در مدار به صورت گرافیکی در گذر زمان می باشد.

نمایش سیگنال های الکتریکی به صورت گرافیکی در طول زمان این امکان را به ما می دهد که دید بهتری نسبت به رفتار مدار به عنوان یک سیستم با سیگنال ورودی الکتریکی و سیگنال خروجی الکتریکی داشته باشیم و از پیچیدگی های مربوط به خروجی مدار به صورت مقادیر گسسته می کاهد.

همچنین در این فاز از پروژه، شما به تحلیل فرکانسی و فازی مدار علاوه بر تحلیل در حوزه ی زمان (Transient) خواهید پرداخت.

یکی دیگر از اهداف این فاز از پروژه، ذخیره سازی پروژه های مربوط به تحلیل مدار های الکتریکی است. در پایان این فاز شما پروژه های کامل تحلیل مدار های الکتریکی خواهید ساخت و نیاز به ذخیره کردن این پروژه ها دارید؛ بدین منظور که توصیف گرافیکی ارایه شده توسط کاربر و خروجی های ذخیره شده مربوط به آن به صورت یک پروژه و فایل های مجزا ذخیره شوند.

همچنین بخش مربوط به شبکه نیز در این فاز قرار داده شده است. با وجود اینکه این قابلیت در نرم افزار های تحلیل مدار های الکتریکی وجود ندارد (و حتی کاربردی هم ندارد!) ولی در درک شما نسبت به شبکه و یادگیری یک مهارت جدید کمک بسیاری می کند.

جزییات مربوط به هرکدام از این بخش ها را در ادامه خواهید دید.

* پی نوشت

توجه داشته باشید که این فاز ترتیب مشخصی نداشته و تا وقتی که برنامه ی شما خروجی مطلوبی داشته باشد مجاز هستید که به هر روش و یا ترتیبی به انجام این فاز از پروژه بپردازید؛ ولی توصیه ی دستیاران درس و مسئُولین این فاز به این است که برای راحتی هرچه بیشتر در انجام پروژه و داشتن یک طراحی درست و اصولی به ترتیب با استفاده از Design Pattern های پیشنهاد شده در طول این دستورالعمل پیش بروید.

داشتن یک طراحی درست در ساخت یک Software از مهم ترین اهداف این درس و پروژه است.

هم چنین پیشنهاد می شود که قبل از شروع پروسه ی Developement نرم افزار، یک بار این Document را به صورت کامل بخوانید تا در ادامه برای طراحی نرم افزار به مشکل نخورید.

مطمین باشید که با طراحی درست زیر بخش های پروژه در آخر برای اتصال این زیر بخش ها به هم و تشکیل پروژه ی نهایی کار بسیار راحتتری خواهید داشت و از نتیجه ی نهایی لذت بیشتری خواهید برد!

۱.۱ کار با SDL

شکل ۱: *بسیاری از Framework های گرافیکی مانند PyGame که در زبان برنامه نویسی Python استفاده می شود، با استفاده از SDL2 Develope شده اند!*

اگر در هر جایی از پروژه کد های مربوط به SDL2 و یا توابع مربوط به آن را فراموش کردید و یا می خواستید نحوه ی درست پیاده سازی یک المان گرافیکی و یا نحوه ی درست Handle کردن Event ها و ... را بدانید، می توانید از طراحان این فاز سوال کرده و یا از منابع زیر استفاده کنید:

(1) *LazyFoo's Introduction to SDL2*

(2) *SDL2's Official Documentation*

(3) *SDL2 Wiki*

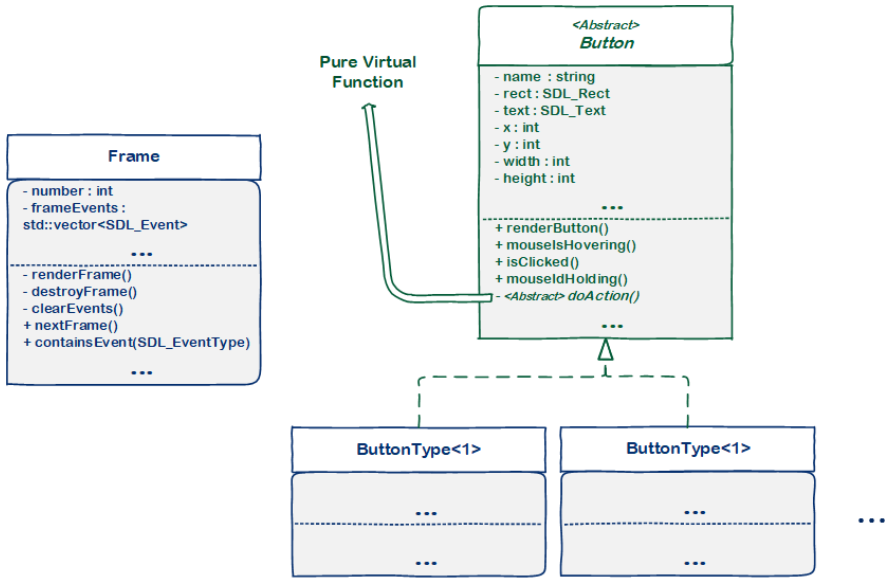
(4) *SDL2 GitHub Repository*

در ادامه توجه داشته باشید که بهترین راه برای Handle کردن گرافیک، Implement کردن المان های گرافیکی مانند Button و ... به صورت Object Oriented است.

با توجه سنگینی پروژه، تعداد بالای منو ها و حجم بالای Process ای که برای Generate کردن هر Frame وجود دارد، در صورتی که گرافیک پروژه را به صورت درست و اصولی Handle نکنید، قطعا برنامه ی شما Crash خواهد کرد و نمره بخش گرافیک را از دست خواهید داد!

در نتیجه به موضوع Handle کردن Event ها، Generate کردن هر Frame و ... بسیار توجه کنید.

از آنجایی که Source مناسبی برای طراحی المان های گرافیکی در SDL به صورت Object Oriented وجود ندارد، مسّولیت این طراحی بر عهده ی شماست. برای مثال برای Handle کردن Frame هایی که Generate می شوند. می توانید یک کلاس به نام `Frame` تعریف کنید و به این صورت هر Frame را Render کرده، Event های آن را Handle کرده و سپس آن را پاک کرده و یک Frame جدید بسازید. (شما می توانید المان های گرافیکی را به هر نحوی طراحی کنید. طراحی داده شده صرفا یک پشنهاد و قالب کلی برای این کار است.)



شکل ۲: نمونه ای از طراحی المان های گرافیکی به صورت Object Oriented

۲ منبع ولتاژ با مقادیر از قبل مشخص شده

یکی از المان های جدیدی که در این فاز از پروژه باید اضافه کنید، منبع ولتاژ با مقادیر خاص در زمان است.
تا اینجای کار برای خروجی یک منبع ولتاژ خاص مانند سینوسی و یا کسینوسی از یک Array با طول خاص استفاده می کردید که مقادیر المان های آن با توجه به تابع مربوطه (مثلا sin و یا cos) مشخص می شد.
حال از شما می خواهیم که یک المان جدید به نام Waveform Source بسازید که می توان مقادیر مربوط به ولتاژ را در Time Sample های مختلف را به صورت دستی برای آن تعریف کرد.
توجه داشته باشید که علاوه بر قرار دادن مقادیر مربوط به آن Time Sample در آرایه، باید اطلاعات مربوط به مدت زمان سیگنال، Sampking Rate (F_s) و ... را نیز ذخیره کنید. در حقیقت در صورت پیاده سازی کلاس `Signal` می توانید در همین کلاس تابعی برای ساخت یک سیگنال با مقادیر خاص Implement کنید.

۳ نمایش سیگنال های الکتریکی به صورت گرافیکی

۱.۳ مقدمه

برای پیاده سازی این بخش از پروژه ابتدا ببینیم که نرم افزار های شاخص در این حوزه مثل PSpice, LTSpice و یا MultiSim به چه صورت این کار را انجام می دهند.

نکته ی مهم درمورد این بخش این است که Plot کردن سیگنال ها در این نرم افزار ها به صورت عددی انجا می شود.

برای درک بیشتر این موضوع به این مثال توجه کنید.

فرض کنید خروجی یک سیستم الکتریکی بدین صورت باشد:

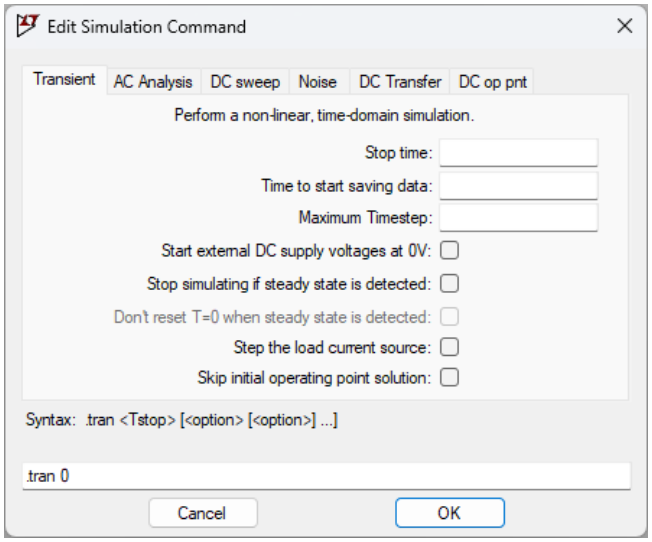
$$y(t) = e^{-t}u(t)$$

نرم افزار های تحلیل و طراحی مدار و تقریبا هر نرم افزاری که به نحوی با Plot کردن یک تابع و یا سیگنال سر و کار دارد از روش عددی و به بیانی گسسته برای این کار استفاده می کنند.

فرض کنید می خواهیم این خروجی را برای زمان های $t_0 = 0s$ تا $t_1 = 2s$ Plot کنیم.

برای انجام این کار به روش عددی و گسسته نیاز به یک مولفه ی از پیش تعیین شده به نام Maximum Timetep داریم.

این مولفه مشخص می کند که هر چند ثانیه یک بار از سیگنال خروجی مربوط به سیستم نمونه برداری شود. به بیانی دیگر فرکانس نمونه برداری از سیگنال خروجی توسط این مولفه مشخص می شود و واضح است که هرچقدر این مقدار کمتر باشد، فرکانس نمونه برداری بالاتر بوده و بدین ترتیب نمایش گرافیکی دقیق تری (نزدیک تر به نمایش پیوسته) از سیگنال خروجی خواهیم داشت.

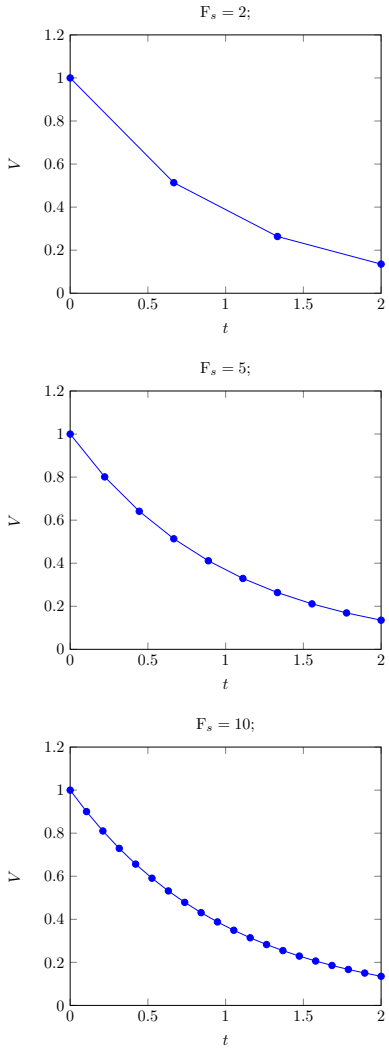


شکل ۳: مولفه های مربوط به تحلیل Transient یا همان تحلیل در زمان در نرم افزار LTSpice

شما برای این بخش از پروژه باید ۳ مولفه ی اول یعنی Stop Time (t_1)، Time to start saving data (t_0) و Maximum Timestep را به عنوان مولفه های ورودی برای Plot کردن سیگنال بگیرید.

