Brezžični senzorji II

Sistemi Daljinskega Vodenja – Laboratorijske Vaje

24. november 2011

Povzetek

Ta naloga je prva v zaporedju nalog, ki se bodo osredotočale na programiranje brezžičnih vozlišč. V nalogi se bomo spoznali z osnovnim konceptom delovanja brezžičnega vozlišča. Videli bomo, da glavni program poteka v neskončni zanki, ki lahko reagira na vhodno/izhodne podatke preko prekinitev. Izhajali bomo iz preprostega okostja programa, nanj pa bomo dodali funkcionalnost preproste komunikacije s periferijo (prižiganje ledic) in implementirali možnost komunikacije preko serijskega porta UART. Bodite pozorni na strukturo pri pisanju programa, saj ga bomo v kasnejših laboratorijskih vajah počasi nadgrajevali, dodajali nove funkcionalnosti in na koncu dobili delujoče "inteligentno" vozlišče. Napišite poročilo, v katerem opišete kaj ste počeli in probleme s katerimi ste se srečevali. Poročilo morate oddate sprintano na naslednjih vajah. V Kolikor ne utegnete oddati v roku, se bo izhodiščna ocena tega poročila znižala na 80%.

1 Zagon preko vmesnika VMWare

Najprej zaženite Kubuntu preko virtualnega stroja VMWare in v njem odprite Terminal:

1. Zaženite VMPlayer s klikom na ikono na vašem namizju. Prikaže se vmesnik VMWare kot na Sliki 1 za izbiro datoteke, ki vsebuje vaš operacijski sistem. Izberite ikono "Open"in odprite datoteko z operacijskim sistemom Kubuntu, oziroma, kliknite kar na napis
   `freshveem Kubuntu 9.04` pod "Recent Virtual Machines".
2. Ko se Kubuntu zažene, vas vpraša za username/password. Vsi imate v izhodišču enako uporabniško ime "SDVx" in geslo "SDVx" (brez navednic).
3. Ko se naloži namizje Kubuntu, lahko nastavite resolucijo (velikost namizja) s klikom na gumb za orodno vrstico v spodnjem levem kotu (ikona kjer je črka "K"). Nato izberite "System Settings", pod "Computer Administration" izberite "Display" in nastavite resolucijo v okencu "Size".
4. Sedaj najprej lahko zaženemo terminalski vmesnik "Terminal". Ponovno zaženite orodno vrstico preko ikone s črko "K" v spodnjem levem kotu. Odpre se okno z večimi zavihki (kot pri Windows "start").
5. Terminal lahko najdete preko grafičnih zavihkov kot je to običajno v operacijskem sistemu Windows, zaradi hitrejšega iskanja pa uporabite okence "Search" v prvem zavihku. V polje pod "Search" vpišite "Terminal".

2 Naš prvi program na vozlišču

V tej nalogi bomo napisali naš prvi program, ki bo demonstriral zelo elementarno funkcionalnost vozlišča in komunikacije preko vmesnika UART. Namen tega programa je seznaniti se s strukturo tipičnih programov, ki lahko tečejo na vozlišču in dobiti občutek

Slika 1: Primer vmesnika VMWare .

za osnovne aplikacijskega vmesnika nanoStack. Izhajali bomo iz minimalnega programa, ki ga imate pripravljenega na spletni strani predmeta `http://vision.fe.uni-lj.si/Classes/SDV-vaje/vaje/2012/ex0.zip`. Paket si prenesite na namizje in ga razpakirajte v mapo `Work/Sensinode/NaniStack/NanoStack-v1.1.0/myProjects/ex0/`. Kot referenco boste uporabljali [4], [3], [2], ter navedene funkcije, ki so opisane v poglavju 2.0.2 .

2.0.1 Opis funkcionalnost programa

Delovanje programa naj bo sledeče. Ko se vozlišče vklopi, najprej posveti z ledico številka 1 in ledico številka 2 za pol sekunde, potem pa ju ugasne. Nato prične utripati na pol sekunde druga ledica (ledica s številko 2). S tem ledica signalizira, da je program v delovanju. Program mora vsebovati možnost komunikacije preko vmesnika UART s programom cuteCom. Program naj omogoča izpis trenutne številke komunikacijskega kanala z ukazom 'w', spreminjanje pa s 'C' in 'c'. Če pritisnemo 'C', se številka kanala poveča, če pritisnemo 'c', pa zmanjša – v obeh primerih se po spremembi preko UART izpiše v cuteCom tudi trenutna številko kanala. **Pazite!** *Številka kanala ne sme pasti pod 11 in ne sme preseči 26, saj tako zahteva definicija izračuna frekvenčnega pasu kanala.* Če v cuteCom pritisnemo tipko 't', nam mora program izpisati koliko časa (**v sekundah**) je preteklo od začetka izvajanja glavne (neskončne) zanke programa.

