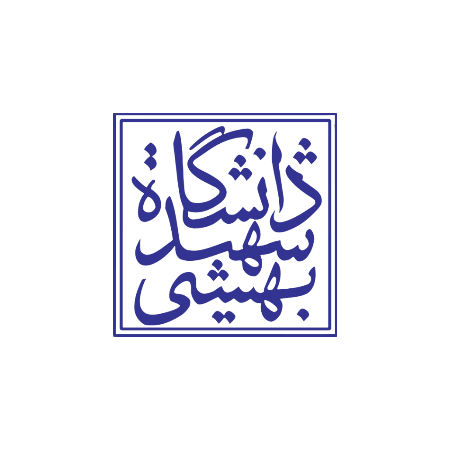
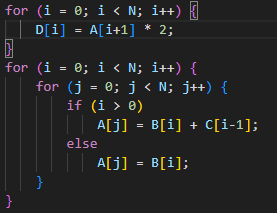
**گرازش تمرین 7**

**بسم الله الرحمن الرحيم**

**درس مبانی سیستم نهفته و بیدرنگ**

سوال اول) قطعه کد C مقابل را در نظر بگیری د و تبدیل های خواسته شده زیر جهت بهینه سازی را یکی پس از دیگری به کد اعمال کنید. هر تبدیل به کد خروجی پیش از آن اعمال شود و توضیح داده شود چگونه به بهبود کارایی کمک میکند ؟

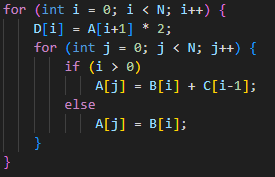


**جواب اول)**

1. ترکیب (fusion)حلقه ها

شامل ترکیب حلقه های مجاور است که در یک محدوده تکرار می شوند و در یک حلقه واحد می شوند. این امر سربار کنترل حلقه را کاهش می دهد و موقعیت داده ها را بهبود می بخشد.

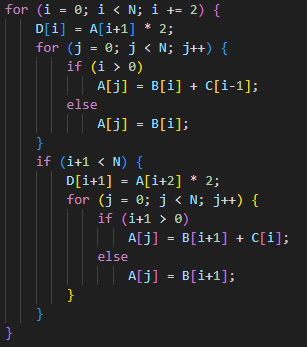
در این قسمت دسترسی به آرایه A راحتر شده است.



1. باز کردن (unroll )حلقه بیرونی با ضریب N=2

Unrolling با اجرای چندین تکرار از بدنه حلقه در یک پاس، تعداد تکرارها را کاهش می دهد.

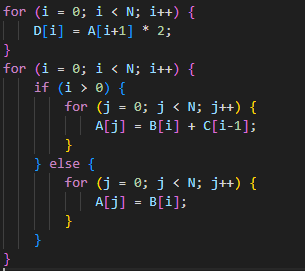
تعداد تکرارهای حلقه را کاهش می دهد و با استفاده بهتر از pipelinesدستورالعمل، عملکرد پردازنده های مدرن را بهبود می بخشد.



1. تبدیل loop splitting/nesting

این تبدیل شامل بازسازی حلقه های تو در تو برای بهبود وضوح یا فعال کردن بهینه سازی های دیگر است.

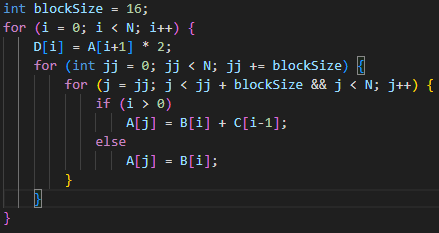
شرایط کنترل حلقه را ساده می کند و کد را ساختارمندتر و بهینه سازی برای کامپایلرها آسان تر می کند.



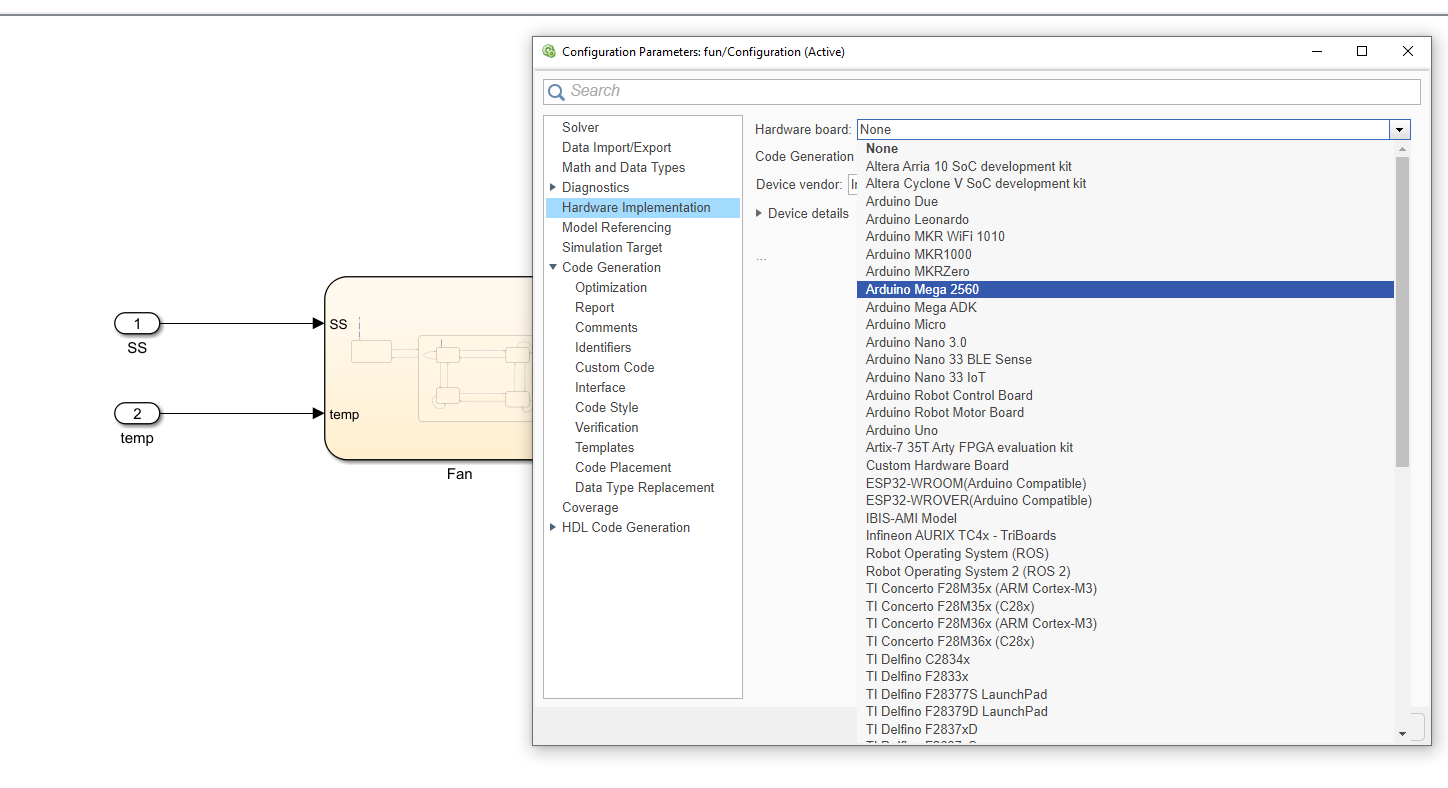
1. تبدیل blocking/tiling loop برای حلقه داخلی به صورت پارامتری با پیش فرض block size = 16

مسدود کردن تکرارهای حلقه را به تکه های کوچکتر (کاشی) تقسیم می کند تا موقعیت داده ها را بهبود بخشد.

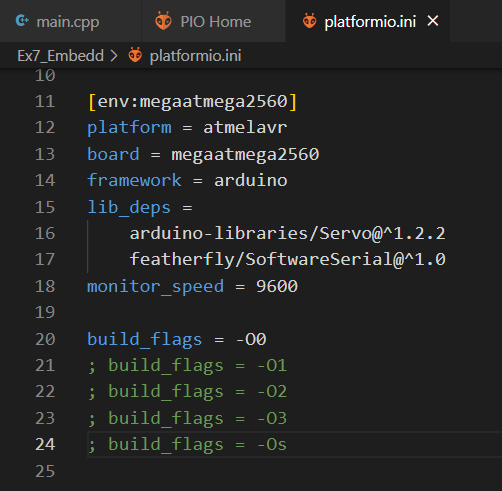
عملکرد cache را با کار بر روی تکه های کوچکتر داده که در حافظه نهان قرار می گیرند، افزایش می دهد و از دست رفتن حافظه پنهان را کاهش می دهد.