جزییات کامل مربوط به منوی تنظیمات شبیه سازی را در بخش منو های پروژه خواهید دید.

برای مثال در این مثال Plot تشکیل شده برای سیگنال خروجی به ازای مقادیر مختلف Maximum Timestep بدین صورت می باشد.



در اینجا به صورت واضحی سیگنال نمایش داده شده با افزایش فرکانس نمونه برداری به یک نمایش پیوسته از سیگنال نزدیک تر می شود و خروجی گرافیکی ما مطلوب تر خواهد بود؛ در نتیجه برنامه ی شما باید توانایی Handle کردن فرکانس های نمونه برداری بزرگ (یعنی تعداد داده های گسسته ی خیلی زیاد!) را داشته باشد.

همچنین برای نزدیک کردن خروجی نمایش داده شده به یک خروجی پیوسته نمونه های برداشته شده از سیگنال را در زمان های مشخص با یک خط به هم متصل می کنیم. به این کار Linear Interpolation می گویند.

توجه داشته باشید که در این پروژه موظف به نمایش خروجی های شبیه سازی های Transient، AC Sweep و Phase Sweep به صورت گرافیکی هستید.

در نتیجه طبق روند توضیح داده شده باید خروجی های عددی به دست آمده در فاز قبل را به صورت گرافیکی نمایش دهید.

۲.۳ انتخاب خروجی مورد نظر

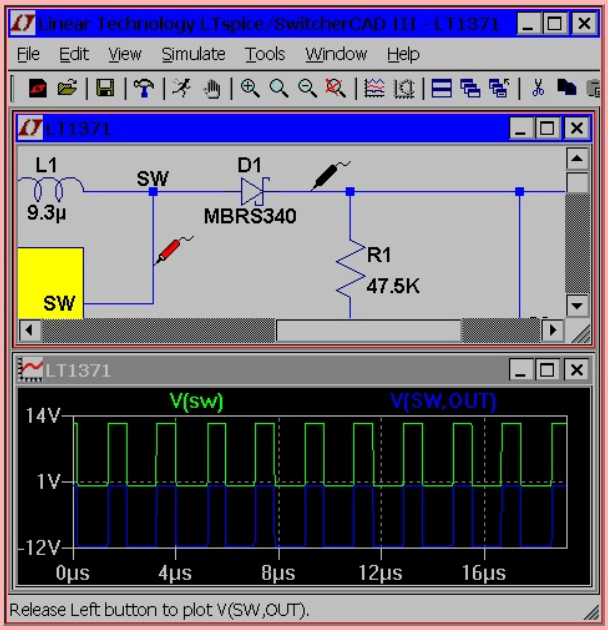
برای انتخاب خروجی مورد نظر می توانید مانند پیاده سازی ای که در فاز ۱ انجام داده بودید با انتخاب ولتاژ و یا جریان مورد نظر از یک منو و یا انتخاب جریان و یا ولتاژ مورد نظر به صورت متنی و با استفاده از Command در منوی Plot آن را رسم کنید.

۱.۲.۳ انتخاب سیگنال مورد نظر با استفاده از Probe

یکی از قابلیت های پیاده سازی شده در نرم افزار های شبیه سازی مدار ابزار Probe است.
Probe ها در این نرم افزار ها در ۳ نوع مختلف پیاده سازی شده اند.

* Voltage Probe
* Current Probe
* Power Probe

برای استفاده از این ابزار با انتخاب Probe مورد نظر و انتخاب المان و یا Node مورد نظر می توان ولتاژ، جریان و یا توان مربوط به آن المان را خوانده و آن را به صورت گرافیکی نمایش داد یا برای ولتاژ یک Node نسبت به Ground Node همین کار را انجام داد.
شما در این پروژه می توانید این ۳ نوع Probe را پیاده سازی کرده و به کاربر این امکان را بدهید که با انتخاب Probe مورد نظر و کلیک کردن برروی المان و یا Node مشخص ولتاژ، جریان و یا توان مربوط به آن المان را خوانده و به صورت گرافیکی رسم کند.



شکل ۴: نمونه ای از استفاده از Probe ها در نرم افزار LTSpice

۳.۳ آماده سازی خروجی عددی برای Plot کردن سیگنال

برای استفاده از داده های عددی به دست آمده در فاز قبل برای Plot کردن سیگنال ها همان طور که در مقدمه ذکر شد نیاز به داشتن مولفه های Sampling Rate و Stop Time داریم.

برای خواندن داده های عددی از فایل مربوطه می توان دو رویکرد مختلف را در نظر گرفت:

(۱) خواندن داده ها به صورت یک جا از روی فایل و ذخیره سازی آن ها در یک `vector`

(۲) خواندن داده ها به صورت یکی یکی یا Chunk هایی به طول n و پردازش داده های موجود در فایل به صورت مرحله به مرحله

هر کدام از این رویکرد ها مزایا و معایبی دارد که در ادامه به آن ها می پردازیم.

از مزایای خواندن داده ها به صورت یک جا این است که احتمالا سرعت پردازش داده ها بالا خواهد رفت چرا که داده ها فقط یک بار خوانده می شوند.

ولی مشکل بزرگ این رویکرد حافظه است! برای اینکه نسبت به این مسیله شهود بیشتری داشته باشید به این مثال توجه کنید.

(*) مثال

فرض کنید میخواهیم یک سیگنال را با $t = 10s$ Stop Time و با $F_s = 10000$ Sampling Rate Plot کنیم.

بدین ترتیب به سادگی می توان محاسبه کرد که برای ذخیره سازی این داده ها به یک `vector` به طول $t \times F_s = 10 \times 10000 = 100000$ نیاز داریم.

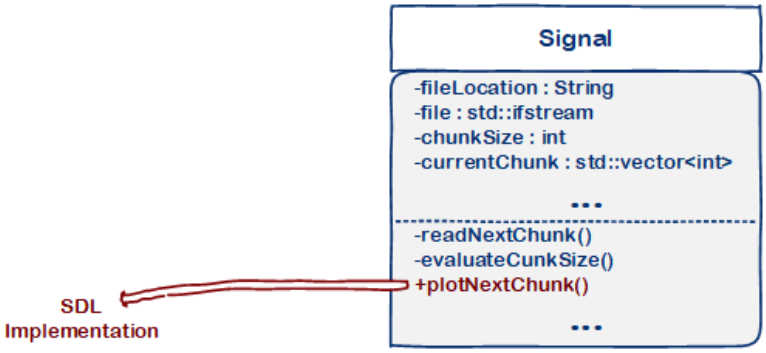
بدین ترتیب به نظر می رسد که خواندن تمام داده ها از فایل به صورت یکجا و ذخیره سازی مستقیم آن ها در یک `vector` کار منطقی ای نباشد.

بنابراین می توان ساختار داده ی شخصی سازی شده ی خود را به صورت یک Class طراحی کرده و داده ها را به صورت Chunk هایی به طول n (می توانید برای مشخص کردن مقدار n از الگوریتم مشخصی که با توجه به اندازه ی داده ی موجود در فایل تغییر می کند استفاده کنید!) ذخیره می کند طراحی کنید.

یکی دیگر از مزایای ذخیره کردن نتایج خروجی های مدار به صورت فایل این است که این خروجی هارا می توانید در Folder مربوط به پروژه نگه داری کرده و هنگام باز کردن دوباره ی پروژه آن ها را بازیابی کنید. (درمورد این موضوع در بخش مربوط به ذخیره سازی اطلاعات کامل تری موجود است.)

توجه کنید که پیاده سازی این بخش کاملا در اختیار شماست و می توانید از هر روشی برای Handle کردن این بخش استفاده کنید. روش داده شده صرفا یک روش پیشنهادی است و صرفا خروجی این بخش بررسی خواهد شد.

برای مثال می توانید Data Type خود را به صورت کلاس Signal و با طراحی ای مشابه این UMl Diagram انجام دهید.



شکل ۵: UML Diagram for "Signal"

۴.۳ نمایش چند سیگنال روی یک Plot

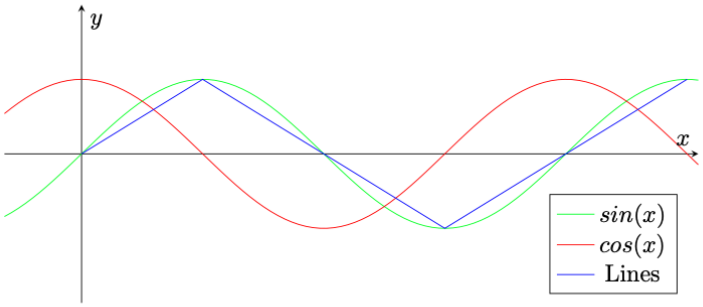
یکی از قابلیت های بسیار مهم در نرم افزار های تحلیل مدار های الکتریکی، نمایش چند سیگنال مربوط به ولتاژ یا جریان چند المان مختلف است. این قابلیت این امکان را به ما می دهد که چند سیگنال مختلف (برای مثال سیگنال ورودی و خروجی مربوط به سیستم الکتریکی) را در یک Plot و در یک بازه ی زمانی مشخص با هم ببینیم و مقایسه کنیم. این قابلیت در افزایش دید و شهود ما نسبت به عملکرد سیستم بسیار کمک می کند.

برای این کار لازم است که مقدار هر کدام از سیگنال ها را در یک نمونه ی خاص روی محور عمودی مربوط به آن زمان نمونه برداری رسم کنیم.

برای این کار می توان رویکرد های مختلفی را در نظر گرفت. نکته ی مهم این است که روش انتخابی برای انجام این کار تا حد ممکن Memory Efficient و Time Efficient باشد؛ چرا که در این نرم افزار با داده هایی با اندازه های خیلی زیاد سر و کار داریم.

همچنین برای تمایز بین سیگنال های نمایش داده شده و افزایش قابلیت تفکیک سیگنال ها از یک دیگر می توان رنگ و Label مربوط به هر سیگنال را در منوی مربوط به آن تغییر داد.

برای مثال با کلیک کردن برروی سیگنال مورد نظر می توان رنگ مربوط به آن سیگنال را تغییر داد. جزییات مربوط به پیاده سازی این بخش در اختیار شماست و تا زمانی که قابلیت تغییر رنگ و قرار دادن Legend در برنامه ی شما وجود داشته باشد، نمره ی این بخش را می گیرید. (منظور از داشتن Legend، منویی است که در آن مشخص شود هر سیگنال با چه رنگی نمایش داده شده است.)



شکل ۶: نمونه ای از Legend در نمایش گرافیکی چند سیگنال.

همچنین درصورتی که کاربر به صورت دستی این رنگ ها را تغییر نداد، شما باید به صورت پیش فرض از رنگ های متمایز برای نمایش سیگنال ها استفاده کنید و فقط وقتی که کاربر رنگ آن ها را به صورت دستی تغییر داد رنگ آن ها را عوض کنید.

برای این کار می توانید از رنگ های پیش فرض و یا Random استفاده کنید!

۵.۳ انجام عملیات ریاضی روی سیگنال ها

نتیجه انجام عملیات ریاضی روی چند سیگنال که در فاز قبل پیاده سازی کرده بودید نیز در این بخش باید نمایش داده شود. در نتیجه باید امکان انجام عملیت ریاضی مثل جمع و یا تفریق با ضریب ثابت مشخص را در این بخش (بخش Scope) پیاده سازی کنید. برای انجام این کار می توانید ۲ رویکرد مختلف را درنظر بگیرید.

* تعریف عملیات ریاضی به صورت یک Mathematical Expression
* تعریف عملیات ریاضی با استفاده از یک منوی جداگانه و Option های پیش فرض

روش اول رویکردی است که اکثر نرم افزار های SPICE Based پیاده سازی شده است. در این نرم افزار ها شما می توانید در منوی Scope یک سیگنال جدید اضافه کرده و آن سیگنال را به صورت یک Mathematical Expression تعریف کنید.

