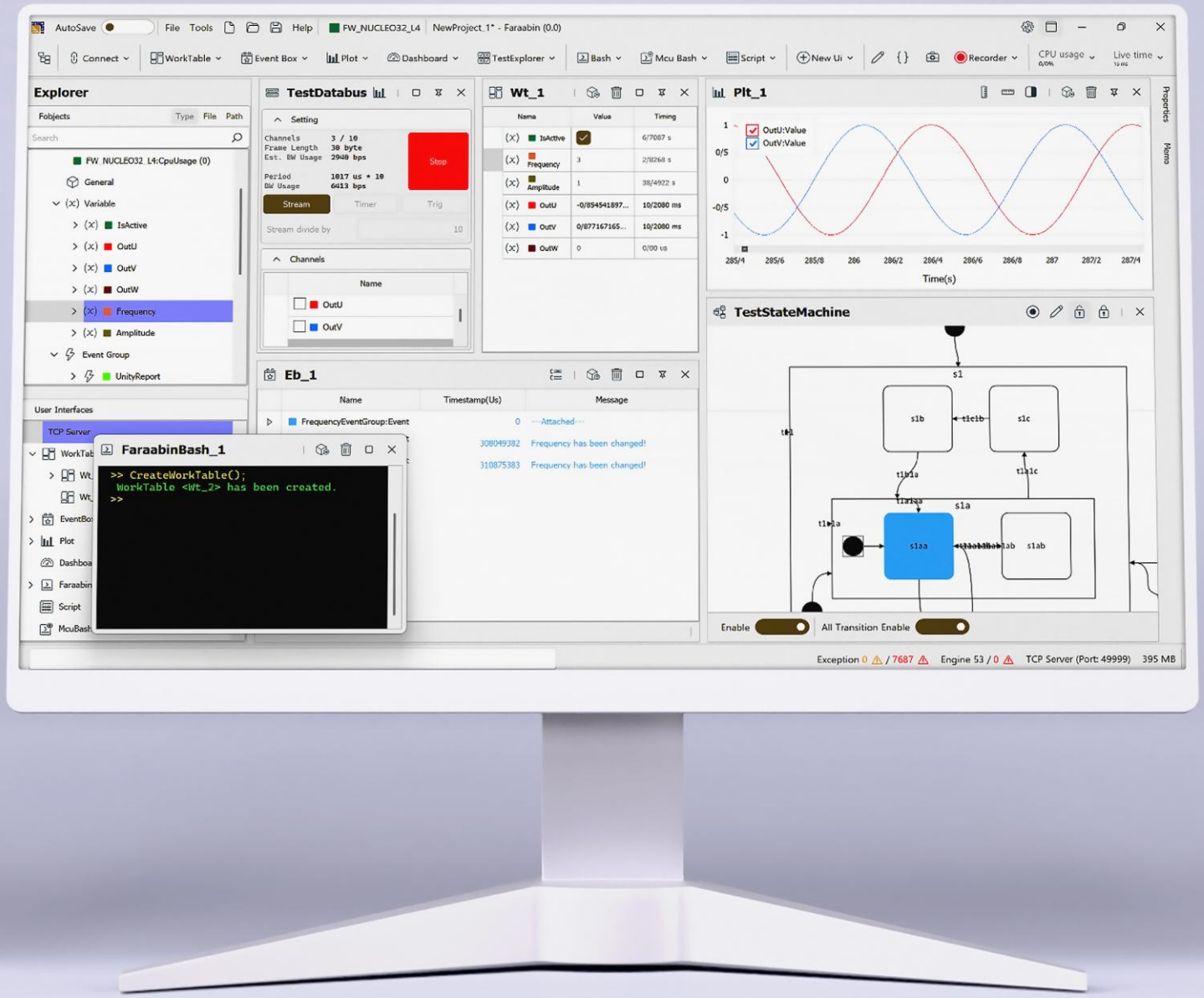
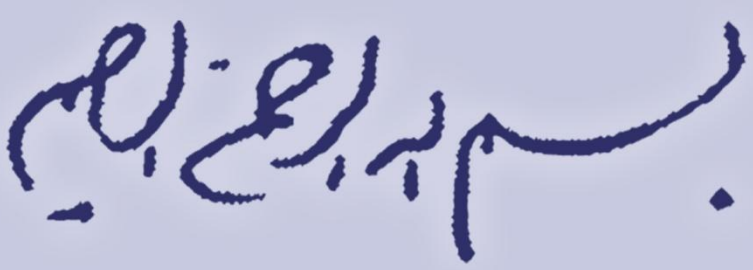
فرابین

باچشمانی بـــاز
تـــــوسعه دهید





فرآیند عیب‌یابی نرم‌افزار نهفته (Embedded Software) پیچیده، زمان‌بر و طاقت‌فرسا است. به‌ویژه هنگامی که دید کاملی از آنچه در داخل پردازشگر اتفاق می‌افتد، وجود ندارد. اینکه در هر لحظه مقادیر متغیرهای داخل کد چه هستند، چه رویدادهایی در حال وقوع هستند، زمان‌بندی اجرای بخش‌های مختلف کد چگونه است و اینکه بتوان این اطلاعات را همزمان و با زمان‌بندی دقیق مشاهده کرد، نیازی اساسی برای هر برنامه‌نویس نرم‌افزار نهفته است.

آیا تا به حال به این فکر کرده‌اید که روش‌های مرسوم پایش و عیب‌یابی نرم‌افزار نهفته، چقدر محدود و ناکافی هستند؟

فرابین محصول تلاش برنامه‌نویسانی است که راضی به عیب‌یابی سیستم‌های نهفته از طریق حدس زدن نبوده‌اند. آن‌ها می‌خواهند در هر لحظه بدانند داخل پردازشگر چه می‌گذرد و کدی که نوشته‌اند چگونه رفتار می‌کند. همچنین، نمی‌خواهند برای کوچکترین تغییری در رفتار نرم‌افزار، مجبور باشند کد را تغییر دهند.

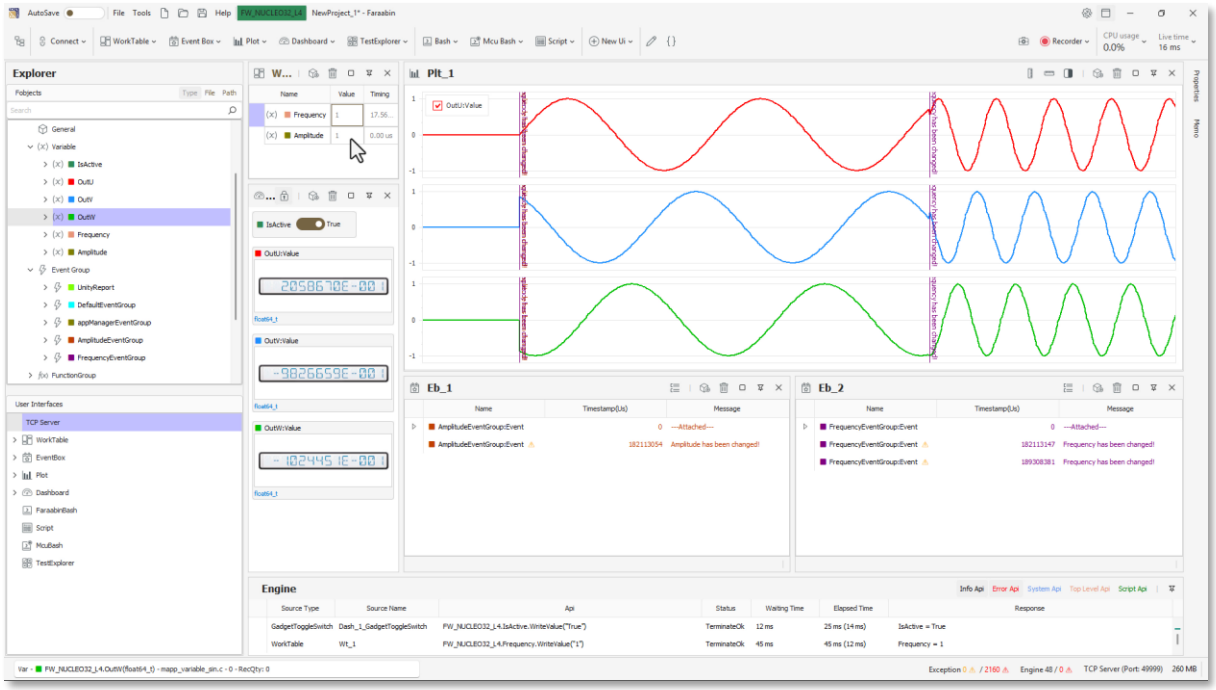
این مهندسین با استفاده از اصل پایش نرم‌افزاری، ابزاری را برای خود توسعه داده‌اند که نیازهایشان را برطرف کند. به مرور زمان و با افزایش پیچیدگی پروژه‌های انجام شده توسط این تیم، این ابزار نیز تکامل یافته و قابلیت‌های متنوعی به آن اضافه شده است.

گستردگی استفاده از فرابین در بیش از ۱۰ صنعت متفاوت و بیش از ۴۰۰ محصول و همچنین استقبال مهندسین و کارفرمایانی که با استفاده از فرابین، دیگر نمی‌خواستند به روش‌های سنتی برگردند، سبب شد فرابین با هدف کمک به فعالین حوزه سیستم‌های نهفته، برای عرضه عمومی بازطراحی شود.

فرابین از ابتدا بر اساس نیاز و خواست برنامه‌نویسان نهفته توسعه داده شده است. در نتیجه، قابلیت‌ها و قالب رابط کاربری آن به گونه‌ای طراحی شده که پاسخگوی نیازهای این مهندسان باشد.

