بسم الله الرحمن الرحیم

گزارش تمرین 7

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 ؟

```
for (i = 0; i < N; i++) {
    D[i] = A[i+1] * 2;
}
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (i > 0)
            A[j] = B[i] + C[i-1];
        else
            A[j] = B[i];
    }
}
```

جواب اول)

أ. 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.

```
for (int i = 0; i < N; i++) {
    D[i] = A[i+1] * 2;
    for (int j = 0; j < N; j++) {
        if (i > 0)
            A[j] = B[i] + C[i-1];
        else
            A[j] = B[i];
    }
}
```

ب. 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.

```
for (i = 0; i < N; i += 2) {
    D[i] = A[i+1] * 2;
    for (j = 0; j < N; j++) {
        if (i > 0)
            A[j] = B[i] + C[i-1];
        else
            A[j] = B[i];
    }
    if (i+1 < N) {
        D[i+1] = A[i+2] * 2;
        for (j = 0; j < N; j++) {
            if (i+1 > 0)
                A[j] = B[i+1] + C[i];
            else
                A[j] = B[i+1];
        }
    }
}
```

ت. 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.

```
for (i = 0; i < N; i++) {
    D[i] = A[i+1] * 2;
}
for (i = 0; i < N; i++) {
    if (i > 0) {
        for (j = 0; j < N; j++) {
            A[j] = B[i] + C[i-1];
        }
    } else {
        for (j = 0; j < N; j++) {
            A[j] = B[i];
        }
    }
}
```

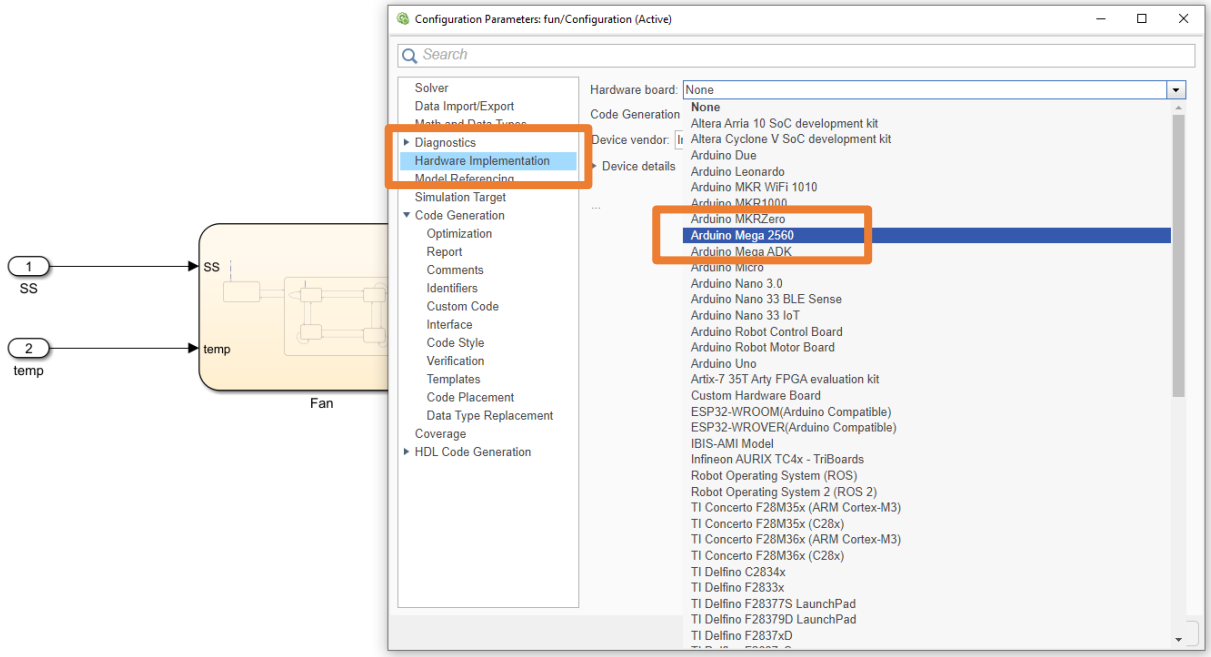
ث. تبدیل blocking/tiling loop برای حلقه داخلی به صورت پارامتری با پیش فرض block size = 16

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.

```
int blockSize = 16;
for (i = 0; i < N; i++) {
    D[i] = A[i+1] * 2;
    for (int jj = 0; jj < N; jj += blockSize) {
        for (j = jj; j < jj + blockSize && j < N; j++) {
            if (i > 0)
                A[j] = B[i] + C[i-1];
            else
                A[j] = B[i];
        }
    }
}
```

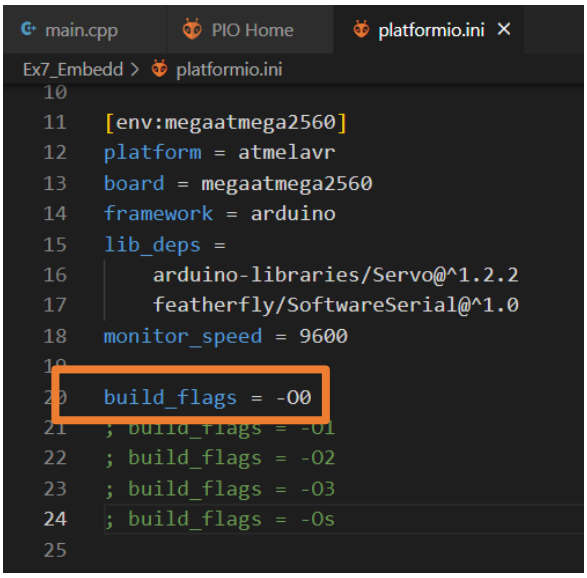
جواب دوم)

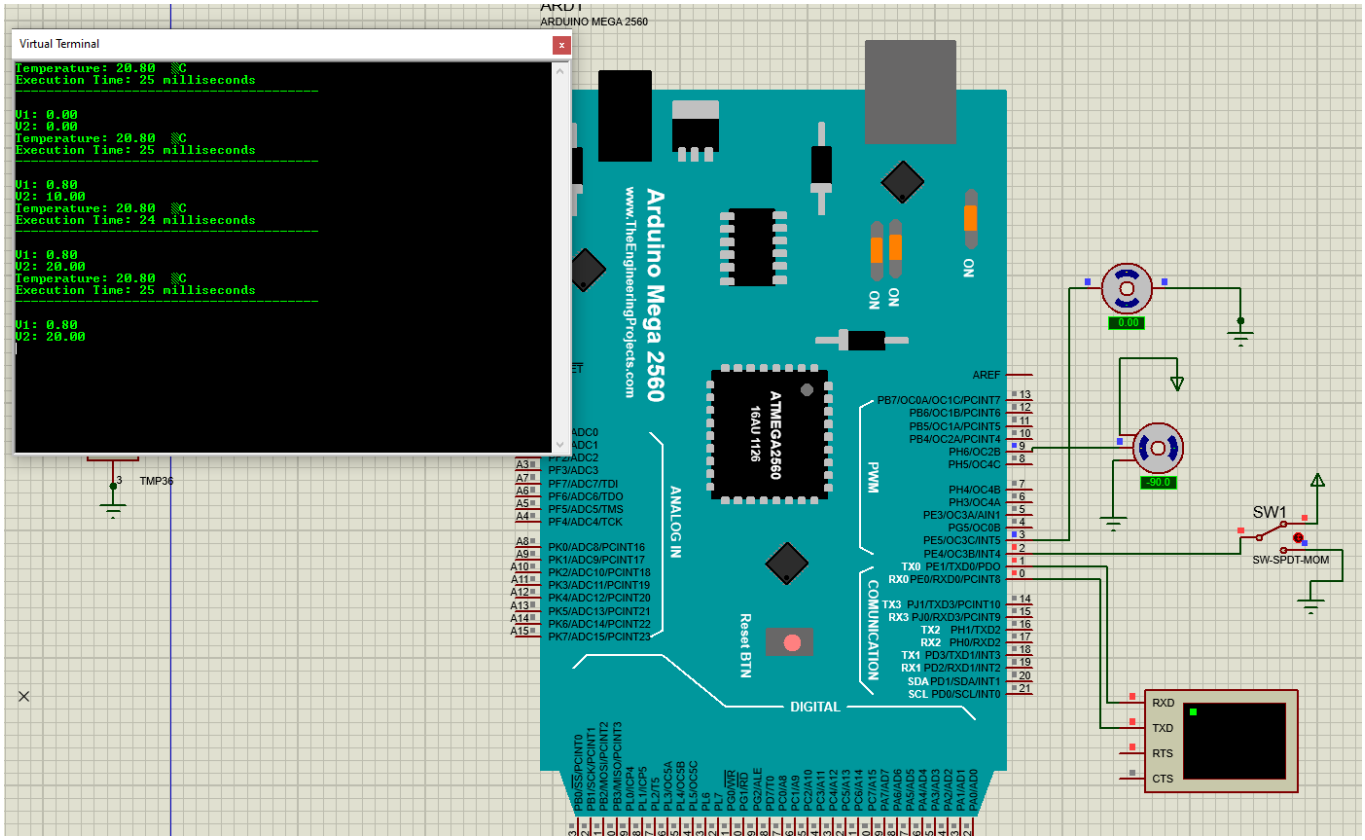
بخش الف) کد مدل را برای بورد Arduino Mega 2560 در Simulink بصورت زیر تولید میکنیم.