برای مثال اگر در مدار یک مقاومت با نام R_1 وجود داشته باشد، می توان با یک Mathematical Expression مشابه دستور زیر سیگنال مربوط به ولتاژ دو سر مقاومت را با یک ضریب ثابت نشان داد.

```
K * (V{R1}(+) - V{R1}(-))
```

که در اینجا K ضریب ثابت است.

همچنین می توانید برای پیاده سازی این بخش یک منوی جداگانه در نظر گرفته و عملیات های ریاضی مختلف را در آن به صورت Option قرار دهید. سپس پس از انتخاب عملیات ریاضی مورد نظر (Operator) امکان انتخاب سیگنال های مربوط به المان های مختلف که در این عملیات ریاضی شرکت داده شوند (Operands) به کاربر داده می شود.

اما در صورتی که این بخش را به صورت یک منوی جداگانه پیاده سازی کردید باید توجه داشته باشید که امکان اعمال دوباره ی عملیات ریاضی برروی سیگنال حاصل از یک عملیات ریاضی وجود داشته باشد. چرا که در غیر این صورت به انجام عملیات ریاضی بین فقط ۲ سیگنال محدود خواهیم شد.

برای مثال برای جمع سیگنال های مربوط به ولتاژ ۲ سر ۳ مقاومت در این دو پیاده سازی باید به این صورت عمل کنیم.

پیاده سازی نوع اول

```
(V{R1}(+) - V{R1}(-)) + (V{R2}(+) - V{R2}(-)) + (V{R3}(+) - V{R3}(-))
```

پیاده سازی نوع دوم

جمع سیگنال های مربوط به مقاومت های R_1 و R_2 در منوی مربوطه و گرفتن سیگنال $OUT(\mathbb{I})$.

جمع سیگنال های $OUT(\mathbb{I})$ و سیگنال مربوط به ولتاژ دو سر مقاومت R_3.

۶.۳ Vertical and Horizontal Scaling

یکی از قابلیت های مهم در نمایش سیگنال ها به صورت گرافیکی توانایی Scale کردن آن ها به صورت افقی و عمودی می باشد. همانطور که در Oscilscope های موجود در آزمایشگاه مدار های الکتریکی (۱) دیده اید یکی از کار هایی که می توان انجام داد این است که روی سیگنال نمایش داده شده به صورت افقی و عمودی Scale یا به بیانی دیگر Zoom کنیم.

برای انجام این کار نیاز به دو مولفه ی Vertical Scale و Horizontal Scale داریم.

محیط نمایش سیگنال های گرافیکی در نرم افزار های تحلیل مدار های الکتریکی مثل Oscilscope های موجود در آزمایشگاه است. در این محیط Grid مربوطه دارای تعداد مشخص و ثابتی از خانه های افقی و عمودی می باشد.

در اینجا شما با تنظیم اندازه یا به بیانی Scale مربوط به هرکدام از این خانه ها در جهات افقی و عمودی می توانید سیگنال را در جهات افقی و عمودی Scale کنید.

توجه داشته باشید که واضحا این مولفه ها برای تمام سیگنال های نمایش داده شده تغییر می کنند؛ نه برای یک سیگنال خاص.

۱.۶.۳ Auto Zoom

برای این بخش می توانید قابلیت Auto Zoom را نیز پیاده سازی کنید. با فعال سازی این قابلیت سیگنال و یا سیگنال های نمایش داده شده باید در محور عمودی روی Maximum و Minimum سیگنال قرار گرفته باشند. یعنی سیگنال ها به صورت کامل و بدون Cut شدن بخشی از آن ها به صورت عمودی نمایش داده شده و همچنین فضای اضافی (نوار های سیاه بالا و پایین در نمایش سیگنال) ظاهر نشود.
منطقا در این حالت Vertical & Horizontal Scale را باید به صورت خودکار و با توجه به مقادیر مربوط به سیگنال تنظیم کنید.

۷.۳ Cursors

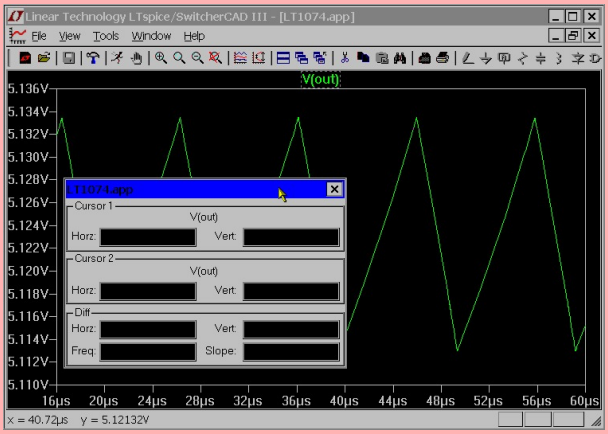
شما باید در منوی مربوط به نمایش گرافیکی سیگنال، ابزار Cursor را نیز قرار دهید.

کارکرد ابزار Cursor بدین صورت است که شما با کلیک کردن بر روی یک نقطه از سیگنال نمایش داده شده، می توانید به مقدار سیگنال در آن نقطه دسترسی پیدا کنید. (یعنی مقدار سیگنال را در آن نقطه بخوانید.)

در اصل این ابزار این اجازه را به شما می دهد تا با مشخص کردن مقدار مربوط به محور افقی (زمان، فرکانس و ...) مقدار مربوط به محور عمودی سیگنال (ولتاژ، جریان و ...) دسترسی پیدا کنیم.

۸.۳ Double Cursor

یکی از قابلیت هایی که ابزار Cursor می تواند داشته بشاد، قابلیت Double Cursor است.
در این حالت کاربر با انتخاب ۲ نقطه برروی سیگنال، علاوه بر به دست آوردن مقدار مربوط به محور عمودی در سیگنال برای ۲ نقطه ی مشخص شده، مقادیر مربوط به اختلاف دو Cursor در محور عمودی و افقی هم نمایش داده می شود.
این قابلیت مخصوصا برای محاسبه ی دوره ی تناوب (T) سیگنال بسیار کاربردی است.
برای درک بهتر ابزار Double Cursor به این مثال توجه کنید.



شکل ۷: نمونه ای از استفاده ی Double Cursor

در اینجا دو Cursor انتخاب شده با استفاده از کلیک موس را می توانید مشاهده کنید.
در منوی باز شده، اطلاعات مربوط به مقدار هرکدام از Cursor ها در محور افقی و عمودی نمایش داده شده است.
همچنین در بخش `diff` اطلاعات مربوط به فاصله ی بین دو Cursor در دو محور افقی و عمودی نمابش داده شده است.
در این بخش از پروژه صرفا کافی است بخش های Horizontal Difference و Vertical Difference و Slope را پیاده سازی کنید. نیازی به پیاده سازی بخش Frequency نیست.

۴ ساخت تک قطبی

۱.۴ مقدمه

یکی از قابلیت هایی که در اکثر نرم افزار های تحلیل وطراحی مدار قرار داده شده است، قابلیت قرار دادن یک System از قبل طراحی شده در یک Black Box و اضافه کردن آن به Node Library برای استفاده های بعدی است.

این قابلیت این امکان را به ما می دهد که یک لایه ی جدیدی از Abstraction را به طراحی مدار های خود اضافه کنیم.

در این پروژه از شما انتظار می رود که این قابلیت را به کاربر بدهید که با انتخاب ۲ Node از Node های تعبیه شده در مدار طراحی شده در یک پروژه، آن را به عنوان یک المان جدید به Node Library اضافه کند.

واضح است که این Subcircuit ها نیازی به داشتن زمین و ... ندارند و خطاهای مربوط به این موارد را دریافت نمی کنند. تنها خطایی که در این بخش باید Handle کنید، خطای مربوط به مشخص کردن Node های اصلی است.

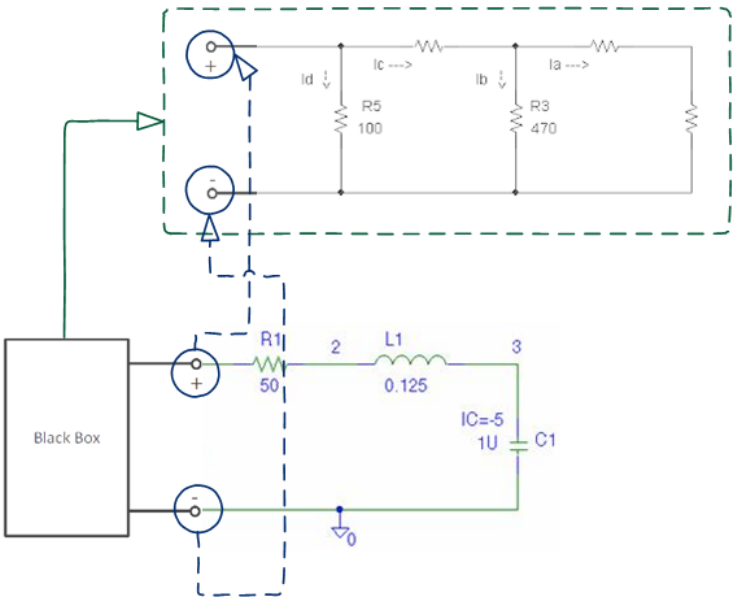
در کل درمورد تئوری تک قطبی ها و تحلیل های مربوط به آن ها در درس تئوری مدار های الکتریکی خواهید آموخت.

۲.۴ مشخص کردن Node ها

برای ذخیره سازی مدار طراحی شده به عنوان یک Subciruit، باید ۲ Node را از آن مشخص کنید که از آن ۲ Node به بقیه ی مدار متصل شود.

این بخش را باید به صورت گرافیکی و به هر روشی که برای شما راحت تر است پیاده سازی کنید. (می توانید Node ها را با نام مربوط به آن ها به صورت Command مشخص کنید و یا با استفاده از یک جفت Probe این دو Node اصلی را مشخص کنید.)

در ادامه در منوی مربوط به File ها و Save کردن پروژه، باید امکان ذخیره سازی پروژه را به عنوان یک Subcircuit به کاربر بدهید.



شکل ۸: نحوه ی عملکرد تک قطبی ها

۳.۴ ذخیره سازی Subcircuit

درمورد ذخیره سازی پروژه ها و بقیه ی مواردی که باید در این پروژه ذخیره کنید به طور مفصل در بخش مربوط به ذخیره سازی صحبت خواهیم کرد.

درمورد ذخیره سازی Subcircuit ها توجه کنید که این المان های ساخته شده را باید در یک Folder جداگانه (مثلا با نام `lib`) به صورت فایل های جدا ذخیره کرده و برای استفاده در تعریف مدار، آن ها را از روی فایل مربوطه Load کنید.

برای اینکه Handle کردن این Subcircuit ها راحتتر باشد، بهتر است که یک کلاس با نام `SubCircuit` طراحی کنید که از کلاس `Element` Derive شده است. بدین ترتیب می توانید به راحتی این Object ها را ذخیره و سپس Load کنید و همچنین مانند یک المان مداری با آن ها رفتار کنید.

۴.۴ نمایش گرافیکی

لازم به ذکر است که حتما بایستی در منو المان ها تک قطبی مورد نظر باشد اما در صورتی که نمیخواهید امتیازی بزنید در صورت اضافه کردن تک قطبی به صفحه در نمایش دقیق آن مجاز هستید.

برای نمایش گرافیکی Subcircuit ها در مدار، استفاده از یک مستطیل با اسم مربوط به آن Subcircuit و ۲ Port مربوطه کافی است.

