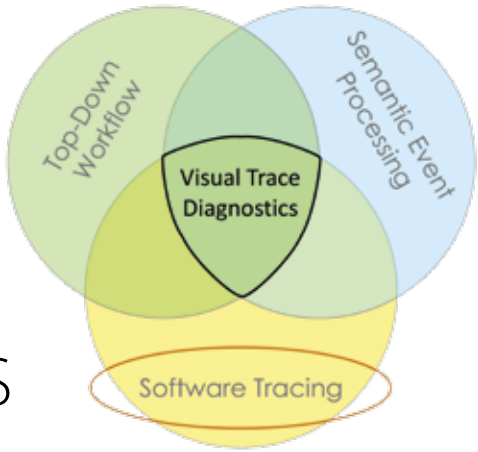
Säkerhet och tillförlitlighet

i Sakernas Internet med Zephyr RTOS



Visuell spårningsdiagnostik kan liknas vid en övervakningskamera som registrerar hur programkoden uppträder under körning och upprättar en grafisk tidslinje som kan användas som utgångspunkt för felsökning och mer detaljerad analys.

Projektet Zephyr (www.zephyrproject.org) har ägnat de senaste fem åren åt att utveckla ett öppet realtidsoperativsystem (RTOS) för IoT applikationer. Zephyr RTOS har i skrivande stund stöd för över 200 utvecklingskort med flera olika processorfamiljer inklusive Arm, RISC-V, Tensilica, NIOS och Arc, och stödjer även multipla processorkärnor. Det finns också inbyggt stöd för trådlös kommunikation över Bluetooth LE, Wifi och 802.15.4 Matter (det som tidigare hette Zigbee), samt stöd för en rad andra standardgränssnitt som Ethernet, USB, CAN-buss, och 6LoWPAN.

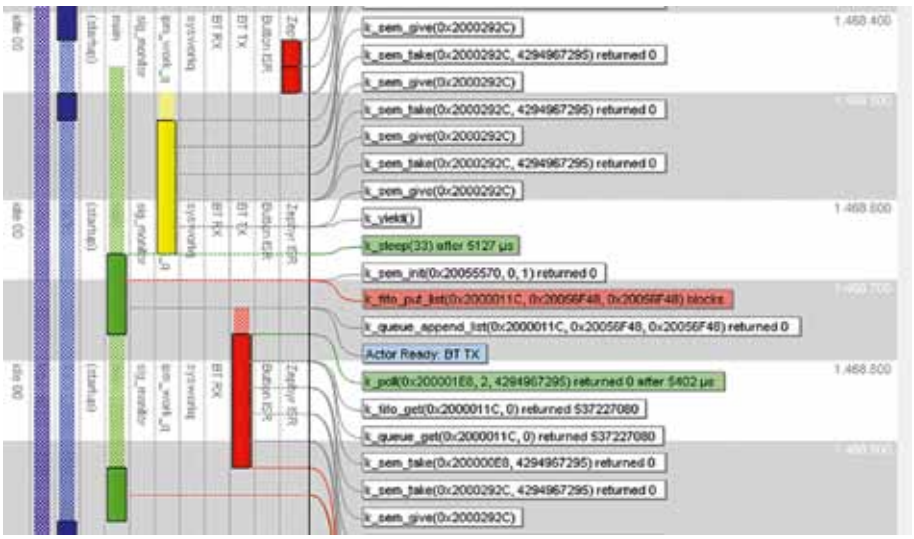
En viktig detalj i sammanhanget är att utvecklingen av Zephyr RTOS har fokus på säkerhet. Projektet drivs inom The Linux Foundation, vilket bland annat innebär att det finns ett akutteam med uppgift att ta emot och agera på rapporter om säkerhetsproblem och att kodbasen hålls i ett sådant skick att den ska klara en säkerhetscertifiering. Uppdaterade versioner av operativsystemet publiceras ungefär var tredje månad och det finns även en LTS-version (”Long Term Support”) för användare som vill ha en stabil plattform med enbart säkerhetsuppdateringar.