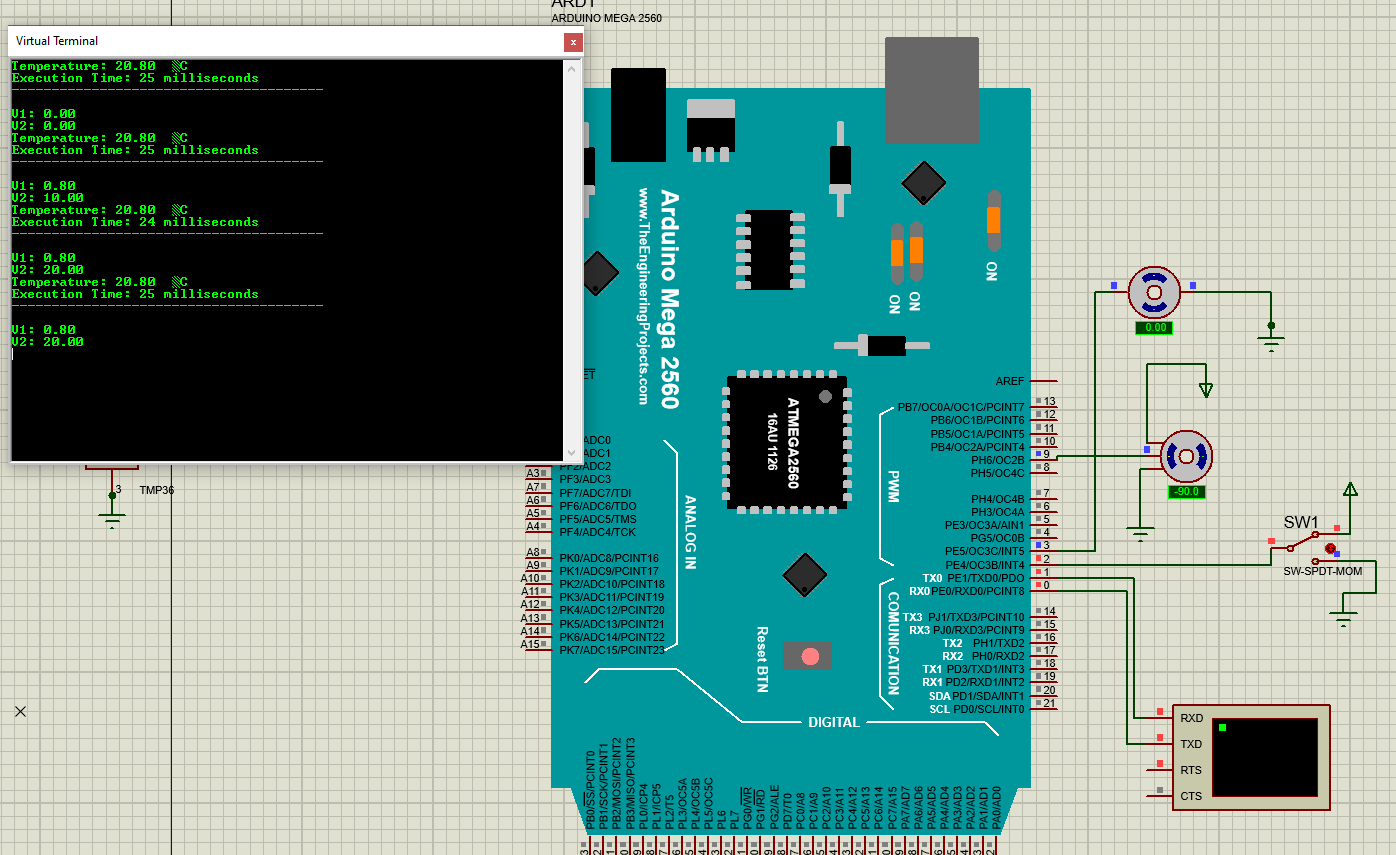
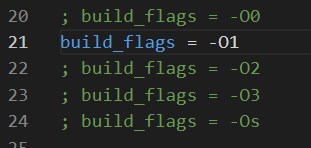


جواب دوم)

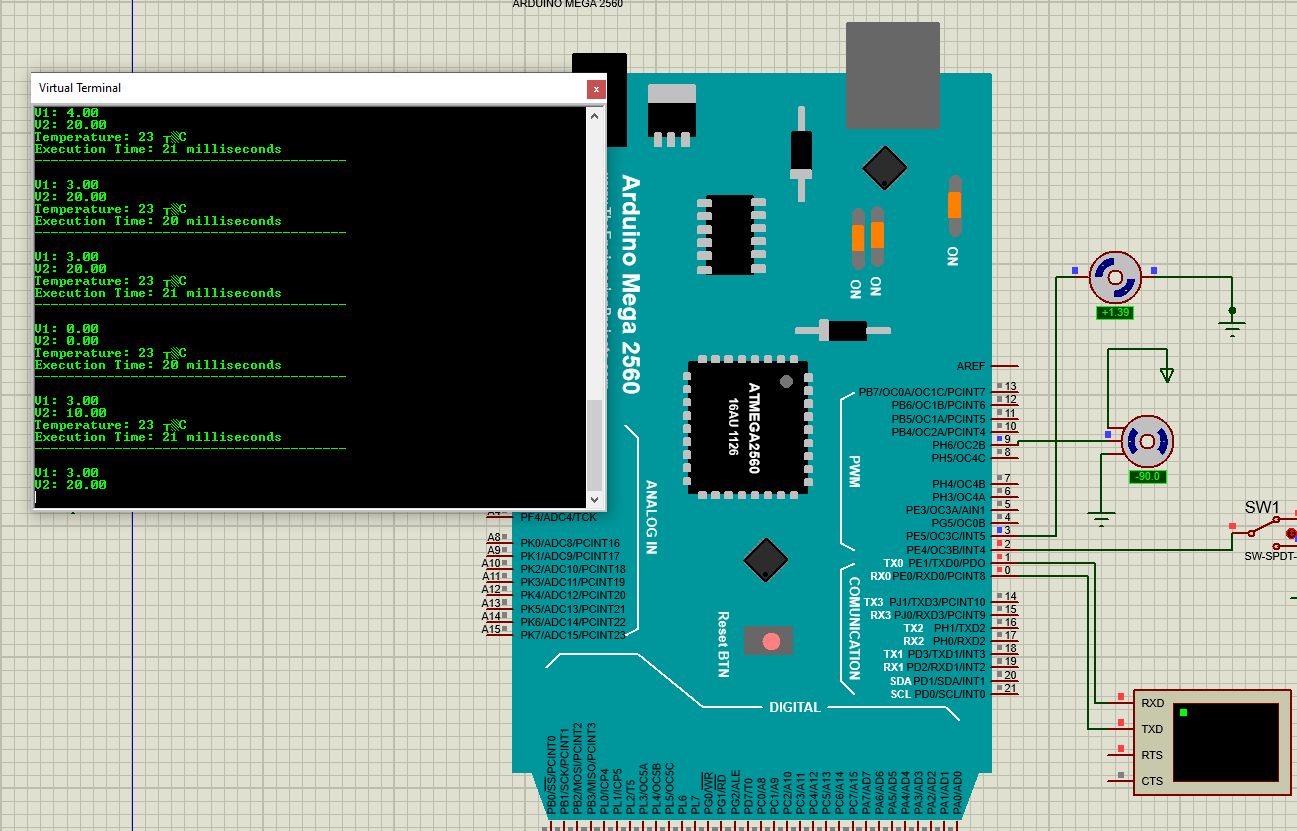
بخش الف) کد مدل را برای بورد Arduino Mega 2560 در Simulink بصورت زیر تولید میکنیم. 

بعد از تولید کد با استفاده از PlatformIO پروژه خود را برای سطوح مختلف بهینه سازی کامپایلر (-O0, -O1, -O2, -O3, -Os) شبه سازی می کنیم و میانگین زمان اجرا هر کدام شان را در جدول اضافه میکنیم.

