UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

Seminarski rad
Implementacija NuttX RTOS-a na BeagleBone Black razvojnoj platformi za ugradbene sisteme

PREDMET:

- RAZVOJ SOFTVERA ZA UGRADBENE SISTEME -

Aladžuz Azra, 2129/18931
Vladavić Salko, 2128/18979

Sarajevo, septembar 2024.g.

Sadržaj

1. Uvod

Ugradbeni sistemi predstavljaju računare sa posebnom namjenom koji služe za nadgledanje i upravljanje nekog procesa. Uviđa se rastući trend upotrebe ugradbenih sistema za različite real-time aplikacije, pogotovo u raznim sferama industrijske automatizacije, te za potrebe IoT uređaja. Dosta često, projektovani sistemi zahtjevaju brze prenose podataka, međusobnu komunikaciju, detekciju kvarova i sl. Kako bi se ostvarile sve navede, a i ostale nenavedene funkcionalnosti potrebno je obezbjediti ugradbenom sistemu dobru hardversku podlogu (veliki broj ulaznih/izlaznih pinova različite namjene, velika brzina, mogućnost komunikacije...), a nakon toga implementirati kvalitetan operativni sistem sposoban da podrži zahtjeve real-time rada. Tema ovoga rada jesu primjeri jedne takve platforme i jednog takvog operativnog sistema. BeagleBone Black je izrazito popularna hardverska platforma koja se koristi u različitim kako inžinjerskim, tako i amaterskim aplikacijama. Ploča je odlikovana izuzetnim performansama i raznim funkcionalnostima. Razvojna ploča bez problema podržava real-time upravljanje, koje je krucijalno za procese industrijske automatizacije. Pored hardvera, navest će se i softverska komponenta koja pruža kvalitetno real-time upravljanje, a riječ je o NuttX operativnom sistemu. Spomenuti OS okarakterisan je zavidnim karakteristikama o kojima će biti više govora u nastavku. Nakon predstavljanja korištenih hardverdskih i softverskih alata, preći će se na izradu projekta koristeći NuttX i BeagleBone Black ploče.

2. NuttX operativni sistem

NuttX je operativni sistem koji radi u realnom vremenu ili skraćeno RTOS (eng. Real Time Operating System). Pri dizajnu OS-a, posebna pažnja je data na njegovoj kompatibilnosti sa standardima, kao i na malom footprint - u. Mali footprint znači da su ve dostupne funkcije dizajnirane imajući u vidu memorijsku ograničenost ugradbenih sistema. Stoga, sve funkcije koje nisu od ključne vrijednosti su izbrisane. Ponuđeni operativni sistem ima osobinu skalabilnosti od 8-bitnog do 64-bitnog okruženja mikrokontrolera, pri čemu se osnovni standard bazira na POSIX i ANSI standardu. Dodatni standardni API-ji porijeklom iz Unix-a i drugih uobičajenih RTOS-a (kao što je VxWorks) su usvojeni zbog ostvarivanja funkcionalnosti koje nisu dostupne prema osnovnim standardima.
U dokumentaciji kreatori NuttX-a ističu i ciljeve ovog operativnog sistema o čemu će biti govora u nastavku [1].

- **Mali footprint**. Moguće je upotrijebiti u više-manje svim mikrokontrolerskim okruženjima, pri čemu je operativni sistem dizajniran za rad na izrazito malim, kompaktnim ugradbenim sistemima.
- **Operativni sistem krcat različitim funkcijama** Pri kreaciji ovog sistema, cilj je bio obezbjediti kvalitetnu implementaciju klasičnog POSIX OS interfejsa, koji nadalje podržava višenitno ugradbeno razvojno okruženje za razne ugradbene procesore.
- Autori dokumentacije ističu da ovdje nije cilj obezbjediti isti nivo funkcije kao što obezbjeđuje Linux, imajući u vidu ograničenja memorije. NuttX se može zamisliti kao "mini" vezija Linux sistema sa drastično smanjenim setom funkcija.
- **OS je također i visokoskalabilan**, imajući raspon od 8-bita do 64-bita. Ovo je realizovano korištenjem malih source file-ova, linkova iz statičke biblioteke, te upotrebom izrazito podesivih, slabih simbola kada su dostupni.
- Bitno je ne zaboraviti da NuttX teži visokom nivou podudaranja sa standardima, fokusirajući se na POSIX i ANSI standarde. Dodatni standardni API-ji iz Unix-a i ostalih često korištenih RTOS-a su preuzeti i ukomponovani u kreirani standard. **Time je obezbjeđen jednostavan prenos softvera kreiranog za Linux na NuttX OS.**
- **Operativni sistem je real-time.**
- **Otvorena, neograničavajuća Apache licensa.**
- Kompatibilni GNU alati koji se baziraju na buildroot su dostupni za besplatno preuzimanje kako bi obezbjedili potpuno razvijajuće okruženje.

