Napredni operativni sistemi - Projekat 2

Student: Danilo Veljović, 1120

Katedra za računarstvo i informatiku
Univerzitet u Nišu, Elektronski fakultet

22. februar 2021.

Sadržaj

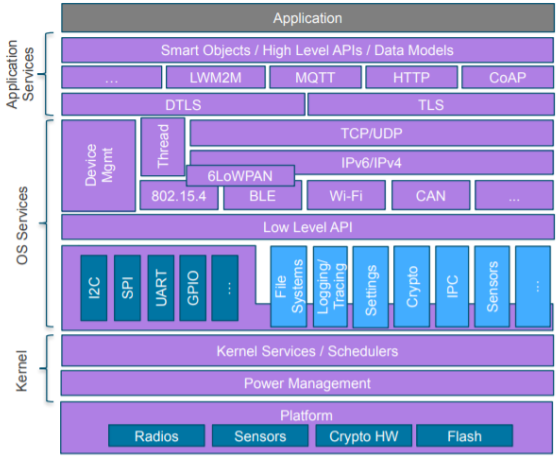
IoT OS arhitektura

Zephyr OS ima 'small-footprint' kernel koji se koristi na embedded uređajima (od senzora do embedded kontrolera, pametnih satova i IoT wireless aplikacija). Zephyr kernel podržava više arhitektura:

- ARC EM i HS
- ARMv6-M, ARMv7-M, and ARMv8-M (Cortex-M)
- Intel x86 (32- and 64-bit)
- NIOS II Gen 2
- RISC-V (32- and 64-bit)
- SPARC V8

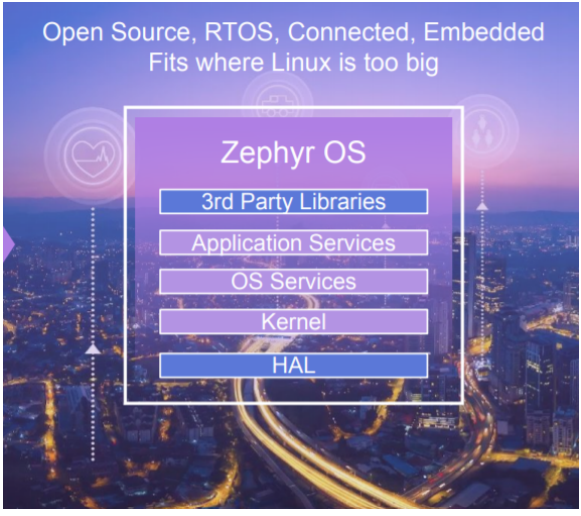
Puna lista ploča koje podržava Zephyr OS može se naći na linku:
https://docs.zephyrproject.org/latest/boards/index.html

IoT OS arhitektura



Slika: Arhitektura Zephyr OS

Komponente



Slika: Komponente Zephyr OS

Razvoj aplikacija

Build sistem Zephyra se zasniva na CMake. Build sistem je 'application-centric' tj zahteva da Zephyr aplikacije izbuildaju kernel source tree. Build aplikacije kontrolise konfiguraciju i build proces i aplikacije i samog Zephyr OS, i kompajlira ih u jedan binarni fajl. Fajlovi u application direktorijumu povezuju Zephyr sa aplikacijom. Ovaj direktorijum sadrži sve fajlove specifične za aplikaciju kao što su konfiguracioni fajlovi i source kod.
Pri kreiranju aplikacije potrebno je kreirati direktorijum gde će aplikacija biti smeštena, tj najćešće je to u home direktorijumu. Svi source kodovi se smeštaju u folderu src. Nakon smeštanja c fajlova u src direktorijum treba napraviti i CMakeLists.txt fajl koji sadrži dodatna podešavanja aplikacije. Kreira se proj.conf fajl koji je Kernel config fajl. Nakon toga se projekat izbuilda naredbom: **west build -b tip-ploče/naziv-aplikacije**. West je alat koji se instalira u Zephyr OS i koji olakšava razvoj i buildarnje Zephyr aplikacija. Kada se aplikacija gura na ploču, jednostavno se ploča poveže na računar i izda se komanda **west flush**.