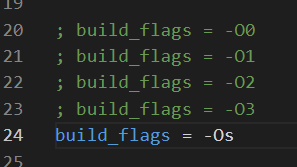
**تست حالت –O0**

**مدت زمان اجرا در حالت –O0** (حالت عادی) بین 26 تا 24 میلی ثانیه است.

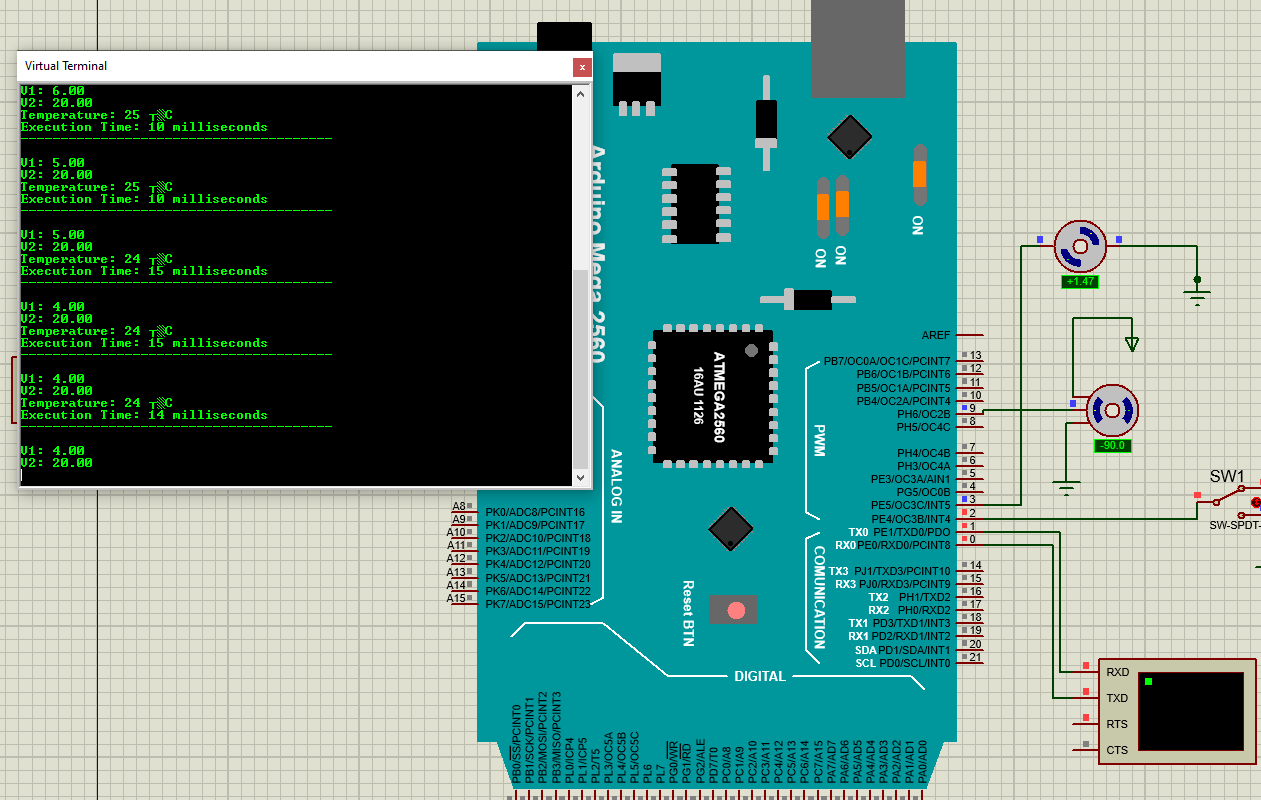
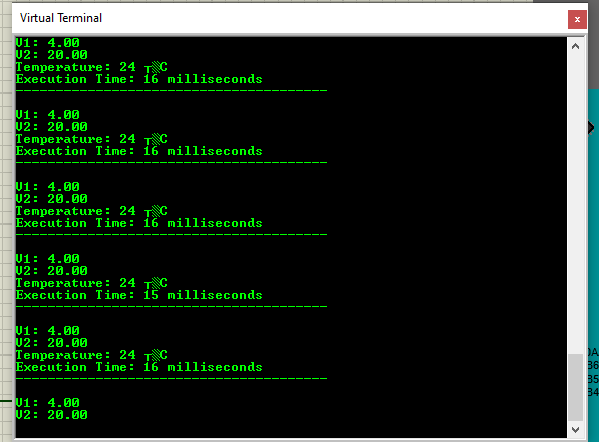
**تست حالت –O1**

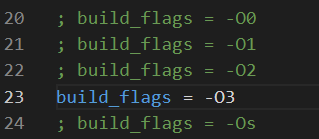


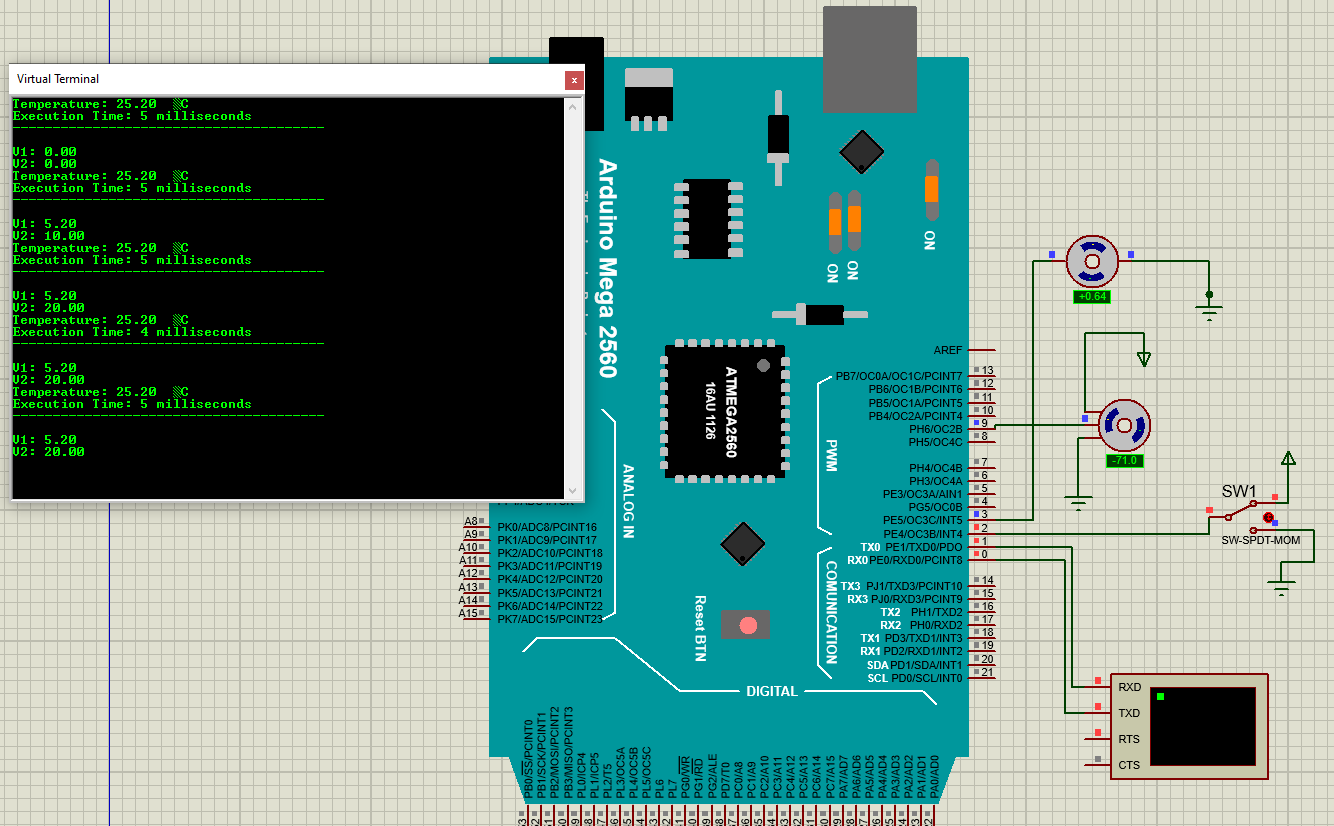
**مدت زمان اجرا در حالت –O1** (بهینه‌سازی اولیه) بین 21 تا 20 میلی ثانیه است.

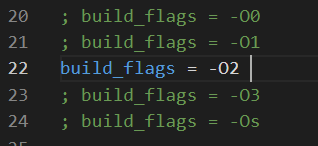
**تست حالت –Os**

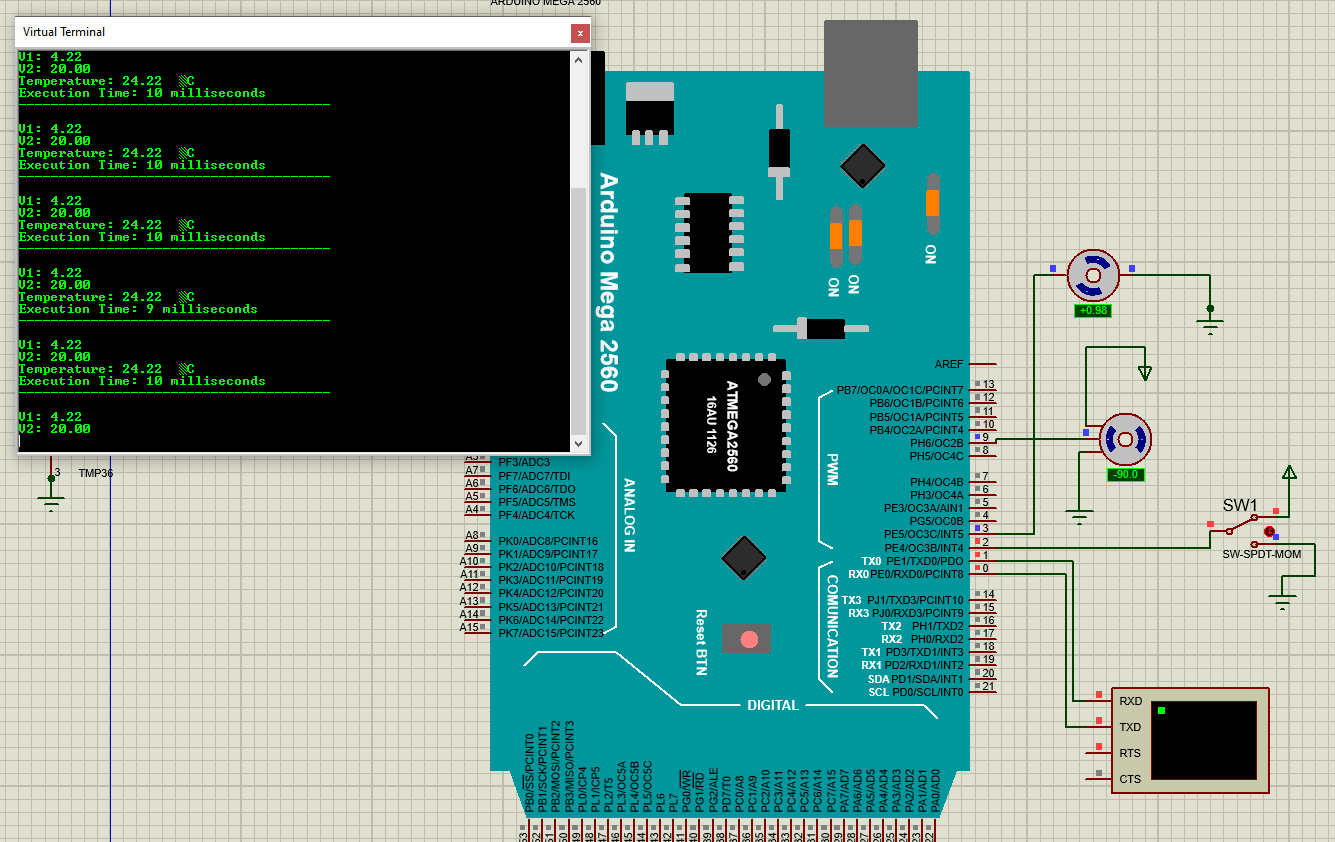
**مدت زمان اجرا در حالت –Os** (بهینه‌سازی برای اندازه کوچکتر)