U dokumentaciji se nalazi detaljni spisak funkcionalnosti NuttX-a koja se ovdje neće navoditi. Međutim, dokumentacija je opsežna i podrazumijeva veliki broj funkcija. Postavlja se prirodno pitanje: Kako je moguće da OS dizajniran za ugradbene sisteme sadrži sve ove funkcije?
Veliki broj funkcija daje veće mogućnosti, no te funkcije uvode dodatnu potrebu za memorijom. Filozofija NuttX-a je sljedeća. Ako funkcije nisu u upotrebi, onda nema smisla da usporavaju ili na bilo koji drugi način "otežavaju" rad programa. Ono što je zanimljivo jeste da pri korištenju NuttX-a svaka funkcija koja nije upotrijebljena neće biti navedena u konačnu izvršnu binarnu datoteku. Suštinski ugradbeni sistem će koristiti isključivo funkcije koje imaju neku funkciju. Pored navednog skrenut će se pažnja i na skalabilnost NuttX-a. Neke manje zahtjevne aplikacije omogućuju da NuttX radi sa 32k memorije (kod i podaci), dok one najzahtjevnije idu i do

100k. Predstavljeni operativni sistem nema ni najmanje problema sa ovim rasponom, što povećava njegovu primjenjivost. Detaljno objašnjenje načina rada NuttX-a je dato u dokumentaciji. Zainteresovani čitaoci mogu provjeriti referenciranu literaturu.

3. BeagleBone Black razvojna platforma

BeagleBone je open-source mikrokontroler koji postaje izrazito popularan [2] u svijetu ugradbenih sistema zbog odličnog omjera snage, fleksibilnosti i cijene. Pruža opsežan spektar primjene, od jednostavnih projekata elektronike do složenih ugradbenih sistema u industrijskim i obrazovnim okruženjima. Njegov skup funkcija čini ga potencijalnim konkurentom drugim računarima opšte namjene kao što je Raspberry Pi. Glavne prednosti koje nudi BeagleBone Black platforma je izrazito kvalitetan hardverski interfejs, obrada u realnom vremenu i detaljna kontrola hardvera, odnosno kodiranje na niskom nivou.

3.1 Dizajn i arhitektura

Jezgro BeagleBone Black je 1 GHz ARM Cortex-A8 CPU, što predstavlja AM335x procesor [3]. Ima 512MB DDR3 RAM-a, što ploču čini sposobnom za više zadataka koji su umjereno zahtjevne težine. Cortex-A8 arhitektura, iako je malo starija u odnosu na najnovije ARM jezgre, idalje je vrlo efikasna u različitim ugradbenim i stvarnim IoT aplikacijama. Uz to, integrisani PowerVR SGX530 GPU dopunjuje sposobnost ploče da rukuje 3D grafikom i izvršava zadatke koji se odnose na grafičke interfejse ili jednostavne aplikacije za igre.
Jača strana BeagleBone Black-a je njegova arhitektura skladištenja. Sadrži 4 GB ugrađene eMMC flash memorije, unaprijed instalirane s Linux distribucijom baziranom na Debianu.
Time je osigurano momentalno pokretanje sistema i početak rada na njemu direktno bez pripreme ikakvog operativnog sistema na eksternoj microSD kartici. U slučaju da se koristi više prostora za skladištenje ili da se koristi neki drugi operativni sistemi, kao što je NuttX upotrijebljen u ovom radu, BeagleBone Black također ima slot za microSD karticu. Na taj način korisnici mogu lako promijeniti operativni sistem koji koriste ili proširiti memoriju kada se bave složenijim projektima.