Primeri delovanja:
V cuteCom vpišemo: 'w', Vozlišče izpiše: 'Trenutna vrednost kanala je 22.'
V cuteCom vpišemo: 'c', Vozlišče izpiše: 'Trenutna vrednost kanala po spremembi je 21.'
V cuteCom vpišemo: 'C', Vozlišče izpiše: 'Trenutna vrednost kanala po spremembi je 22.'
V cuteCom vpišemo: 't', Vozlišče izpiše: 'Preteklo sekund od zagona zanke: 120'

2.0.2 Opis API, ki ga boste uporabljali v tej nalogi

Spodaj je spisek nekaterih funkcij in predefiniranih spremenljivk, ki jih boste uporabljali. Sicer so vse te funkcije opisane v dokumentaciji Sensinoda ali na spletni strani FreeRTOS, (v goole poiščite `FreeRTOS API.pdf`), vendar jih vseeno zaradi preglednosti povzemamo tukaj.

1. `LED1_ON(void), LED1_OFF(void), LED1_TOGGLE(void)` ... za delo z ledico LED1[1].
2. `LED2_ON(void), LED2_OFF(void), LED2_TOGGLE(void)` ... za delo z ledico LED2.
3. `void vTaskDelay( portTickType xTicksToDelay )` ... blokira izvajanje za `xTicksToDelay` procesorskih ciklov. Natančen opis najdete na strani freeRTOS [1].

[1]Operacija `OFF` ugasne ledico, `ON` jo prižge, `TOGGLE` pa spremeni stanje.

4. `portTICK_RATE_MS` ... je predefinirana spremenljivka, ki pove koliko milisekund traja en procesorski cikel.
5. `portTickType xTaskGetTickCount(void)` ... za branje števila procesorskih cikov od zagona programa. Natančen opis najdete na strani freeRTOS [1].
6. `int8_t debug( char *str )` ... izpiše na UART polje znakov. `str` je kazalec na polje znakov. Primer uporabe, ki vam bo prišel prav: `debug(''moj tekst'')` (glej [3]).
7. `uint8_t * debug_int(int y)` ... izpiše na UART celoštevilsko vrednost `y` (glej [3]).
8. `int16_t debug_read_blocking( uint32_t n_cycles)` ... blokira za `n_cycles` procesorskih

   ciklov in na izbranem portu UART. Funkcija implementira klic `uart0_get_blocking(time)` ali `uart1_get_blocking(time)`. Z UART vsakič prebere po en bajt in ga vrne v spremenljivki kot `int16_t` (`unsigned char`). Če podatkov ni, vrne `-1` – glej API za `uart1_get_blocking()` v [3][2].
9. `int8_t mac_current_channel()` ... vrne trenutno številko kanala, na katerem posluša vozlišče.
10. `portCHAR mac_set_channel(uint8_t channel_number)` ... postavi kanal na vrednosti `channel_number`.

Ukazi za spreminjanje in preverjanje komunikacijskega kanala so napisani v [4], v poglavju 6.1 – poglejte si to poglavje[3].

2.1 Pravilna inicializacija strojne in programske opreme, ter prvi program (50%)

Program bo najprej inicializiral strojno opremo in programski sloj na vozlišču. Ko se vozlišče prižge, bo prižgal ledici št.1 in 2 (LED1, LED2), počakal 500ms, ju ugasnil, nato pa vstopi v neskončno zanko. Znotraj neskončne zanke boste dodali kodo, ki bo na vozlišču prižigala in ugašala LED2 na pol sekunde.

1. Povežite vozlišče na programator s priloženimi priključnimi kabli (UART in PGR priključek). S priključkom USB priklopite programator na računalnik in preverite številko USB porta, ki mu jo dodeli operacijski sistem (`dmesg | tail`). Kasneje ne pozabite pred prenosom programa na vozlišče preveriti, če se številka USB priključka (`BSL_PORT`) ujema s tisto v datoteki z navodili za programiranje (`app.rules`).
2. Zaženite `Terminal` in se postavite na `Work/Sensinode/NaniStack/NanoStack-v1.1.0/myProjects/ex0/`. V tej mapi se nahaja okostje programa, s katerega bomo izhajali, in vse potrebne dodatne datoteke, vključno za datoteko `make`. Za preventivo najprej zaženite `make clean`. Z urejevalnikom besedil Kate odprite glavno datoteko s programom, `main.c` (`kate main.c&`).
3. Poglejte v [4] na strani 7 in 8 (samo do začetka poglavja 3.1) Katera koda v funkciji `main()` inicializira strojno opremo, razhoroščevalnik in programsko opremo na vozlišču. Inicializacijo si oglejte v `main.c` – to je tista koda, ki pride pred klicem `xTaskCreate(...)`.
4. Postavite se v glavno funkcijo, kjer bo tekel vaš program za prižignaje, ugašanje ledic in komunikacijo preko serijskega porta. To funkcijo smo določili za glavno funkcijo na začetku v `main()` preko `xTaskCreate()` in je v grobem sestavljena iz dveh delov. Na začetku so potrebne definicije in klici za nastavitev parametrov, potem pa sledi