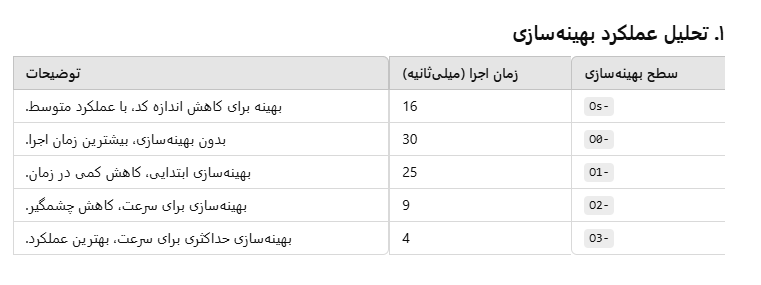
بین 16 تا 15 میلی ثانیه است.

**تست حالت –O****3 : مدت زمان اجرا در حالت –O3** (بهینه‌سازی حداکثری) بین 5 تا 4 میلی ثانیه است



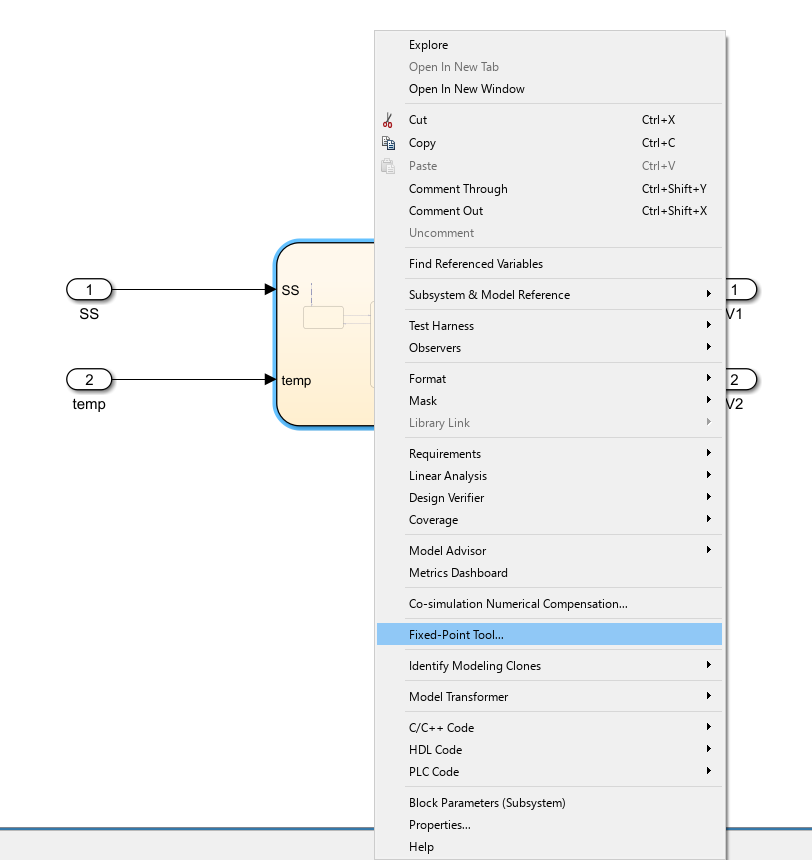
**تست حالت –O2 : مدت زمان اجرا در حالت –O2** (بهینه‌سازی برای سرعت) بین 10 تا 9 میلی ثانیه است