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.

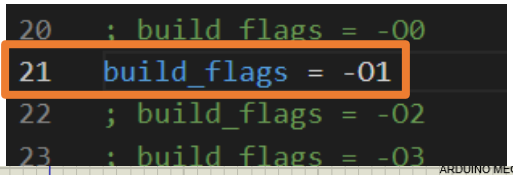
تست حالت O0–

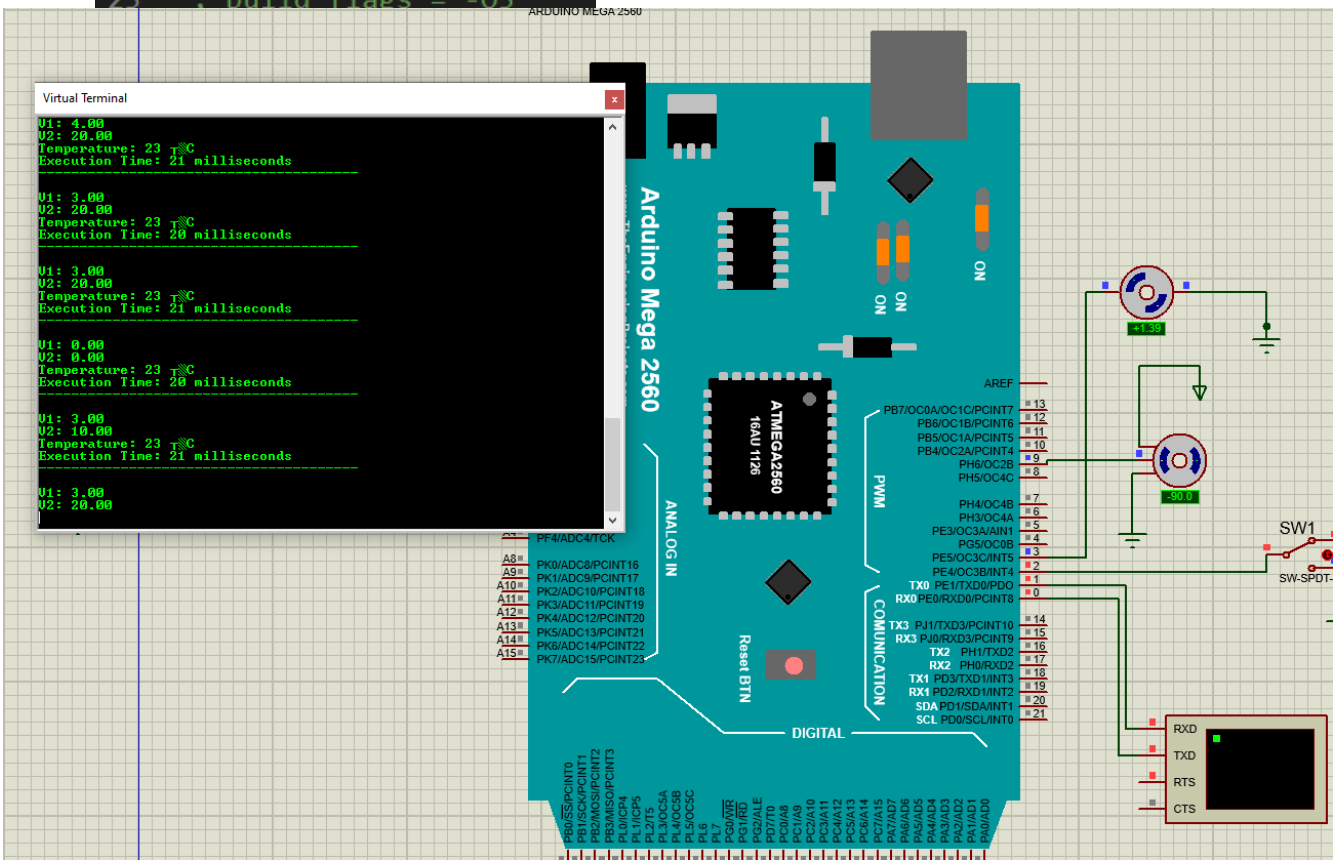




مدت زمان اجرا در حالت O0– (حالت عادی) بین 26 تا 24 میلی ثانیه است.

تست حالت O1–





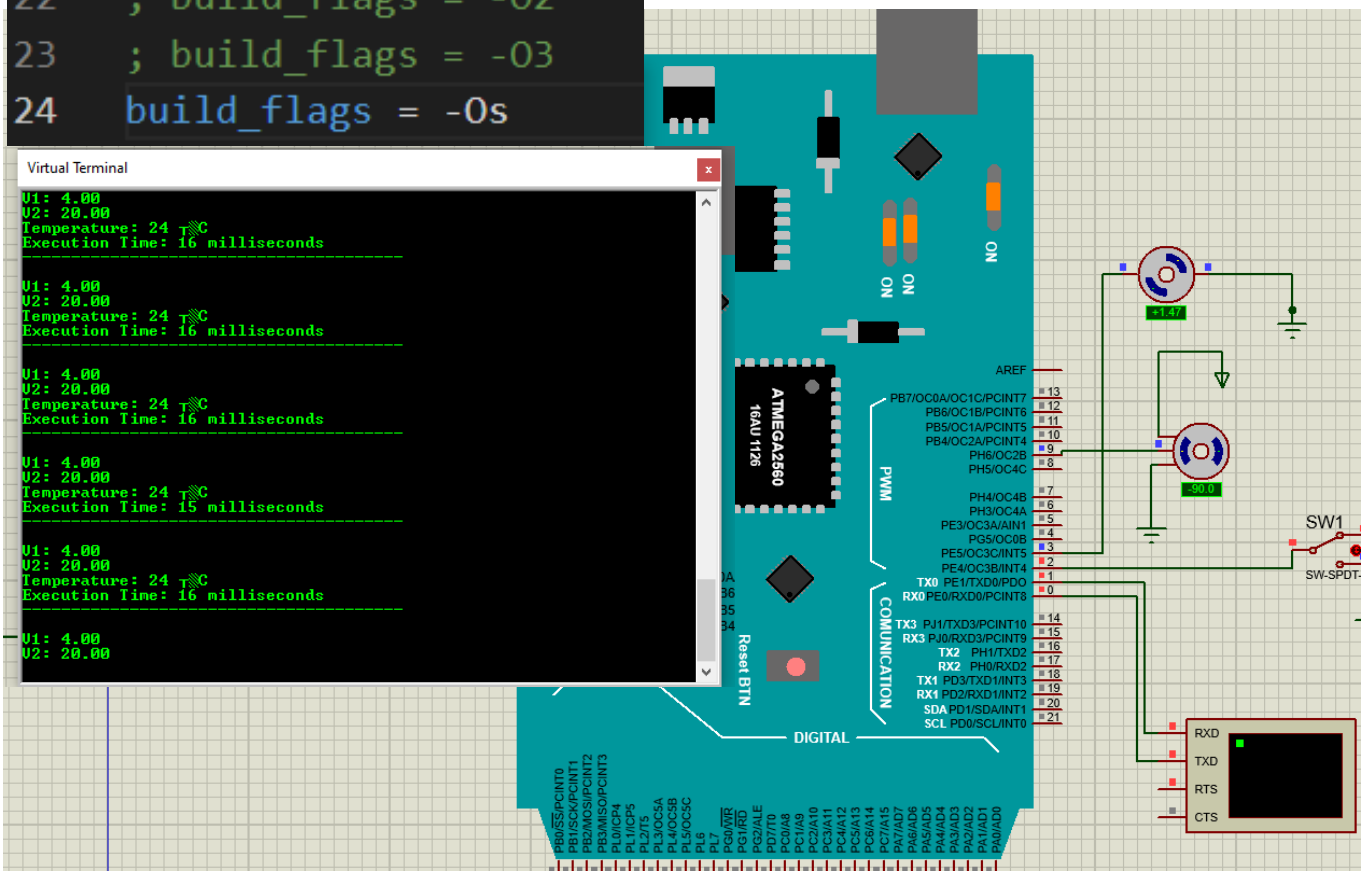
مدت زمان اجرا در حالت O1– (بهینه‌سازی اولیه) بین 21 تا 20 میلی ثانیه است.