مرور اجمالی

فرابین ابزاری است برای مشاهده و کنترل رفتار نرم‌افزار نهفته در زمان اجرا بر روی پردازشگر. فرابین با استفاده از یک پورت ارتباطی و بدون نیاز به سخت‌افزار جانبی (مانند پروگرامر و دیباگر)، دسترسی کاملی را جهت مشاهده و کنترل آنچه در داخل نرم‌افزار نهفته در جریان است به کاربر می‌دهد.



با استفاده از فرابین تمام جنبه‌های مختلف نرم‌افزار نهفته در زمان اجرا قابل مشاهده، پایش و کنترل است:

- لایه‌های پایین (درایورنویسی امکانات پردازشگر و قطعات بورد)
- بخش‌های منطقی لایه کاربرد مانند ماشین حالت، تشخیص رویداد و ...
- بخش‌های محاسباتی لایه کاربرد مانند حلقه‌های کنترلی، محاسبات، فیلترها و ...
- زمان‌بندی اجرای بخش‌های مختلف کد مانند توابع، اینتراپت‌ها و تسک‌ها

فرابین اطلاعات و رویدادهای مربوط به کد کاربر را با برچسب زمانی دقیق (در حد میکروثانیه) و با استفاده از طیف گسترده‌ای از قالب‌های متنی و گرافیکی برای کاربر نمایش می‌دهد.

با استفاده از فرابین :

- می‌توانیم فرآیند تست و عیب‌یابی را با چشمان باز و مشاهده آنچه در حال وقوع است، انجام دهیم.
- می‌توانیم رفتار نرم‌افزار نهفته را به صورت برخط تغییر دهیم.
- می‌توانیم منشأ مشکلات نرم‌افزار را ببینیم. (به جای آنکه حدس بزنیم)
- می‌توانیم توسعه نرم‌افزار را با سرعت بیشتر و هزینه کمتر به پایان برسانیم.
- و بالاخره می‌توانیم تعداد زیادی از ایرادات و خطاها را در همان ابتدا، کشف و اصلاح کرده و کیفیت نرم‌افزار را افزایش دهیم.

کاربردها

- مشاهده، تحلیل و صحت‌سنجی رفتار نرم‌افزار نهفته
- توسعه نرم‌افزار نهفته در تمام سطوح، از درایورنویسی تا کاربرد و سناریوی عملکرد
- پایش و عیب‌یابی نرم‌افزار نهفته
- دسترسی به سیستم نهفته از راه دور
- تست دستی و اتوماتیک نرم‌افزار نهفته
- ذخیره رفتار نرم‌افزار نهفته در طول زمان
- ساخت رابط کاربری سازمانی برای سطوح مختلف بوسیله رابط کاربری فرابین یا انواع زبان‌های برنامه‌نویسی به صورت اپلیکیشن و تحت وب

مزایای استفاده

- کاهش زمان عیب‌یابی و در نتیجه افزایش سرعت توسعه
- افزایش سرعت کشف ایراد و علت آن و در نتیجه افزایش کیفیت نرم‌افزار
- دیدن منشأ ایراد به‌جای حدس زدن
- ساده و سریع شدن طراحی رابط کاربری سازمانی برای محصولات
- مشاهده گرافیکی رفتار سطح بالای سیستم
- امکان تنظیم برخط حلقه‌های کنترلی با مشاهده برخط تمام سیگنال‌های موجود
- صحت‌سنجی طراحی سناریو عملکرد و پیاده‌سازی آن
- بی‌نیازی از ابزارهای ابتدایی مشاهده‌پذیری مانند printf بر روی پورت سریال، اعمال سیگنال‌های داخلی بر روی DAC یا استفاده از GPIO جهت مشاهده زمان‌بندی بخشی از کد

مخاطبان

- برنامه‌نویس نهفته
- طراح سناریوی عملکرد نرم‌افزار نهفته (مهندس سیستم، مهندس کنترل، مهندس هوافضا و . . .)
- طراح و اپراتور تست نرم‌افزار نهفته در سطوح مختلف
- کاربر نهایی محصول

مشاهده‌پذیری، پایش و عیب‌یابی

مشاهده‌پذیری یکی از مفاهیم اصلی برای بررسی صحت عملکرد یک سیستم و همچنین عیب‌یابی آن است. مشاهده‌پذیری به معنی دسترسی به اطلاعات بیشتر از داخل سیستم است تا درک عمیق‌تر از عملکرد بخش‌های داخلی و همچنین پیدا کردن منشأ رفتارهای نامطلوب آن به دست آید. در زندگی روزمره، از مفهوم مشاهده‌پذیری استفاده‌های متعددی می‌شود. مثلا:

- دیاگ زدن خودرو یا باز کردن درب کاپوت خودرو و رفتن زیر خودرو توسط مکانیک
- تجویز آزمایش و عکس توسط پزشک
- استفاده از مولتی‌متر و اسیلوسکوپ برای مشاهده سیگنال بخش‌های مختلف سخت‌افزار

در تمام این موارد، با استفاده از ابزار مناسب، اطلاعات بیشتری از داخل سیستم استخراج شده و با مشاهده و تحلیل این اطلاعات، رفتار و مشکلات داخلی سیستم نمایان می‌شود.

مشاهده‌پذیری در نرم‌افزار نهفته از اهمیت بیشتری برخوردار است. زیرا:

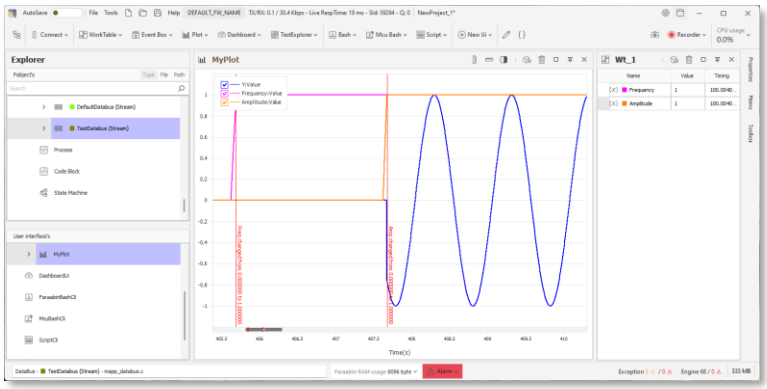
- نرم‌افزار دارای ساختاری پیچیده و چند وجهی است و تمام بخش‌های آن بر روی هم تاثیرگذارند.
- در سیستم‌هایی که توسط نرم‌افزار نهفته کنترل می‌شوند، قابلیت اطمینان اهمیت بالایی دارد.
- دسترسی به رفتار داخلی نرم‌افزار نهفته در حین عملکرد عادی آن سخت است و زیرساخت مناسبی بدین منظور وجود ندارد.

موارد زیر، بخشی از تلاش برنامه‌نویسان نهفته جهت مشاهده رفتار داخلی نرم‌افزار است که اگرچه در برخی موارد راهگشا هستند، ولی به دلیل محدودیت‌هایی که در حجم اطلاعات ارسالی، نرخ ارسال، قالب نمایش، یکپارچه نبودن، نیاز به ابزارهای سخت‌افزاری جانبی و ... دارند، هرگز امکان یک مشاهده‌پذیری جامع را فراهم نمی‌کنند:

- دیباگ سخت‌افزاری (بوسیله اتصال دیباگر به پردازشگر)
- استفاده از یک یا چند LED برای نمایش برخی رویدادها و شرایط داخلی سیستم
- استفاده از مبدل دیجیتال به آنالوگ (DAC) برای مشاهده مقدار متغیر نرم‌افزار با سرعت بالا (این کار بین مهندسان کنترل مرسوم بوده و برای مشاهده سیگنال‌های داخلی مانند خطای حلقه کنترلی استفاده می‌شود. برخی پردازشگرها تعداد محدودی کانال DAC دارند که در صورت آزاد بودن، بدین منظور استفاده می‌شوند).
- استفاده از پین‌های پردازشگر (GPIO) و تغییر وضعیت آن در بخش‌های مختلف کد و مشاهده سیگنال تولیدی در اسیلوسکوپ یا لاجیک آنالایزر.
- ارسال پیام متنی در بخش‌های مختلف کد با استفاده از دستور printf بر روی یک پورت سریال و مشاهده پیام‌ها بر روی ترمینال در کامپیوتر.

کنترل‌پذیری در نقطه مقابل مشاهده‌پذیری قرار دارد. بدین معنی که کاربر امکان ایجاد تغییر در رفتار و پارامترهای داخلی نرم‌افزار را در حین عملکرد عادی آن داشته باشد. این قابلیت به‌خصوص در زمان تست نرم‌افزار و سیستم‌ها کاربرد فراوانی دارد. روش‌های فوق در این زمینه نیز ناکارآمد هستند.

فرابین



فرابین ابزاری است برای ایجاد مشاهده‌پذیری و کنترل‌پذیری در نرم‌افزار نهفته، در حین عملکرد عادی آن، بدون استفاده از سخت‌افزار جانبی و بدون وابستگی به کد کاربر.

فرابین پنجره‌ای بر روی پردازشگر می‌گشاید که از طریق آن کاربر می‌تواند اطلاعات و رویدادهای داخل نرم‌افزار را به صورت برخط، با برچسب زمانی دقیق، در قالب‌های بصری گوناگون مشاهده کند و از طرف دیگر، تغییراتی در رفتار و پارامترهای نرم‌افزار نهفته ایجاد نماید.

