Digitalni prikazovalnik - verzija 2: Uporaba GUI-O vmesnika

V tej knjigi najdete poglavje, kjer sem opisal projekt Digitalni prikazovalnik. Uporabniški vmesnik tega digitalnega prikazovalnika je bil sestavljen iz spletne strani, do katere je uporabnik lahko dostopal in kjer bi vnesel besedilo, ki ga želi prikazovalnik prikazati.

Dolga sporočila bi se počasi pomikala v eno ali drugo smer. Ker bi digitalni prikazovalnik občasno prikazoval tudi čas, je obstajal tudi način za prilagoditev ure realnega časa prikazovalnika na pravilen čas. Slika 1 je fotografija prikazovalnika, ki prikazuje čas.

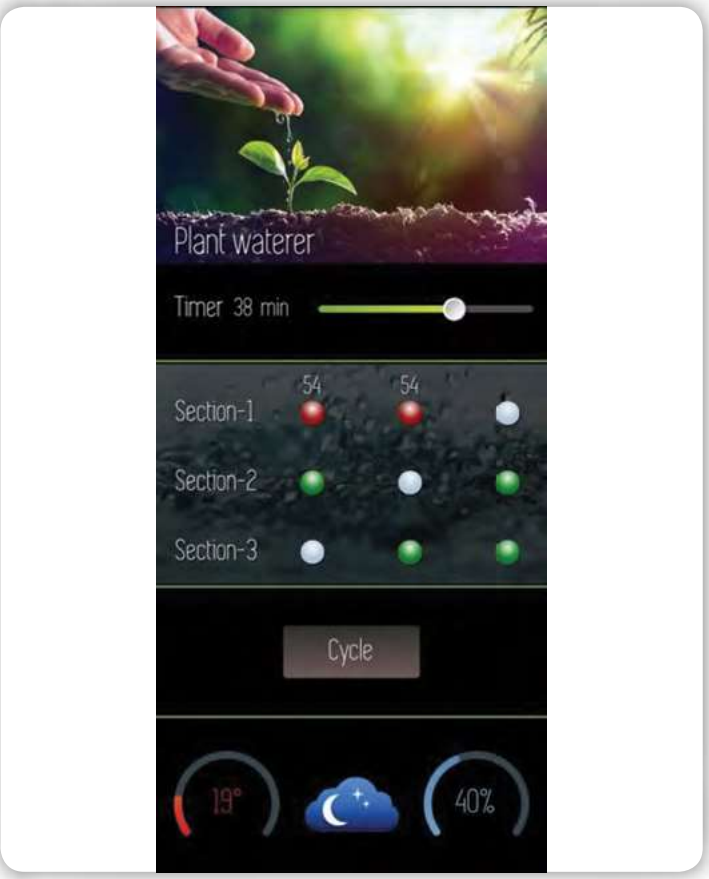
Slika 1

Digitalni prikazovalnik je kot krmilnik uporabljal ESP8266, saj lahko zlahka gosti spletni strežnik. Napisal sem dve različici programske opreme:

- *ESP8266 bi bil nastavljen v načinu dostopne točke. V tem načinu bi moral uporabnik izbrati ESP8266 kot delujočo Wi-Fi dostopno točko in se z brskalnikom pomakniti na spletno stran. Ker ni bilo neposredne povezave z internetom, je ta različica vsebovala čip - uro realnega časa za spremljanje časa.*
- *ESP8266 bi se povezal s katero koli Wi-Fi dostopno točko, ki je bila na voljo v bližini. V tem primeru uporabniku ne bi bilo treba izbrati nove Wi-Fi dostopne točke, temveč bi se le pomaknil na spletno stran prikazovalnika. V tem primeru je bila nastavitev časa digitalnega prikazovalnika izvedena prek interneta z uporabo NTP časovnega strežnika.*

Mislil sem, da je zasnova tega projekta precej praktična in poceni. Pred kratkim sem izvedel za programsko opremo GUI-O, ki jo je napisalo slovensko podjetje. Programska oprema GUI-O je v bistvu aplikacija, ki deluje na Android pametnem telefonu ali tabličnem računalniku. Z MCU, ki je uporabljen v vašem projektu po meri, lahko komunicira na različne načine, od katerih je večina brezžična (na primer Bluetooth).