Detta ger snabbare utveckling av uppkopplade inbyggda system och med hög säkerhet, samt stor flexibilitet i valet av både hårdvaruplattform och molnleverantör. De två ledande alternativen FreeRTOS och Azure RTOS ThreadX är knutna till de två dominerande molnleverantörerna, Amazon Web Services och Microsoft Azure, medan Zephyr erbjuder ett helt oberoende alternativ.

Den pågående pandemin har skapat nya tillämpningar av Sakernas Internet, exempelvis kontaktspårningsutrustning som avståndsmätare och även ”smarta” säkerhetsskor. Med hjälp av Zephyr RTOS har olika tillverkare kunnat ta fram den här typen av prylar på bara några månader, tack vare operativsystemets tillförlitlighet och integrerade kommunikationsstackar.

Av Johan Kraft, Percepio

Dr. Johan Kraft är vd och grundare av Percepio AB. Han har också utvecklat den första versionen av Percepio Tracealyzer, ett verktyg för visuell spårningsdiagnostik som ger utvecklarna insikt i programvarans beteende under körning och underlättar utvecklingen av inbyggnadssystem. Innan Percepio grundades 2009 arbetade Johan Kraft med utveckling av inbyggda system hos ABB Robotics. Han har doktorerat i datavetenskap.



En visualiserad spårningssession som visar schemaläggningen i Zephyr, tillsammans med bl a API-anrop.

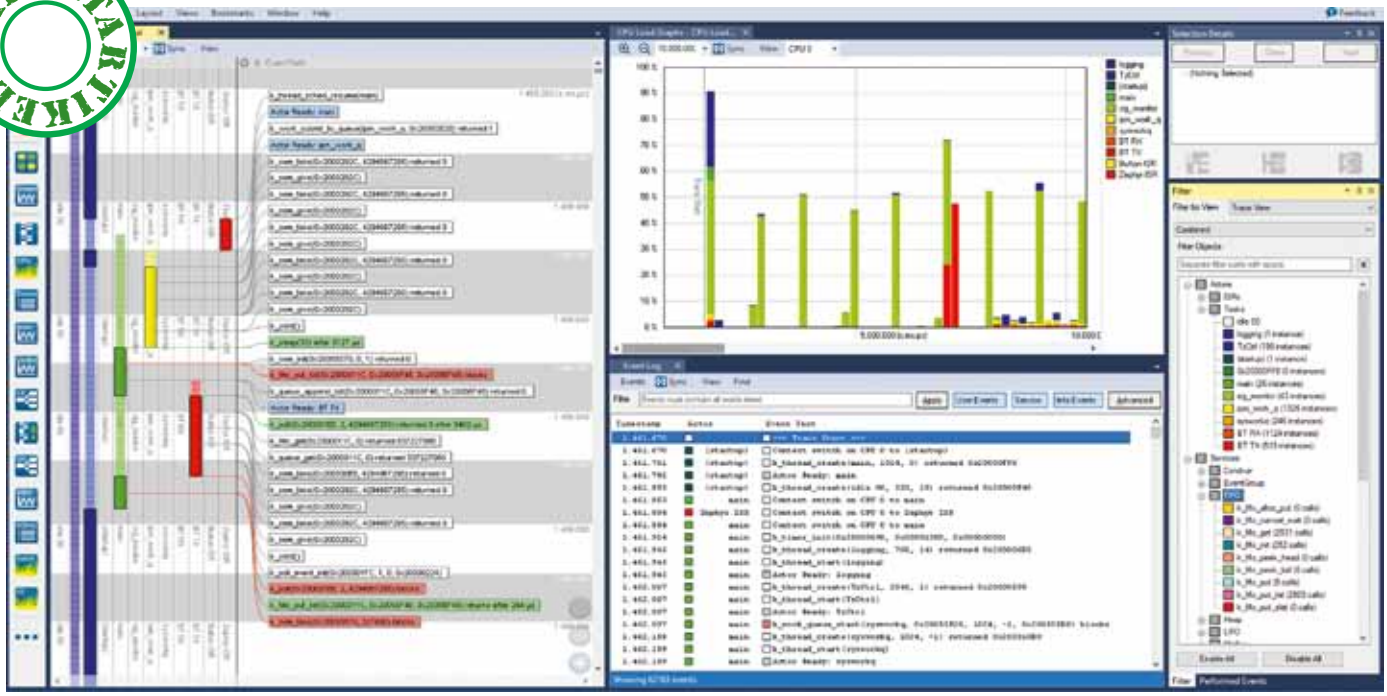
För att ta säkerhetsskorna som ett exempel: de upplyser bäraren, genom att skon börjar vibrera, om hen riskerar att komma för nära en kollega. Bäraren kan då till exempel sätta på sig sin ansiktsmask eller flytta sig längre bort.