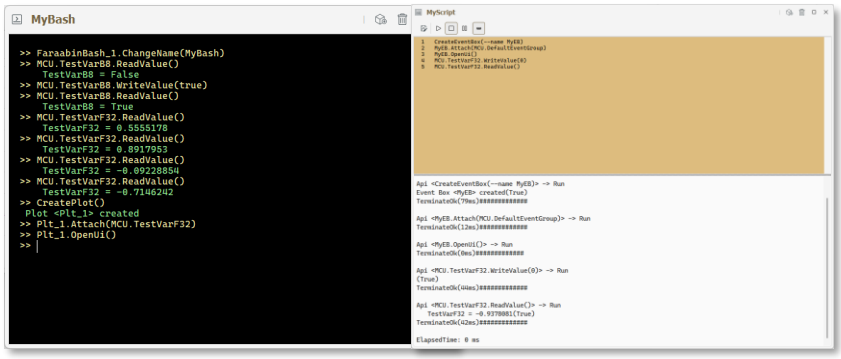
فرابین به وسیله پورت‌های ارتباطی مرسومی که در سخت‌افزار نهفته استفاده می‌شوند، می‌تواند پردازشگر را به کامپیوتر متصل کرده و از طریق ابزارهای موجود در رابط کاربری خود، رفتار، سیگنال‌ها، پیام‌ها و اطلاعات سیستم نهفته را نمایش داده و یا فرامین دریافتی از طرف کاربر را به کد منتقل کند.

فرابین با هدف کمک به فعالین حوزه سیستم‌های نهفته توسعه پیدا کرده است. مشکلاتی که کاربران این حوزه با آن درگیر بودند، راه‌حل‌های ابتدایی که استفاده می‌شد و عیوبی که این راه‌حل‌ها داشتند، زمینه‌ساز توسعه فرابین شد. به دلیل آنکه فرابین از ابتدا برای کدنویسان توسعه داده شده و قابلیت‌های آن مطابق با نیازهای آنان و بنا به درخواست آن‌ها توسعه یافته است، قابلیت‌های عمومی که به صورت متداول در این حوزه‌ها مورد نیاز است، در آن موجود است. علاوه بر آن، قابلیت‌ها و ابزارهای ویژه‌تری که برای تولید نرم‌افزار نهفته مورد نیاز است، در آن تعبیه شده است. ساختار فرابین به گونه‌ای توسعه یافته است که اضافه کردن ویژگی‌ها و قابلیت‌های جدید به سهولت و سرعت انجام می‌شود.

فرابین با کنار زدن محدودیت‌های موجود در روش‌های مرسوم:

- امکان مشاهده تعداد نامحدودی از سیگنال‌های سیستم را فراهم کرده است.
- رویدادهای رخ داده در نرم‌افزار را نمایش می‌دهد.
- زمان‌بندی اجرای نرم‌افزار را اندازه‌گیری کرده و نمایش می‌دهد.
- امکان مشاهده و تغییر در رفتارهای سطح بالای سیستم مانند ماشین حالت را فراهم کرده.

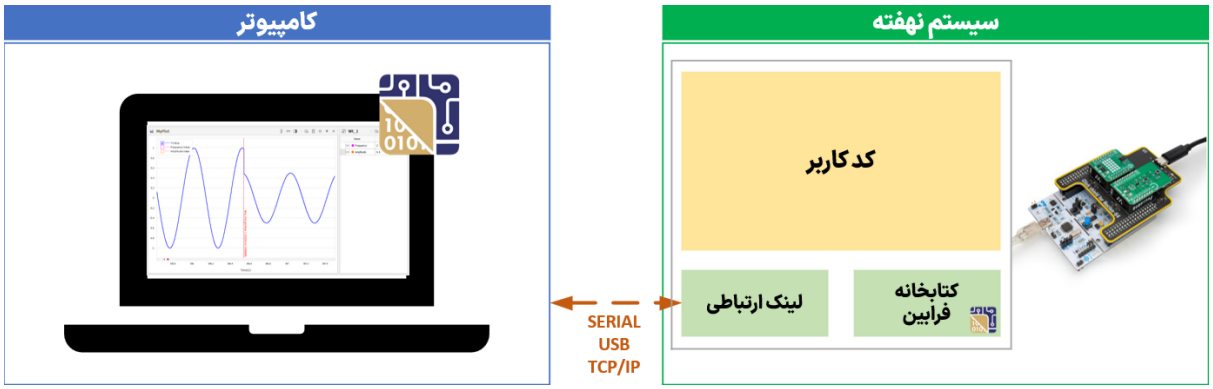
و تمام این کارها را با برچسب زمانی دقیق انجام می‌دهد.



فرابین با قابلیت‌هایی مانند بش، اسکریپت‌نویسی و پشتیبانی از کتابخانه‌هایی مانند Unity، امکانات وسیعی را جهت طراحی، مدیریت و اجرای تست‌های نرم‌افزار نهفته از سطح یونیت تست تا سطح سیستم را به صورت دستی و اتوماتیک فراهم می‌کند.

ساختار فرابین

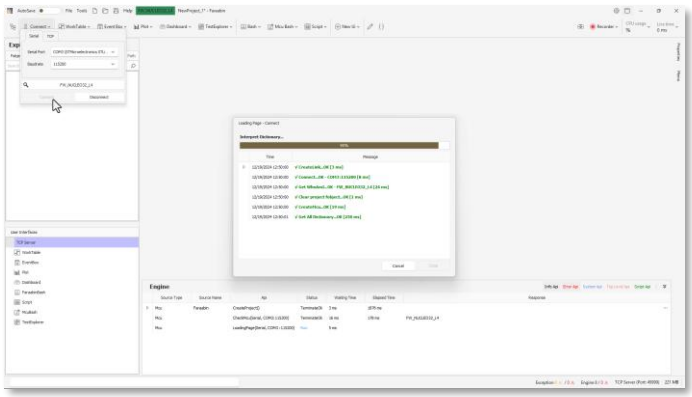
فرابین از دو بخش کتابخانه و نرم‌افزار تشکیل شده است.



کتابخانه به زبان C نوشته شده و در پروژه‌های C/C++ قابل استفاده است. این کتابخانه در کنار پروژه کاربر قرار گرفته و از طریق لینک ارتباطی سریال، اترنت (یا WiFi) و USB (در مد VCP) به نرم‌افزار کامپیوتر متصل می‌شود. کاربر بوسیله توابع و ماکروهایی که کتابخانه در اختیارش قرار می‌دهد، می‌تواند بخش‌های مختلف کد مانند متغیرها، رویدادها، پیام‌ها، توابع، زمان‌بندی‌های بخش‌های مختلف کد و حجم اِشغال شده از زمان پردازشگر را به تفکیک بخش‌های مختلف، به فرابین معرفی کرده و از طریق نرم‌افزار به آن‌ها دسترسی داشته باشد.

```
faraabin.c
src > library > sw_faraabin > lib > faraabin.c > ...
80   /**
84    *
85    * @return result Can be one of FARAABIN_RET va
86    */
87   uint8_t fFaraabin_Init(void) {
88
89     if(fFaraabinLinkSerializer_Init(fFaraabin_Get
90       return FARAABIN_SERIALIZER_INIT_FAILED;
91     }
92
93     if(fFaraabinLinkHandler_Init(fFaraabin_GetRxBu
94       return FARAABIN_LINK_HANDLER_INIT_FAILED;
95     }
96
```

نرم‌افزار در ویندوز اجرا می‌شود و با برقراری ارتباط با کد کاربر، رفتارهای داخلی آن را نمایش داده و امکان تغییر در رفتار آن را فراهم می‌کند.



نرم‌افزار با دارا بودن رابط‌های متنی مانند بش و اسکریپت، رابط‌های گرافیکی مانند جدول، نمودار و دیاگرام و همچنین امکان ایجاد رابط‌های کاربری اختصاصی، محیط منعطف و قدرتمندی را جهت برقراری ارتباط با کد کاربر فراهم می‌کند .

ویژگی‌ها

- قابل استفاده در پروژه های C و C++
- مستقل از نوع پردازشگر نهفته و IDE مورد استفاده
- دارای کتابخانه منطبق بر MISRA-C
- کمترین تاثیر در روند اجرای کد کاربر
- کمترین استفاده از منابع پردازشگر نهفته
- دارای نرم‌افزار رابط کاربری در ویندوز
- بروزرسانی و توسعه سریع و پیوسته قابلیت‌ها

قابلیت‌های پایه

- اتصال کامپیوتر به پردازشگر
  - اتصال به پردازشگر از طریق یکی از پورت‌های سریال، اترنت (یا WiFi) و USB (در مد VCP)
  - قابلیت اتصال به پردازشگر در حین عملکرد عادی
  - قابلیت تعریف رمز جهت دسترسی ایمن
- اطلاعات قابل دسترس
  - خواندن برخط مقدار متغیرهای موجود در نرم‌افزار (با نرخ قابل تنظیم) و نوشتن بر روی آن‌ها
  - امکان دسترسی به متغیرها با سرعت بیش از پهنای باند لینک ارتباطی به صورت برون‌خط
  - پشتیبانی از متغیرهای struct و enum
  - مشاهده پیام‌های کاربر و رویدادهای رخ داده
  - تزریق رویداد یا ارسال پیام به نرم‌افزار کاربر
  - ثبت اطلاعات و رویدادها با برچسب زمانی (در حد میکروثانیه)
  - امکان ایجاد تعداد نامحدودی ترمینال مجازی بین پردازشگر و کامپیوتر (که هر یک می‌توانند به صورت یک پورت سریال مجازی برای انتقال داده استفاده شوند)
- رابط کاربری
  - انواع رابط‌های گرافیکی (نمودار، جدول، داشبورد، دیاگرام و . . .) و متنی (CLI) جهت دسته‌بندی، مدیریت و نمایش اطلاعات
  - امکان تفسیر و نمایش قالب‌های مختلف ارسال پیام مانند ANSI Code و HTML Tag
  - قابلیت نمایش اطلاعات مختلف نسبت به یکدیگر با مقیاس زمانی یکسان جهت مقایسه و تحلیل رفتار بخش‌های مختلف نرم‌افزار نسبت به هم
  - ذخیره تمام اطلاعات، سیگنال‌ها و رخدادهای نرم‌افزار نهفته
  - قابلیت اسکریپت‌نویسی
  - قابلیت شخصی‌سازی رابط کاربری
  - قابلیت ذخیره پروژه شخصی‌سازی شده و استفاده مجدد
  - نمایش خطاهای کاربر در استفاده از کتابخانه فرابین در کد خود