3.2 Opsežne mogućnosti povezivanja

Za potrebe povezivanja, BeagleBone Black koristi 10/100 Ethernet port, što znači da se brzina prijenosa podataka kreće u opsegu od 10 Mb u sekundi sve do 100 Mb u sekundi. Sistemi koji se oslanjaju na brzi prijenos podataka mogu kvalitetno iskoristiti ovu osobinu. Primjeri ovakvih slučajeva se mogu naći u oblastima industrijske automatizacije ili za potrebe IoT (eng. Internet of Things) uređaja koji zahtjevaju konstantnu povezanost. Pored navedenog, ploča sadrži i USB host i USB klijent port. USB host port se može koristiti za povezivanje tastature, miša ili drugih eksternih uređaja za skladištenje, dok USB klijent port omogućava da se BeagleBone Black napaja preko USB veze, ili da se koristi USB uređaj. Ovo uveliko olakšava programiranje preko računara, pojednostavljuje provjeru funkcionalnosti sistema, te čini detekciju i ispravku grešaka trivijalnim zadatkom.

3.3 Ulazno/izlazne mogućnosti BeagleBone Black ploče

Ploča raspolaže sa čak 92 konfigurabilna I/O pina, koje služe različite funkcionalnosti: GPIO, PWM, ADC, SPI, I2C, UART, CAN. Raspon mogućnosti potvrđuje da je BeagleBone Black ploča idealni kandidat za regulaciju sistema koji zahtijevaju real-time upravljanje i prikupljanje podataka. Mogućnost očitavanja signala direktno sa senzora ili drugog hardvera iz analognog čini BeagleBone Black posebno korisnim u aplikacijama kao što je nadzor okoline, pri čemu senzori mogu dati podatke u analognom obliku. Također poseduje PWM za fino podešavanje

motora, servo motora i drugih aktuatora, proširujući njegovu upotrebu i na oblasti robotike. PWM izlaze mogu koristiti programeri da upravljaju brzinom DC motora, položajem servo uređaja ili drugih hardverskih komponenti. Navedene karakteristike, zajedno sa raznim podržanim komunikacijskim protokolima, čine BeagleBone Black svestranim alatom za projekte koji uključuju kompleksnu integraciju senzora, pogon motora ili komunikaciju s drugim uređajima.

3.4 Softverski ekosistem

Pored unaprijed instaliranog operativnog sistema, ploča podržava razne druge OS-eove, a za potrebu ovog ovog rada upotrijebljen je NuttX. Programski, BeagleBone Black podržava popularne programske jezike kao što su Python, JavaSCcript C i C++ [4], a ovo posljednje dvoje je od velikog značaja za real-time aplikacije. Pored navedenih konkretnih prednosti, treba imati u vidu da BeagleBone Black ploča osmišljena je sa open-source filozofijom. Posljedično, stvorena je cijela zajednica inžinjera i programera koji aktivno koriste ovu ploču te pružaju podršku svim korisnicima, što dovodi do postojanja raznih tutorijala, gotovih projekata, opsežne dokumentacije. Dakle, ova ploča je izrazito prijateljski nastrojena prema novim korisnicima. Također, politika otvorenosti pruža uvid u dizajn ploče, što je od velike vrijednosti inžinjerima koji dizajniraju svoje vlastite ploče, ili za sve one koji žele dobiti bolji uvid u rad sistema.

Kombinacija niske cijene, kvalitetne izrade i opsežnog domena primjene čini ovu ploču izrazito popularnom. Široko je upotrebljavaju kako studenti, tako i profesionalni inžinjeri, što dodatno svjedoči o kvalitetu platforme.

4. Implementacija NuttX-a na BeagleBone Black platformi

Seminarski rad se fokusira na implementaciju NuttX-a, operativnog sistema u realnom vremenu, na BeagleBone Black, popularnoj razvojnoj platformi za ugradbene sisteme. Proces je izveden korištenjem Windows podsistema za Linux (WSL) i Ubuntu-a. U nastavku su detaljno opisani koraci za instalaciju NuttX-a, njegovo konfigurisanje za BeagleBone Black i pripremu SD kartice za pokretanje za ploču.

