Real Time operativni sitem na platformi ESP32 (FreeRTOS)

Uopšteno o real time operativnim sistemima

Real-time operativni sistemi (RTOS) su operativni sistemi koji su dizajeni da pruže visoko precizno i predvidivo vreme odgovora na događaje. RTOS je ključan za aplikacije koje zahtevaju da se operacije izvrše unutar striktnih vremenskih ograničenja. U real-time okruženju, sistem mora da odgovori na događaje u tačno određenom vremenskom okviru, bez obzira na druge aktivnosti u sistemu.

Ključne karakteristike Real-Time Operativnih Sistema:

1. Predvidljivost:
	* RTOS mora biti vrlo predvidljiv, sa veoma niskim kašnjenjem (latencijom) između prijema događaja i početka odgovora na taj događaj.
	* Vreme odgovora mora biti dosledno, što znači da mora biti poznato koliko vremena je potrebno da se sistem odazove na određeni događaj.
2. Zadovoljavanje vremenskih ograničenja:
	* RTOS je dizajniran da ispuni striktne vremenske rokove za izvršavanje zadataka, poznate kao deadline. Ovi rokovi su od esencijalnog značaja za sigurnost i tačnost sistema (npr. u industrijskoj automatizaciji, automobilskim sistemima ili medicinskim uređajima).
3. Prioriteti zadataka:
	* RTOS obično koristi sistem prioriteta za upravljanje zadacima. To znači da su neki zadaci (ili procesi) važniji i treba ih obraditi pre drugih, obično sa višim prioritetom.
	* Preemptivni scheduling omogućava RTOS-u da prekinu trenutno izvršavanje zadatka (task) i prebaci se na zadatak sa višim prioritetom, čak i u toku njegovog izvršavanja.
4. Multitasking:
	* RTOS obično podržava multitasking, što znači da može izvršavati više zadataka u isto vreme, raspoređujući ih prema prioritetima. Međutim, RTOS mora to raditi na način koji je predvidiv i ne krši vremenska ograničenja.
5. Minimalno kašnjenje (latencija):
	* Real-time sistemi imaju vrlo nisku latenciju u reakcijama na događaje. To znači da je vreme između nastanka događaja i njegove obrade minimalno i konstantno.

Tipovi Real-Time Operativnih Sistema:

1. Hard Real-Time OS:
	* Ovaj tip RTOS-a mora strogo da ispunjava vremenske rokove. Ako operacija ne bude izvršena unutar definisanog vremenskog okvira, može doći do ozbiljnih grešaka u sistemu ili nesreća.
	* Primeri: Sistemi u avionima, medicinski uređaji, vozila sa autonomnim sistemima.
	* Primer: Vremenski rok za pokretanje vazduhoplovnog sistema mora biti tačno zadovoljen kako bi leteli bez opasnosti.
2. Soft Real-Time OS:
	* U soft real-time sistemima, vremenski rokovi su važni, ali nisu kritični. To znači da ako zadatak ne završi na vreme, neće doći do ozbiljnih problema, ali bi to moglo smanjiti performanse ili kvalitet sistema.
	* Primeri: Multimedijalni sistemi, video igre, online transakcije.
	* Primer: Kašnjenje u video streamingu može smanjiti kvalitet, ali neće izazvati kritične posledice.
3. Firm Real-Time OS:
	* Ovaj tip RTOS-a se nalazi između hard i soft real-time sistema. Vremenski rokovi nisu striktni kao kod hard real-time sistema, ali su još uvek važni i kašnjenje može negativno uticati na sistem.
	* Primeri: Industrijski automatski sistemi, robotika.

Osnovni koncepti i tehnologije RTOS-a:

1. Task Scheduling:
	* RTOS koristi scheduler za odlučivanje koji zadatak će se izvršavati u određenom trenutku. Scheduler može biti preemptivan ili kooperativan.
		+ Preemptivni scheduler omogućava da proces sa višim prioritetom preuzme CPU od trenutnog procesa.
		+ Kooperativni scheduler omogućava da procesi ne preuzimaju CPU sve dok ga ne oslobode.
2. Interruption Handling (obrada prekida):
	* RTOS mora efikasno upravljati prekidima i brzo reagovati na događaje koji dolaze od spoljnog hardvera (kao što su tastatura, miša, senzor itd.).
	* Prekidni servisi moraju biti vrlo brzo implementirani, jer obradom prekida može da se odgovori na spoljne događaje (npr. očitavanje podataka sa senzora).
3. Semaphore i Mutexes:
	* Semafori i mutex objekti se koriste za sinhronizaciju zadataka. Semafori omogućavaju zadacima da kontrolišu pristup deljenim resursima kako bi se izbegle konfliktne situacije.
4. Time Management:
	* RTOS mora omogućiti precizno upravljanje vremenom, jer je praćenje vremena ključno za obavljanje zadataka u okviru tačno definisanih vremenskih ograničenja.
5. Resource Management:
	* U real-time sistemima, resursi poput procesora, memorije i uređaja moraju biti pažljivo upravljani kako bi se obezbedila maksimalna efikasnost i minimalno kašnjenje.

Popularni Real-Time Operativni Sistemi:

1. FreeRTOS:
	* Jedan od najpopularnijih besplatnih real-time operativnih sistema. Koristi se za ugrađene sisteme i male uređaje.
2. VxWorks:
	* Komercijalni RTOS koji se koristi u mnogim industrijskim aplikacijama, vojnim i aeronautičkim sistemima.
3. RTEMS (Real-Time Executive for Multiprocessor Systems):
	* Open-source RTOS koji podržava više procesora i koristi se u industrijskim, istraživačkim i svemirskim aplikacijama.
4. QNX:
	* Popularni komercijalni RTOS koji se koristi u automobilskim, industrijskim, medicinskim i drugim kritičnim aplikacijama.
5. MicroC/OS-II:
	* Lagan, efikasan RTOS koji je popularan u ugrađenim sistemima i industrijskim aplikacijama.

Primene Real-Time Sistema:

* Avionika: U avionima se koriste RTOS za upravljanje kritičnim funkcijama kao što su navigacija, kontrola leta i komunikacija.
* Medicina: Medicinski uređaji kao što su pacemakeri, sistemi za nadzor pacijenata, MRI skeneri, itd.
* Automobilski sistemi: Autonomna vozila, sistem za kontrolu stabilnosti vozila, ABS kočnice.
* Industrijska automatizacija: Roboti, upravljanje procesima, CNC mašine, sistemi za kontrolu proizvodnje.
* Telekomunikacije: RTOS se koristi u mrežnim uređajima, kao što su ruteri i preklopnici, koji moraju obraditi podatke u realnom vremenu.
* Video igre: RTOS se koristi za obrada korisničkog unosa i renderovanje grafike sa minimalnim kašnjenjem.

Real-time operativni sistemi su ključni za aplikacije koje zahtevaju visoko predvidivo ponašanje i preciznu kontrolu vremena. RTOS-ovi omogućavaju efikasno upravljanje resursima i prioritetima, što je od esencijalne važnosti u aplikacijama kao što su medicina, automobilski sistemi, industrijska automatizacija i mnogi drugi kritični sistemi.

FreeRTOS je besplatan i otvoreni real-time operativni sistem koji se često koristi u ugrađenim sistemima, kao što je ESP32. ESP32 je popularna razvojna ploča sa ugrađenim Wi-Fi i Bluetooth mogućnostima, i često se koristi za IoT projekte. FreeRTOS na ESP32 omogućava upravljanje multitaskingom, upravljanje vremenskim ograničenjima, i efikasno upravljanje resursima, što je idealno za real-time aplikacije.

Korišćenje FreeRTOS-a na ESP32

ESP32 koristi FreeRTOS kao osnovni operativni sistem za multitasking, tako da ne morate instalirati posebnu verziju FreeRTOS-a; on je već integrisan u ESP-IDF (ESP32 Development Framework). ESP32 koristi preemptivni multitasking, što znači da sistem može efikasno prebacivati između zadataka sa različitim prioritetima.

Osnovni primer sa FreeRTOS na ESP32

Evo jednostavnog primera kako koristiti FreeRTOS za multitasking na ESP32. U ovom primeru kreiraćemo dva zadatka koji će se periodično izvršavati i obraditi podatke.

1. Uključivanje potrebnih biblioteka: FreeRTOS već dolazi sa ESP32 bibliotekom, pa ćemo je koristiti za kreiranje zadataka.
2. Kreiranje zadataka (task): FreeRTOS omogućava kreiranje više zadataka, svaki sa svojim prioritetima i funkcijama.
3. Upravljanje vremenom: FreeRTOS omogućava korišćenje delay funkcija za upravljanje vremenskim intervalima između izvršavanja zadataka.