قابلیت‌های پیشرفته

- دسترسی به توابع کد کاربر به صورت رابط متنی (CLI)
- پشتیبانی از کتابخانه ماشین حالت تک‌سطحی (FSM) و چندسطحی (HSM). نمایش وضعیت ماشین حالت به صورت گرافیکی، تاریخچه رویدادهای ماشین حالت و امکان کنترل کامل آن.
- پشتیبانی از کتابخانه پروفایلر برای نمایش درصد اِشغال پردازشگر به صورت کلی و به تفکیک المان‌های اجرایی (اینتراپت یا تسک). قابلیت سنجش زمان اجرا و دوره اجرای بخش‌های مختلف کد کاربر.
- پشتیبانی از کتابخانه یونیتی (Unity) برای طراحی و پیاده‌سازی یونیت تست و مشاهده گزارش تست در فرابین.
- قابلیت اتصال به نرم‌افزار فرابین از طریق TCP/IPو HTTP جهت دسترسی به میکرو از طریق نرم‌افزارهای ثالث و زبان‌های برنامه‌نویسی دیگر.
- دارای کتابخانه در MATLAB Simulink و پشتیبانی از Embedded Coder.
- قابلیت رمزنگاری شخصی لینک ارتباطی

منابع مصرفی

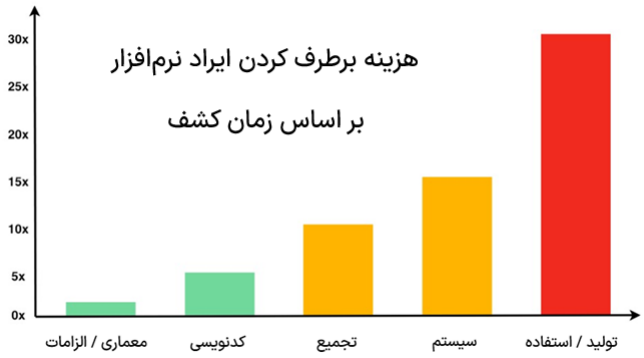
کتابخانه فرابین برای اجرا نیازمند منابع زیر است :

- تیک سخت‌افزاری : به عنوان مرجع زمان‌سنجی ماژول کرونو (معمولا توسط یکی از تایمرهای میکروکنترلر ایجاد می‌شود)
- پورت ارتباطی
- حافظه دیتای پردازشگر (RAM) : برای بافر ارسال و همچنین برای هر یک از پروب‌های نرم‌افزاری تعریف شده توسط کاربر. میزان رم مصرف شده قابل مدیریت توسط کاربر است. در نرم‌افزار فرابین، آمار دقیق و تفکیکی رم استفاده شده نمایش داده می‌شود.
- حافظه کد پردازشگر (FLASH) : در تنظیمات کتابخانه امکاناتی برای غیرفعال‌سازی بخش‌های استفاده نشده جهت مدیریت حجم کد وجود دارد.
- زمان پردازشگر : فرابین تابعی دارد که باید به صورت دوره‌ای فراخوانی شود. معمولا این تابع در پایین‌ترین اولویت نرم‌افزار کاربر فراخوانی می‌شود. بنابراین از زمان‌های خالی پردازشگر استفاده می‌کند و تاثیر حداقلی در زمان‌بندی کد کاربر دارد. API هایی هم که کاربر در متن کد خود فراخوانی می‌کند، به تناسب، زمان کمی برای اجرا نیاز دارند که در مستندات اشاره شده است.

وابستگی‌ها

کتابخانه فرابین برای اجرا به کتابخانه کرونو نیاز دارد. این کتابخانه در فولدر فرابین موجود است.

پیدا کردن زودهنگام اشتباهات



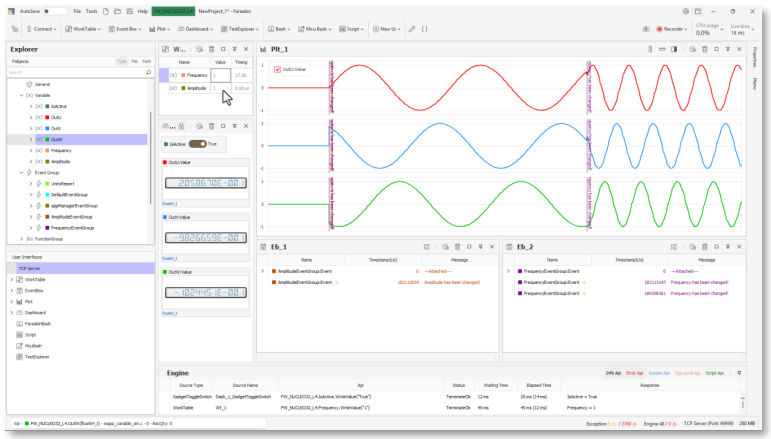
دلیل استفاده از ابزارهای مشاهده‌پذیری و کنترل‌پذیری از اولین مراحل کدنویسی این است که به برنامه‌نویسان کمک می‌کند تا ایرادات را همان زمان که ایجاد می‌شوند، تشخیص دهند. هزینه تشخیص و برطرف کردن ایرادهای نرم‌افزار با گذشت زمان به‌طور نمایی افزایش می‌یابد.

اکثر کاربران منتظر وقوع مشکل می‌مانند تا شروع به ردیابی کنند، اما اگر برنامه‌تان را به‌طور دوره‌ای در طول توسعه ردیابی کنید، بلافاصله رفتارهای عجیب یا اشتباهات را در طی پیاده‌سازی پیدا خواهید کرد. در نتیجه، زمان کمتری صرف اشکال‌زدایی خواهد شد که به زمان کمتر توسعه و هزینه‌های کمتر منجر می‌شود.

در مرحله کدنویسی ماژول‌ها که هنوز حجم کدها کم است، نوشتن تست و پیدا کردن ایرادها راحت‌تر است، چون منشا مشکل مشاهده شده، در حد همان ماژول است. با گذشت زمان و افزایش حجم کدها و تعامل بین ماژول‌ها، ایرادهای کوچک در یک ماژول می‌توانند به اَشکال دیگری ظاهر شوند. این موضوع، یافتن منشأ مشکل را دشوار می‌کند. با تجمیع ماژول‌های بیشتر، پیدا کردن منشأ خطاها پیچیده‌تر می‌شود.

پس از تولید و تحویل سیستم به مشتری، دسترسی به نرم‌افزار و مشاهده وضعیت آن در زمان خطا به‌طور قابل‌توجهی محدود می‌شود و زمان و هزینه کشف و رفع ایراد به‌صورت نمایی افزایش می‌یابد. علاوه بر این، در چنین شرایطی هزینه‌های اعتباری و آبرویی نیز به هزینه‌های قبلی اضافه می‌شود.

بر این اساس، اگر از همان مراحل اولیه کدنویسی با دقت نظارت کنیم که چه اتفاقی در حال رخ دادن است، ایرادها پیش از تبدیل شدن به باگ‌های سطح سیستم، کشف و برطرف می‌شوند. "توسعه تست‌محور" یکی از روش‌هایی است که دقیقاً همین هدف را دنبال می‌کند و در توسعه نرم‌افزارهای نهفته استفاده می‌شود.



فرابین از اولین روزهای کدنویسی تا زمان تست سیستم و حتی پس از تحویل محصول به مشتری، زیرساختی فراهم می‌کند برای مشاهده آنچه در درون نرم‌افزار می‌گذرد. به تبع آن، زیرساختی برای پیدا کردن منشأ مشکلات برای برنامه‌نویسان، مهندسان سیستم، تست‌کننده‌ها و تیم خدمات پس از فروش فراهم می‌کند.

درک روند اجرای برنامه با مشاهده رویدادها

روند اجرای برنامه با مشاهده رویدادها قابل درک و صحت‌سنجی است.

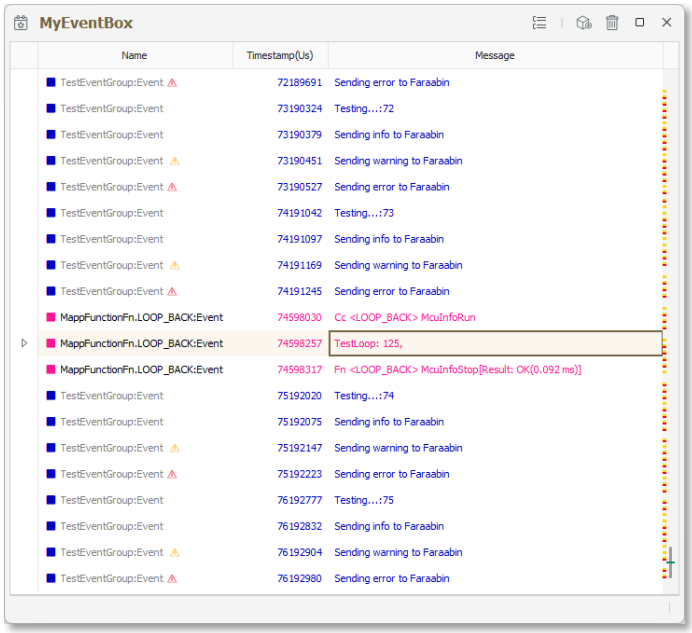
نرم‌افزار معمولا از یک سری الگوریتم، فلوچارت، ماشین حالت و . . . تشکیل شده که عمدتا رویدادمحور هستند. به بیان دیگر، بخش مهمی از رفتار یک نرم‌افزار نهفته شامل رویدادها و پاسخ نرم‌افزار به آن‌ها است. بنابراین بدیهی است که مشاهده روند رویدادها درک خوبی از رفتار نرم‌افزار در حال اجرا می‌دهد.