تست حالت Os–

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

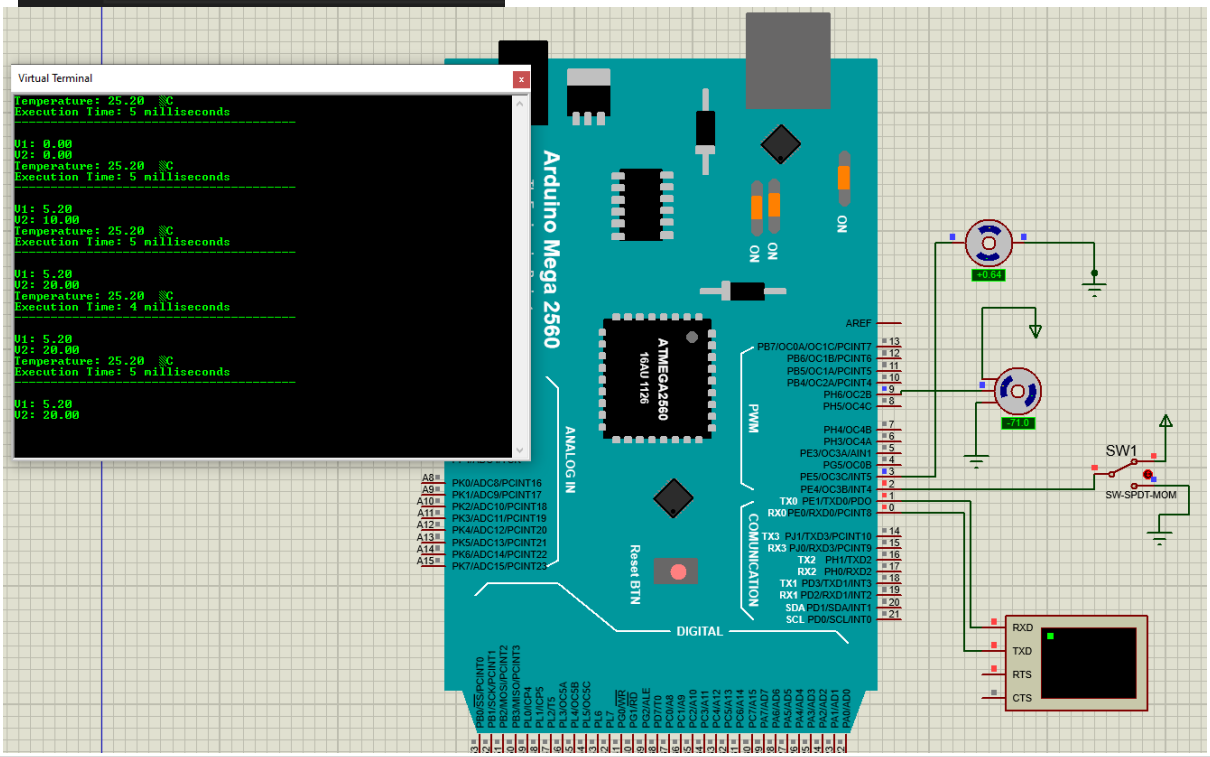
بین 16 تا 15 میلی ثانیه است.

```
; build_flags = -O0
; build_flags = -O1
; build_flags = -O2
; build_flags = -O3
build_flags = -Os
```



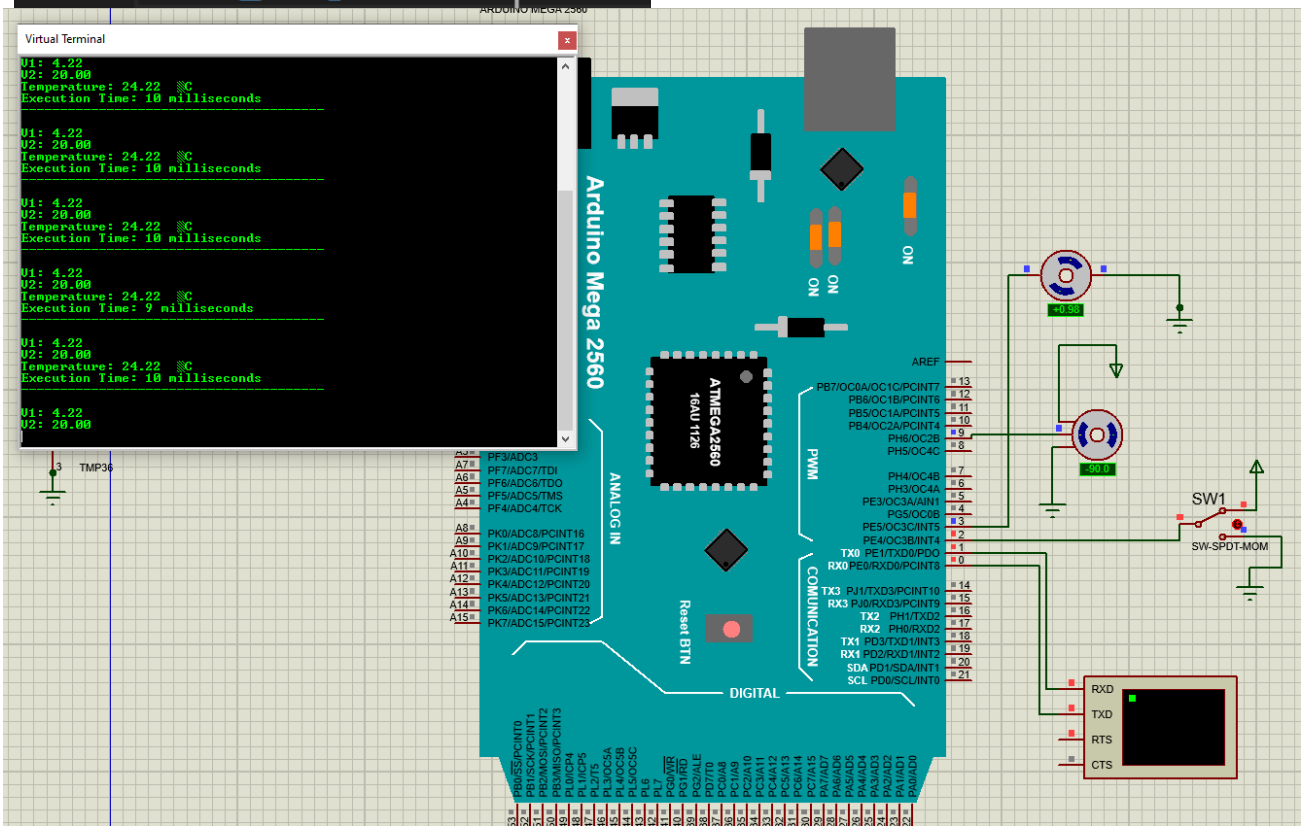
تست حالت O3– : مدت زمان اجرا در حالت O3– (بهینه‌سازی حداکثری) بین 5 تا 4 میلی ثانیه است

```
; build_flags = -O0
; build_flags = -O1
; build_flags = -O2
build_flags = -O3
```