۵.۴ مدار معادل تونن و نورتن

یکی از کارهایی که می تواند تحلیل مدار های شامل Subcircuit ها را تسریع کند، تحلیل Subcircuit ساخته شده قبل از ذخیره سازی آن می باشد. می توانیم برای سادگی تحلیل Subcircuit قبل از ذخیره سازی آن را تحلیل کرده و مدار معادل تونن و یا نورتن آن را به جای کل Subcircuit قرار دهید. این کار یک مرحله از تحلیل و Process کردن مدار را به بخش ذخیره سازی Subcircuit منتقل می کند.

درصورت پیاده سازی این بخش تضمین می شود که Subcircuit های ذخیره شده فقط مدار های مقاومتی بوده و شامل خازن و سلف نیستند. این قضیه باعث می شود که مدار های معادل تونن و نورتن Derive شده فقط شامل منبع و مقاومت باشند.

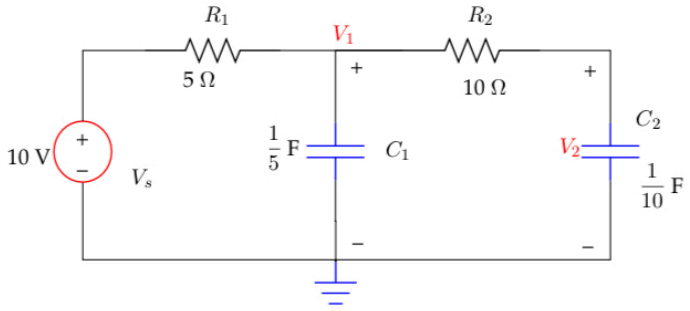
۵ توصیف گرافیکی مدار های الکتریکی

۱.۵ مقدمه

در این بخش باید شماتیک کلی و عناصر مختلف مدار، باید به صورت گرافیکی پیاده‌سازی شوند. نیاز ما دیدن مدار و اطلاعات کلی مربوط به عناصر آن در یک نگاه برای فهم بهتر کارکرد مدار است. لذا هرگونه پیاده‌سازی این بخش که این نیاز را برآورده کند قابل قبول است و از خلاقیت شما در جلوه‌ی بصری استقبال می‌شود. البته موارد مشخصی باید در این پیاده‌سازی وجود داشته باشد که در ادامه آمده است.

برای درک بهتر این موضوع به توصیف یک مدار الکتریکی به صورت گرافیکی و به Format نوشتاری مطابق فاز اول توجه کنید.

```
add R R1 0 1 5
add R R2 1 2 10
add C C1 1 3 0.2
add C C2 2 3 0.1
add GND 3
add VoltageSource VS 0 3 10
```



توجه داشته باشید که خطاهای مربوط به توصیف گرافیکی مدار ها را که در فاز قبل پیاده سازی کرده بودید را باید در این بخش هم اجرا کنید.

۲.۵ ویژگی‌های گرافیکی عمومی

در این پروژه از شما انتظار پیاده سازی مواردی مانند Zoom In / Zoom Out و یا حرکت در صفحه ی مربوط به توصیف گرافیکی مدار را نداریم.

Shematic مربوط به توصیف گرافیکی مدار یک صفحه ی Grid دار (برای مشخص کردن جای Node ها) خواهد بود که المان های مداری را برروی Node ها جاگذاری خواهید کرد.

البته که می توانید طراحی را بدون صفحه ی Grid دار انجام دهید. ولی در این صورت برای متصل کردن Node ها به هم با استفاده از سیم و یا با استفاده از المان ها و حتی قرار دادن المان ها بین ۲ Node با چالش رو به رو خواهید شد.

۳.۵ عناصر مداری

برای نمایش گرافیکی عناصر گرافیکی می توانید از هر روشی استفاده کنید. تا وقتی که روش شما عملی باشد و به درستی کار کند نمره ی این بخش را خواهید گرفت.

می توانید برای عناصر مداری از عکس و یا حتی یک مستطیل با نام عنصر مربوطه استفاده کنید. ولی عناصر باید از هم قابل تفکیک باشند.

برای پیاده سازی این بخش باید علاوه بر امکان انتخاب المان مورد نظر از Node Library، امکان اضافه کردن عناصر با استفاده از کلید میانبر (مثلا R برای مقاومت و ...) را نیز پیاده سازی کنید.

توجه داشته باشید که شماتیک گرافیکی تمام المان هایی که در فاز قبل پیاده سازی کرده اید را باید در این بخش طراحی کنید.

۴.۵ اتصالات

برای اتصال Node ها به یک دیگر باید از المانی به اسم Wire استفاده کنید.

از آنجایی که در صفحه ی Node Shematic ها بو صورت نقطه در Grid مشخص هستند می توانید Node ها را با استفاده از سیم به یکدیگر متصل کنید.

همچنین توجه داشته باشید که امکان اتصال یک سیم به سیم دیگر باید وجود داشته باشد.

۶ پرچسب گزاری گره ها

یکی از ویژگی های بسیار مهم در نرم افزار های تحلیل مدار، امکان برچسب گزاری گره ها است.

برچسب گزاری گره ها به ما این امکان را می دهد که بدون سیم کشی در مدار، یک Node را به یک Node دیگر صرفا با استفاده از Label Name مشترک متصل کنیم.

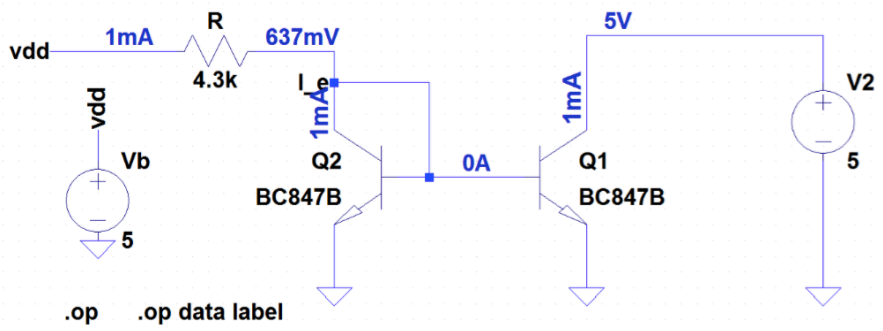
پیاده سازی این بخش را می توانید به ۲ صورت مختلف انجام دهید.

برای راحتی می توانید که Node هایی که اسم مشترک دارند را به هم متصل کنید. این کار می تواند باعث اشتباه در توصیف مدار شود ولی کارساز است.

همچنین برای این کار می توانید که یک المان به نام Node Label را طراحی کرده و با Node های مد نظر خود متصل کنید. سپس هر کدام از Node هایی که به یک Node Label با اسم مشترک متصل هستند را به هم وصل کنید.

در کل می توانید درمورد Label ها مثل یک سیم نامرئی فکر کنید!

در کل نحوه ی پیاده سازی این بخش اهمیتی ندارد و کارکرد آن مورد ارزیابی قرار خواهد گرفت.



شکل ۹: نمونه ای از استفاده از Net Label ها در نرم افزار LTSpice. در اینجا منبع ولتاژ به یک Label با نام V_{dd} متصل شده است. همچنین یکی از Node های موجود در مدار به یک Label با همین نام متصل شده است. بدین ترتیب این دو Node با یک اتصال مجازی که به صورت گرافیکی نمایش داده نمی شوند به هم متصل هستند.

۷ AC Sweep & Phase Sweep

۱.۷ مقدمه

در این فاز از پروژه علاوه بر تحلیل در حوزه ی زمان یا همان Transient که در بخش قبل پیاده سازی کرده اید، از شما انتظار می رود که تحلیل در حوزه ی فرکانس و تحلیل در حوزه ی فاز را در این بخش پیاده سازی کنید.

تحلیل فرکانسی یکی از مهم ترین (حتی شاید مهم ترین) نوع تحلیل مداری است.

این تحلیل به ما اجازه می دهد رفتار سیستم الکتریکی مورد نظر را در فرکانس های مختلف بررسی کنیم. این کار مخصوصا برای تحلیل فیلتر های فرکانسی بسیار مفید می باشد.

۲.۷ AC Sweep

در انجام تحلیل فرکانسی، قصد داریم که اندازه ی ولتاژ و یا جریان مربوط به یک المان و یا Node را در یک فرکانس خاص محاسبه کنیم.

با قرار دادن مقادیر به دست آمده در فرکانس های مختلف می توان به پاسخ فرکانسی مدار دست پیدا کرد.

برای پیاده سازی این نوع تحلیل باید یک منبع ولتاژ با نام AC Voltage طراحی کنید که ورودی $\cos(\omega t)$ را به سیستم الکتریکی مورد نظر ما می دهد.

سپس در تحلیل AC کافی است که فرکانس این ورودی را از ω_{start} تا ω_{stop} تغییر داده و اندازه ی پاسخ مدار (اندازه ی ولتاژ و یا جریان المان و یا Node مد نظر) را در آن فرکانس خاص رسم کنید.

توجه داشته باشید که در این پروژه فقط باید یک منبع AC در مدار وجود داشته باشد و فقط فرکانس مربوط به این منبع تغییر بکند.

بدین ترتیب برای انجام این تحلیل باید پارامتر های زیر را در منوی مربوط به تنظیمات شبیه سازی تعیین کنید.

نماد	توضیح	واحد
ω_{start}	کران پایین بازهٔ فرکانس	$\text{rad}\,\text{s}^{-1}$
ω_{stop}	کران بالا بازهٔ فرکانس	$\text{rad}\,\text{s}^{-1}$
N	تعداد گام‌ها	—

۳.۷ Phase Sweep

در این بخش شما باید تحلیل مدار را در حوزه ی فاز انجام دهید. این تحلیل مانند تحلیل حوزه ی فرکانس است با این تفاوت که در اینجا به جای تغیر فرکانس، فاز را تغییر می دهیم.

در این بخش باید با تعریف المان Phase Voltage یک ولتاژ $\cos(\omega_{base}t + \phi)$ را به سیستم الکتریکی بدهید.

توجه کنید که در اینجا باید پارامتر ω_{base} را نیز مشخص کنید. در این تحلیل فرکانس ثابت بوده و فاز مربوطه تغییر خواهد کرد. بدین ترتیب با محاسبه ی اندازه ی ولتاژ یا جریان المان یا Node مربوطه در فاز مورد نظر تحلیل در حوزه ی فاز را انجام دهید.

نماد	توضیح	واحد
ϕ_{start}	کران پایین بازهٔ فاز	rad
ϕ_{stop}	کران بالا بازهٔ فاز	rad
ω_{base}	فرکانس پایه	$\text{rad}\,\text{s}^{-1}$
N	تعداد گام‌ها	—

۸ منو های برنامه

در این بخش درمورد منوهایی که در این پروژه باید پیاده سازی کنید خواهیم پرداخت.

۱.۸ منوی تنظیمات شبیه سازی

در این منو باید پارامتر های مربوط به شبیه سازی را تعیین کنید.

در این پروژه از شما انتظار می رود که ۳ نوع شبیه سازی را پیاده سازی کنید.

* شبیه سازی در حوزه زمان (Transient)
* شبیه سازی در حوزه فرکانس (AC Sweep)
* شبیه سازی در حوزه فاز (Phase Sweep)

درمورد هرکدام از این شبیه سازی ها پارامتر های زیر در منوی تنظیمات شبیه سازی باید قابل تنظیم و Set کردن باشند.