مواردی مانند :

- اجرای یک تابع یا ورود و خروج آن
- گذر برنامه از نقطه‌ای خاص
- نتیجه اجرای بخشی از کد
- روند بالا آمدن برنامه و راه‌اندازی بخش‌های مختلف سخت‌افزار و نرم‌افزار
- وقوع شرایط خاصی در سیگنال‌های سیستم
- دریافت یک فریم از پورت ارتباطی

مثال‌هایی از رویدادهای مرسوم در نرم‌افزارهای نهفته هستند. پرینت کردن پیام‌ها در نقاط مختلف برنامه بر روی پورت سریال و مشاهده پیام‌ها در ترمینال کامپیوتر، از روش‌های قدیمی، مرسوم و کارآمد جهت مشاهده روند اجرای برنامه است. اما دلایل زیر، باعث محدودیت استفاده از این روش می‌شود:

- زمان زیادی که پرینت پیام از پردازشگر می‌گیرد.
- عدم امکان دسته‌بندی و مدیریت پیام‌ها.
- عدم امکان اندازه‌گیری فاصله زمانی بین پیام‌های مختلف.
- عدم امکان مقایسه بقیه پارامترها و سیگنال‌های سیستم در لحظه دریافت پیام.
- اشغال پهنای باند زیاد لینک ارتباطی.
- عدم امکان استفاده در رویدادهای سریع (مانند ورود و خروج تابع، اینتراپت یا تسک)

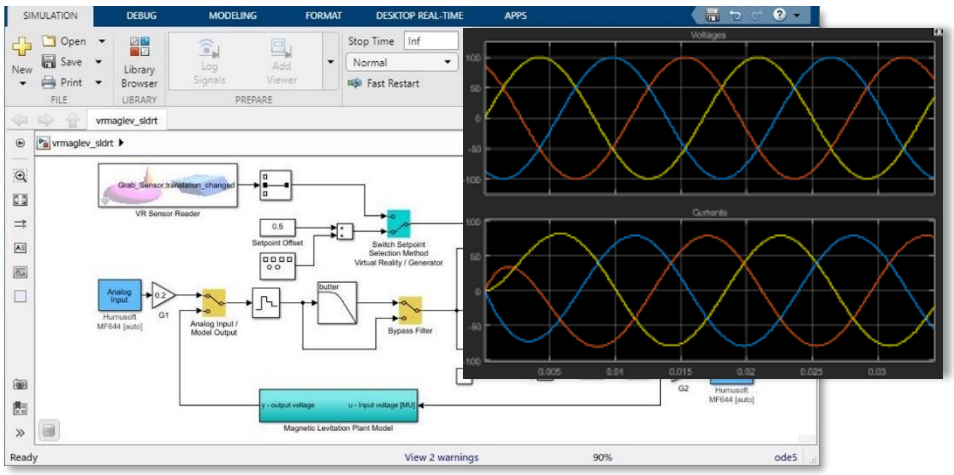


فرابین علاوه بر فراهم کردن تجربه قدیمی پرینت پیام، با برطرف کردن محدودیت‌ها، اضافه کردن قابلیت‌های جدید جهت مدیریت و بهره‌برداری بهتر از رویدادها و با ترکیب این اطلاعات با سایر داده‌های به‌دست‌آمده از کد کاربر، دید جامعی از رفتار پردازشگر ارائه می‌دهد.

فرابین بدین منظور، امکاناتی را نظیر تعریف هر تعداد دسته رویداد با قابلیت رنگ‌بندی و فیلتر کردن، مشاهده رویدادها با برچسب زمانی، اندازه‌گیری زمان بین رویدادها و مشاهده همزمان رویدادها با سایر سیگنال‌های سیستم در قالب‌های گرافیکی مختلف فراهم کرده است.

پیاده‌سازی با امکانات شبیه‌سازی

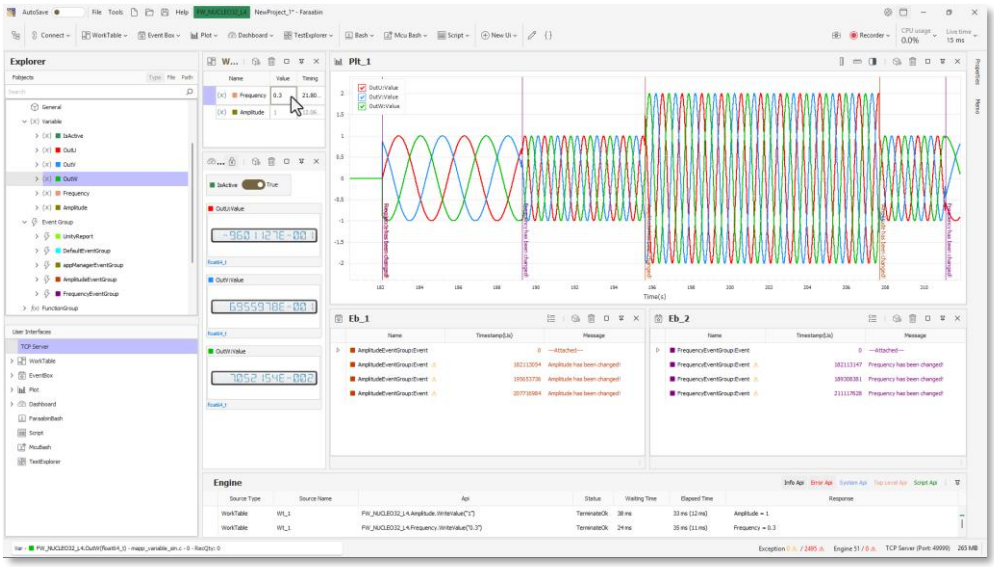
رفتار نرم‌افزاری که برنامه‌نویس نهفته باید پیاده‌سازی کند، توسط مهندسان سیستم مربوطه - مانند مهندسان کنترل، هوافضا، الکترونیک قدرت و . . . - طراحی می‌شود. این مهندسان در ابتدا، با ابزارهایی مانند MATLAB، الگوریتم‌ها و محاسبات خود را شبیه‌سازی کرده و عملکرد آن را بررسی می‌کنند. در این محیط‌های شبیه‌سازی، ابزارها و امکانات قدرتمندی وجود دارد، به گونه ای که طراحان می‌توانند در تمام لحظات، جزئیات سیگنال‌ها و رویدادها را با نرخ بالا مشاهده کنند، پارامترهای سیستم را به صورت برخط تغییر دهند و اثر این تغییرات را بر روی خروجی سیستم مشاهده کنند.



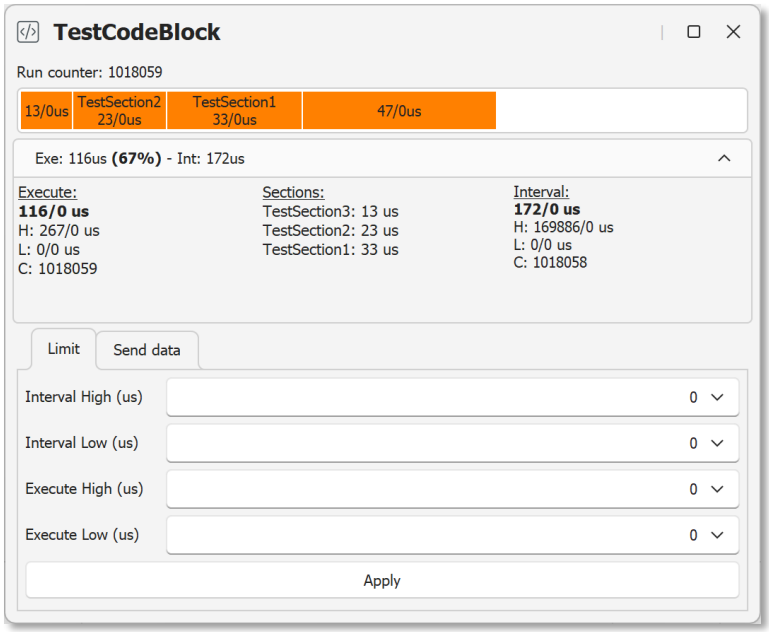
ولی پس از پیاده‌سازی نرم‌افزار و اجرای آن بر روی سیستم نهفته، دیگر از آن امکانات مشاهده‌پذیری و کنترل‌پذیری محیط شبیه‌سازی خبری نیست و باید با ابزارهای محدودی مانند پین‌های میکرو و ارسال سیگنال‌ها روی مبدل دیجیتال به آنالوگ، شروع به عیب‌یابی و تست نرم‌افزار کنند.

ماجرا زمانی سخت‌تر می‌شود که معلوم نباشد مشکل از الگوریتم‌های طراحی شده است و یا از پیاده‌سازی اشتباه آن الگوریتم‌ها توسط برنامه‌نویسان نهفته.

فرابین در زمان اجرای برنامه روی سیستم نهفته، همان امکانات محیط شبیه‌سازی – و حتی بیشتر - و همان آزادی عمل را برای مهندسان سیستم و برنامه‌نویسان نهفته فراهم می‌کند.