Säkerhet kontra komplexitet

Men även med säkerhet i centrum kan förstås säkerhetsproblem uppstå, både i Zephyr-kärnan och i applikationskod. Zephyr-projektet redovisar öppet upptäckta sårbarheter och i listan kan man exempelvis hitta ”BadAlloc”, en minnesallokeringsbugg som påverkade en lång rad realtidsoperativsystem, tillsammans med problem i USB- och Bluetooth-biblioteken.

Ett inbyggt system med säker anslutning till molntjänster kräver mycket kod för kommunikationsstackarna, vilket ironiskt nog ökar komplexiteten och därmed riskerna. Ett ”Hello World”-projekt för anslutning till AWS genom MQTT och TLS producerar hundratals kilobyte körbar kod. Därtill tillkommer komplexiteten i själva applikationen och dess interaktioner med omvärlden, som också kan ha defekter och sårbarheter. Dessa kan utnyttjas till att få tillgång till enheten och i värsta fall också resten av nätverket. Ett system som inte är säkrat mot intrång (security) är inte heller funktionellt säkert (safety). Det är inte svårt att föreställa sig vilka konsekvenser ett intrång kan få i exempelvis industristyrsystem eller medicinska system.

Ett realtidsoperativsystem som Zephyr lägger till ytterligare ett lager av komplexitet: flertrådade applikationer. Ett av huvudskälen att använda ett RTOS är möjligheten att dela upp koden i separata trådar som i princip exekveras oberoende av varandra. I praktiken finns det dock ofta beroenden mellan trådarna som påverkar beteendet och ökar komplexiteten. Vissa beroenden är både avsiktliga och nödvändiga, då en del trådar behöver kommunicera mellan varandra, men det förekommer också oavsiktliga beroenden som kan ställa till problem – det är inte uppenbart utifrån källkoden att



Ett visuellt spårningsverktyg kan visa ett stort urval av analysvyer för Zephyr RTOS utan att det krävs några ändringar i operativsystemets källkod.

de finns där, än mindre hur de eventuellt påverkar exekveringen.

Det här blir en särskild utmaning för de mikrokontrollersystem som används i Sakernas Internet eftersom dessa ofta inte använder minnesskydd. Ett fel eller ett intrång i en del av systemet kan därmed påverka alla delar.

Determinism och testbarhet

De flesta RTOS, inklusive Zephyr, använder prioritetsbaserad schemaläggning av trådar, där en tråd kan avbrytas nästan när som helst för att lämna plats åt en annan. Om applikationen inte är designad på rätt sätt kan detta öka komplexiteten ytterligare, då den exakta exekveringsordningen inte tydligt framgår ur källkoden. Tillsammans med avsaknaden av minnesskydd och beroenden mellan trådar kan resultatet bli ett ickedeterministiskt beteende som kan leda till svårfångade fel.

Deterministisk exekvering är nödvändig för att systemet ska vara möjligt att testa på ett bra sätt, och därmed också för säkerheten. Det kräver att variationen i körtider hålls nere till ett minimum och framför allt inte påverkar den inbördes ordningen mellan viktiga händelser i mjukvaran. Annars finns det risk att systemet får oerhört många potentiella händelseordningar, mycket fler än vad som realistiskt kan testas i praktiken. Funktionell testning av sådana applikationer riskerar att bara skrapa på ytan av de möjliga fall som kan inträffa.