4.1 Instalacija NuttX-a

Za početak, postavljeno je okruženje zasnovano na Linuxu korištenjem Windows podsistem za Linux (WSL). WSL omogućava pokretanje kompletne Linux distribucije uz Windows operativni sistem bez potrebe za dvostrukim pokretanjem ili virtualizacijom. Upotreba WSL-a i Ubuntua bila je neophodna jer su NuttX i njegovo okruženje za izgradnju dizajnirani da rade na sistemima sličnim Unixu. Sam Windows izvorno ne podržava alate potrebne za razvoj NuttX-a, kao što su make i gcc-arm, tako da je WSL pružio efikasno rješenje za pokretanje ovih Linux alata. Sljedeći korak je bio instaliranje NuttX-a slijedeći upute dostupne na službenoj web stranici NuttX-a. Korištene komande su navedene u nastavku.

Instalacija i podešavanje alata potrebnih za pokretanje Apache NuttX-a obavljeni su korištenjem:

```
sudo apt update
sudo apt install build-essential gdb
sudo apt install cmake
$ sudo apt install \
bison flex gettext texinfo libncurses5-dev libncursesw5-dev xxd \
gperf automake libtool pkg-config build-essential gperf genromfs \
libgmp-dev libmpc-dev libmpfr-dev libisl-dev binutils-dev libelf-dev \
libexpat-dev gcc-multilib g++-multilib picocom u-boot-tools util-linux
```

Instalacija alata na Ubuntu/WSL okruženju, uključujući kconfig-frontends za konfiguraciju kernel modula, kao i Python biblioteke kconfiglib, pyelftools i cxxfilt koje su potrebne za rad sa konfiguracijom sistema i analizom binarnih fajlova obavljena je komandama:

```
sudo apt install kconfig-frontends
pip install kconfiglib
pip install pyelftools cxxfilt
```

Gcc-arm-none-eabi i binutils-arm-none-eabi alati, koji su neophodni za kompajliranje i kreiranje binarnih fajlova za ARM arhitekturu instaliraju se komandom:

```
sudo apt install gcc-arm-none-eabi binutils-arm-none-eabi
```

Naredne komande kreiraju radni direktorij nuttx-workspace, zatim kloniraju Apache NuttX repozitorij sa GitHub-a u direktorij nuttx, kao i repozitorij Apache NuttX aplikacija u direktorij apps.

```
mkdir nuttx-workspace
cd nuttx-workspace
git clone https://github.com/apache/nuttx.git nuttx
git clone https://github.com/apache/nuttx-apps apps
```

Direktorij apps sadrži aplikacije na nivou korisnika za NuttX, dok nuttx sadrži NuttX kernel i jezgro sistema. Ova struktura direktorija je neophodna za NuttX, jer odvaja jezgro sistema od aplikacija koje se pokreću na njemu. Nakon instalacije NuttX-a, potrebno ga je kofigurisati posebno za BeagleBone Black, što je ključan korak jer različiti ugradbeni sistemi zahtevaju različite konfiguracije na osnovu njihovih hardverskih specifikacija. Naredne naredbe (unutar nuttx-workspace/nuttx foldera) konfigurišu NuttX za BeagleBone Black sa NSH shell-om pomoću skripte configure.sh i na kraju kompajliraju sistem koristeći make.

```
./tools/configure.sh beaglebone-black/nsh
make
```