Transient

Stop Time (t) (1)

Time to start saving data (t_0) (2)

Maximum Timestep (t_{step}) (3)

AC Sweep

Type of sweep (1)

Octave

Decade

Linear

Start Frequency (ω_s) (2)

Stop Frequency (ω_e) (3)

Number of Points (N) (4)

Phase Sweep

Base Frequency (ω_0) (1)

Start Phase (ϕ_s) (2)

Stop Phase (ϕ_e) (3)

Number of Points (N) (4)

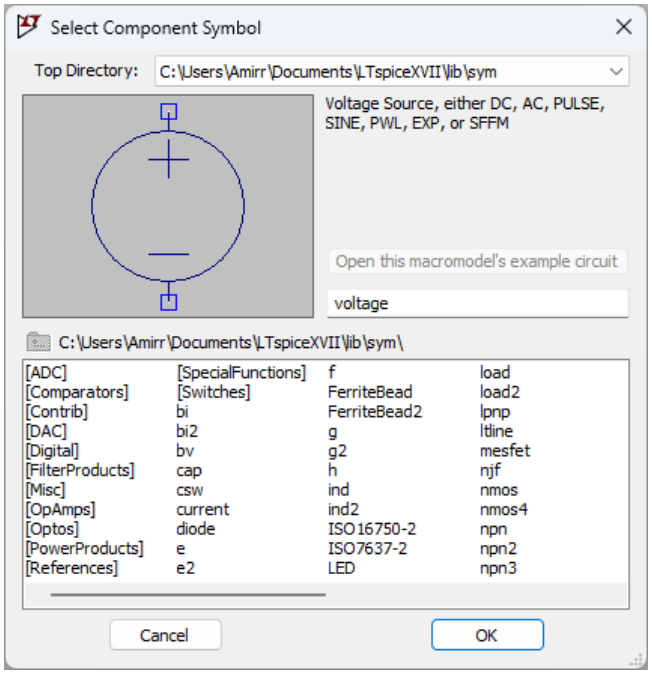
درمورد جزیّات مربوط به شبیه سازی های AC Sweep و Phase Sweep به بخش های مربوطه مراجعه کنید.

۲.۸ منوی Node Library

از آنجایی که در فاز ۱ برای قرار دادن المان های مداری در مدار خود از دستورات متنی استفاده می کردید، در این فاز لازم است که منویی به نام Node Library را پیاده سازی کنید که قابلیت انتخاب و قرار دادن المان های مداری در مدار را به صورت گرافیکی به ما بدهد.

در نتیجه تمام المان هایی که در فاز ۱ پیاده سازی کرده اید را باید در این منو برای انتخاب و جاگذاری قرار دهید.

همچنین المان هایی که خوتان ساخته اید (Subcircuits) هم باید در این منو قابل دسترسی باشند.



شکل ۱۰: نمونه ای از Node Library پیاده سازی شاده در نزم افزار LTSpice

۳.۸ منوی فایل

این منو هم در فاز یک پیاده سازی شده است. شما در این فاز صرفا آن را به صورت گرافیکی پیاده سازی می کنید و Shematic های از قبل ساخته شده را در آن نمایش داده و قابلیت انتخاب آن ها را به این منو اضافه کنید.

همچنین امکان ذخیره سازی پروژه و همچنین ذخیره سازی پروژه به صورت Subcircuit باید در این منو قرار بگیرد.

۴.۸ منوی Library

در اینجا باید Subcircuit های ساخته شده توسط کاربر از `lib` Folder بارگزاری شود و امکان اضافه کردن آن ها به Shematic فراهم شود. (یعنی علاوه بر اضافه کردن Subcircuit ها به Node Library، باید آن ها را به این منو هم اضافه کنید تا از المان های پایه تفکیک شوند.)

۵.۸ منوی Scope

درمورد این منو که برای نمایش گرافیکی سیگنال ها استفاده می شود در بخش نمایش گرافیکی سیگنال ها به صورت مفصل توضیح داده شده است. در نتیجه برای این منو ویژگی های مربوط به آن بخش را پیاده سازی کرده و سیگنال را به صورت گرافیکی نمایش دهید.

۹ ذخیره سازی

۱.۹ مقدمه

از آنجایی که در این پروژه، در جاهای مختلف مانند ذخیره سازی المان های ساخته شده (Subcircuits)، ذخیره سازی پروژه ها و داده های خروجی، ذخیره سازی توصیف گرافیکی مدار و ... نیاز به ذخیره سازی اطلاعات و یا به طور کلی ذخیره سازی Object ها داریم، داشتن یک ساختار مناسب برای ذخیره سازی این اطلاعات کمک شایانی به Handle کردن این اطلاعات می کند.

در نرم افزار های شبیه سازی مدار، این کار به صورت ذخیره یک پروژه در یک Folder با نام مشخص اتفاق می افتد که هرکدام از Field هایی که باید ذخیره شوند (مثلا توصیف گرافیکی مدار، داده های خروجی، المان های ذخیره شده و ...) در یک فایل جداگانه در Folder پروژه قرار می گیرند.

در این پروژه از شما انتظار می رود موارد زیر را ذخیره سازی و سپس بازیابی کنید.

* توصیف گرافیکی مدار
* اطلاعات مربوط به تحلیل مدار (پیاده سازی شده در فاز ۱)
* خروجی های گرفته شده از مدار (کلاس Signal)
* Subcircuit های تعریف شده (ذخیره در Library)

در کل، پروژه باید به نحوی ذخیره سازی شود که اولا توصیف گرافیکی مدار پس از باز کردن دوباره ی پروژه بازیابی شود. همچنین نیازی به حل دوباره عددی و یا پرامتری مدار (تشکیل ماتریس های مربوطه و ...) نباشد و اینکه در صورتی که خروجی ای از قبل گرفته شده بود، در Folder پروژه ذخیره سازی شود.

همچنین با ساخت یک فولدر به نام Library در یک مکان مشخص، Subcircuit های تعریف شده را نیز به صورت فایل های جداگانه ذخیره کنید.

۲.۹ رویکرد های ذخیره سازی Object ها

برای ذخیره سازی Object ها میتوان رویکرد های مختلفی را در نظر گرفت. درمورد این رویکرد ها یک Trade-off ای بین راحتی پیاده سازی آن رویکرد و Performance آن هم از نظر زمانی و هم از نظر حافظه وجود دارد.

در این پروژه برای اجتناب از پیچیدگی های بی مورد و راحتی کار Performance رویکردی که در پیش میگیریم را درنظر نگرفته و راحتی پیاده سازی Protocol مربوطه را در الویت قرار می دهیم.

رویکرد پیشنهادی طراحان پروژه برای ذخیره سازی Object ها در این پروژه استفاده از Object Serialization است که در بخش بعد به صورت مفصل درمورد آن توضیح خواهیم داد.

۳.۹ Object Serialization

یکی از رویکرد هایی که برای ذخیره سازی Object ها می توان در نظر گرفت ذخیره کردن Field ها و ویژگی های هر Object در یک فایل با فرمت مشخص است.

برای مثال الگوریتم ذخیره سازی به روش json به همین صورت عمل می کند.

برای مثال ذخیره سازی پارامتر های مربوط به یک مدار مغناطیسی در فرمت json بدین صورت خواهد بود.

```
{
  "circuit": {
    "coil1": {
      "turns": 100,
      "current_A": 5.2
    },
    "coil2": {
      "turns": 100,
      "current_A": 0.0
    },
    "air_gap_mm": 0.1,
    "inductance": {
      "self_inductance_H": 05.0,
      "mutual_inductance_H": 045.0,
      "coupling_coefficient": 9.0
    }
  }
}
```

این دویکرد دو ایراد اساسی دارد.

(1) درصورتی که در Object ذخیره شده، Object دیگری وجود داشته باشد، این قضیه پیچیدگی ذخیره و بازیابی Object را به این صورت بسیار زیاد می کند و همچنین احتمال خطا در این کار بسیار بالا می رود.

(2) از آنجایی که Design مربوط به هرکلاس متفاوت است برای ذخیره سازی هر کلاس باید یک فرمت جداگانه و منحصر به فردی را در نظر بگیریم. این کار سرعت Developement پروژه را به شدت کاهش می دهد.

در نتیجه روشی که برای ذخیره سازی هر Object ای یکسان باشد بسیار کارآمد بود و سرعت Developement پروژه را بسیار افزایش می دهد و از پیچیدگی های بی مورد در طراحی جلوگیری می کند.

برای حل این مشکل و یکپارچه سازی مسئله روش پیشنهادی ما استفاده از Object Serialization است.

در این روش اطلاعات (Object ها) به شکل و Format ای در می آیند که برای انتقال و ذخیره سازی راحت باشند. به این تبدیل Serialization می گوییم که از این روش معمولا برای انتقال اطلاعات در یک شبکه استفاده می شود.

پس از انتقال داده ی Serialize شده باید این داده به فرمت اصلی خودش بازگردد. به این کار De-serialization می گوییم.

Object ها مانند هر Datatype دیگری در کامپیوتر به صورت یک رشته از 0 ها و 1 ها در Memory ذخیره می شوند. ما با ذخیره کردن این اطلاعات باینری و تبدیل دوباذه آن ها به Object عمل Serialization را انجام می دهیم.

...	0x0003	0x0002	0x0001	0x0000	Address
...	0xa2	0xef	0xcd	0xab	Value

جدول ۱: ساختار فرضی اطلاعات در Memory

Serialization در C++ باید به صورت دستی و Manual انجام شود و مانند Java و یا Python امکان انجام این کار به صورت خودکار وجود

ندارد.

اما از آنجایی که Manual Serialization فقط برای Object هایی قایل انجام است که POD-Type هستند (یعنی Object دیگری در خود ندارند!) باید از Library های آماده و روش های خودکار استفاده کینم.

برای انجام این کار، بهتر است که یک Method مربوط به Serialize، ذخیره کردن و خواندن از Database در کلاس مربوط به Database تعریف کنید. از آنجایی که Serialization و De-serialization برای همه ی Object ها یکسان است می توانید این توابع را به صورت Generic تعریف کنید.

در ادامه یک ساختار کلی برای انجام این کار داده می شود. پیاده سازی دقیق این بخش را باید خوتان با توجه به ساختار پروژه ی خودتان انجام دهید.

۴.۹ کتابخانه ی Cereal

شکل ۱۱: *جالب است بدانید کلمه ی Cereal به معنای غلات صبحانه می باشد. ولی از آنجایی که تلفظ آن مشابه کلمه ی Serial در Serialization می باشد، به عنوان اسم این کتابخانه انتخاب شده است!*

کتابخانه ی Cereal یک کتابخانه ی جدید و Modern است که برای ذخیره سازی Object ها در Format های مختلف مثل Binary, JSON, XML, ... کاربرد دارد.

در این پروژه ما قصد داریم که Object ها را به صورت Binary در یک فایل باینری (`.bin`) ذخیره کنیم.

فایل های باینری `.bin`، اطلاعات را به صورت باینری خالص و Human Unreadable ذخیره می کنند. این کار حجم فایل را تا حد خوبی نسبت به یک فایل `.txt` کاهش می دهد.

برای دانلود و استفاده از این کتابخانه از این لینک GitHub اقدام کنید.

GitHub

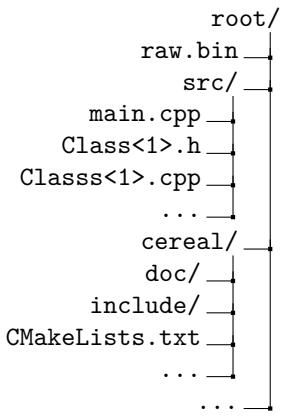
می توانید فایل مربوط به کتابخانه را به صورت یک فایل `.zip` دانلود کنید و یا کل Repository مربوطه را در Repository مربوط به پروژه تان Clone کنید.

همچنین لینک مربوط به Documentation این کتابخانه و قطعه کد های مربوط به راه اندازی آن در همین صفحه ی GitHub موجود است. البته که پیشنهاد می شود برای این کار از قطعه کد های موجود در همین فایل استفاده کنید.

۵.۹ نصب و راه اندازی Cereal

برای نصب و راه اندازی این کتابخانه، از آنجایی که پروژه ی ما نسبتا کوچک است نیازی به استفاده از Package Manager ها نداریم و می توانیم فایل `zip`. مربوط به کتابخانه را به صورت مستقیم از صفحه ی GitHub مربوط به آن دانلود کرده و در پروژه ی خود قرار دهیم.