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

```
; build_flags = -O0
; build_flags = -O1
build_flags = -O2
```



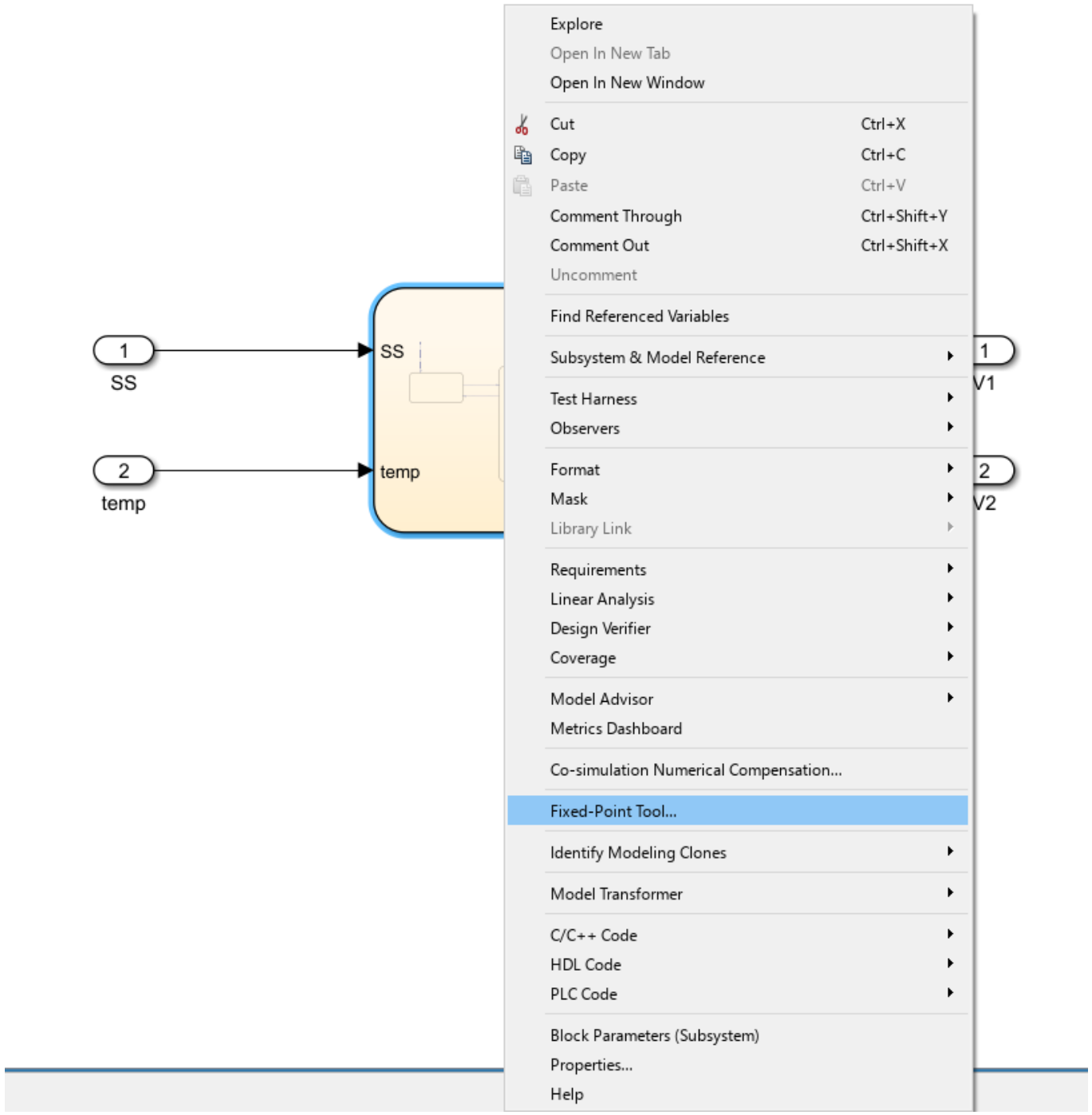
۱. تحلیل عملکرد بهینه‌سازی

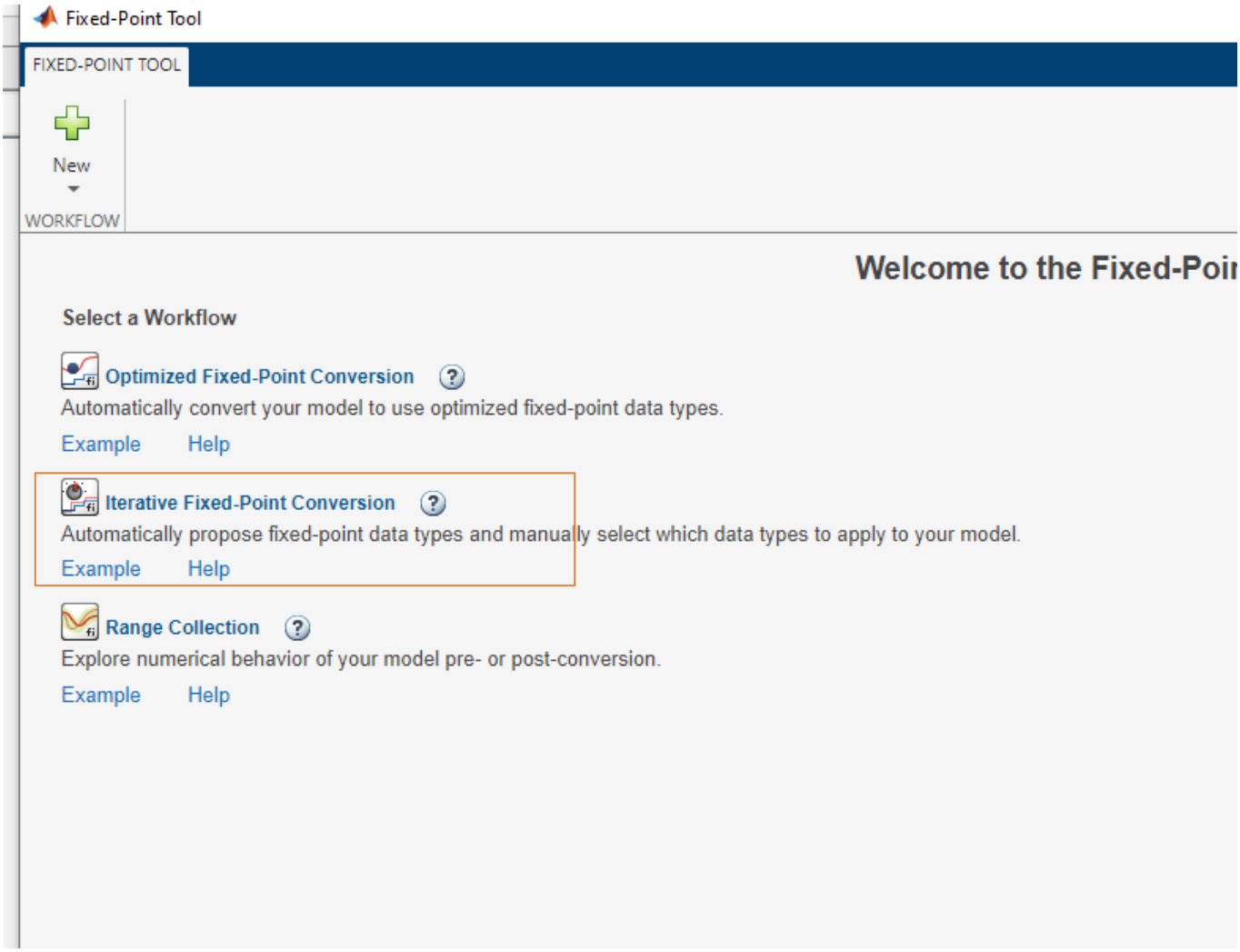
سطح بهینه‌سازی	زمان اجرا (میلی‌ثانیه)	توضیحات
Os-	16	بهینه برای کاهش اندازه کد، با عملکرد متوسط.
O0-	30	بدون بهینه‌سازی، بیشترین زمان اجرا.
O1-	25	بهینه‌سازی ابتدایی، کاهش کمی در زمان.
O2-	9	بهینه‌سازی برای سرعت، کاهش چشمگیر.
O3-	4	بهینه‌سازی حداکثری برای سرعت، بهترین عملکرد.