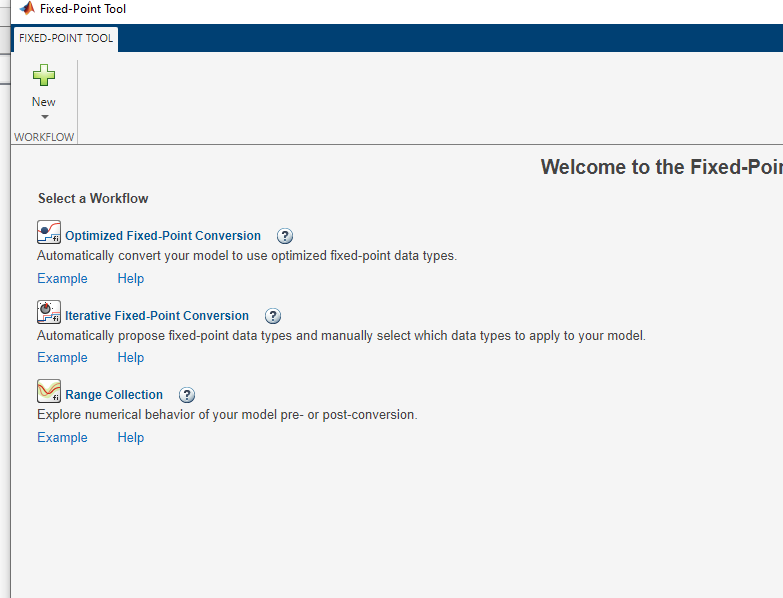


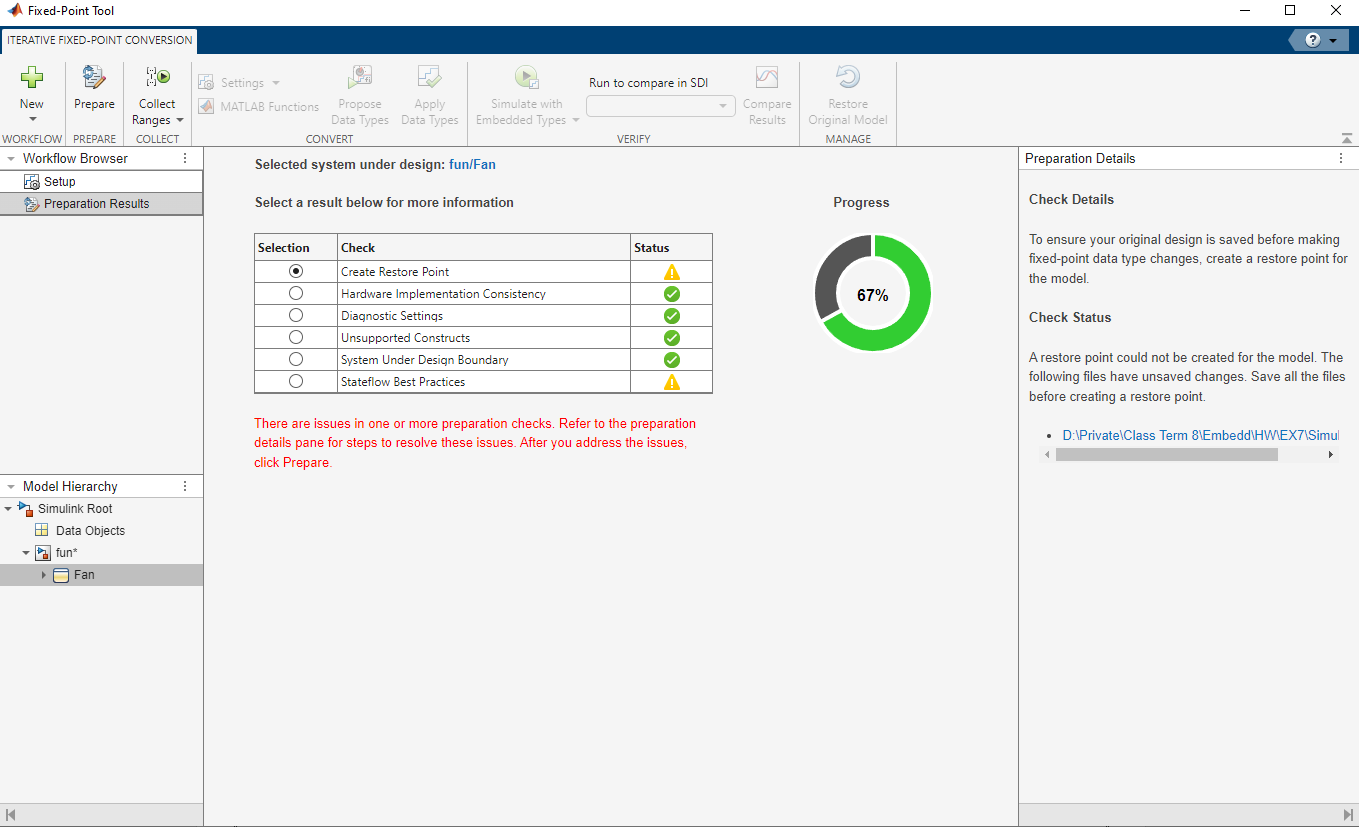


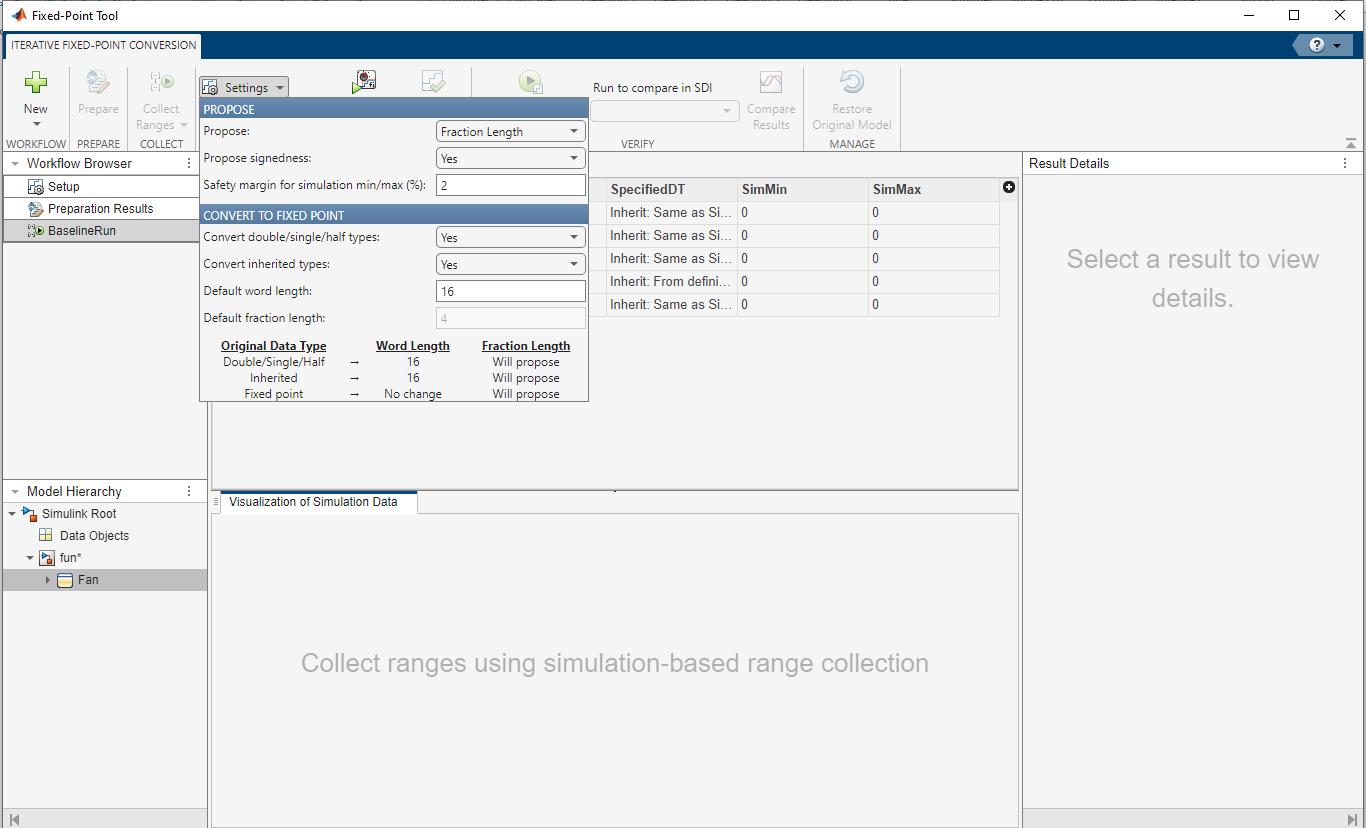
**روند تغییرات**

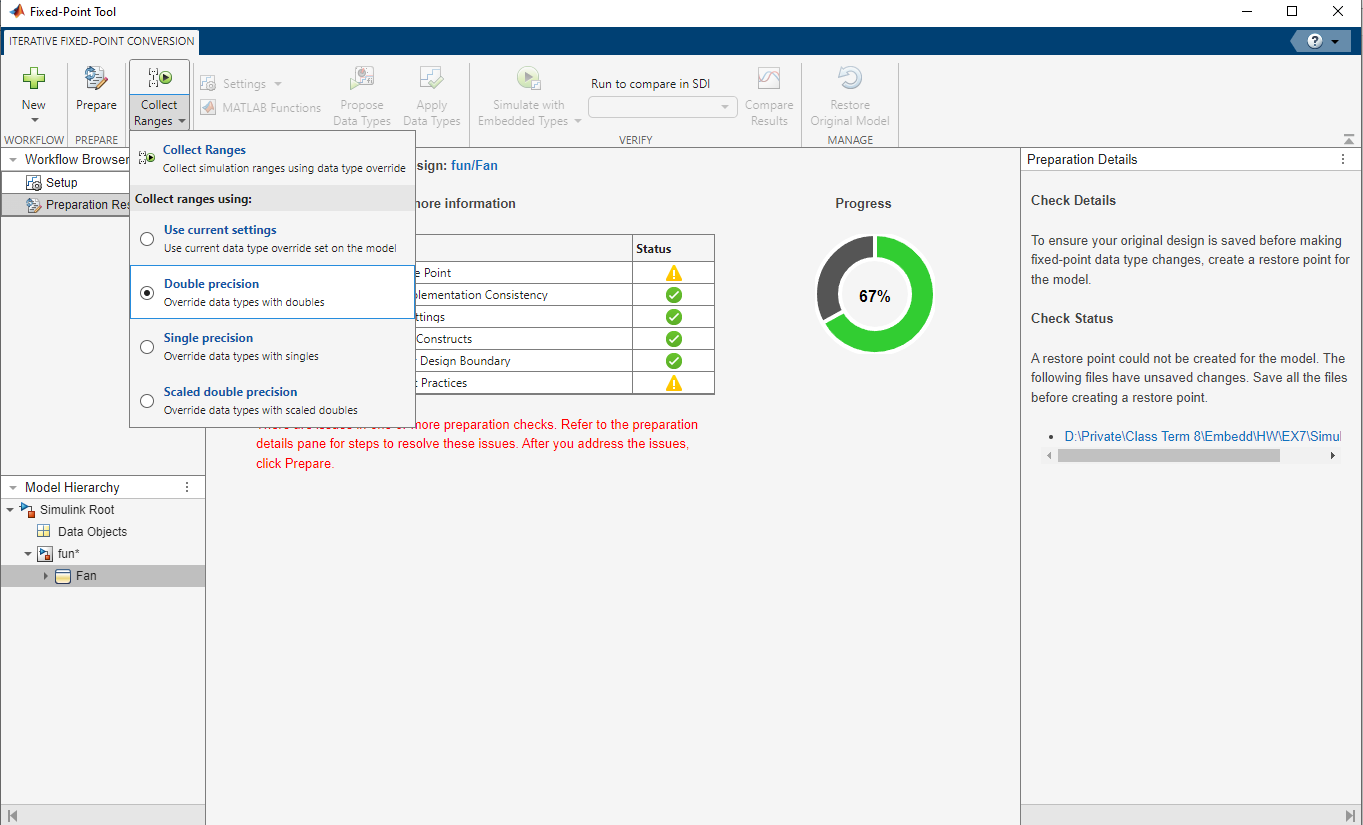
* **بدون بهینه‌سازی (-O0)**:
  + زمان اجرای طولانی نشان می‌دهد که کد مستقیماً بدون هرگونه بهینه‌سازی اجرا شده است.
* **بهینه‌سازی ابتدایی (-O1)**:
  + کاهش کمی در زمان اجرا، بیشتر مربوط به حذف دستورهای غیرضروری است.
* **بهینه‌سازی برای سرعت (-O2)**:
  + بهبود چشمگیر در عملکرد با کاهش زمان اجرا از 30 به 9 میلی‌ثانیه.
* **بهینه‌سازی حداکثری (-O3)**:
  + بهترین زمان اجرا (4 میلی‌ثانیه)، نشان‌دهنده کارایی بالای این سطح بهینه‌سازی برای عملیات شما است.
* **بهینه‌سازی برای اندازه (-Os)**:
  + زمان اجرا 16 میلی‌ثانیه است که کمتر از -O0 اما بیشتر از -O2 است. این سطح بهینه‌سازی بیشتر برای کاهش اندازه فایل نهایی طراحی شده و نه سرعت.