بررسی زمان‌بندی اجرای برنامه

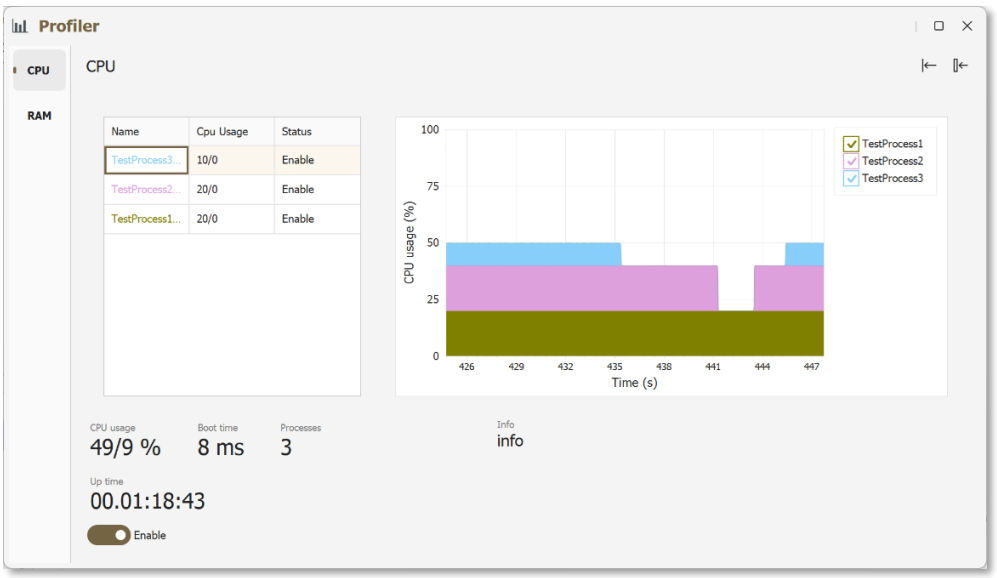


متاسفانه بسیاری از برنامه‌نویسان نرم‌افزارهای نهفته‌ای که برنامه‌های بلادرنگ (Real-Time) می‌نویسند، از اینکه برنامه‌هایشان نیازهای زمانی را برآورده می‌کند یا نه، اطلاعی ندارند. در مراحل اولیه‌ی طراحی، تجزیه و تحلیل نرخ مونوتونیک (RMA) تایید می‌کند که آیا برنامه زمان‌بندی محقق خواهد شد یا نه. اما پس از شروع فاز پیاده‌سازی، نتایج واقعی به ندرت با فرضیات طراحی اولیه تطابق خواهند داشت.

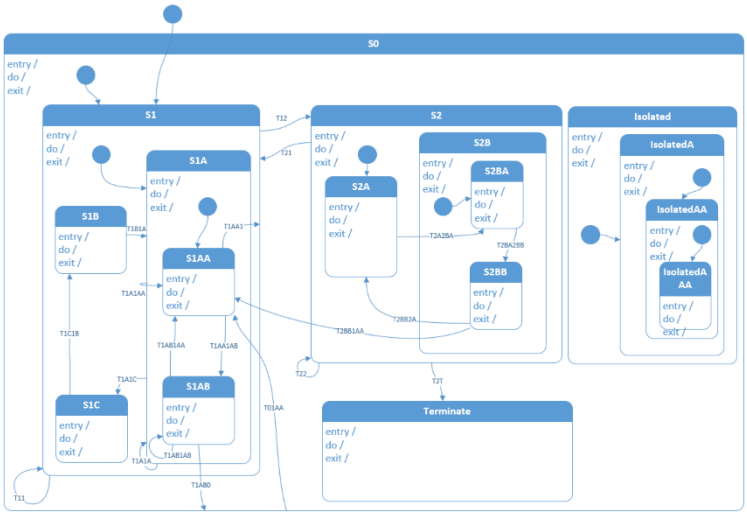
ما نیاز داریم که زمان‌بندی اجرای بخش های مختلف کد مانند تسک‌ها (در صورت استفاده از کرنل)، اینتراپت‌ها، زمان پاسخ سیستم به رویدادها و . . . را در زمان اجرای برنامه ببینیم و با نیازهای اولیه مقایسه کنیم. همچنین باید بتوان درصد اِشغال پردازشگر و سهم هر یک از بخش‌های برنامه در آن را اندازه‌گیری و مشاهده کرد.

معمولا از ابزارهای ساده‌ای مانند صفر و یک کردن پین‌های پردازشگر در زمان‌های مورد نظر و مشاهده و تحلیل آن از طریق اسکوپ یا لاجیک‌آنالایزر بدین منظور استفاده می‌شود. علاوه بر آنکه نمی‌توان اتفاقات ناگهانی در زمان‌بندی را بدین روش مشاهده کرد، امکان همزمانی این اطلاعات با بقیه رویدادها و سیگنال‌های سیستم هم وجود ندارد.

فرابین با استفاده از کتابخانه پروفایلر، امکان مشاهده برخط زمان‌بندی تمام المان‌های اجرایی (مانند تسک، اینتراپت و یا هر بخشی از کد) و اطلاعات آماری آن‌ها را فراهم می‌کند. همچنین قابلیت تعریف بازه مجاز برای زمان اجرا و دوره اجرا و بررسی عبور زمان‌بندی از این حدود را در اختیار کاربر قرار می‌دهد.

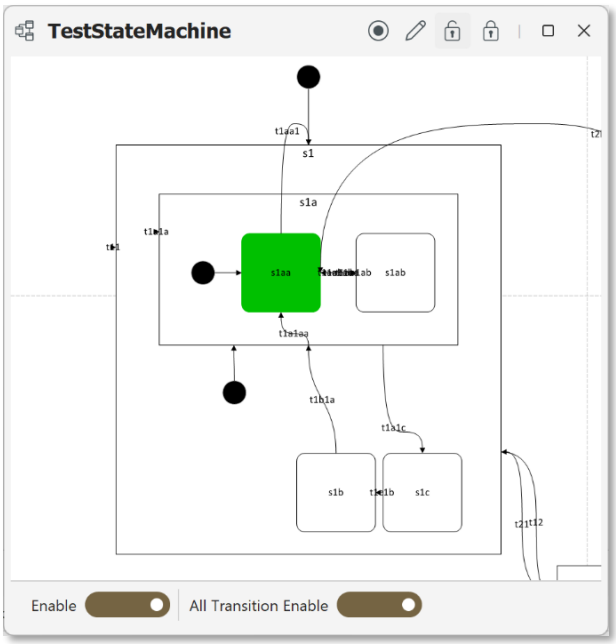


ماشین حالت



ماشین حالت ابزاری برای مدل‌سازی رفتار سطح بالای سیستم است. در اکثر نرم‌افزاره‌ای نهفته، رفتار سطح بالا با ماشین حالت مدل‌سازی می‌شود و استفاده از این مفهوم در این سیستم‌ها اجتناب‌ناپذیر است.

پیاده‌سازی تمام قابلیت‌های ماشین حالت، کاری پیچیده است. معمولا کدنویسان برای پیاده‌سازی آن از ابزارهای ساده‌ای مانند if else یا switch case استفاده می‌کنند که مفهوم کلی ماشین حالت را محقق می‌کند. ولی برای پیاده‌سازی مفاهیمی مانند Entry, Exit, Transition و . . . مناسب نیستند و باید قابلیت‌های دیگری به کد اضافه شود که کار را سخت می‌کند. این موارد برای ماشین حالت تک‌سطحی (FSM) است. در صورتی که نیاز باشد از ماشین حالت چندسطحی (HSM) استفاده شود (ذیل هر حالت چند حالت دیگر تعریف می‌شود) - که در سیستم‌های متوسط به بالا اکثرا چنین نیازی وجود دارد - این روش‌ها جوابگو نیستند.

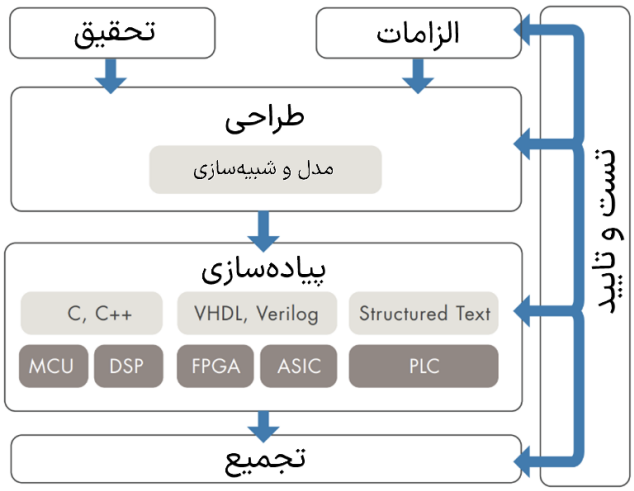


فرابین با پشتیبانی از کتابخانه جامع ماشین حالت، ابزاری قدرتمند را برای پیاده‌سازی هر نوع ماشین حالت تک‌سطحی و چندسطحی، فعال/غیرفعال کردن بخش‌های مختلف آن، تزریق رویداد، جلوگیری از سرایت رویداد و . . . فراهم می‌کند.

کاربر می‌تواند دیاگرام ماشین حالت طراحی شده را به صورت برخط مشاهده کرده و با فعال/غیرفعال کردن حالت‌ها و انتقال‌ها و همچنین فرمان انتقال به حالت موردنظر، رفتار ماشین حالت را به صورت برخط کنترل کند.

نمایش تاریخچه رویدادهای ماشین حالت با زمان‌بندی دقیق و در مقایسه با بقیه رویدادها و سیگنال‌های سیستم از دیگر قابلیت‌های فرابین در این بخش است.