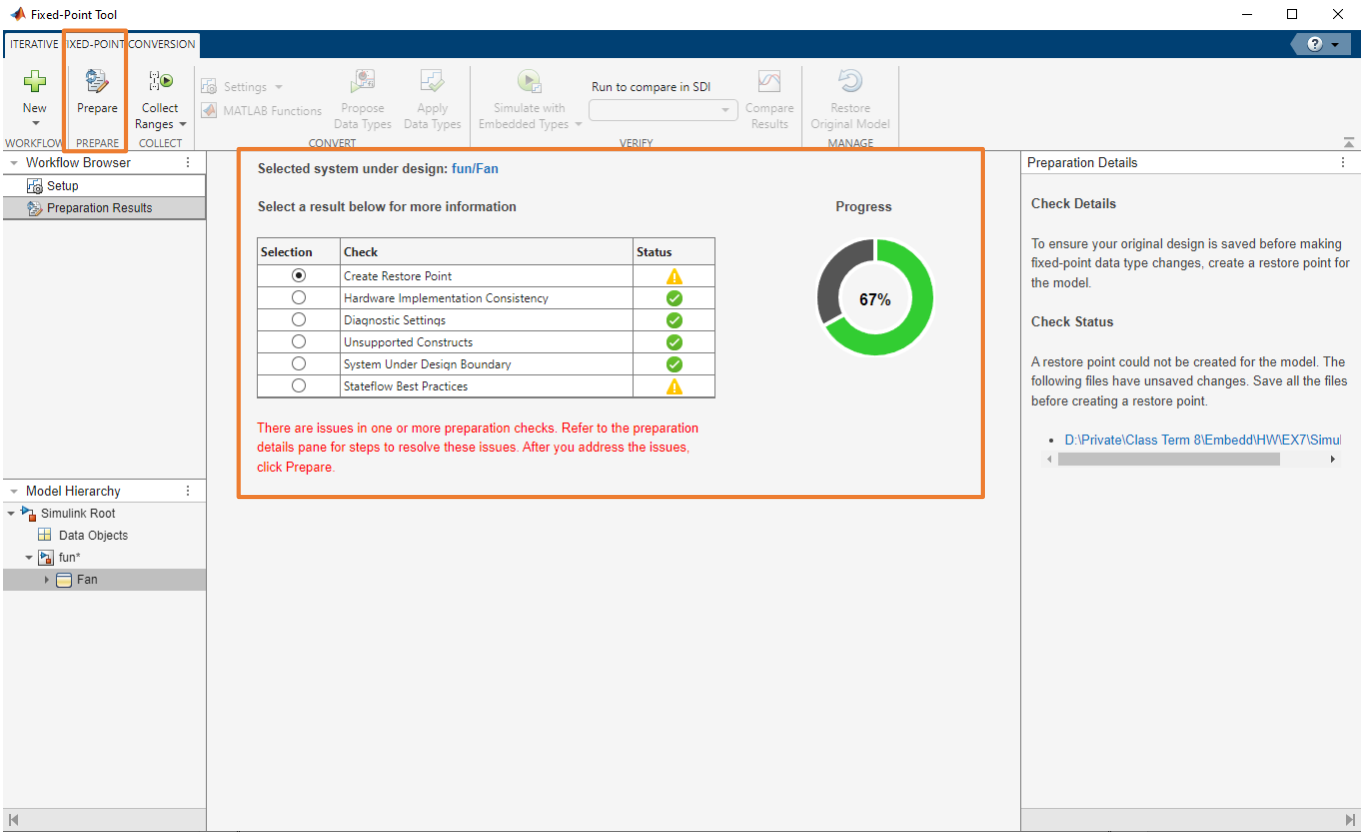
روند تغییرات

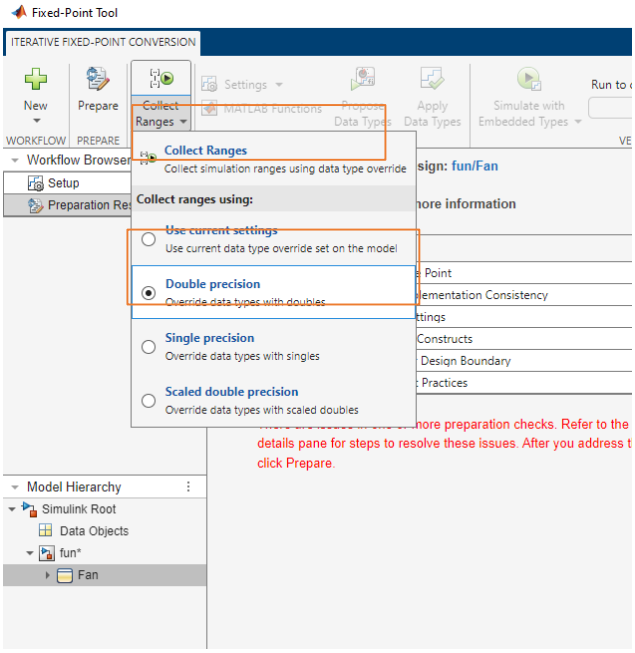
- **بدون بهینه‌سازی:(O0-)**
  - زمان اجرای طولانی نشان می‌دهد که کد مستقیماً بدون هرگونه بهینه‌سازی اجرا شده است.
- **بهینه‌سازی ابتدایی:(O1-)**
  - کاهش کمی در زمان اجرا، بیشتر مربوط به حذف دستورهای غیرضروری است.
- **بهینه‌سازی برای سرعت:(O2-)**
  - بهبود چشمگیر در عملکرد با کاهش زمان اجرا از 30 به 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.