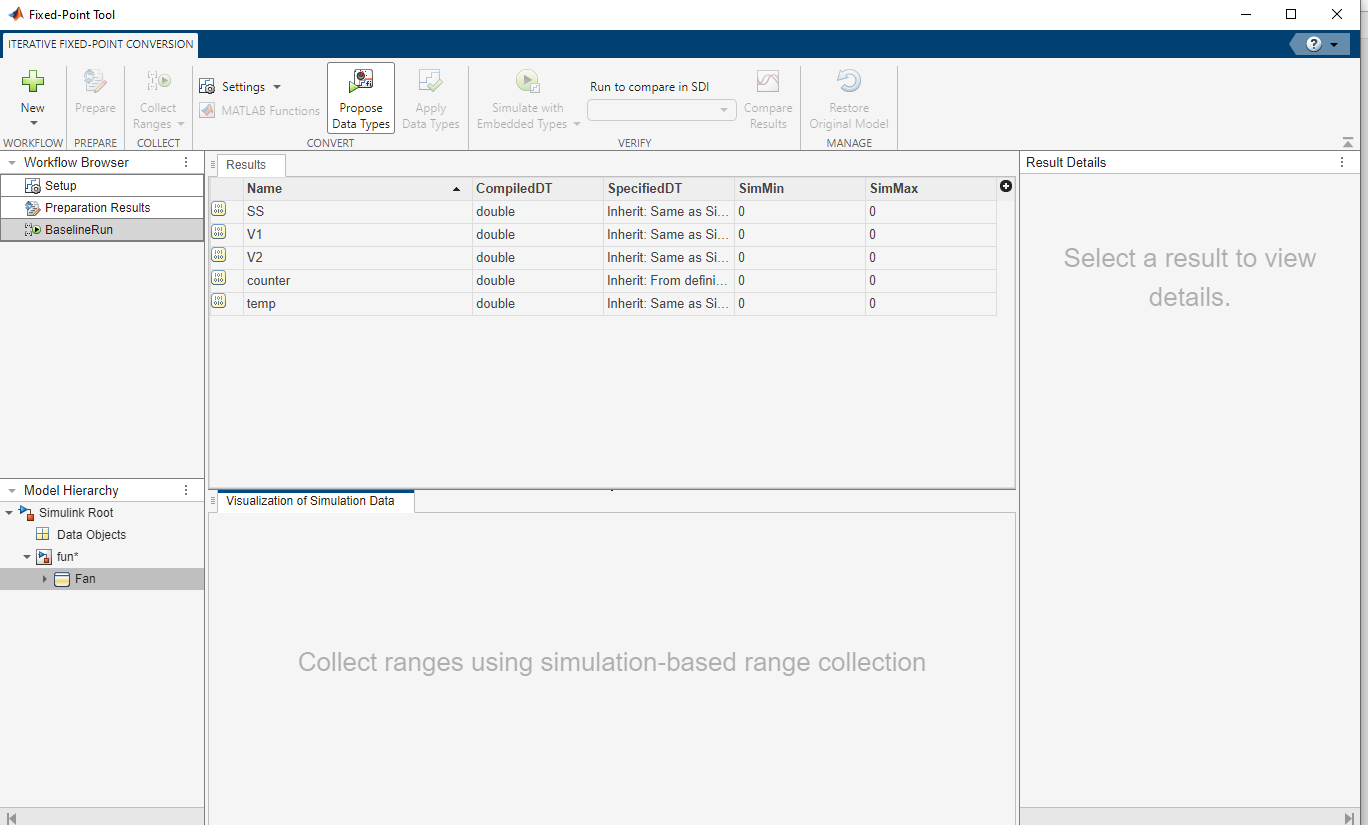
بخش ب) مدل را با استفاده از ابراز Fixed-Point   


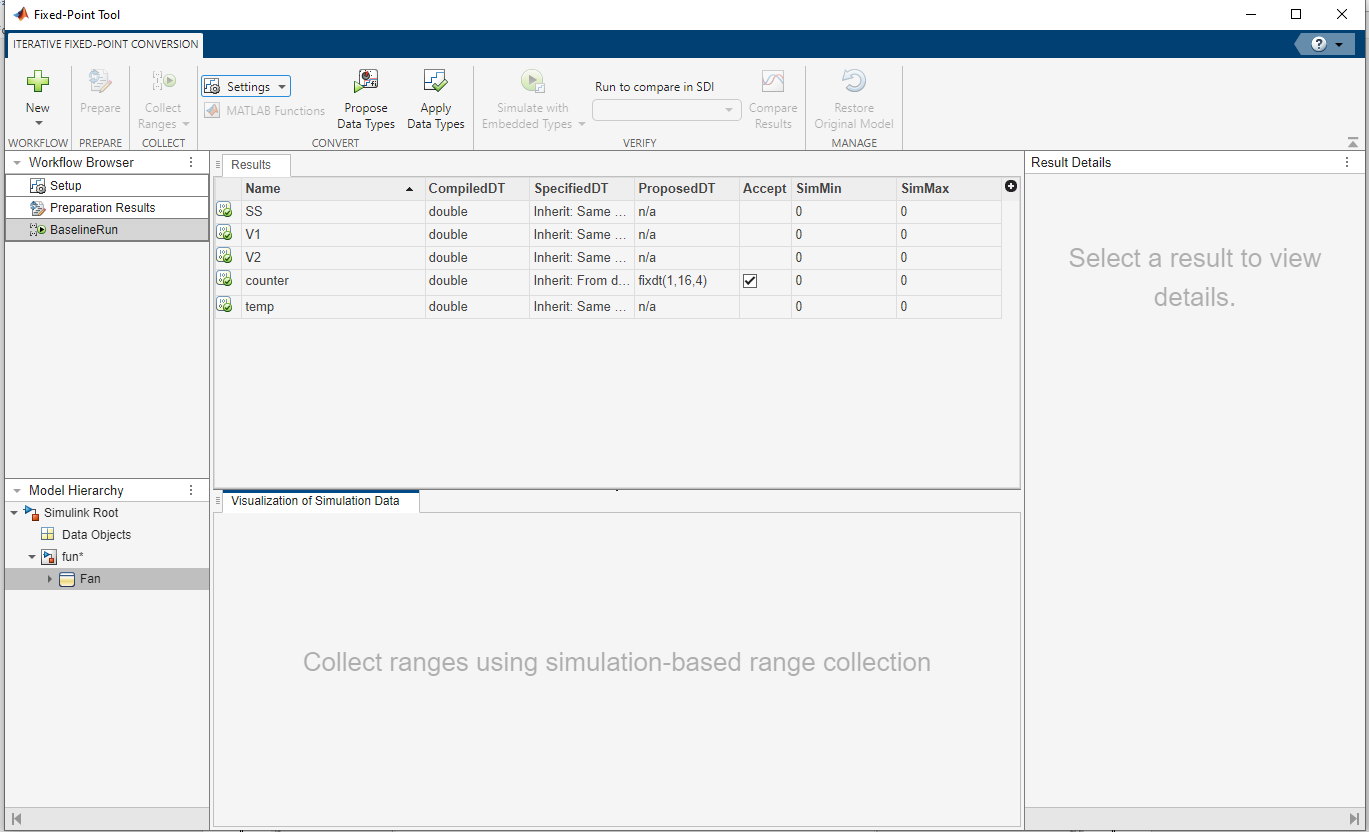












3

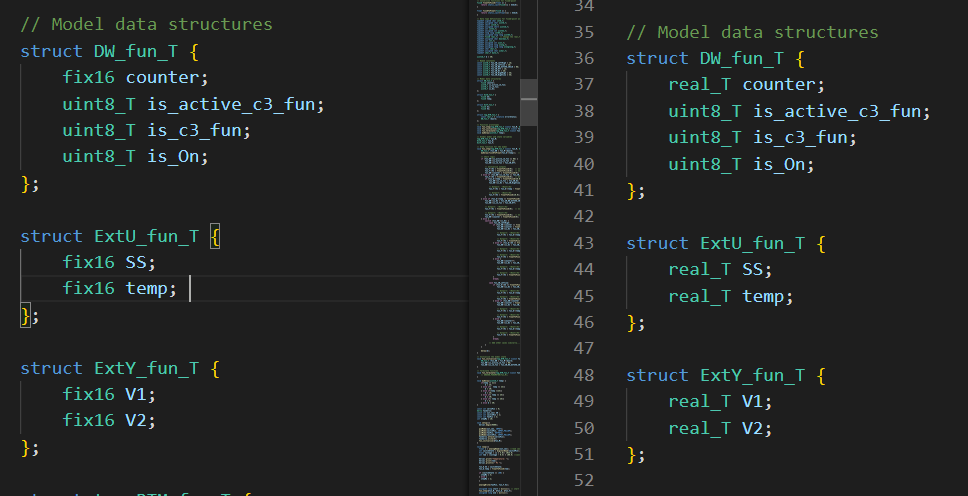
2

1

مراحل بالا را بصورت مرتب اجرا میگیریم.