پشتیبانی از "طراحی مبتنی بر مدل"



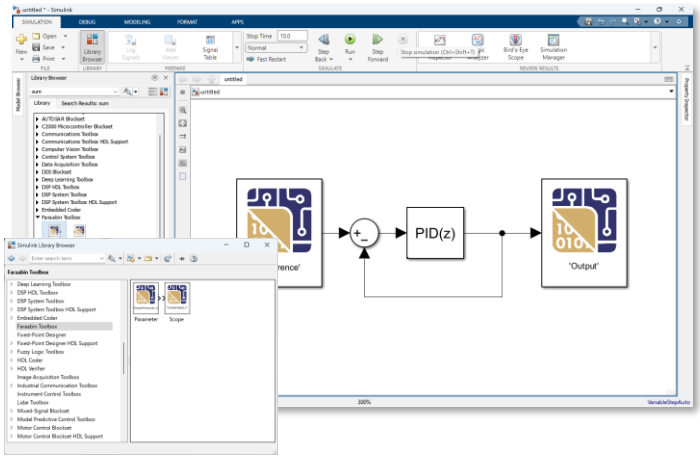
"طراحی مبتنی بر مدل" (Model-Based Design) یک روش توسعه است که در زمینه‌های مهندسی مختلف، به ویژه در مهندسی سیستم‌ها و محصولات پیچیده‌ای که دارای نرم‌افزار نهفته هستند، استفاده می‌شود. این روش توسعه به کاربران امکان می‌دهد تا سیستم‌ها را با استفاده از مدل‌ها طراحی، شبیه‌سازی و بررسی کنند و سپس آن‌ها را به صورت خودکار به کد اجرایی تبدیل کنند.

Embedded Coder یکی از ابزارهای نرم‌افزار متلب است که برای تولید کد خودکار از مدل‌های سیمولینک استفاده می‌شود. این ابزار امکان تولید کد C و C++ بهینه‌شده را فراهم می‌کند که می‌تواند به پروژه نهفته اضافه شده و اجرا شود.

با استفاده از این روش و ابزارهای مرتبط، مهندسان می‌توانند به طور مستقیم مدل‌های طراحی شده را به کد تبدیل کنند و در سیستم‌های واقعی اجرا کنند. (عکس‌های هواپیما از شرکت Embraer و قطار از شرکت Bombardier که در بالا نشان داده شده است، از پروژه‌های معروفی هستند که با استفاده از این روش توسعه داده شده‌اند.)

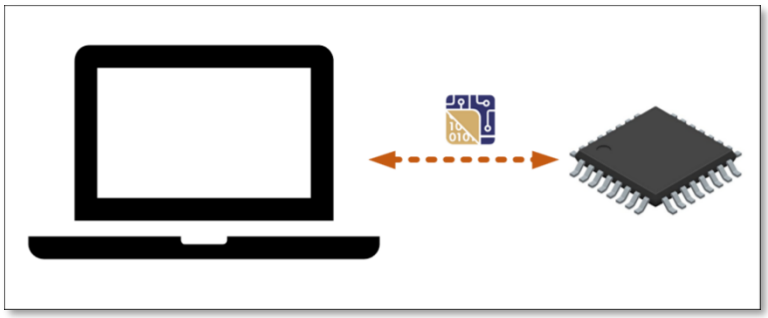
استفاده از این روش باعث می‌شود :

- کد اجرایی به طور خودکار و بهینه تولید شود که زمان و تلاش مورد نیاز برای کدنویسی دستی را کاهش می‌دهد و باعث کاهش خطا و افزایش دقت می‌شود.
- مدل‌ها و کدهای تولید شده بتوانند به راحتی برای پروژه‌های دیگر مجدداً استفاده شوند.
- مهندسان سیستم (مانند کنترل، هوافضا، الکترونیک قدرت و . . .) بتوانند مستقیما و بدون نیاز به برنامه‌نویس نهفته با نرم‌افزار تعامل برقرار کرده و آن را تغییر دهند.
- نرم‌افزار توسعه داده شده دارای مستندات و مدل صددرصد منطبق با واقعیت باشد.



فرابین با ارائه یک کتابخانه در محیط سیمولینک، از این ابزار پشتیبانی می‌کند. کاربر با استفاده از بلوک‌های ارائه شده در این کتابخانه، می‌تواند سیگنال‌های خود در مدل را در فرابین مشاهده کرده و در حین اجرا به آنها دسترسی داشته باشد.

فراتر از ابزار عیب‌یابی



فرابین فراتر از یک ابزار برای مشاهده‌پذیری، کنترل‌پذیری و عیب‌یابی است. این ابزار یک کانال ارتباطی بین کامپیوتر و پردازشگر ایجاد می‌کند که امکان دسترسی به رفتار داخلی پردازشگر را در حالت عادی فراهم می‌سازد. این کانال ارتباطی و پروتکل آن به گونه‌ای طراحی شده که با کمترین سربار و استفاده بهینه از پهنای باند، زیرساختی منعطف برای انتقال اطلاعات ارائه می‌دهد که به ساختار کد کاربر و نوع اطلاعات وابسته نیست. این کانال ارتباطی، دریچه‌ای به روی کاربر می‌گشاید و امکان ایجاد کاربردهای متنوعی مانند پایش و تحلیل، تست اتوماتیک، سخت‌افزار در حلقه و رابط‌های کاربری مختلف را فراهم می‌سازد.

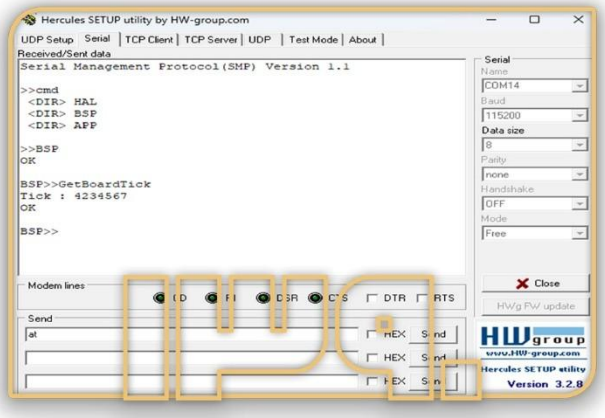
فرابین بر پایه رابط کاربری متنی (Command Line Interface) طراحی شده است. این امکان به کاربر اجازه می‌دهد تا با استفاده از ابزارهای اسکریپت‌نویسی و برنامه‌نویسی، به کد دسترسی داشته و توابع خود را از طریق رابط کاربری متنی فراخوانی کند.

برای این منظور، قابلیت‌هایی پیش‌بینی شده است که کاربر می‌تواند با استفاده از ارتباط TCP/IP و HTTP و از طریق دستورات متنی و قالب JSON، به نرم‌افزار فرابین متصل شده و در محیط مورد نظر خود از این کانال ارتباطی با میکرو استفاده کند.

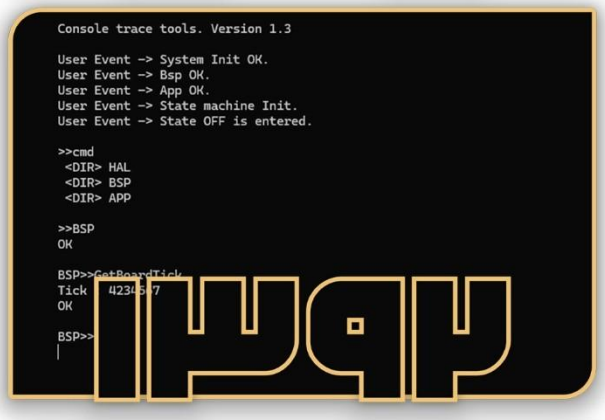
به عبارت دیگر، کاربر می‌تواند از کانال ارتباطی ایجاد شده بین کامپیوتر و پردازشگر در زبان برنامه‌نویسی یا نرم‌افزار خود استفاده کند. و به جای استفاده از رابط گرافیکی فرابین، یا علاوه بر آن، می‌تواند از زبان‌های پایتون، جاوا اسکریپت، سی‌شارپ، سی‌پلاس‌پلاس یا هر زبان و نرم‌افزار دیگری برای نوشتن برنامه‌ها و ارتباط با پردازشگر بهره ببرد.

در زیر به چند نمونه از کاربردهای این قابلیت اشاره می‌شود:

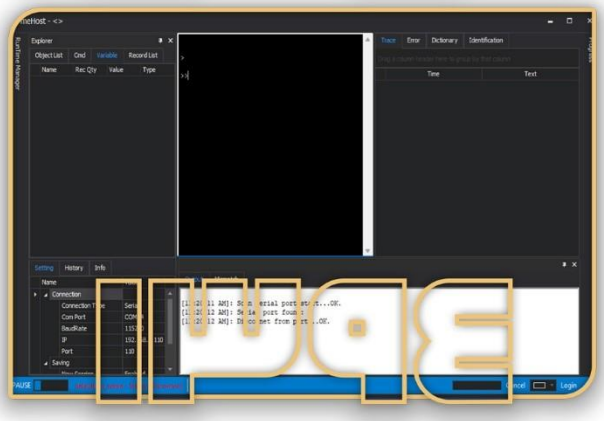
- ایجاد رابط کاربری شخصی در هر نرم‌افزار و زبان برنامه‌نویسی
- ایجاد نرم‌افزارهای تحت وب و دسترسی به فرابین از طریق شبکه و اینترنت
- برنامه‌نویسی سطح بالا جهت ارتباط با پردازشگر
- استفاده از ابزارهای پیشرفته تست اتوماتیک (مانند کتابخانه‌های موجود در پایتون) بر روی پردازشگر نهفته
- توسعه اپلیکیشن‌های شخصی بر روی فرابین جهت تحلیل، پردازش، رابط‌های گرافیکی جدید و . . .



