Univerza v Ljubljani

Fakulteta za elektrotehniko

Žiga Bobnar

Wi-Fi radio

Projektna naloga

Modul B - Vgrajeni sistemi

Univerzitetni študijski program prve stopnje Elektrotehnika

Ljubljana, Junij 2021

Vsebina

1 Predstavitev projekta

V tej projektni nalogi bom predstavil izdelavo predvajalnika glasbe z uporabo mikrokrmilnika. Pri tem je bil cilj kot izziv uporabiti brezžično komunikacijo preko Wi-Fi za prenos zvočnih podatkov. Hkrati pa sem si zastavil nekakšno omejitev, da za celoten projekt uporabim samo material, ki ga imam že na voljo doma, to pa pomeni brez uporabe kakšnih zelo namenskih čipov (na primer MP3 dekodirnik), z izjemo čipov, ki jih je možno enostavno "reciklirati"iz izrabljenih oziroma za današnje standarde zastarelih naprav (na primer star avtoradio).

1.1 Zahteve projekta

Končano prvo fazo projekta (domena te projektne naloge) tako interpretiram kot nekakšno osnovno implementacijo ideje (ang. proof of concept), ki lahko brezžično predvaja glasbo iz strežnika, na zahtevo ustavi predvajanje oziroma ga nadaljuje, zvok pa naj bo zadosti kvaliteten recimo za uporabo v kakšni delavnici (torej primerljiv z FM sprejemnikom, ne neka avdiofilska kvaliteta).

Končan izdelek naj bo tudi prenosljiv, torej napajanje iz baterije, za enostavnost pa predpostavljam da je polnjenje Li-Ion baterij izvedeno zunaj izdelka, saj bi bilo sicer potrebno poskrbeti za ustrezno zaščito baterij in elektronike.

1.2 Osnovna struktura projekta

Celoten projekt sem tekom razvoja razdelil na tri ločene segmente. Prvi del je razvojna ploščica Arduino Due (v nadaljevanju Due), na kateri je mikrokrmilnik Atmel SAM3X8E. To je 32-bitni ARM mikrokrmilniški sistem, ki je jedro projekta in opravlja večino nalog, na primer krmiljenje LCD zaslona, komunikacija z modulom za brezžično povezavo ter predvajanje zvoka preko vgrajenega DA (digitalno-analognega) pretvornika. Drugi del predstavlja modul za brezžično komunikacijo, in sicer sem uporabil modul ESP-01, na katerem je mikrokrmilnik Espressif ESP8266EX (v nadaljevanju samo ESP8266). Ploščico Due sem z modulom ESP8266 povezal preko UART vmesnika. Tretji in zadnji del projekta pa je enostaven računalniški program, ki deluje kot strežnik za pretakanje glasbe (z ustreznim kodiranjem in vzorčno frekvenco), obenem pa omogoča tudi na primer prenos podatkov o trenutnem času, s čimer se izognem potrebi po ročnem nastavljanju ure.

1.3 Seznam uporabljenih elementov:

- Razvojna ploščica Arduino DUE (s krmilnikom Atmel SAM3X8E)
- Modul ESP-01 (s krmilnikom Espressif ESP8266EX)
- Dupont žičke za hitro povezovanje prototip (ang. Dupont jumper wire)
- Linearni regulator Motorola 7805CT (za zagotavljanje ustrezne 5 V napetosti za Arduino)
- Integrirani audio ojačevalnik TDA2030 (izvor neznan, verjetno iz nekega avtoradia)
- LCD zaslon, kompatibilen z vmesnikom Hitachi HD44780 (vzeto iz starega DVD predvajalnika)
- 6 mm pritisni gumb (taktilna tipka, ang. tactile button, reciklirano iz starega avtoradia)

- Li-Ion 3.7 V baterije tipa (dimenzije) 18650 z nominalno kapaciteto okrog 2400 mAh (baterije so pogosto uporabljene kot celice v starejših akumulatorjih prenosnih računalnikov, kjer lahko ena slaba baterija povzroči, da postane na prvi pogled celoten akumulator neuporaben, v resnici pa so nekatere posamezne celice lahko še povsem dobre, sploh za uporabo v nezahtevnih vezjih)

- Zvočnik (v mojem primeru bo to manjši 4 ohmski zvočnik)

- Ohišje (odločil sem se za 3D tiskanje ohišja, možna pa bi bila tudi uporaba lesa, plastičnih plošč, pločevine...)

1.4 Projekt due-radio (Arduino Due)

Za projekt sem uporabil znanje, enako programsko okolje (program Code::Blocks) ter kodo, ki smo jo pisali tekom laboratorijskih vaj pri predmetu Programiranje vgrajenih sistemov (Modul B na univerzitetnem programu elektrotehnike). Koda je napisana v jeziku C.