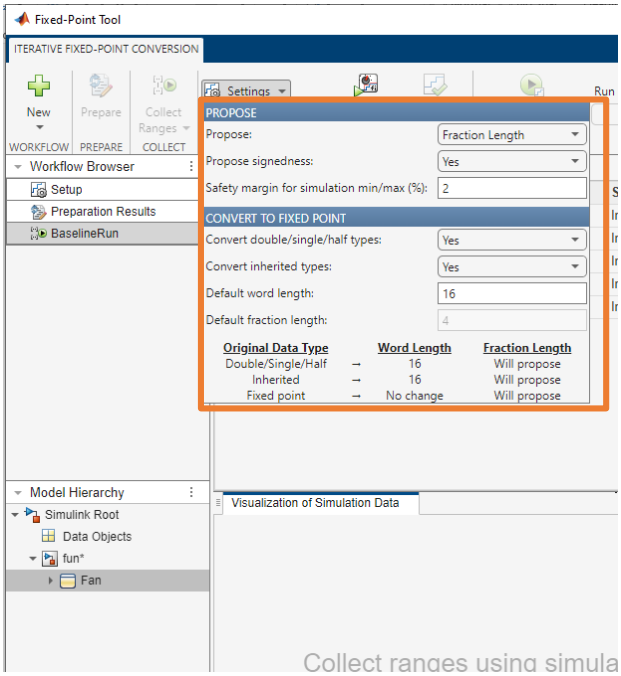
بخش ب) مدل را با استفاده از ابزار Fixed-Point

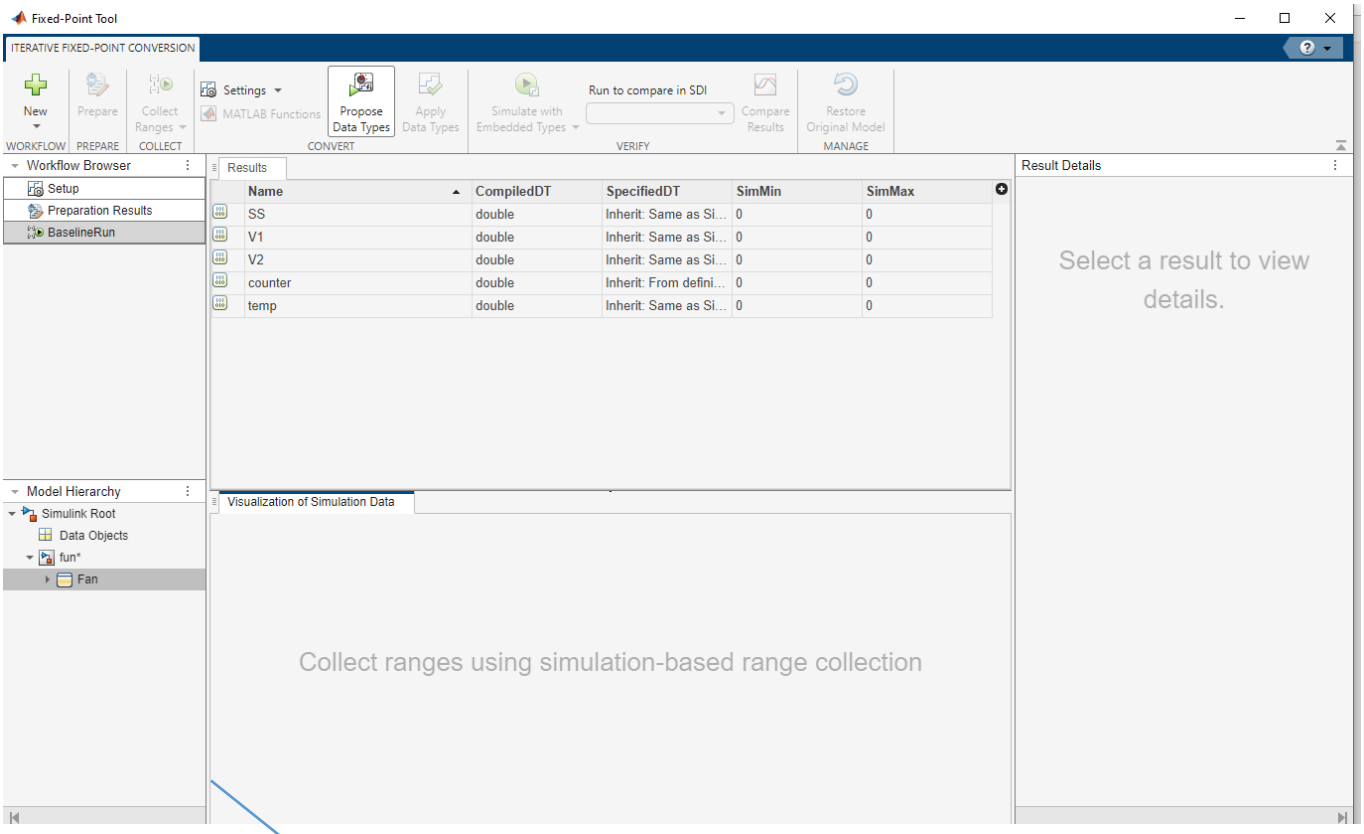


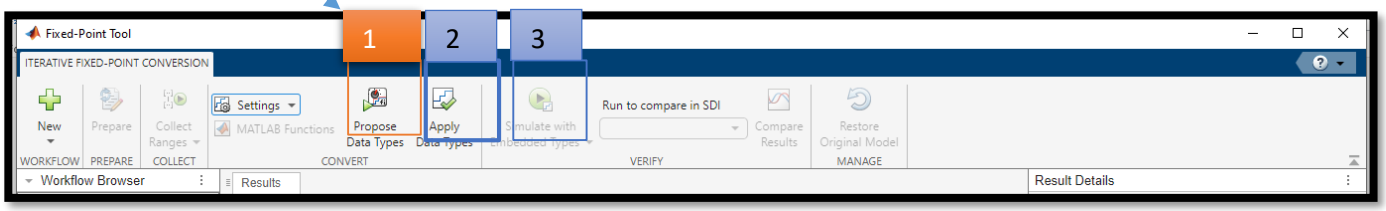

Fixed-Point Tool
FIXED-POINT TOOL
New
WORKFLOW
Welcome to the Fixed-Poi
Select a Workflow
Optimized Fixed-Point Conversion
Automatically convert your model to use optimized fixed-point data types.
Example Help
Iterative Fixed-Point Conversion
Automatically propose fixed-point data types and manually select which data types to apply to your model.
Example Help
Range Collection
Explore numerical behavior of your model pre- or post-conversion.
Example Help


Fixed-Point Tool
ITERATIVE FIXED-POINT CONVERSION
New
Prepare
Collect Ranges
Settings
MATLAB Functions
Propose Data Types
Apply Data Types
Simulate with Embedded Types
Run to compare in SDI
Compare Results
Restore Original Model
WORKFLOW
PREPARE
COLLECT
CONVERT
VERIFY
MANAGE
Workflow Browser
Setup
Preparation Results
Model Hierarchy
Simulink Root
Data Objects
fun*
Fan
Selected system under design: fun/Fan
Select a result below for more information
Progress
67%
Selection
Check
Status
Create Restore Point
Hardware Implementation Consistency
Diagnostic Settings
Unsupported Constructs
System Under Design Boundary
Stateflow Best Practices
There are issues in one or more preparation checks. Refer to the preparation details pane for steps to resolve these issues. After you address the issues, click Prepare.
Preparation Details
Check Details
To ensure your original design is saved before making fixed-point data type changes, create a restore point for the model.
Check Status
A restore point could not be created for the model. The following files have unsaved changes. Save all the files before creating a restore point.
D:\Private\Class Term 8\Embedd\HW\EX7\Simul









مراحل بالا را بصورت مرتب اجرا میگیریم.