Det finns förvisso ett antal metoder att verifiera säkerhet, men det finns ingen heltäckande metod för att hitta alla potentiella sårbarheter. För att säkerställa deterministisk exekvering i ett RTOS-baserat system behövs detaljerad analys av exekveringen, inklusive variationer i körtider och i exekveringen av trådar och API-anrop.

Ett sätt att undersöka hur flertrådade applikationer beter sig under körning på Zephyr RTOS är att använda ett verktyg för programvaruspårning. Sådana verktyg använder strategiskt utplacerade anrop, eller spårningspunkter, i kärnans kod till att spela in händelser under körning. Källkoden till Zephyr behöver inte modifieras för att göra detta, det räcker med att aktivera spårning i inställningarna för Zephyr och kompilera om. I Zephyr 2.6.0 har det inbyggda stödet för spårning utökats med fler spårningspunkter och funktioner för att bättre kunna analysera vad som händer under körning, samt stöd för spårningsverktyget Tracealyzer från Percepio.

Spårningsverktyg kan visa inspelade händelser på en grafisk tidslinje, som många gånger är en utmärkt ingång till felsökning, men också visa använd processortid samt minnesförbrukning för stack och arbetsminne, uppdelat per tråd. Sådan information kan också användas till att upptäcka källor till ickedeterministiskt beteende, till exempel stor variation i tidsåtgång för vissa uppgifter. Målet är som sagt förbättrad tillförlitlighet och säkerhet, och vägen dit är en stabilare och mer testbar applikation.

Programvaruspårning kan också användas till att registrera API-anrop och användardefinierade händelser i applikationen, vilket gör att tekniken kan användas brett, till exempel för felsökning av låsningar och minnesläckor.

Verktygsstöd för avancerad analys och visualisering av spårningsdata – det som vi kallar visuell spårningsdiagnostik – realiserar något som närmast kan liknas vid en övervakningskamera för inbyggda system. Utvecklarna kan zooma ut bilden för att få överblick över de trådar som körs, och sedan zooma in på detaljnivå. Spårningsdata kan visualiseras ur flera perspektiv och på olika nivåer av abstraktion, vilket möjliggör att man arbetar sig utifrån och inåt: studera helhetsbilden för att se möjliga avvikelser och undersök sedan avvikelserna i mer detaljerade vyer.

Spårning kan realiseras helt i mjukvara, endera genom att spara de senaste händelserna i en ringbuffert i målsystemet eller genom att kontinuerligt strömma spårningsdata till en ansluten dator, exempelvis genom en TCP/IP-länk eller genom en avlusningsprob. På så vis kan man övervaka ett system över långa tidsperioder och fånga även mycket sällsynta problem.

Mjukvarubaserad spårning går att använda tillsammans med i stort sett alla inbyggnadsprocessorer och utvecklingsverktyg, eftersom det inte kräver speciellt hårdvarustöd för spårning.

Sammanfattning

Förmågan att tidigt och snabbt identifiera möjliga problem är avgörande för att hålla tidsramarna i ett utvecklingsprojekt, och för att leverera en produkt med hög kvalitet.

Tillförlitlighet och säkerhet är viktiga krav för uppkopplade inbyggda system inom Sakernas Internet. Zephyr 2.6.0 inkluderar ett utökat stöd för programvaruspårning som underlättar felsökning och bidrar till bättre säkerhet och tillförlitlighet.

Visuell spårningsdiagnostik gör det lättare att detektera ickedeterministiskt beteende, ett fenomen som kan dölja defekter i koden och därmed påverkar testbarheten negativt.

Med visuell spårningsdiagnostik kan man uppnå högre produktivitet, främst genom att felsökning går fortare, och man förbättrar testbarheten vilket har en positiv inverkan på kritiska faktorer som säkerhet och tillförlitlighet. ■