[2]Izpisovanje preko UART vmesnika sicer implementira več funkcij, odvisno kakšnega tipa podatke želimo izpisovati. Te funkcije in njihov API najdete v [3], `NanoStack-v1.1.0/Common/include/` in `NanoStack-v1.1.0/Platform/nano/`

[3]Bodite pa pozorni, da je v specifikaciji tiskarski škrat pri funkciji za nastavljanje kanala.

neskončna zanka `for`, ki se nepretrgoma izvaja. V to zanko torej želimo vstaviti funkcionalnost vozlišča. Vidimo, da je pred neskončno zanko že vključena koda, ki prižge in ugasne ledice, vendar ji manjka pravilna zakasnitev. Tukaj dodajte manjkajoči del, ki bo zakasnil za *500ms*. Nato v glavno zanko na koncu, kjer je označeno, dodajte kodo, ki blokira izvajanje za $500ms$, in spremeni stanje ledici LED2. Pazite, da daste funkciji za blokiranje zakasnitev v ciklih, in ne v milisekundah!

5. Program prevedite in zapišite na vozlišče (`make`, `make program`). Ob pravilnem delovanju, bi morala ledica s številko dva utripati na pol sekunde. Rezultat pokažite asistentu, da si zabeleži uspešno opravljeni del naloge. Naredite tudi primerne printscreene in jih vključite v vaše poročilo.

2.2 Komunikacija preko vmesnika UART (50%)

Sedaj bomo dodali komunikacijo z vozliščem preko vmesnika UART, ki jo pogosto potrebujemo za razhroščevanje in jo bomo še posebej potrebovali pri kasnejših nalogah. V glavno zanko `for` boste vstavili kodo, ki blokira za 100ms, in bere na vmesniku UART (`debug_read_blocking()`, glej poglavje 2.0.2). Če v času 100ms ne prejme nobenega znaka (če funkcija vrne `-1`), potem nadaljuje z izvajanjem glavne zanke. V nasprotnem primeru, pa se na znak odzove, kot je opisano v spodnjih alinejah. Nekatere opise APIja za razhroščevanje si lahko pregledate v [3] (`Files->File List->debug.h`), več pa jih najdete v `NanoStack-v1.1.0/Common/include/` in `NanoStack-v1.1.0/Platform/nano/`.

Namig: Pri izpisu preko UART boste vsako sporočilo izpisali v svojo vrsto. Kot ukaz za novo vrstico uporabite `debug(''\r\n'')`. Preverjanje vhodnih znakov pa implementirajte s klasičnim C-jevskim `switch/case` stavkom (za uporabo glej, npr. [2]). V tem delu naloge boste uporabili funkcije, ki so opisane v poglavju 2.0.2 pod alinejami 6–10.

1. Če dobi znak 'w', izpiše trenutno vrednost številke komunikacijskega kanala.
2. Če dobi znak 'c', *najprej zmanjša* vrednost kanala za ena (ob tem ne pusti, da vrednost pade pod 11), *postavi kanal* na to vrednost, nato pa vrednost kanala izpiše na UART.
3. Če dobi znak 'C', *najprej poveča* vrednost kanala za ena (ob tem ne pusti, da vrednost preseže 26), *postavi kanal* na to vrednost, nato pa vrednost kanala izpiše na UART.
4. Če prejmemo znak 't', se izpiše *v sekundah* čas, ki je pretekel od začetka zagona *zanke* (bodite pozorni, da je vrednost `portTICK_RATE_MS` dana v milisekundah).

Program prevedite in prenesite na vozlišče. Zaženite in skonfigurirajte cuteCom kot ste to storili pri nalogi s predhodnih vaj (`Device: /dev/ttyUSBx`). Preizkusite komunikacijo preko vmesnika UART. Pozor: Če želite ponovno programirati vozlišče morate najprej ustaviti komunikacijo s CuteCom (`Close Device`). Če program deluje kot je zahtevano, pokličite asistenta, da preveri. Po opravljeni nalogi naredite printscreen ekrana tako, da se vidi cuteCom in glavna funkcionalnost. To sliko dajte v poročilo, kjer boste na kratko povzeli današnje delo. Poročilo prinesite na naslednje vaje.

Literatura

[1] *The FreeRTOS Project*, http://www.freertos.org/, October 2009.

[2] E. Huss, *The C Library Reference Guide*, 1997.

[3] Sensinode, Work/Sensinode/NanoStack/NanoStack-v1.1.0/Docs/html/index.html, *The NanoStack documentation*, 0.6 ed., 2008.

[4] Sensinode Ltd., *NanoStack Reference*, 2008.