Projekt sem tako izdelal po predlogi, ki vsebuje nastavitve prevajalnika in razhroščevanja za krmilnik SAM3X9E. Preko JTAG priključkov priključen vmesnik Olimex ARM-USB-OCD-H, ki deluje kot programator in razhroščevalnik in omogoča enostavnejše programiranje.

V tem projektu sem uporabljal določene knjižnice ASF (ang. Advanced software framework, prej tudi Atmel Software Framework), kar omogoča hitrejši razvoj funkcionalnosti z uporabo že pripravljenih metod, namesto direktne interakcije z registri (v nekaterih primerih pa je le-ta hitrejša). Te knjižnice so na primer *clock*, *delay*, *ioport*, *serial*, *dacc*, *pio*, *tc*, *uart* ter *usart*.

Poleg teh knjižnic sem napisal oziroma dopolnjeval nekaj lastnih, ki smo jih razvijali na laboratorijskih vajah, primer tega so na primer gonilnik za LCD, FIFO vrsta (ang. first in, first out), gonilnik za DAC (bazira na obstoječem *dacc* gonilniku iz ASF) ter serijski vmesniki za uporabo UART in USART komunikacije na višjih nivojih.

1.4.1 Logična zgradba projekta

Ker je s časom postalo dodajanje daljših kosov kode v datoteke, ki sicer nimajo veze z dejansko prvotno mišljeno funkcijo te datoteke (na primer beleženje napak v dnevnik dogodkov znotraj gonilnika za komunikacijo z Wi-Fi, za kar hitro postane precej nepraktično vsakič podvajati celotno kodo za pisanje neposredno v medpomnilnik dnevnika), sem se odločil za lažji razvoj postaviti nekakšno logično delitev in abstrakcijo kode.

Tako je možno sedaj razdeliti projekt na naslednje module (vse, kar je potrebno za uporabo kode enega modula v drugem modulu naj bi bila vključitev datoteke *.h* modula in klic ustrezne funkcije):

- *esp_module.h*: Wi-Fi komunikacija, že glede na naslov projektne naloge verjetno najbolj ključen del celotnega projekta. Ta omogoča komunikacijo z modulom ESP-01, z uporabo strukturiranih ukazov in odzivov, preko serijskega vmesnika. Glavni problem te komunikacije je bila zagotovo visoka odzivnost, saj je za občutek realnočasnosti predvajanja posnetka potrebno zvočne vzorce v ustreznih intervalih nastavljati na izhod.

- *lcd.h*: LCD gonilnik za 16x2 zaslone, kompatibilne z HD44780 protokolom. Ta vsebuje funkcije za inicializacijo zaslona, čiščenje, nastavljanje besedila, poleg tega pa funkcije, ki so po funkcionalnosti zelo podobne funkciji *printf* in omogoča zelo enostavno formatiranje besedila (vstavljanje vrednosti dinamičnih argumentov v predpripravljeni obliki).

- *dac.h*: DAC gonilnik, omogoča inicializacijo ter nastavljanje izhodnih vrednosti na izbrani kanal digitalno-analognega pretvornika.

- *fifo.h*: FIFO vrsta, omogoča ciklično branje do zadnjega zapisanega elementa ter pisanje do zadnjega prebranega elementa. Poleg tega pa omogoča tudi "kukanje"(ang. peek), kar poenostavi preverjanje naslednjega podatka brez odstranjevanja.

- *console.h in console_definitions.h*: Serijska konzola. En od problemov programiranja tega projekta (več o tem v nadaljevanju), je omejeno število

znakov, ki se lahko na zaslonu prikažejo. Za namene enostavnejšega pregleda nad dejanskim dogajanjem sem zato napisal vmesnik za komunikacijo z računalnikom preko serijske konzole. Ta vmesnik omogoča poleg pisanja besedila formatiranega podobno kot pri LCD gonilniku s stilom *printf* tudi uporabo ukazov VT100, ki v večini modernih programov omogočajo spreminjanje barve besedila, torej je možno izdelati manj monoton pregled nad trenutnim stanjem.

- *timeguard.h*: Za potrebe omejevanja dovoljenega časa izvajanja nekaterih ukazov (na primer povezovanje v omrežje Wi-Fi, ki lahko traja), sem uporabil časovne števce, vgrajene v mikrokrmilnik. Z uporabo funkcij pa je možno pridobiti vrednost ustrezno skalirano v milisekunde oziroma sekunde. To omogoča, da lahko program, ki se izvaja v neskončni zanki prekine izvajanje, če je le-to predolgo.

- *buttons.h*: Gonilnik za odzivanje na pritiske gumbov. Uporabljen za človeško interakcijo s programom.