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

قبل از اجرا fixed-Point

```
36  struct DW_fun_T {
37      real_T counter;
38      uint8_T is_active_c3_fun;
39      uint8_T is_c3_fun;
40      uint8_T is_On;
41  };
42
43  struct ExtU_fun_T {
44      real_T SS;
45      real_T temp;
46  };
47
48  struct ExtY_fun_T {
49      real_T V1;
50      real_T V2;
51  };
52
```

بعد از اجرا fixed-Point

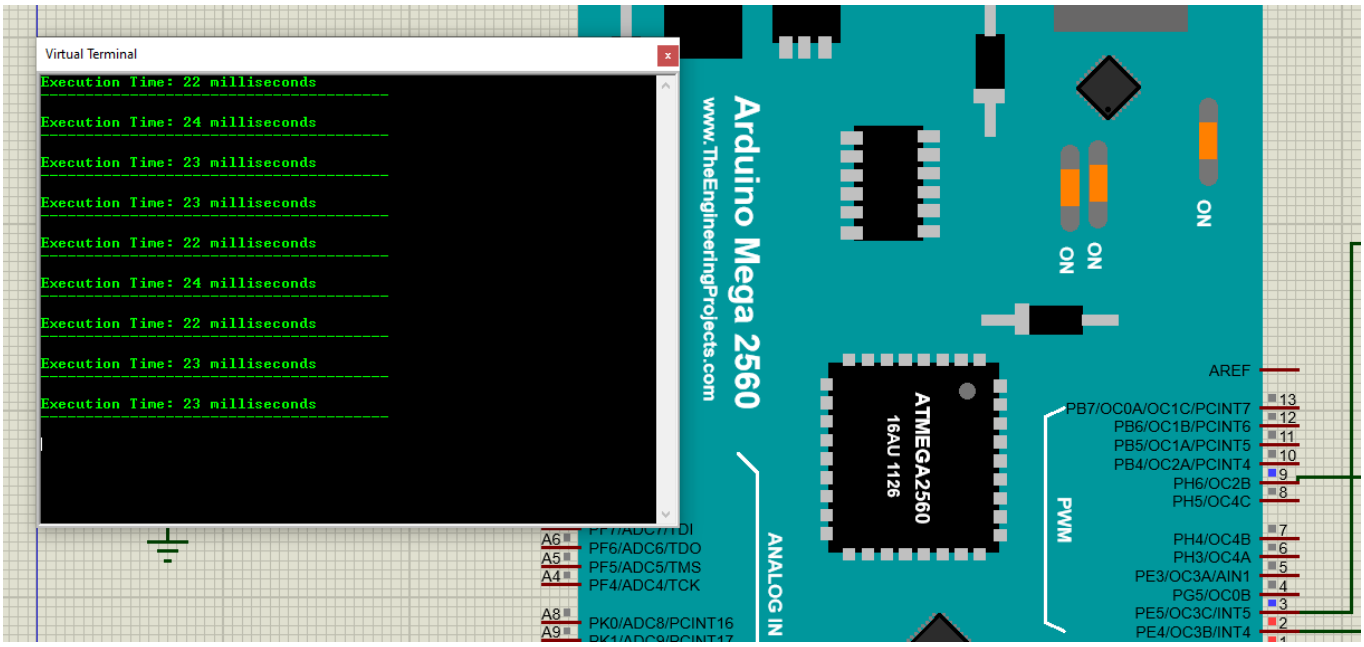
```
struct DW_fun_T {
    fix16 counter;
    uint8_T is_active_c3_fun;
    uint8_T is_c3_fun;
    uint8_T is_On;
};

struct ExtU_fun_T {
    fix16 SS;
    fix16 temp;
};

struct ExtY_fun_T {
    fix16 V1;
    fix16 V2;
};
```

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

```
249
250     unsigned long start = millis();   // start time
251     fun_step(&fun_M, &fun_U, &fun_Y);
252     unsigned long end = millis();     // finish time
253
254     Serial.print("Execution Time: ");
255     Serial.print(end - start);
256     Serial.println(" milliseconds");
257     Serial.println("----------------------------------------");
258     Serial.println();
259
```