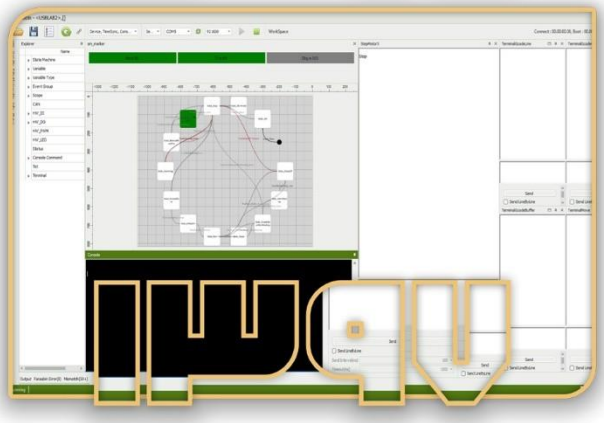
ایده اولیه فرابین به نام (Serial Management Protocol) SMP در این سال خلق شد. این ابزار تنها یک کتابخانه ساده در پردازشگر بود و از طریق پورت سریال با نرم‌افزارهای کنسول کامپیوتر ارتباط برقرار می‌کرد.



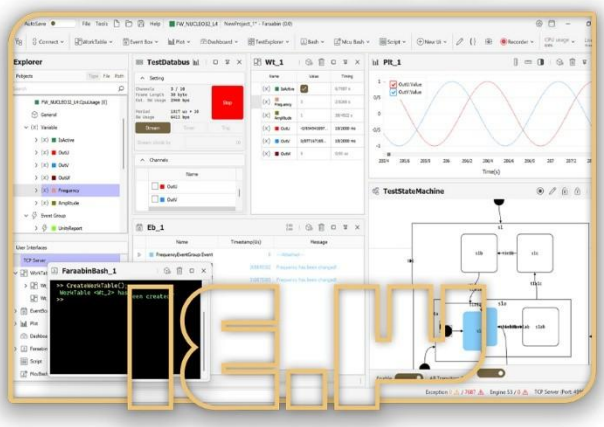
نرم‌افزار کامپیوتری مشابه Command Prompt اضافه شد و نام آن به Console تغییر پیدا کرد. نرم‌افزار اختصاصی کامپیوتر باعث ایجاد آزادی عمل در تعیین قالب نمایش متن‌ها شد.



رابط کاربری گرافیکی و قابلیت ارتباط از طریق اترنت اضافه شدند و نام آن Host انتخاب شد. نمایش نمودار، جدول رویدادها و جدول متغیرها از قابلیت‌هایی است که در این نسخه اضافه شده است.



رابط کاربری گرافیکی بهبود یافت و نام فرابین برای آن انتخاب شد. این ویرایش بیشترین استفاده را در صنعت داشته است. همچنین، پشتیبانی از ماشین حالت تک‌سطحی در این ویرایش اضافه شد.



بازطراحی کامل با هدف عرضه عمومی انجام شد. این ویرایش شامل تغییراتی مانند رابط کاربری متنی، اسکریپت‌نویسی، تعریف پروژه، رابط API برای نرم‌افزارهای ثالث، پشتیبانی از ماشین حالت چندسطحی، پشتیبانی از پروفایلر، پشتیبانی از MATLAB Embedded Coder و مستندات کامل است.

نسخه پایه

رایگان

- بدون محدودیت زمانی
- بدون محدودیت نصب
- تعداد نامحدود دیتاباس
- تعداد نامحدود نمودار
- تعداد نامحدود جعبه رویداد
- تعداد نامحدود داشبورد
- تعداد نامحدود ترمینال
- تعداد نامحدود میزکار
- تعداد نامحدود اسکریپت و بش
- پشتیبانی از آرایه
- چهل متغیر پایه
- سه متغیر struct / enum (با ۱۰ عضو)
- سه دسته رویداد
- ده تابع
- پورت ارتباطی سریال
- پنج دقیقه ثبت اطلاعات

نسخه حرفه‌ای

تماس بگیرید

→ علاوه بر موارد طرح قبلی

- ۱ ساله
- نصب بر روی دو کامپیوتر
- تعداد نامحدود متغیر پایه
- تعداد نامحدود متغیر struct / enum
- تعداد نامحدود دسته رویداد
- تعداد نامحدود تابع
- ثبت اطلاعات نامحدود

+ افزونه

- تمدید
- تعداد نصب اضافه
- ماشین حالت
- پروفایلر
- پورت ارتباطی اترنت
- رابط API از طریق TCP/IP و HTTP

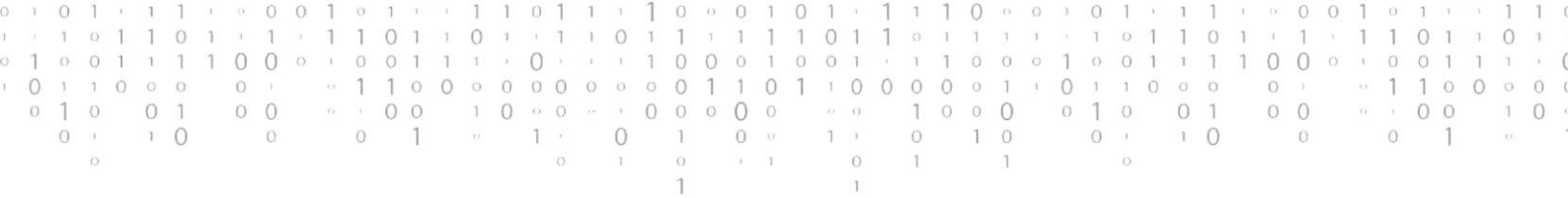
نسخه سازمانی

تماس بگیرید

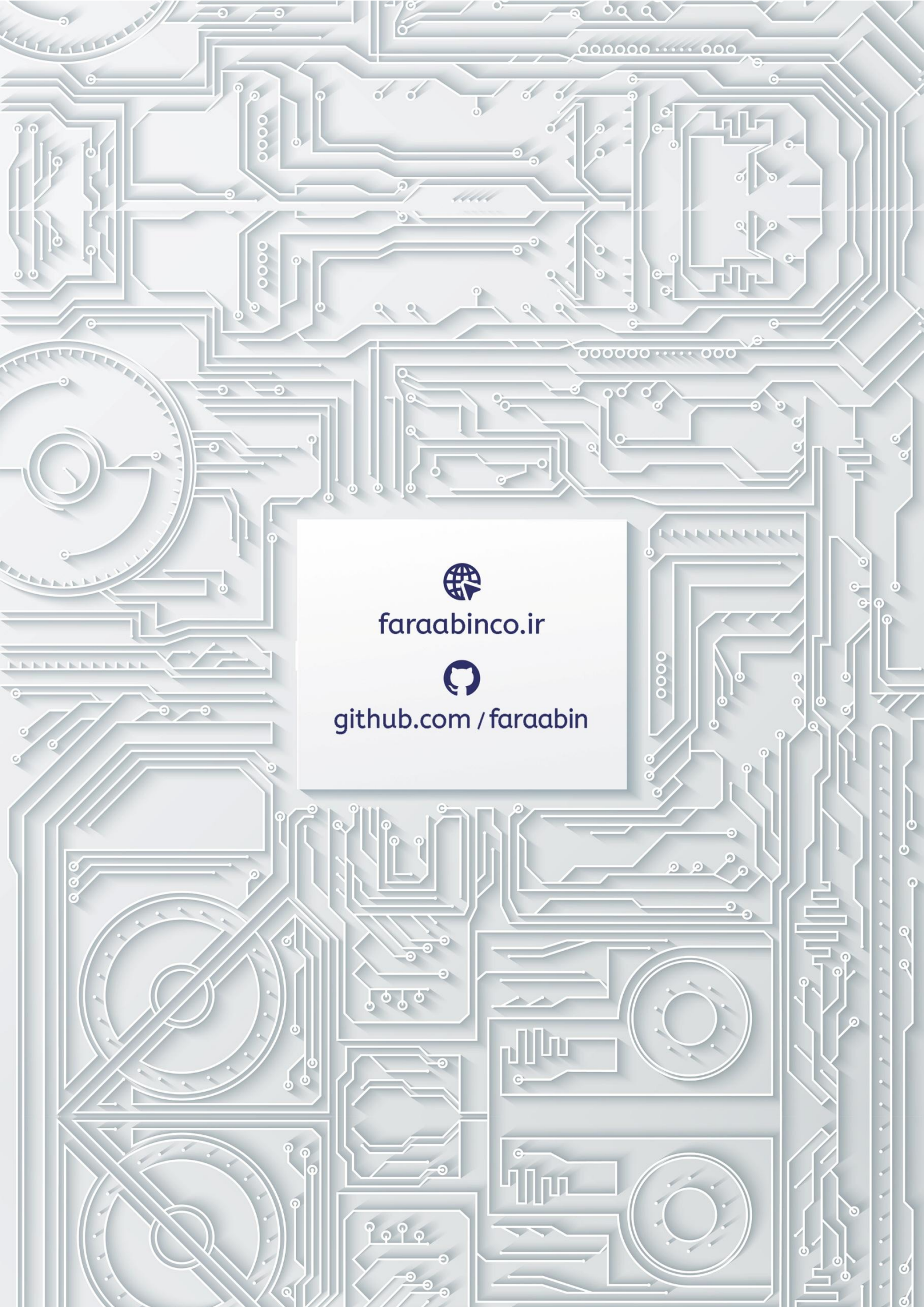
→ علاوه بر موارد طرح قبلی

- بدون محدودیت زمانی

لینک دانلود نرم افزار فرابین




Explorer
TestDatabus
Wt_1
Plt_1
Eb_1
TestStateMachine
FaraabinBash_1
>> CreateWorkTable();
WorkTable <Wt_2> has been created.
>>
Time(s)
Enable
All Transition Enable


faraabinco.ir
github.com / faraabin