Primer koda:

#include <Arduino.h>

// Funkcija koja će biti izvršena kao zadatak 1

void Task1(void \*parameter) {

 while (true) {

 Serial.println("Taks 1: Hello from Task 1");

 vTaskDelay(1000 / portTICK\_PERIOD\_MS); // Pauza od 1 sekunde

 }

}

// Funkcija koja će biti izvršena kao zadatak 2

void Task2(void \*parameter) {

 while (true) {

 Serial.println("Task 2: Hello from Task 2");

 vTaskDelay(500 / portTICK\_PERIOD\_MS); // Pauza od 500 ms

 }

}

void setup() {

 // Početno pokretanje serijske komunikacije

 Serial.begin(115200);

 while (!Serial);

 // Kreiranje dva zadatka

 xTaskCreate(Task1, "Task 1", 1000, NULL, 1, NULL); // Kreiraj Task 1

 xTaskCreate(Task2, "Task 2", 1000, NULL, 1, NULL); // Kreiraj Task 2

}

void loop() {

 // Funkcija loop() ostaje prazna, jer se sav posao obavlja u zadacima

}

Objašnjenje koda:

1. Kreiranje zadataka:
	* xTaskCreate() je funkcija koja kreira novi zadatak. Ona prima nekoliko parametara:
		+ Funkcija koja će biti izvršena kao zadatak (u ovom slučaju Task1 i Task2).
		+ Naziv zadatka.
		+ Veličina stoga (stack size) - ovo je prostor u memoriji koji je rezervisan za zadatak. U ovom primeru imamo 1000 bajtova.
		+ Parametri - u ovom primeru ne šaljemo nikakve parametre, pa je NULL.
		+ Prioritet - u ovom slučaju oba zadatka imaju isti prioritet (1).
		+ Task handle - ovo je opcionalni pokazivač na strukturu koja čuva informacije o zadatku, ali je u ovom primeru nepotreban, pa je NULL.
2. Task1 i Task2:
	* Svaki zadatak se ponaša kao beskonačna petlja, u kojoj se periodično izvršava neka radnja (ispisivanje u serijski monitor).
	* vTaskDelay() se koristi za pauzu između izvršavanja, a parametar portTICK\_PERIOD\_MS se koristi za konvertovanje vremena u ticks (jedinica vremena u FreeRTOS-u).
3. Serial.println():
	* Ova funkcija se koristi za ispisivanje poruka u serijski monitor, kako bi se pratilo ponašanje oba zadatka.
4. Funkcija loop():
	* U ovom primeru funkcija loop() je prazna jer sav rad obavljaju zadaci. FreeRTOS upravlja multitaskingom.

Kako pokrenuti ovaj kod?

1. Preuzmite i instalirajte Arduino IDE ako to već niste učinili.
2. Instalirajte ESP32 podršku u Arduino IDE:
	* Otvorite File > Preferences i u polje "Additional Boards Manager URLs" unesite URL:
	https://dl.espressif.com/dl/package\_esp32\_index.json
	* Idite na Tools > Board > Boards Manager, pretražujte "esp32" i instalirajte paket.
3. Odaberite odgovarajući ESP32 model iz menija Tools > Board.
4. Priključite ESP32 na računar i postavite serijsku brzinu na 115200 baud rate.
5. Kompajlirajte i otpremite kod na vaš ESP32 uređaj.

Šta se dešava tokom izvršenja?

* U serijskom monitoru ćete videti poruke:
	+ "Task 1: Hello from Task 1" svakih 1 sekundu.
	+ "Task 2: Hello from Task 2" svakih 500 milisekundi.

Slobodno se igrajte sa različitim prioritetima zadataka, vremenskim intervalima (vTaskDelay), ili dodajte još zadataka. Takođe možete koristiti semafore i mutex-e za sinhronizaciju između zadataka kada je to potrebno.

Dodatne funkcionalnosti u FreeRTOS na ESP32:

* Task synchronization: FreeRTOS omogućava korišćenje semafora, mutexa i događaja za sinhronizaciju između zadataka.
* Queues (redovi): Omogućavaju zadacima međusobnu komunikaciju.
* Timeri: Za pokretanje zadataka u tačno definisanim vremenskim intervalima.