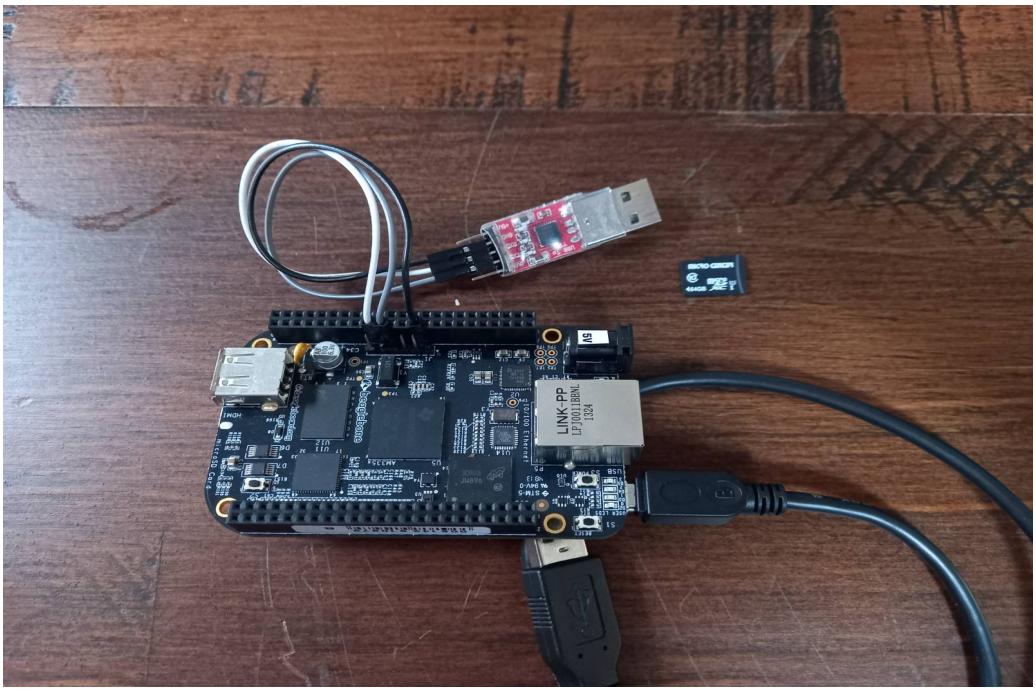
Ove komande određuju hardversku platformu, koja je u ovom slučaju BeagleBone Black. Ovo osigurava da će generirani binarni fajlovi (kao što je nuttx.bin) biti kompatibilni s hardverom. Konfiguracija također definira postavke pokretačkog programa, dodjelu memorije i periferne postavke koje odgovaraju arhitekturi BBB-a. Kada je NuttX ispravno konfigurisan, sljedeći korak je bio kompajliranje sistema. Ovaj proces je pokrenut korištenjem naredbe make. Izlaz ovog procesa je binarni fajl nazvan nuttx.bin. Ova datoteka je u suštini izvršni kod koji će se učitati na BeagleBone Black za pokretanje NuttX operativnog sistema. Za pokretanje NuttX-a na BeagleBone Black, korištena je SD kartica. BBB podržava pokretanje sa SD kartice formatirane kao FAT32, zbog čega je prvi korak u pripremi SD kartice uključivao njeno formatiranje. Nakon formatiranja SD kartice, datoteka nuttx.bin je kopirana u root direktorij kartice. S pripremljenom SD karticom koja sadrži datoteku nuttx.bin, sljedeći korak uključuje pokretanje BeagleBone Black ploče.

4.2 Povezivanje i Konfiguracija BeagleBone Black-a

Za povezivanje i instalaciju BeagleBone Black-a bilo je potrebno instalirati odgovarajuće drajvere sa zvanične web stranice. Drajveri za BeagleBone Black često ne rade ispravno na Windows 11, ali je ovaj problem moguće riješiti tako što se omogući kompatibilnost sa prethodnim verzijama operativnog sistema. Ova opcija omogućava Windowsu da koristi drajvere dizajnirane za starije verzije sistema, što je neophodno za uspješno povezivanje BBB-a. Nakon instalacije drajvera, BeagleBone Black je povezan sa laptopom koristeći USB kabl koji služi i za napajanje. Ploča je pokrenuta sa ugrađene memorije (eMMC) nakon što je prepoznata od strane sistema.
Za komunikaciju sa BeagleBone Black-om, instaliran je program PuTTY. PuTTY je softver za serijsku komunikaciju koji omogućava korisniku da se poveže sa različitim uređajima putem serijskog porta, SSH-a, telnet-a ili raw protokola. U ovom slučaju, PuTTY je korišten za uspostavljanje serijske komunikacije sa BeagleBone Black-om preko serijskog adaptera. Za uspješno korištenje NuttX-a sa BeagleBone Black-om, potreban je serijski adapter (USB-to-TTL adapter) koji omogućava serijsku komunikaciju između laptopa i ploče. Ovaj adapter je neophodan jer BeagleBone Black ne podržava direktnu serijsku komunikaciju putem običnog USB kabla, koji je rezervisan za napajanje i prenos podataka. Adapter služi za nisku razinu komunikacije, posebno kada se komunicira sa bootloaderom, kao što je U-Boot, ili prilikom debagiranja. Za korištenje ovog adaptera potrebno je instalirati odgovarajuće drajvere. Serijski adapter je povezan sa BeagleBone Black-om putem njegovih GPIO pinova, i to na sljedeći način:

- RX (na adapteru) povezan sa TX (na BBB-u),
- TX (na adapteru) povezan sa RX (na BBB-u),
- GND (na adapteru) povezan sa GND (na BBB-u).

Na slici 1 je prikazan BeagleBone Black sa serijskim adapterom, SD karticom i USB kablom.



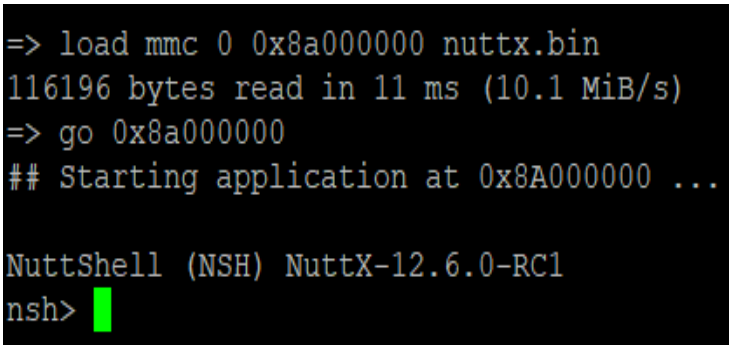
Slika 1: BeagleBone Black sa serijskim adapterom, SD karticom i USB kablom

4.3 Komunikacija sa BeagleBone Black-om i pokretanje Nuttx-a

Nakon povezivanja adaptera i BeagleBone Black-a sa laptopom, pokrenut je PuTTY i postavljeni su parametri za serijsku komunikaciju, uključujući COM port i brzinu prijenosa podataka (baud rate), koji je tipično postavljen na 115200. Zatim je priključen USB kabl kako bi se omogućilo napajanje za BeagleBone Black, čime je odmah započeo proces pokretanja sistema. Tokom ovog procesa, putem PuTTY terminala, prekinut je U-Boot, što je omogućilo dalje upravljanje pločom. Nakon što je U-Boot prekinut, u BeagleBone Black je ubačena prethodno pripremljenu SD kartica koja sadrži NuttX binarni fajl. Korištenjem odgovarajućih komandi preko terminala, učitan je nuttx.bin fajl sa SD kartice:

```
load mmc 0 0x8a000000 nuttx.bin
go 0x8a000000
```

Na ovaj način, NuttX je pokrenut na BeagleBone Black-u, slika 2.



Slika 2: Izgled PuTTY terminala nakon učitavanja nuttx.bin fajla

4.4 Kreiranje aplikacija u NuttX-u

Nakon što je sve ispravno postavljeno i konfigurisano, kreirana je custom aplikacija u NuttX-u koristeći Ubuntu/WSL okruženje. Prva aplikacija, pod nazivom Custom Hello, napravljena je prateći instrukcije sa zvanične NuttX stranice. Osnovni koraci dati su u nastavku.