1.5 Projekt modem-esp8266 (ESP8266)

Drugi del projekta kot že prej omenjeno sestavlja modul ESP-01. To je modul, ki se je na tržišču pojavil okrog leta 2014, vgrajen mikrokrmilnik pa med drugim omogoča povezavo in komunikacijo z Wi-Fi napravami, ki delujejo pri frekvencah 2.4 GHz (to pomeni, da mora biti ta način delovanja vključen na sodobnejših dostopnih točkah, ki omogočajo tudi delovanje pri frekvencah 5 GHz). Teoretično bi lahko uporabil sam mikrokrmilnik na ploščici (ESP8266) za brezžično predvajanje glasbe, ampak modul nima zadosti priključkov. Na različici 01 je samo 8 priključkov, od tega pa so samo 4 priključki na voljo za programirljive izhode, kar pomeni da bi bilo precej oteženo priključiti še kakšne tipke in zaslon.

Zaradi teh omejitev bo modul opravljal samo nekakšno vlogo modema, ki prek serijske povezave sprejema ukaze od Due in jih ustrezno interpretira ter pošlje po brezžičnem omrežju in vrne ustrezno oblikovan rezultat.

Projekt je izdelan v okolju PlatformIO, ki je popularna razširitev urejevalnika

kode Visual Studio Code in omogoča programiranje velikega nabora vgrajenih sistemov. Za enostavnost (in ker sem želel čim večji fokus projekta posvetiti projektu na Due) sem kot osnovo uporabil knjižnice Arduino okolja, ki omogočajo enostavno sestavljanje besedilnih spremenljivk (ang. string). Poleg tega sem uporabil knjižnici *ESP8266WiFi.h* in *ESP8266HTTPClient.h*, napisani s strani proizvajalca čipa, ki omogočata enostavno povezovanje na dostopne točke in povezavno na HTTP strežnik.

Vsa koda projekta se nahaja v datoteki *main.cpp* in vsebuje vse od inicializacije do procesiranja in izvajanja ukazov na enem mestu. Za ta pristop sem se odločil, ker se mi projekt zdi dovolj enostaven, da ločitev na več datotek še ni potrebna.

1.6 Projekt wav-server (HTTP strežnik)

Za delovanje prenosa glasbe, sem na začetku imel v mislih direktno povezavo na že obstoječe internetne radio postaje. Te pogosto prenašajo zvok v neki različici MP3 kodiranja. Problem pa je ne samo v omejeni procesorski zmogljivosti krmilnika, ampak tudi v mojem nepoznavanju celotnega kompleksnega procesa obdelave MP3 signala v surov avdio signal. Možna bi bila uporaba oziroma adaptacija kakšne že obstoječe kode, ki počne to pretvorbo, a sem se odločil za drugačen pristop.

Pri razvoju projekta je bilo potrebno vse skupaj kar velikokrat testirati. To pomeni, da je vsaka dodatna povezava na zunanji strežnik nova spremenljivka, na katero imamo malo vpliva in lahko prinaša frustracije, kadar ne deluje pravilno. Temu sem se sprva poskušal izogniti z lokalno namestitvijo programa, ki bi deloval na enak način kot tisti komercialni programi, ki jih uporabljajo radijske postaje. Problem se pojavi ker imajo vsi ti programi na zastonjski različici zelo omejene funkcionalnosti in je njihova konfiguracija veliko bolj zapletena, kot bi morala biti za enostavno pretakanje glasbe.

Zato sem v projekt dodal še tretji del, ki pravzaprav na zunaj izgleda kot čisto običajen HTTP strežnik, torej deluje na enakem protokolu kot običajne spletne strani. Izkoristil pa sem možnost programskega vračanja odziva tako,

da lahko krmilnik zahteva s strežnika določen krajši segment izbranega posnetka (ang. chunk), vrne pa se kar neposredno zaporedje podatkov, kot bi bilo to kodirano v WAV formatu. WAV format omogoča precej bolj enostavno uporabo za nastavljanje izhoda, saj dejansko predstavlja zaporedne vrednosti v enakomernih intervalih s frekvenco vzorčenja (ang. sampling frequency). To pomneni, da potrebuje krmilnik samo pridobiti vrednosti posnetka in jih zaporedno v ustreznem časovnem intervalu nastavljati na izhod digitalno-analognega pretvornika.

Ker se ta program izvaja na računalniku, ki je dovolj zmogljiv za višjenivojske jezike, sem se odločil za izdelavo projekta na osnovi okolja Node.js in sicer v skriptnem jeziku TypeScript. To je nadnabor (ang. superset) v spletnem svetu znanega jezika JavaScript. Za razliko od slednjega ima TypeScript med drugim sposobnost preverjanja veljavnosti tipa spremenljivk in sintakse, uporabo razredov (ang. class) in vmesnikov (ang. interface). Vse to omogoča precej enostavnejše odkrivanje napak v primerjavi z JavaScript kodo, hkrati pa je še vedno na precej visokem nivoju programiranja, tako da je programiranje relativno hitro.