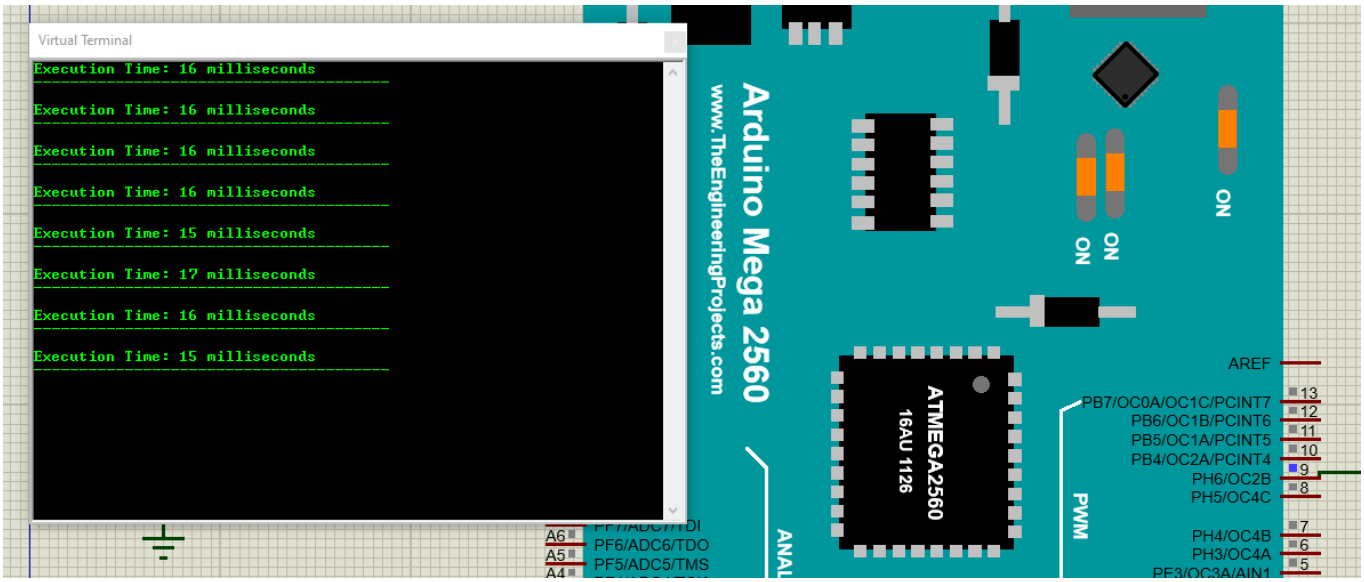
تست حالت O0–

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



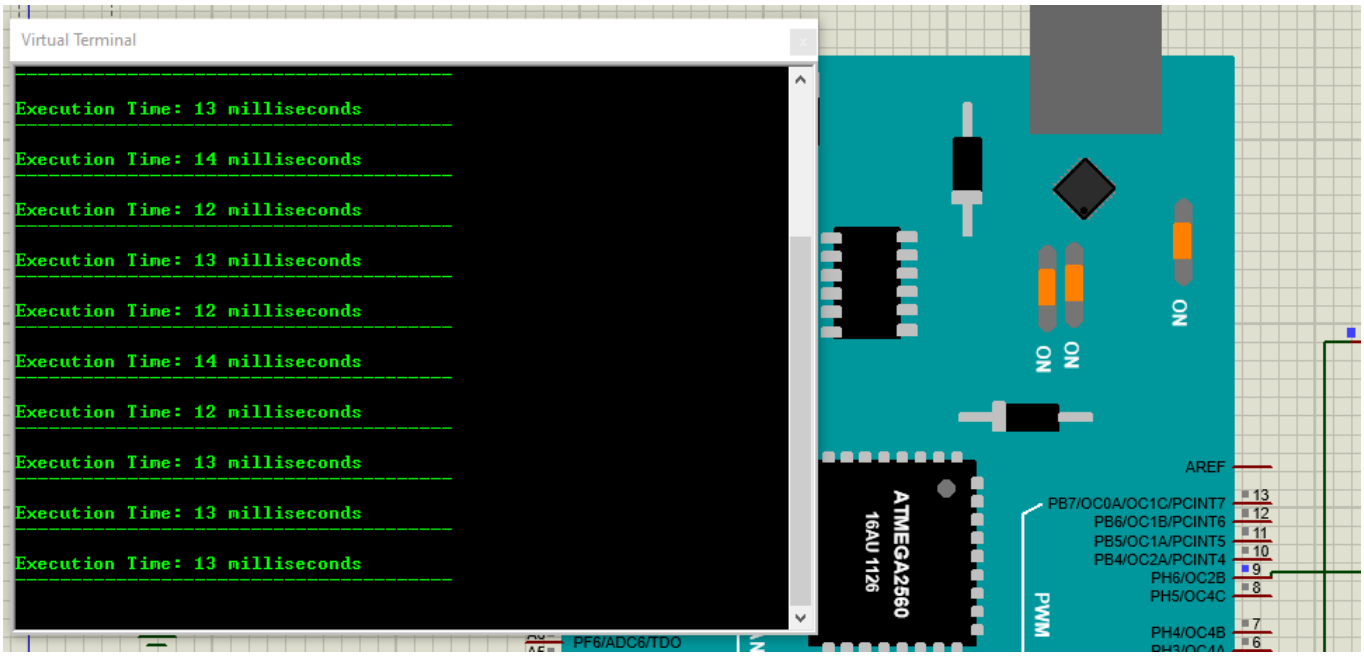
تست حالت O1–

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

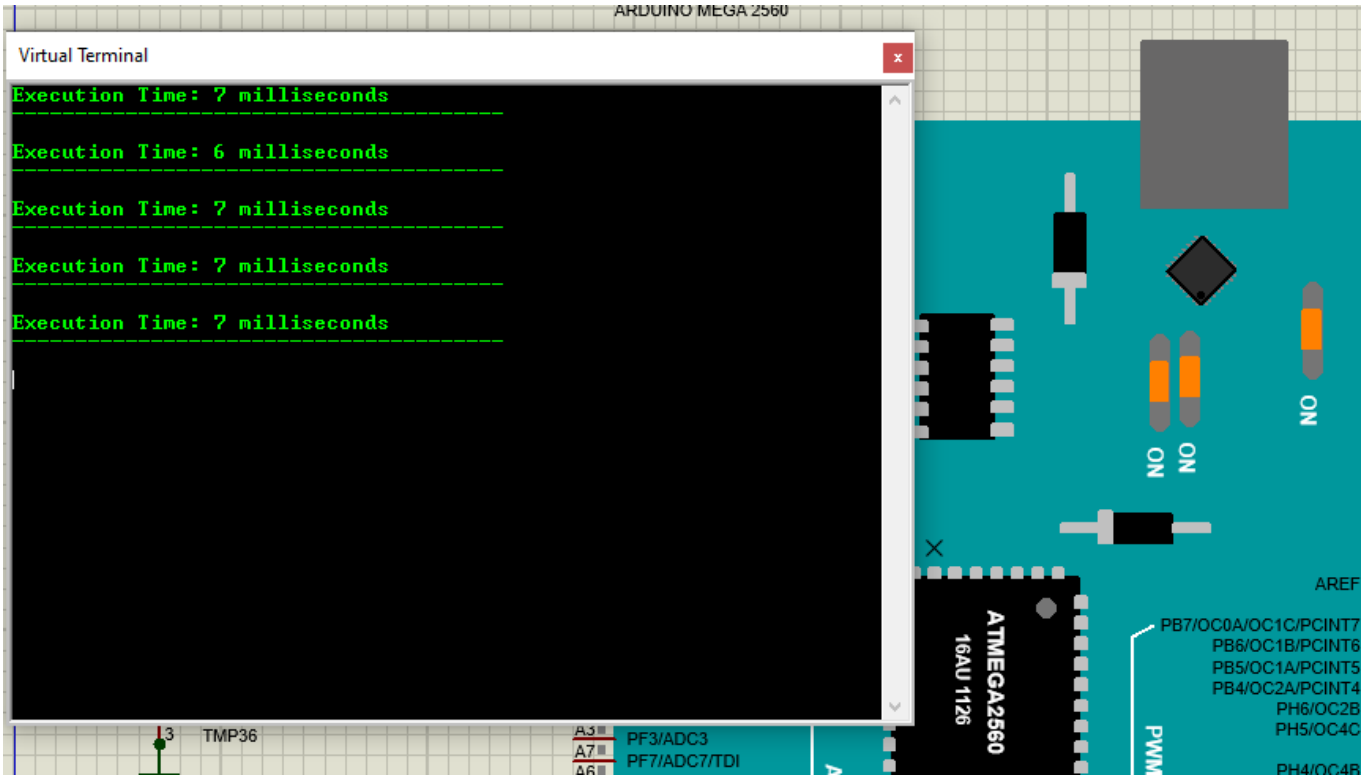


تست حالت –Os

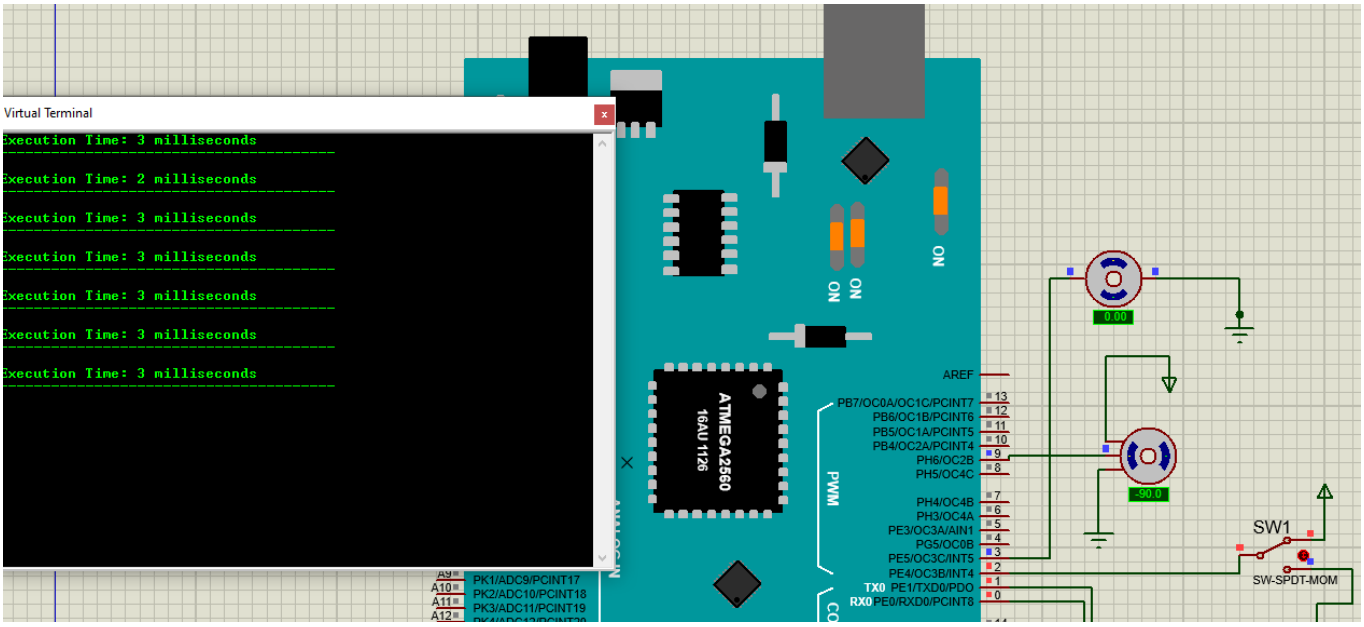
مدت زمان اجرا در حالت –Os :(بهینه‌سازی برای اندازه کوچکتر) بین 12 تا 14 میلی ثانیه است



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
بدون بهینه سازی - 00	30	23	بیشترین زمان اجرا، کمترین تلاش برای بهینه سازی.
بهینه سازی ابتدایی - 01	25	17	بهینه سازی‌های ساده برای کاهش جزئی زمان اجرا.
بهینه متوسط - OS	16	13	کد تولید شده حجم کمتری دارد اما ممکن است سرعت اجرای آن متوسط باشد.
بهینه سازی برای سرعت - 02	9	7	بهینه سازی‌های پیچیده برای افزایش قابل توجه سرعت.
بهینه سازی حداکثری - 03 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-Pointاز 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 بهFixed-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.