برای این کار در Root اصلی پروژه، دو Folder به نام های `src` و `lib` قرار دهید. `src` Folder شامل Source Code اصلی برنامه خواهد بود و `lib` Folder شامل کتابخانه هایی که نصب کرده ایم. (البته که می توانید `lib` Folder را تشکیل نداده و Directory مربوط به کتابخانه ی Cereal را در Root اصلی پروژه قرار دهید.) توجه داشته باشید که پس از دانلود فایل `zip`.، فولدری به نام `mastercereal`- در آن وجود دارد. نام این Folder را به `cereal` تغییر داده و سپس در Main Directory پروژه قرار دهید.



حال در Source Code پروژه و با استفاده از Relative Address مربوط به Header File هایی که از این کتابخانه نیاز داریم، Header های مورد نیاز را در برنامه Include می کنیم.

پس از نصب کتابخانه، می توانید به این صورت آن را به Include Directories خود در CMakesList اضافه کنید. (این کار Folder مربوط به این کتابخانه را به Path مربوطه اضافه می کند و به ما این اجازه را می دهد که با استفاده از (<>|Diamond Operator)، Header File مربوطه را `include` کنیم.

```
include_directories(${CMAKE_SOURCE_DIR}/cereal/include)
```

بدین ترتیب شما با موفقیت کتابخانه ی Cereal را نصب و راه اندازی کرده اید!

۶.۹ Serialization and De-Serialization

برای درک روند Serialization و De-Serialization در این کتابخانه، به قطعه کد زیر توجه کنید.

در اینجا از Nested Object ای به نام `Person` استفاده شده است که یک Object از کلاس `Address` در خود دارد. در اینجا سعی داریم که یک Object از کلاس `Person` را Serialize و سپس De-Serialize کنیم.

```cpp
#include <cereal/archives/binary.hpp>
#include <cereal/types/string.hpp>
#include <fstream>
#include <iostream>
#include <string>

class Address {
public:
    std::string city;
    int zip;

    template <class Archive>
    void serialize(Archive& ar) {
        ar(city, zip);
    }
};

class Person {
public:
    std::string name;
    int age;
    Address address;

    template <class Archive>
    void serialize(Archive& ar) {
        ar(name, age, address);
    }
};
```

در اینجا برای هر کدام از کلاس ها یک تابع `serialize()` تعریف می کنیم و Field هایی از کلاس را که قصد داریم ذخیره کنیم (که درمورد پروژه ی شما، همه ی Attribute ها است!) را به عنوان ورودی `archive` قرار می دهیم.

کلاس `Archive` در اینجا مثل یک مخزن و Container برای داده عمل می کند. از آنجایی که این کلاس یک کلاس Abstract است نمی توان آن را Instantiate کرده و باید از یکی از Derived Class های مربوط به آن استفاده کنیم.

در اینجا چون قصد داریم داده ها را به صورت باینری ذخیره کنیم، از کلاس های `BinaryOutputArchive` و `BinaryInputArchive` که کلاس های فرزند کلاس `Archive` هستند استفاده می کنیم.

توابع `serialize(Archive& ar)` تعریف شده در هر کلاس به صورت خودکار (Implicit) توسط `Archive` Call خواهند شد.

حال برای Serialize کردن Object بدین صورت عمل می کنیم.

```cpp
int main() {
    Person p1{"Ali", 20, {"Tehran", 10001}};

    {
        std::ofstream ofs("../raw.bin", std::ios::binary);
        cereal::BinaryOutputArchive archive(ofs);
        archive(p1);
        ofs.flush();
    }

    Person p2;
    {
        std::ifstream ifs("../raw.bin", std::ios::binary);
        cereal::BinaryInputArchive archive(ifs);
        archive(p2);
    }

    std::cout << " "Name: << p2.name << "\n" Age: << p2.age
              << "\n" City: << p2.address.city << "\n" ZIP: << p2.address.zip << '\n';

    return 0;
}
```

در کل، این کتابخانه هیچ کار خاصی انجام نمی دهد! تنها کاری که این کتابخانه می کند این است که کار ما را برای Serilaize کردن Object ها، مخصوصا Nested Objects بسیار راحتتر می کند.

در اصل این کتابخانه با تبدیل Pointer اشاره کننده به Object به یک `char*` Pointer و یا `void*`، Object مورد نظر را به صورت بایت به بایت می خواند و در یک فایل باینری Write می کند. (البته این کتابخانه، این کار را به صورت بسیار Efficient تری انجام می دهد.)

(*) کمی ماجراجویی!

اگر کمی فضولیتان گُل کرد (!) و خواستید ببینید که این Data به چه صورت در فایل باینری ذخیره می شود، می توانید از یک HEX Converter استفاده کنید تا بتوانید فایل باینری را به صورت Human Readable در آورده و محتوای آن را مشاهده کنید.
در اینجا برای این کار از ابزار `xxd` در Linux استفاده شده است.

ورودی

```
xxd -b raw.bin
```

خروجی

```
00000000: 00000011 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 01000001 01101100 01101001 00010100  ..Ali.
0000000c: 00000000 00000000 00000000 00000110 00000000 00000000  ......
00000012: 00000000 00000000 00000000 00000000 00000000 01010100  .....T
00000018: 01100101 01101000 01110010 01100001 01101110 00010001  ehran.
0000001e: 00100111 00000000 00000000                             ...
```

در اینجا می توان به وضوح مشاهده کرد که این کتابخانه تمامی Attribute های مربوط به کلاس در حال Serialize شدن را به شکل باینری در آورده و به صورت متوالی در یک رشته قرار می دهد. (می توانید ASCII Code هایی که مربوط به کاراکتر های موجود در String هستند را مشاهده کنید!) البته که انجام این کار توسط این کتابخانه بسیار Efficient تر بوده و Nested Object ها را هم به صورت خودکار Handle می کند.

۷.۹ Polymorphism

تا اینجای کار روند Object Serialization به خوبی پیش رفته است. ما توانسته ایم Object های مختلف و یا حتی Nested Obejct ها را Serialize کنیم.

اما آخرین چالشی که باید درمورد این موضوع حل کنیم، مسّله ی Polymorphism است.

مشکلی که وجود دارد این است که کتابخانه ی Cereal برای اینکه از Class Hiearchy برنامه آگاهی داشته باشد و بداند که کلاس ها به چه صورت باهم ارتباط دارند، نیاز دارد که این ارتباطات به صورت دستی (Explicit) تعریف شوند.

در حقیقت در C++، با وجود اینکه RTTI (Run-Time Type Information) وجود دارد (Type یک Object و یا Data Type دیگر می تواند در زمان اجرای برنامه Evaluate (Runtime) شود. همین قضیه باعث می شود که بتوانید از auto Keyword برای Type متغیر ها استفاده کنید.) ارتباط بین کلاس ها برای کتابخانه ی مد نظر ما نا مشخص است.

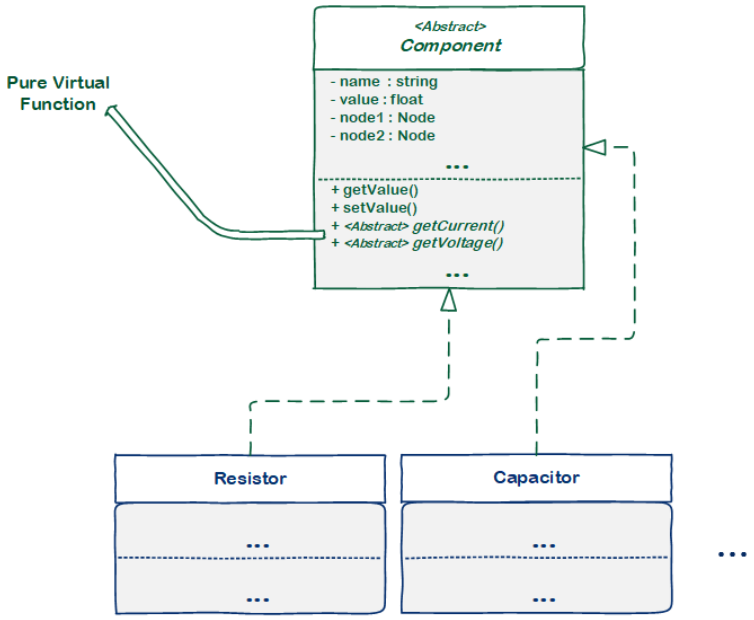
بدین ترتیب می توانیم به این صورت کلاس ها را و روابط بین آن ها را Register کنیم.

برای مثال در این مورد، کلاس های Circle و Rectangle از کلاس Shape Derive شده اند. بدین ترتیب برای Register کردن کلاس ها و روابط بین آن ها باید از Built-in Macro های زیر استفاده کنیم.

```
CEREAL_REGISTER_TYPE(Circle);
CEREAL_REGISTER_TYPE(Rectangle);
CEREAL_REGISTER_POLYMORPHIC_RELATION(Shape, Circle);
CEREAL_REGISTER_POLYMORPHIC_RELATION(Shape, Rectangle);
```

در صورتی که این ارتباطات به صورت Explicit تعریف نشوند، در Serialize کردن Object به مشکلی نخواهیم خورد، ولی برای De-Serialize کردن آن Object از فایل باینری، به خطا خواهیم خورد.

اما درمورد پروژه ی شما اگر این مسّله را درست Handle نکنید، یکی از مواردی که در مورد آن قطعا به مشکل خواهید خورد المان های مداری است. شما برای طراحی المان های مداری نیاز به استفاده از کلاس Abstract Component دارید که کلاس های مربوط به المان های مختلف مداری مثل مقاومت، خازن و ... از آن Derive شوند. در اینجا شما به وضوح در حال استفاده از اصل Polymorphism هستید. چرا که تمامی این المان ها دارای ولتاژ و جریان هستند و تنها تفاوت آن ها نحوه ی رفتار آن ها درمورد این دو Attribute است.



شکل ۱۲: نمونه ای از استفاده در Polymorphism در طراحی المان های مداری

۱۰ Networking (*)

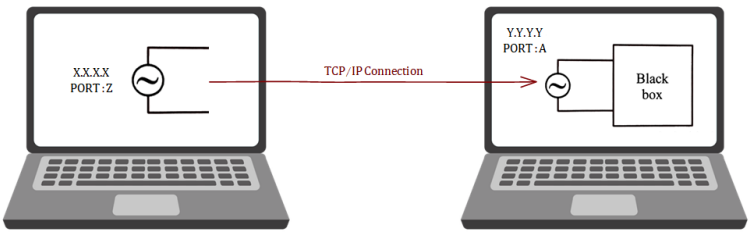
توجه داشته باشید که انجام این بخش برای گروه های ۲ نفره شامل نمره ی امتیازی شده و برای گروه های ۳ نفره اجباری می باشد.

نرم‌افزاری که طراحی می کنید، نه تنها باید مدار را بفهمد، بلکه باید بتواند با بقیه ی لپ تاپ‌ها ارتباط برقرار کند! چیزی فراتر از یک تحلیل، نزدیک به یک جور معاشرت مهندسی!

۱.۱۰ مقدمه

در این بخش از پروژه قرار است تحلیل مدار های الکتریکی را فراتر از صفحه لپ تاپ هایتان برده و وارد فاز Network کنید.

از آنجایی که Networking جز اهداف درس نیست، در این بخش از پروژه از شما انتظار نمی رود که مسلّل پیچیده ی مرتبط با Networking را حل و پچپچیاده سازی کنید. در این بخش از شما می خواهیم که یک نوع Wireless از یک منبع ولتاژ را پیدا سازی کنید! بدین معنا که با قرار دادن یک Instance از آن منبع ولتاژ در یک لپ تاپ، بتوان ولتاژ آن را به Instance دیگر این منبع ولتاژ در لپ تاپ دیگر انتقال داد.

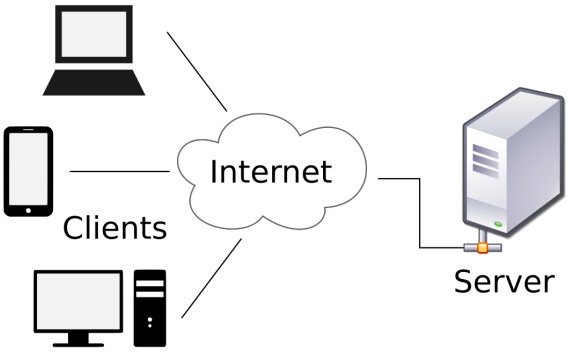


شکل ۱۳: نمایش گرافیکی مدلی که باید پیاده سازی کنید.