Za moje potrebe najpomembnejša lastnost pa je to, da ima že ogromno izdelanih knjižnic s kodo. Z njihovo uporabo lahko zelo hitro izdelamo spletni strežnik, ki vrača sprogramirane dinamične odzive. Poleg tega pa obstajajo knjižnice za branje in operacije z WAV posnetki.

2 Izdelava projekta

V tem poglavju bom predstavil sam proces izdelave projekta. Skozi izdelavo ne bom šel v kronološkem zaporedju dejanskega razvoja, saj se je marsikatera funkcionalnost prepletala z drugimi. Poskusil pa bom opisati proces tako, da mu je možno slediti in ob tem v vsakem naslednjem delu dograjevati izdelek. Torej v vsakem poglavju bom poskusil izdelati nekakšen zaključen funkcionalni del izdelka.

2.1 Krmiljenje LCD zaslona

Prva težava, ki se pojavi je opazovanje, kaj se z izdelkom dogaja. Za povratno informacijo je sicer možno z uporabo razhroščevalnika ustaviti program in spremljati dogajanje. Vendar pa to predstavlja zahtevo po računalniku. Precej bolj enostavno s stališča končnega uporabnika, je imeti že v sam izdelek vgrajen zaslon, ki lahko prikazuje trenutno stanje oziroma morebitne napake.

2.1.1 LCD HD44780

Izbrani LCD uporablja krmilnik kompatibilen s precej pogosto uporabljenim protokolom Hitachi HD44780. Ta krmilnik omogoča, da z ukazi lahko v zaslon zapišemo ustrezne vrednosti za prikaz besedila, po tem pa ne potrebujemo več skrbeti za periodično osveževanje samega zaslona, saj to dela vgrajen krmilnik. Poleg tega omogoča dva načina prenosa podatkov, 8-bitni ali dvakrat po 4-bitni. Zasloni tega tipa se lahko razlikujejo med seboj na različne načine, na primer število vrstic, število stolpcev, število pikslov na znak, barva ozadja, barva

ospredja, možnost osvetlitve in barva osvetlitve. Najpogostejši so zasloni z znaki v 16 stolpcih in 2 vrsticah.

Za samo analizo protokola je najboljše uporabiti kar dokumentacijo proizvajalca (oziroma kar izvornega HD44780), ki jo je možno dobiti na internetu. Za pošiljanje ukazov v sam zaslon so pomembni priključki E (Enable), RS (Register select), R/W (Read/Write) in podatkovne linije D7-D0. Kadar ima RS logično vrednost 0 pomeni uporabo ukaznega registra, pri vrednosti 1 pa je aktiven podatkovni register (za znake). Kadar je R/W enak 0, je zaslon konfiguriran za branje v smeri iz zaslona, za vrednost 1 pomeni pisanje v zaslon (naj bo to ukaz ali pa vrednost znaka). Za prenos vsake vrednosti v zaslon moramo linijo E za kratek čas postaviti na vrednost 1 in nato nazaj na 0.

Tabela 2.1: Ukazi za HD44780 (ang. instruction set)

Instruction	RS	R/W	D7-D0
Clear display	0	0	0000 0001
Return home	0	0	0000 001-
Entry mode set	0	0	0000 01IS
I =¿ I/D=1 - increment			
I =¿ I/D=0 - decrement			
S=1 - Accompanies display shift			
Display on/off control	0	0	0000 1DCB
D=1 - sets display on			
D=0 - sets display off			
C=1 - sets cursor on			
C=0 - sets cursor off			
B=1 - enables cursor blinking			
B=0 - disables cursor blinking			
Cursor or display shift	0	0	0001 SR–
S =¿ S/C=1 - display shift			
S =¿ S/C=0 - cursor move			
R =¿ R/L=1 - shift to the right			
R =¿ R/L=0 - shift to the left			
Function set	0	0	001L NF–
L =¿ DL=1 - 8-bit data length mode			
L =¿ DL=0 - 4-bit data length mode			
N=1 - 2 lines			
N=0 - 1 line			
F=1 - 5x10 dots character font			
F=0 - 5x8 dots character font			
Set CGRAM address	0	0	01AA AAAA
A =¿ ACG - CGRAM address (character gen RAM)			
Set DDRAM address	0	0	1AAA AAAA
A =¿ ADD - DDRAM address (display data RAM)			
Read busy flag address	0	1	BAAA AAAA
B =¿ BF=1 - internally operating			
B =¿ BF=0 - instructions acceptable			
A =¿ AC - Address counter			
Write data to CG or DDRAM	1	0	[Write data value]
Read data from CG or DDRAM	1	1	[Read data value]