- Kreiranje direktorija za custom aplikacije: Prvi korak bio je proširenje postojećeg apps/ direktorija u NuttX sistemu kreiranjem novog direktorija za custom aplikacije, pod nazivom CustomApps.
- Konfiguracija Make.defs fajla: Nakon kreiranja direktorija, potrebno je postaviti Make.defs fajl koji definiše pravila i varijable za izgradnju custom aplikacija. Ovaj fajl osigurava da sistem za izgradnju zna za sve poddirektorije unutar CustomApps direktorija.
- Kreiranje Makefile aajla za CustomApps: Kreiran je Makefile unutar CustomApps/ direktorija, koji sadrži logiku za izgradnju svih aplikacija unutar ovog direktorija. Postavljen je tako da NuttX sistem može rekurzivno pretraživati poddirektorije i graditi aplikacije koje se tamo nalaze.
- Kreiranje direktorija za custom aplikaciju: Kreiran je direktorij za prvu custom aplikaciju pod nazivom CustomHello unutar CustomApps/. Ovaj direktorij sadrži izvorni kod, konfiguracijske fajlove, i sve što je potrebno za izgradnju aplikacije.
- Dodavanje koda za CustomHello: U direktoriju CustomHello, kreiran je fajl CustomHello.c koji sadrži jednostavan C program. Ovaj program, kada se pokrene, ispisuje poruku "Hello, Custom World!!" na terminalu.
- Postavljanje Make.defs fajla za CustomHello: Kako bi se aplikacija ispravno uključila u NuttX build sistem, postavljen je Make.defs fajl unutar direktorija CustomHello. Ovaj fajl sadrži uslov koji provjerava da li je aplikacija CustomHello omogućena u konfiguraciji, i ako jeste, dodaje je na listu aplikacija koje se grade prilikom kompilacije NuttX-a.
- Kreiranje Makefile-a za CustomHello: Makefile unutar CustomHello direktorija definiše način na koji se aplikacija kompajlira. U njemu su postavljene varijable poput imena programa (PROGNAME), prioriteta zadatka (PRIORITY), i veličine steka zadatka (STACKSIZE).
- Postavljanje Kconfig fajla: Kako bi se aplikacija CustomHello mogla upravljati putem menuconfig interfejsa, kreiran je Kconfig fajl. Ovaj fajl omogućava uključivanje ili isključivanje aplikacije u konfiguraciji, kao i postavljanje imena programa, prioriteta, i veličine steka zadatka putem korisničkog interfejsa.
- Izgradnja i pokretanje: Kada su svi fajlovi kreirani, potrebno je integrisati aplikaciju u NuttX build sistem. Prvi korak je čišćenje postojećih build fajlova, a nakon toga koristi se menuconfig da se omogući aplikacija Custom Hello unutar menija za custom aplikacije, te se započinje proces izgradnje NuttX-a sa novom aplikacijom.

Ova aplikacija je uspješno kreirana, a koristeći isti princip, kreirana je i druga custom aplikacija, Custom Counter App, koja kreira dva zadatka (taska) koristeći POSIX niti. Task 1 povećava globalni brojač svake 3 sekunde, dok Task 2 ispisuje vrijednost brojača svakih 7 sekundi. Za obje aplikacije korišteni su isti koraci, a glavna razlika je u sadržaju pojedinih fajlova. Na

kraju, nakon uspješne izgradnje, aplikacije Custom Hello i Custom Counter pokrenute su na BeagleBone Black-u, slika 3.

```
=> load mmc 0 0x8a000000 nuttx.bin
125216 bytes read in 11 ms (10.9 MiB/s)
=> go 0x8a000000
## Starting application at 0x8A000000 ...

NuttShell (NSH) NuttX-12.6.0-RC1
nsh> custom_hello
Hello, Custom World!!
nsh> custom_counter
[DEBUG] 14.28 Task 1 - izvršavanje
[DEBUG] 14.28 Task 2 - izvršavanje
Counter value: 1
[DEBUG] 17.29 Task 1 - izvršavanje
[DEBUG] 20.30 Task 1 - izvršavanje
[DEBUG] 21.30 Task 2 - izvršavanje
Counter value: 3
[DEBUG] 23.31 Task 1 - izvršavanje
[DEBUG] 26.32 Task 1 - izvršavanje
[DEBUG] 28.31 Task 2 - izvršavanje
Counter value: 5
[DEBUG] 29.33 Task 1 - izvršavanje
[DEBUG] 32.34 Task 1 - izvršavanje
[DEBUG] 35.32 Task 2 - izvršavanje
Counter value: 7
[DEBUG] 35.35 Task 1 - izvršavanje
[DEBUG] 38.36 Task 1 - izvršavanje
[DEBUG] 41.37 Task 1 - izvršavanje
[DEBUG] 42.33 Task 2 - izvršavanje
Counter value: 10
[DEBUG] 44.38 Task 1 - izvršavanje
[DEBUG] 47.39 Task 1 - izvršavanje
```