Source Tree Zephyr aplikacije

Source Tree struktura Zephyr projekta:

- CMakeLists.txt
- Kconfig
- west.yml
- arch
- soc
- boards
- doc
- drivers
- dts
- include
- kernel
- lib
- misc
- samples
- scripts
- cmake
- subsys
- tests
- share

Demo aplikacija

Za potrebe projekta razvijena je aplikacija za Zephyr OS za ploču Nordic nRF52DK (https://www.iot-lab.info/docs/boards/nordic-nrf52dk/), koja ima Shield ST X-NUCLEO-IKS01A2 koji ima senzore za temperaturu i vlažnost (HTS221), senzore za atmosferski pritisak (LPS22HB), accelerometer sensor (LSM6DSL) i accelerometer sensor LSM303AGR. Sa ovog shielda se čitaju trenutno očitane vrednosti i nakon očitavanja vrednosti ih štampa. Aplikacija sve vreme računa srednju vrednost pročitanih temperatura i ako je srednja vrednost temperature veća od 22 stepena, upaliće diodu kao aktuaciju, u suprotnom će je ugasiti. Testiranje aplikacije se radi na serveru https://www.iot-lab.info/

Struktura aplikacije

Struktura aplikacije je sledeća:

- inicijalizacija
- očitavanje prisutnih senzora
- očitavanje senzora
- računanje prosečne temperature i aktuacija po potrebi
- štampa očitanih vrednosti

Struktura aplikacije

```
/* Get sensor data */

sensor_channel_get(hts221, SENSOR_CHAN_AMBIENT_TEMP, &temp1);
sensor_channel_get(hts221, SENSOR_CHAN_HUMIDITY, &hum);
sensor_channel_get(lps22hb, SENSOR_CHAN_PRESS, &press);
sensor_channel_get(lps22hb, SENSOR_CHAN_AMBIENT_TEMP, &temp2);
sensor_channel_get(lsm6dsl, SENSOR_CHAN_ACCEL_XYZ, accel1);
sensor_channel_get(lsm6dsl, SENSOR_CHAN_GYRO_XYZ, gyro);
sensor_channel_get(lsm303agr_a, SENSOR_CHAN_ACCEL_XYZ, accel2);
sensor_channel_get(lsm303agr_m, SENSOR_CHAN_MAGN_XYZ, magn);
```

Slika: Očitavanje sa senzora

Komponente

```
cnt_temp++;
HTS221_temp += sensor_value_to_double(&temp1);
HTS221_temp_avg = HTS221_temp / cnt_temp;

if(HTS221_temp_avg > 22){
        led_is_on = true;
        printf("LED is on in iteration %ld\n", cnt_temp);
}
else{
        led_is_on = false;
        printf("LED is off in iteration %ld\n", cnt_temp);
}

gpio_pin_set(dev, PIN, (int)led_is_on);
```

Slika: Računanje proseka očitane temperature i aktuacija

Komponente

```c
/* temperature */
printf("HTS221: Current temperature: %.1f C\n",
      sensor_value_to_double(&temp1));

printf("HTS221: Average temperature: %.1f C\n",
      HTS221_temp_avg);


/* humidity */
printf("HTS221: Relative Humidity: %.1f%%\n",
       sensor_value_to_double(&hum));

/* pressure */
printf("LPS22HB: Pressure:%.3f kpa\n",
       sensor_value_to_double(&press));

/* lps22hb temperature */
printf("LPS22HB: Temperature: %.1f C\n",
       sensor_value_to_double(&temp2));

/* lsm6dsl accel */
printf("LSM6DSL: Accel (m.s-2): x: %.1f, y: %.1f, z: %.1f\n",
       sensor_value_to_double(&accel1[0]),
       sensor_value_to_double(&accel1[1]),
       sensor_value_to_double(&accel1[2]));
```

Slika: Štampanje očitanih vrednosti

The End