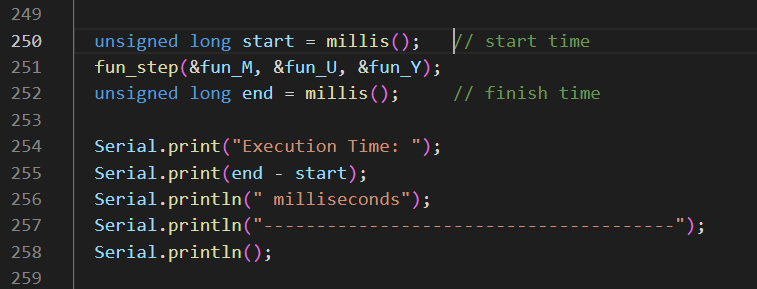
بخش ج) نمونه تغییرات که در کد در ایجاد شده است:

**بعد از اجرا fixed-Point**



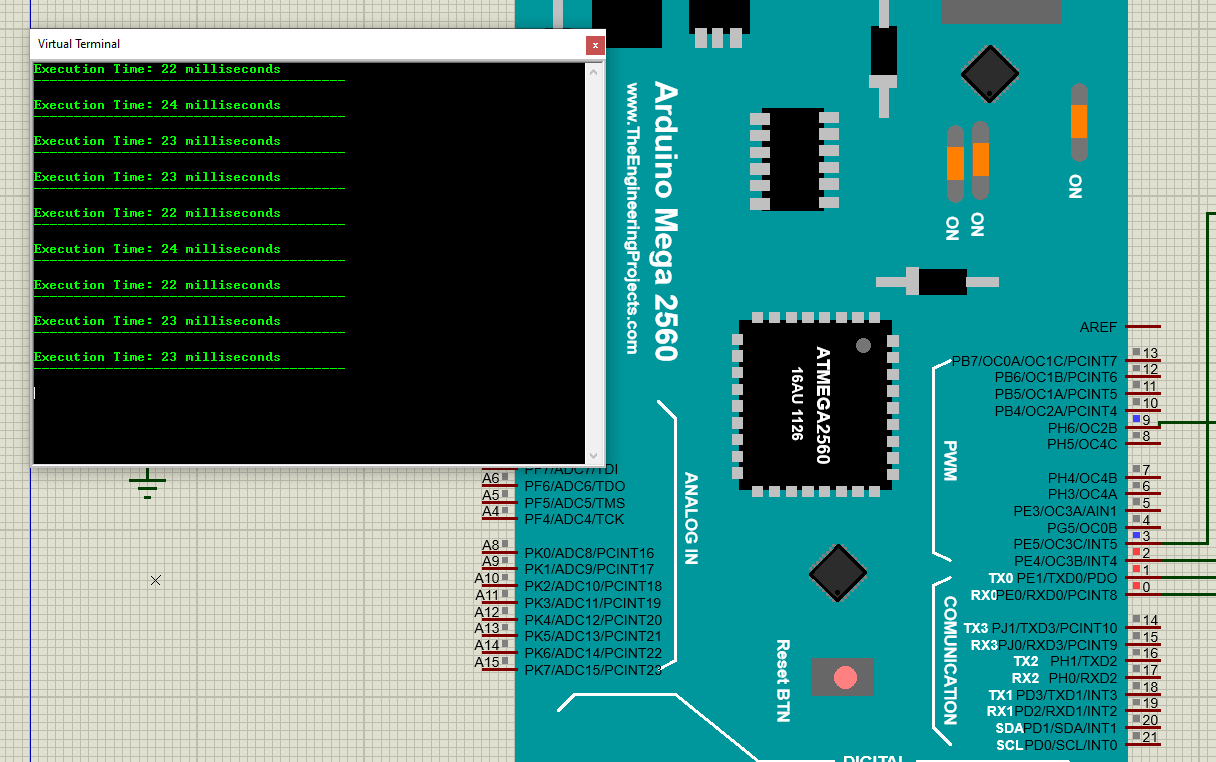
**قبل از اجرا fixed-Point**

**زمان اجرا تمام بهینه سازی ها را با استفاده از روش زیر ثبت می کنیم.**



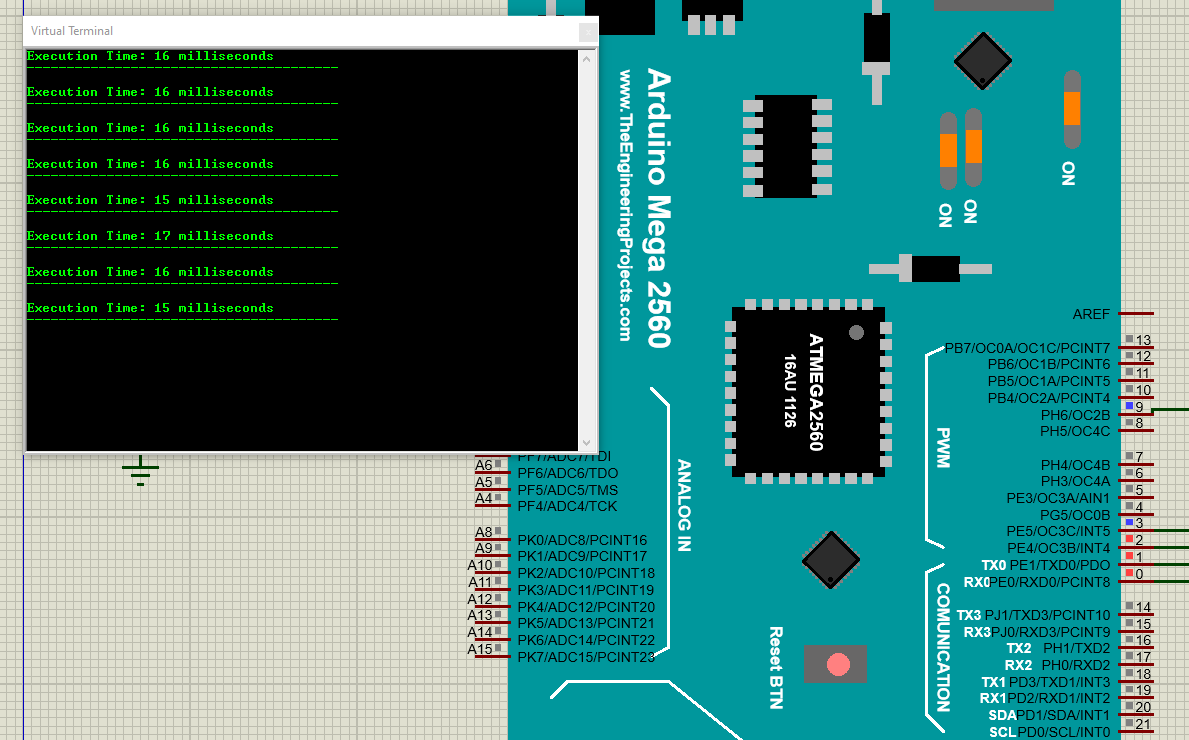
**تست حالت –O0**

**مدت زمان اجرا در حالت –O0** (حالت عادی) بین 24 تا 22 میلی ثانیه است.



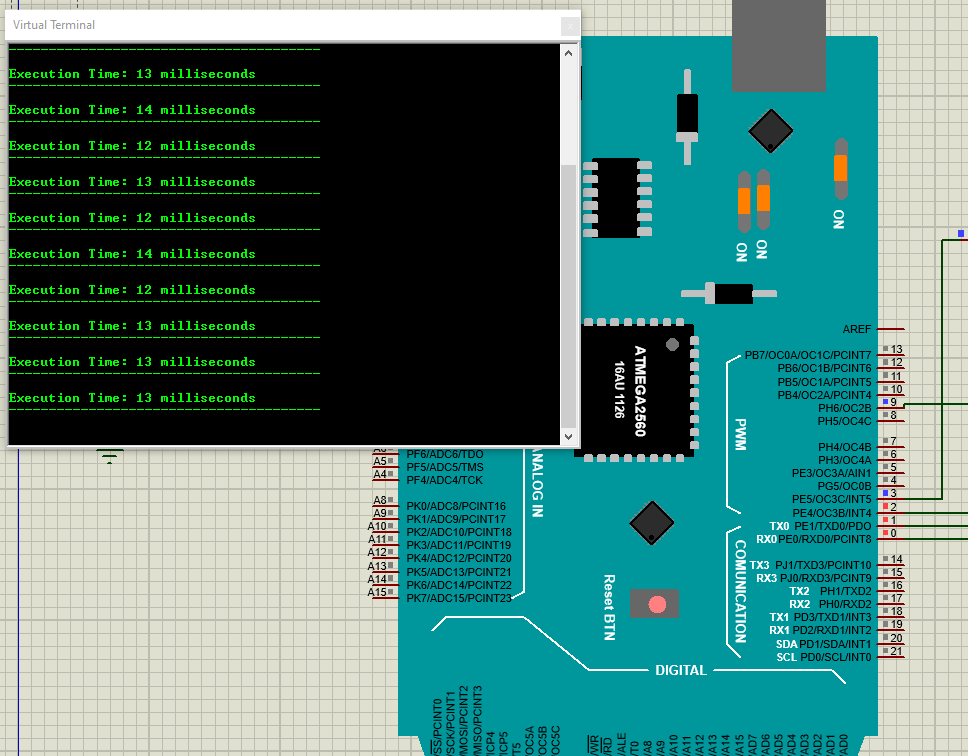
**تست حالت –O1**

**مدت زمان اجرا در حالت –O1** (بهینه‌سازی اولیه) بین 17 تا 15 میلی ثانیه است.

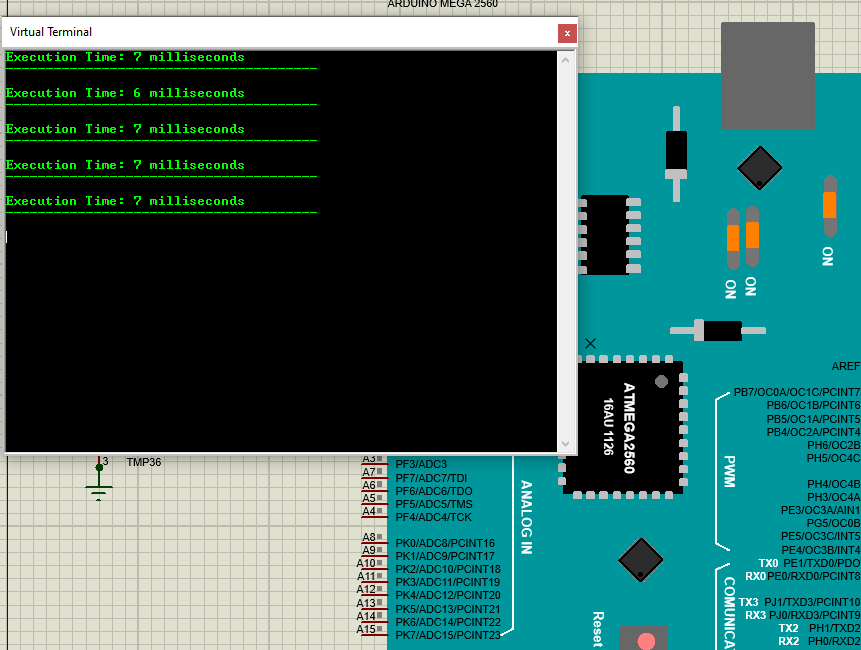


**تست حالت –Os**

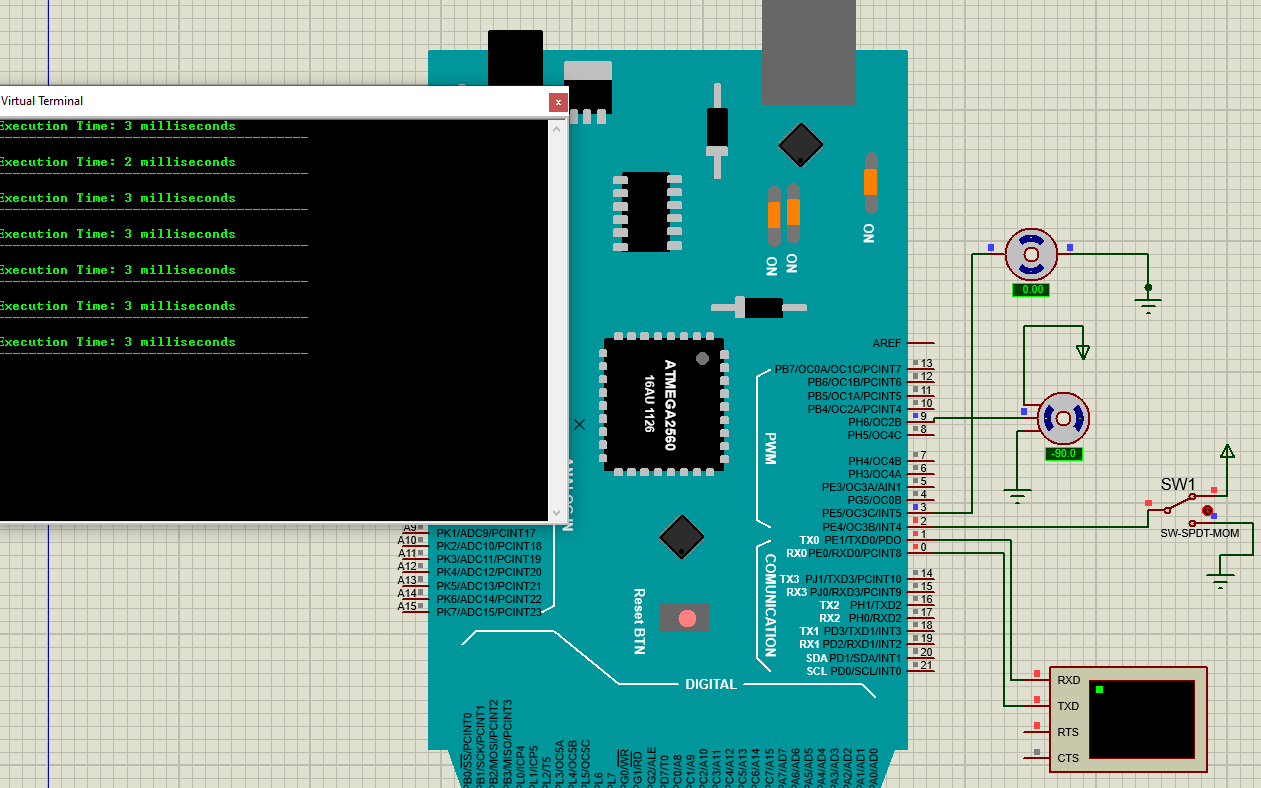
مدت زمان اجرا در حالت –: Os(بهینه‌سازی برای اندازه کوچکتر) بین 12 تا 14 میلی ثانیه است



**تست حالت –O2 : مدت زمان اجرا در حالت –O2** (بهینه‌سازی برای سرعت) بین 7 تا 6 میلی ثانیه است



**تست حالت –O3 : مدت زمان اجرا در حالت –O3** (بهینه‌سازی حداکثری) بین 2 تا 3 میلی ثانیه است



|  |  |  |  |
| --- | --- | --- | --- |
| سطح بهینه سازی | زمان اجرا قبلfixed | زمان اجرا بعد fixed | توضیحات |
| 00 - بدون بهینه سازی | 30 | 23 | بیشترین زمان اجرا، کمترین تلاش برای بهینه سازی. |
| 01 - بهینه سازی ابتدایی | 25 | 17 | بهینه سازی‌های ساده برای کاهش جزئی زمان اجرا. |
| OS - بهینه متوسط | 16 | 13 | کد تولید شده حجم کمتری دارد اما ممکن است سرعت اجرای آن متوسط باشد. |
| 02 - بهینه سازی برای سرعت | 9 | 7 | بهینه سازی‌های پیچیده برای افزایش قابل توجه سرعت. |
| 03 - بهینه سازی حداکثری برای سرعت | 4 | 3 or 2 | بهترین عملکرد، کمترین زمان اجرا، حداکثر تلاش برای بهینه سازی. |