Slika 3: Izgled PuTTY terminala nakon Custom Hello i Custom Counter

Kod za aplikaciju Custom Counter je dat u nastavku.

```c
#include <nuttx/config.h>
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <errno.h>
#include <time.h>
#include <signal.h>

// Global variable for the counter
volatile int counter = 0;
volatile int running = 1;

// Mutex to protect access to counter
pthread_mutex_t counter_mutex = PTHREAD_MUTEX_INITIALIZER;

// Function to print timestamp
void print_timestamp(const char* task_name)
{
    struct timespec ts;
```

```
20      clock_gettime(CLOCK_REALTIME, &ts);
21      double time_in_seconds = ts.tv_sec + ts.tv_nsec / 1e9;
22      printf("[DEBUG] %.2f %s - izvr  avanje\n",
23      time_in_seconds, task_name);
24  }
25
26  // Signal handler to stop the program
27  void handle_sigint(int sig)
28  {
29      printf("Caught signal %d, stopping...\n", sig);
30      running = 0;
31  }
32
33  // Task 1: Increment the counter every 3 seconds
34  void* task1_func(void* arg)
35  {
36      while (running)
37      {
38          print_timestamp("Task 1");
39
40          pthread_mutex_lock(&counter_mutex);
41          counter++;
42          pthread_mutex_unlock(&counter_mutex);
43
44          sleep(3);
45      }
46      return NULL;
47  }
48
49  // Task 2: Print the value of the counter every 7 seconds
50  void* task2_func(void* arg)
51  {
52      while (running)
53      {
54          print_timestamp("Task 2");
55
56          pthread_mutex_lock(&counter_mutex);
57          printf("Counter value: %d\n", counter);
58          pthread_mutex_unlock(&counter_mutex);
59
60          sleep(7);
61      }
62      return NULL;
63  }
64
65
66  int main(int argc, char *argv[])
67  {
68      pthread_t task1;
69      pthread_t task2;
70      int ret;
71
72      // Register signal handler for SIGINT (Ctrl+C)
73      signal(SIGINT, handle_sigint);
74
75      // Create Task 1
76      ret = pthread_create(&task1, NULL, task1_func, NULL);
77      if (ret != 0)
```

```
78      {
79          printf("Failed to create Task 1: %d\n", ret);
80          return -1;
81      }
82
83      // Create Task 2
84      ret = pthread_create(&task2, NULL, task2_func, NULL);
85      if (ret != 0)
86      {
87          printf("Failed to create Task 2: %d\n", ret);
88          return -1;
89      }
90
91      // Wait for tasks to finish
92      pthread_join(task1, NULL);
93      pthread_join(task2, NULL);
94
95      printf("Custom counter app stopped.\n");
96
97      return 0;
98  }
```

Literatura

[1] Apache Software Foundation, *NuttX Documentation*, https://nuttx.apache.org/docs/latest/, 2024, posljednji put pristupljeno: 14.9.2024. [Online]. Available: https://nuttx.apache.org/docs/latest/

[2] Arrow Electronics, Inc., "Comparing beaglebone boards," https://www.arrow.com/en/research-and-events/videos/comparing-beaglebone-boards, 2024, posljednji put pristupljeno: 14.9.2024. [Online]. Available: https://www.arrow.com/en/research-and-events/videos/comparing-beaglebone-boards

[3] BeagleBoard.org Foundation, "Beaglebone black," https://www.beagleboard.org/boards/beaglebone-black, 2024, posljednji put pristupljeno: 14.9.2024. [Online]. Available: https://www.beagleboard.org/boards/beaglebone-black

[4] Makezine, "Get started with beaglebone," https://makezine.com/projects/get-started-with-beaglebone/, 2024, posljednji put pristupljeno: 14.9.2024. [Online]. Available: https://makezine.com/projects/get-started-with-beaglebone/