۲.۱۰ معماری Network

برای انجام این بخش، نیاز به یک دانش حداقلی از نحوه ی کار اینترنت و در حالت کلی Computer Networks خواهید داشت. برای انجام این بخش از پروژه به اطلاعات اضافه ای نیاز نخواهید داشت و هرآنچه که لازم دارید در این سند برای شما فراهم شده است.

معماری ای که برای طراحی Computer Network خود در این پروژه استفاده می کنیم، معماری Server-Client نام دارد. در این معماری Server المان (Object!) سرویس دهنده بوده و Client المان متقاضی سرویس می باشد.



شکل ۱۴: معماری Server-Client

برای درک بهتر این معماری به این مثال توجه کنید.

فرض کنید که تصمیم دارید که برای دوست خود (یک Client دیگر) در یک نرم افزار پیام رسان یک پیام ارسال کنید. برای این کار درخواست خود را از طریق اینترنت به Server ارسال می کند. سپس Server این درخواست را Handle کرده و با توجه به Protocol مختص خود پیام شما را برای دوستتان ارسال می کند.

ارسال اطلاعات بدون استفاده از Server نیازمند معماری های دیگری مانند معماری Peer-to-Peer می باشد که از موضوع این بخش خارج است. شما برای طراحی Network نیاز به یک المان کنترل کننده و Centeralized خواهید داشت که Server نام دارد.

نکته ی مهم درمورد این پروژه این است که شما نیازی به Handle کردن چند Client به صورت همزمان ندارید. در اصل یکی از Machine ها به عنوان Server و دیگری به عنوان Client خواهد بود و انتقال اطلاعات صرفا بین Server و Client اتفاق می افتد. در نتیجه نیازی به Handle کردن چند Client نخواهید داشت. (نکته ی مهم درمورد Handle کردن Client ها این است که در این بخش از پروژه، Server شما منتظر درخواست Connection می ماند و وقتی این درخواست را دریافت کرد، همان درخواست را Handle می کند و منتظر درخواست های جدید نمی ماند. برای Handle کردن چند درخواست نیاز به استفاده از Multithreading است بدین معنا که یکی از Thread ها درخواست فعلی را Handle کرده و یک Thread دیگر منتظر درخواست های جدید می ماند.)

بدین ترتیب ساز و کار Server و Client بدین صورت می باشد.

(Ⅱ) **Server**

---

(1) منتظر ماندن Listening برای درخواست Connection

(2) Connection کردن handle

(3) دریافت و ارسال داده

(4) بستن Connection

(Ⅲ) **Client**

---

(1) ارسال درخواست Connectino

(2) دریافت و ارسال داده

(3) بستن Connection

در این بخش از پروژه، نیازی به دانستن ساختار و معماری خود اینترنت ندارید.

۳.۱۰ IP Address

برای پیدا کردن هرکدام از Node ها در یک شبکه، نیازمند آدرس دهی هر کدام از Node ها هستیم. به این آدرس IP Address می گویند.

در حال حاضر دو نوع از IP Address ها وجود دارد. IP-V4 که نسخه ی ۴ بایتی این مدل آدرس دهی است (هرکدام از آدرس ها ۴ بایت هستند) و IP-V6 که نسخه ی ۶ بایتی است.

$$IPV4 \Rightarrow X.X.X.X$$

$$IPV6 \Rightarrow X.X.X.X.X.X$$

توجه داشته باشید که Server هم مانند Client یک Machine و یکی از Node های شبکه است که برای ارسال و دریافت داده ها، نیازمند داشتن IP Address است.

۴.۱۰ DNS Servers

اما اگر برای اتصال به Server ها نیازمند IP Address آن ها هستیم، چگونه با استفاده از Domain Address مثلا Google.com به IP Address متناسب با آن Server متصل می شویم؟

در اینجا Domain Name Systems (DNS) ها به میان می آیند. کار این System این است که Domain Name ها را به IP Address متناسب با آن ها وصل می کند. مانند یک Map ولی بسیار پیچیده تر!

۵.۱۰ PORTs

اولین جایی در کامپیوتر شما که پاسخ Server مبتنی بر درخواست شما را دریافت می کند سیستم عامل شماست.

این قضیه از این جهت اهمیت دارد که کامپیوتر شما در هر لحظه ممکن است چندین پاسخ از Server های مختلف را دریافت کند. تشخیص اینکه کدام یک از این پاسخ ها مربوط به کدام یک از نرم افزار های موجود در کامپیوتر شماست وظیفه ی سیستم عامل است.

سیستم عامل برای انجام اینکار از یک سیستم Internal Addressing در خود کامپیوتر استفاده می کند بدین معنی که به هر کدام از نرم افزار های شما (Process) که از سیستم عامل این درخواست را داشته، یک PORT Number نسبت می دهد که مثل یک آدرس Local عمل می کند.

وقتی که یک درخواست را به Server ارسال می کنید در حقیقت IP Address و PORT Number مربوطه را هم همراه با داده های مورد نظر ارسال می کنید و Server در هنگام ارسال پاسخ با استفاده از IP Address ارسال شده توسط شما، پاسخ را برای شما ارسال کرده و با قرار دادن PORT Number ارسال شده توسط شما، به سیستم عامل این امکان را می دهد که پاسخ را برای Process مورد نظر ارسال کند.

۶.۱۰ Sockets

Socket ها یک مدل برای نرم افزار ها و Process های شما هستند که به یک Server متصل شده و Connection را برقرار کرده و به انتقال داده می پردازند.

در اصل تمامی مواردی که در قبل به آن ها اشاره شد مانند برقراری Connection و ... توسط Handle Socket می شود.

به Socket ها مانند یک Object نگاه کنید که با گرفتن IP Address مربوط به Server ارتباط را برقرار کرده و داده ها را انتقال می دهند. همچنین Server با دریافت همان Socket پاسخ ها را ارسال می کند.

البته که در این پروژه نیازی به انتقال داده به صورت ۲ طرفه وجود ندارد بلکه صرفا یکی از طرفین اطلاعات مربوط به ولتاژ را به طرف دیگر ارسال می کند.

مثال

برای اینکه مفاهیمی که تا اینجای کار آموخته اید را مرور کنیم، به این مثال توجه کنید.

فرض کنید که قصد دارید وبسایت دانشکده را برروی مروگر خود Load کنید.

در ابتدا با تایپ کردن Domain Name مربوط به وبسایت دانشکده، مروگر با جستجو در DNS Server ها، IP Address متناظر با آن Domain Name را می کند. سپس یک Connection از نوع TCP/IP بین Machine شما و آن Server ایجاد می کند. پس از ایجاد این Connection یک درخواست با استفاده از پروتوکل HTTP به Server ارسال کرده و پاسخ Server را که به صورت یک پاسخ HTTP است برروی صفحه ی مروگر شما نمایش می دهد.

در حقیقت، شما با نوشتن `http://` در قبل از Domain Name مربوط به یک وبسایت، مشخص می کنید که قصد دارید یک درخواست HTTP به آن Server بدهید.

(*) کمی ماجراجویی!

سعی کنید با استفاده از IP Address مربوط به سیستم ایمیل دانشکده، وارد صفحه ی Login ایمیل دانشکده شوید.

IP Address مرتبط با این Server 81.31.170.116 می باشد.

با تایپ کردن `http://81.116.170.31` در مروگر خود، یک درخواست HTTP به Server ارسال کرده و سپس پاسخ Server در مروگر شما نمایش داده خواهد شد.

(IP Address مربوط به Server ایمیل دانشکده را از طریق وبسایت هایی که امکان جستجو در DNS Server ها را می دهد پیدا کرده ایم!)

در صورتی که خواستید IP Address مربوط به وبسایت های دیگر را نیز پیدا کنید، می توانید از دستور `nslookup` در CMD استفاده کنید.

```
$> nslookup ee.sharif.ir

Server:  UnKnown
Address:  172.20.10.1

Non-authoritative answer:
Name:    ee.sharif.ir
Address:  81.31.170.116
```

۷.۱۰ استفاده از Serialization

همانگونه که در بخش مربوط به Serialization ذکر شد، یکی از کاربرد های بسیار مهم Serialization استفاده ی آن در Network های کامپیوتری برای انتقال داده است.

در این بخش از پروژه می توانید برای انتقال سیگنال مربوط به ولتاژ منبع ولتاژ بی سیم، یک Object از جنس Signal را Serialize کرده، به کامپیوتر دیگر ارسال کنید و Data ارسال شده را در کامپیوتر دیگر De-Serialize کنید.

۸.۱۰ Socket Programming in C++

در این بخش به موضوع Socket Programming به صورت ویژه در زبان برنامه نویسی C++ می پردازیم.

توجه داشته باشید که قطعه کدی که در ادامه می بینید، مربوط به سیستم عامل Windows می باشد. بدین ترتیب برای Windows این Header File ها را Include می کنیم.

```
#include <winsock2.h>
```

در ابتدا باید Class مربوط به TCP Server را تعریف کنیم.

```
class TCPServer {
public:
    TCPServer(int port);
    ~TCPServer();
    void start();

private:
    int port;
    SOCKET serverSocket, clientSocket;
    void handleClient();
};
```

این مدل مربوط به PORT Number Server ای که از سیستم عامل دریافت کرده است را نگه داشته و Socket های مربوط به Server و Client را در خود نگه می دارد.

همچنین `start()` Method Server را شروع کرده و منتظر درخواست از طرف Client می ماند.

پس از برقراری ارتباط `handleConnection()` Method اجرا می شود.

در حقیقت تنها کاری که Server Socket انجام می دهد این است که منتظر یک درخواست Connection می ماند. پس از برقراری ارتباط تمامی انتقال داده ها توسط Client Socket انجام می شود.

```
"ws2_32.lib") ,comment(lib #pragma

TCPServer::TCPServer(int port) {
    this >- port = port;
    this >- serverSocket = INVALID_SOCKET;
    this >- clientSocket = INVALID_SOCKET;
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);
}

void TCPServer::start() {
    sockaddr_in serverAddr{};
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(port);

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    bind(serverSocket, (sockaddr*)&serverAddr, sizeof(serverAddr));
    listen(serverSocket, SOMAXCONN);

    std::cout << " port on listening "Server << port << "...\n";

    sockaddr_in clientAddr{};
    int clientSize = sizeof(clientAddr);
    clientSocket = accept(serverSocket, (sockaddr*)&clientAddr, &clientSize);

    if (clientSocket != INVALID_SOCKET) {
        handleClient();
    }
}

void TCPServer::handleClient() {
    char buffer[1024] = {};
    recv(clientSocket, buffer, sizeof(buffer), 0);
    std::cout << " "Received: << buffer << "\n";

    std::string reply = server!" from "Hello;
    send(clientSocket, reply.c_str(), reply.size(), 0);
}
```

در ادامه توضیح مختصری درمورد قطعه کد نوشته شده خواهیم داد.

تابع | start()

Data Structure sockaddr_in یک ساختمان داده برای نگه داری IP Address می باشد. AF_INET مشخص کننده ی نسخه ی Addressing استفاده شده می باشد. (IP-V4) همچنین در دو خط بعدی PORT Number گرفته شده از سیستم عامل و IP Address مشخص می شود. (IP Address همین سیستم داده می شود.)

تا اینجا آدرس منطبق با Server Socket را مشخص کرده ایم.

در ادامه Server عملیات Listening را انجام می دهد بدین معنا که منتظر درخواست Connection می ماند.

سپس با توجه به درخواست های وارد شده، اگر درخواست Valid ای مطابق با برقراری Connection وجود داشت، Connection برقرار شده و Construct Client Socket می شود.