تبدیل کد به **Fixed-Point** زمان اجرای برنامه را کاهش می‌دهد زیرا عملیات Fixed-Point ساده‌تر و سریع‌تر از عملیات Floating-Point است. این به دلایل زیر رخ می‌دهد:

1. **پیچیدگی کمتر در سخت‌افزار:**
   * عملیات Floating-Point مانند ضرب و تقسیم، به سخت‌افزار پیچیده‌تر و دستورالعمل‌های بیشتری نیاز دارد.
   * در مقابل، عملیات Fixed-Point می‌تواند با استفاده از دستورالعمل‌های ساده‌تر (جمع، تفریق، شیفت) انجام شود که در پردازنده‌های معمولی سریع‌تر اجرا می‌شوند.
2. **کاهش مصرف منابع پردازشی:**
   * Fixed-Point از اعداد صحیح استفاده می‌کند، که نیاز به محاسبات کمتری نسبت به Floating-Point دارد.
   * این امر باعث کاهش تعداد چرخه‌های پردازنده در هر عملیات می‌شود.
3. **بهینه‌سازی حافظه:**
   * اعداد Floating-Point فضای بیشتری در حافظه مصرف می‌کنند (مانند 32 یا 64 بیت برای اعداد IEEE 754)، در حالی که اعداد Fixed-Point معمولاً می‌توانند در فضای کوچکتری ذخیره شوند.
   * این منجر به کاهش تأخیر در عملیات حافظه می‌شود.
4. **هماهنگی بیشتر با معماری سیستم:**
   * بسیاری از میکروکنترلرها و پردازنده‌های توکار (embedded processors) برای عملیات صحیح (integer) بهینه شده‌اند. استفاده از Fixed-Point به آن‌ها اجازه می‌دهد از توانایی‌های سخت‌افزاری خود بهتر استفاده کنند.

### نتیجه:

پس از تبدیل کد به Fixed-Point، عملیات ساده‌تر و سریع‌تر شده و زمان اجرا کاهش یافته است، در حالی که عملکرد کلی کد بهبود یافته است. البته این تبدیل نیاز به تحلیل دقیق دارد تا دقت محاسبات حفظ شود و بهینه‌سازی قابل قبول باقی بماند.