برای انتقال داده در تابع handleClient() باید از یک Buffer استفاده کنیم. Buffer ها ساختمان داده هایی هستند که برای انتقال داده در سطح شبکه استفاده می شوند.

بدین ترتیب لازم است که سایز Buffer را به نحو مناسبی تعیین کنید. در این مورد در ادامه به صورت مفصل صحبت خواهیم کرد.

حال کلاس `Client` را بررسی می کنیم.

```
class TCPClient {
public:
    TCPClient(const std::string& ip, int port);
    ~TCPClient();
    void sendMessage(const std::string& message);

private:
    std::string serverIP;
    int port;
};

```

این کلاس نیازی به Handle کردن Connection ندارد. این کار وظیفه ی Server می باشد.

```
"ws2_32.lib") ,comment(lib #pragma

TCPClient::TCPClient(const std::string& ip, int port) {
    this >- serverIP = ip;
    this >- port = port;
    WSADATA wsaData;
    WSAStartup(MAKEWORD(2, 2), &wsaData);
}

void TCPClient::sendMessage(const std::string& message) {
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in serverAddr{};
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(port);
    inet_pton(AF_INET, serverIP.c_str(), &serverAddr.sin_addr);

    if (connect(sock, (sockaddr*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        std::cerr << failed. "Connection\n";
        closesocket(sock);
        return;
    }

    send(sock, message.c_str(), message.size(), 0);
    char buffer[1024] = {};
    recv(sock, buffer, sizeof(buffer), 0);
    std::cout << " says: "Server << buffer << "\n";

    closesocket(sock);
}

```

تمام توضیحات مربوط به این کلاس، مشابه کلاس `TCPServer` می باشد.

همچنین توجه داشته باشید IP Address و PORT Number ای که در `Client` مشخص می کنید، مربوط به Server است. IP Address مربوط به `Client` برای دریافت پاسخ Server توسط خود Socket ارسال می شود.

۹.۱۰ ارسال داده

برای ارسال داده می توانید Approach های مختلفی را در نظر بگیرید.

یکی از این Approach ها Serialize کردن Object مربوط به سیگنال و ارسال آن توسط شبکه است.

اما مشکل بزرگی که این Approach دارد این است که به دلیل حجم بالای داده ها، انتقال داده نسبتا وقت گیر خواهد بود.

برای مثال فرض کنید که قرار است یک سیگنال با $F_s = 1000$ و $t = 2s$ را ارسال کنیم. بدین ترتیب Object مربوط به سیگنال ما اگر Attribute های دیگر این Object را در نظر نگیریم، به اندازه ی $1000 \times 2 \times S$ داده ارسال خواهد شد.

اگر اعداد ولتاژ ارسال شده از جنس `float` باشند، در نتیجه $2000 \times S = 2000 \times 4 = 8kB$ یا ۸ کیلوبایت داده ارسال خواهد شد که می تواند چالش برانگیز باشد.

البته که پیاده سازی به این روش مشکلی ندارد و مشمول نمره ی این بخش خواهد شد، هرچند که روش بهتر این است که داده ها به صورت تکه تکه ارسال شده (منظور از داده ها، ولتاژ ارسالی در طول زمان می باشد.) و این داده ها به صورت Process Real-Time شوند. همین Protocol انتقال داده است که باعث می شود شما بتوانید یک ویدیو را به صورت آنلاین و Real-Time تماشا کنید!

۱۰.۱۰ پیدا کردن IP Address دستگاه

پس از پیاده سازی این بخش، برای اتصال به کامپیوتر هم گروهیتان به عنوان Server نیازمند داشتن IP Address مرتبط با آن دستگاه و PORT Number ای که Server از سیستم عامل گرفته است هستید.

PORT Number به صورت دستی مشخص شده و به سادگی می توانید برای آن یک مقدار مشخص قرار دهید، ولی IP Address یک آدرس از قبل مشخص شده می باشد.

بدین ترتیب برای پیدا کردن IP Address مربوط به دستگاهی که به عنوان Server کار می کند، می توانید از دستور `ipconfig` در CMD ویندوز استفاده کنید.

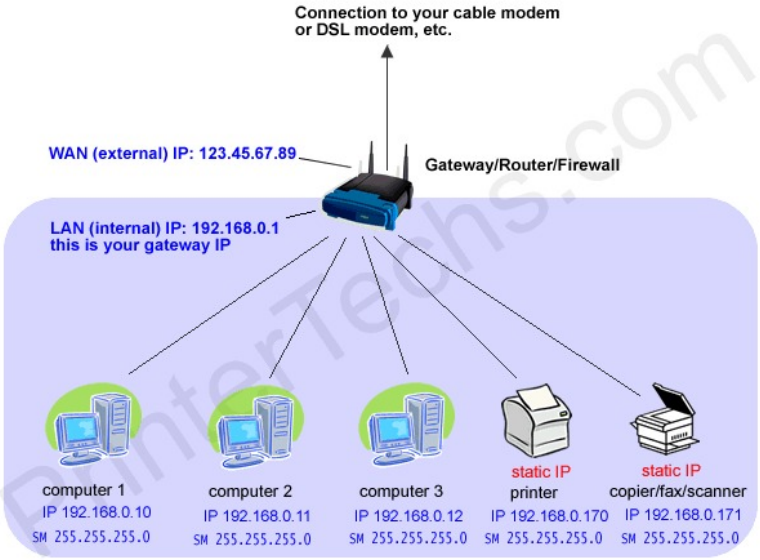
در ادامه IP Address مربوط به Wireless LAN Adapter را خوانده و به عنوان Server IP Address قرار دهید.

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : sharif.ir
   Link-local IPv6 Address . . . . . : XXXX::XXXX:XXXX:XXXX:XXXX
   IPv4 Address. . . . . . . . . . . : X.X.X.X
   Subnet Mask . . . . . . . . . . . : 2550.240.255.
   Default Gateway . . . . . . . . . : X.X.X.X
```

در اینجا برای ایجاد یک TCP Connection، باید از IP-V4 استفاده کنید.

Default Gateway در اصل IP Address مربوط به Router شماست که اطلاعات را به خارج از Local Network منتقل می کند.



شکل ۱۵: Network Diagram

(*) توجه داشته باشید که در این بخش از پروژه، شما قرار است ۲ دستگاه که به یک Local Network متصل هستند را به هم وصل کنید. بدین ترتیب باید از Local IP دستگاه ها برای انجام این کار استفاده کنید. یعنی حتی اگر مودم شما به اینترنت متصل نباشد، دو کامپیوتر از طریق این Local Network به هم متصل شده و اطلاعات را جا به جا می کنند.

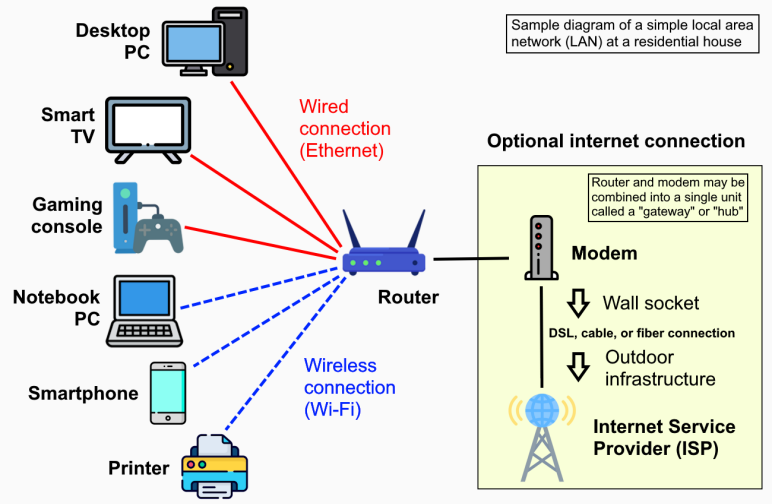
Local IP به فرمت $192.168.X.X$ **می باشد.**

(*) پی نوشت

توجه داشته باشید که برای اتصال به کامپیوتر هم گروهیتان باید به یک Local Network متصل باشید. (یعنی برای مثال به یک Router و یا Modem متصل باشید.) در غیر این صورت امکان انتقال داده تحت Local Network را نداشته و باید انتقال اطلاعات را با استفاده از شبکه ی WAN انجام دهید که باعث پیچیدگی های بی مورد خواهد شد.

همچنین پیدا کردن Public Address نیز به سادگی پیدا کردن Local Address نمی باشد، چرا که باید Public Address مودم را که توسط شرکت تامین کننده ی اینترنت شما (ISP) تعیین شده است را در تنظیمات مودم پیدا کنید.

انگار انتقال اطلاعات در یک شبکه ی داخلی صورت می گیرد که با اینترنت و شبکه ی بیرون هیچ ارتباطی ندارد. به این نوع از شبکه ها LAN (Local Area Network) گفته می شود. همچین شبکه ای می تواند به اینترنت متصل باشد و یا نباشد. (گاهی برای موارد امنیتی، توصیه می شود که شبکه های داخلی تا حد امکان به اینترنت متصل نباشند.)



شکل ۱۶: LAN Diagram

۱۱.۱۰ اتصال منبع ولتاژ از طریق شبکه

در ورژن ساده استفاده از شبکه از شما میخواهیم تا صرفا یک نوع از منبع ولتاي را در یکی از سیستم های خود و مدار را در سیستم دیگر پیاده سازی کنید و منبع تان را از طریق شبکه به سیستم مداری تان انتقال داده و شبیه سازی را انجام دهید.

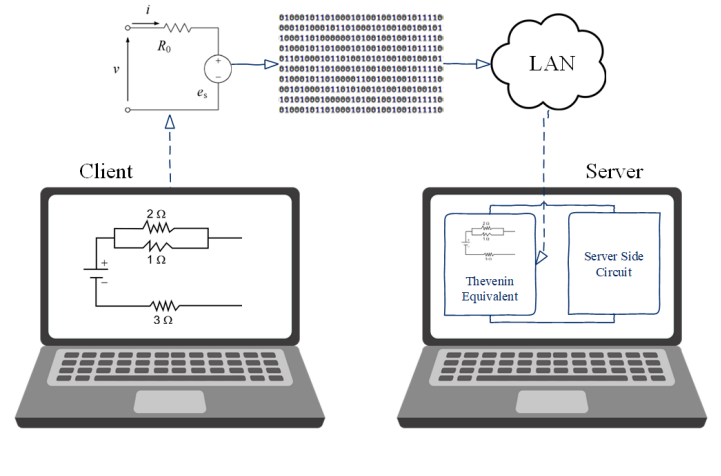
۱۲.۱۰ متصل کردن ۲ مدار از طریق شبکه

پس از پیاده سازی مدار معادل تونن و نورتن در بخش مربوطه، می توانید تک قطبی تولید شده را از طریق شبکه انتقال داده و به یک تک قطبی دیگر متصل کنید. بدین ترتیب شما یک مدار کامل خواهید داشت و می توانید آن را در یکی از طرفین تحلیل کنید!

برای این کار کافی است که مدار معادل تونن و یا نورتن سمت Client را Serialize کرده و از طریق شبکه به Server منتقل کنید، سپس پس از De-Serialize کردن اطلاعات ارسال شده Subcircuit را به مدار سمت Server متصل کرده و آن را تحلیل کنید.

انتقال مدار معادل تونن یا نورتن باعث می شود حجم داده ی منتقل شده توسط شبکه به صورت چشم گیری کاهش پیدا کند.

توجه داشته باشید که همانطور که در بخش مربوط به مدار معادل تونن نورتن توضیح داده شد برای این بخش از پروژه از شما انتظار داریم که فقط مدار معادل تونن یا نورتن مدار های مقاومتی را محاسبه کنید. بدین ترتیب در سمت CLient تنها مدار های مقاومتی برای اتصال به سمت Server قرار داده خواهد شد.



شکل ۱۷: نمایه ی کلی ای از نحوه ی انتقال Subcircuit تحت شبکه