Aplikacija GUI-O je v bistvu tolmač: bere ukaze, ki temeljijo na besedilu, ki ga pošlje MCU vašega projekta, in jih prikaže z uporabo najrazličnejših zaslonskih pripomočkov. S prilagajanjem ukazov, ki jih pošljete v GUI-O aplikacijo, lahko na Android napravi prikažete »Nadzorno ploščo«, ki vsebuje veliko ali malo pripomočkov, pač kolikor jih potrebujete za upravljanje/nadzor vaše zunanje naprave. Ti pripomočki so bodisi vhodni objekti, občutljivi na dotik, kot so drsniki, bodisi statusni objekti, ki prikazujejo stanje, ki je prisotno v zunanji napravi. Slika 2 prikazuje primer »nadzorne plošče« GUI-O z drsnikom, nekaterimi indikatorji/merilniki stanja in lepo sliko.



Slika 2

Nekdo iz GUI-O ekipe me je kontaktiral, ko je prebral moj članek o Digitalnem prikazovalniku v reviji Svet Elektronike, in predlagal, da bi bila GUI-O aplikacija še en način za izvedbo mojega projekta. V bistvu bi lahko opustil idejo o spletnem mestu in uporabil GUI-O aplikacijo na Android telefonu za nadzor digitalnega prikazovalnika, na primer prek Bluetooth. Strinjal sem se, vendar je prišlo do enega zapleta - GUI-O je trenutno samo aplikacija za Android, jaz pa uporabljam samo IOS (iPhone in iPad) naprave.

Vendar sem imel srečo – obrisal sem prah s stare Samsung Galaxy Note 10.1 tablice, ki je nisem uporabljal že vrsto let – Samsung je opustil posodabljanje tablice Note 10.1 nazaj na Android 4. Z iskanjem po Youtube sem odkril, kako posodobiti operacijski sistem na tem starem tabličnem računalniku na Android 9, s pomožnim nalaganjem Lineage 16 operacijskega sistema. S tem novim operacijskim sistemom sem lahko prenesel brezplačno demo različico GUI-O iz trgovine Google Play.

Po nalaganju enega od vzorcev na spletnem mestu GUI-O v ESP32 razvojno ploščo, mi je uspelo zagnati GUI-O aplikacijo in komunicirati z ESP32 z uporabo Bluetooth, vse dokaj hitro. To je bilo impresivno, zato sem se odločil, da poskusim svoj projekt Digitalni prikazovalnik pretvoriti v nadzor z GUI-O.

GUI-O komunikacijski protokoli

Preden opišem, kako sem projekt Digitalni prikazovalnik pretvoril v GUI-O, je koristno razpravljati o različnih metodah povezave, za katere je mogoče konfigurirati GUI-O. Te metode so naslednje:

1. *Bluetooth (klasični način z uporabo serijskega perifernega profila ali SPP)*
2. *Bluetooth z nizko porabo energije (BLE)*
3. *Wi-Fi (z razpoložljivo dostopno točko)*
4. *Ethernet (žični)*
5. *USB (žični)*

Za telefon ali tablico sta najpogosteje uporabljeni metodi 1 in 2. Metodi 3 in 4 sta v bistvu načina povezave, pri katerih se telefon/tablični računalnik Android povežeta z MQTT strežnikom, ki gostuje v internetu, in vaša naprava po meri se poveže z istim MQTT strežnikom, bodisi brezžično ali prek Etherneta. Številka pet, način USB povezave deluje samo, če vaš telefon lahko zagotovi emulirana COM vrata z uporabo USB vrat, ki so namenjena tudi polnjenju baterije.

Ker sem že zasnoval spletni vmesnik za Digitalni prikazovalnik, sta bili samo metodi 1 in 2 zanimivi možnosti. Moram priznati, da čeprav sem naredil veliko projektov z ESP8266 in ESP32 podjetja Espressif, sem pozabil, da ESP8266 upravlja samo Wi-Fi, ne pa tudi Bluetooth. To je pomenilo, da ESP8266, ki sem ga uporabil v prvotnem projektu, ne bi deloval z Bluetooth GUI-O metodo povezave. Vendar sem bil navdušen nad preizkusom GUI-O, zato sem se odločil zgraditi novo krmilno ploščo za Digitalni prikazovalnik z uporabo ESP32.

Klasični Bluetooth proti BLE

Ker redno uporabljam iPhone/iPad, sem imel nekaj izkušenj s poskusom povezovanja lastnih naprav po meri s temi IOS napravami. Glede na to, kar sem ugotovil, bodo IOS naprave delovale z BLE, vendar ne delujejo s klasičnim Bluetoothom – vsaj ne z uporabo Serial Peripheral Profile, ki ga najraje uporabljam. Android naprave očitno delujejo z obema Bluetooth protokoloma, saj je GUI-O aplikacija samo za Android in podpira oba.

Vsekakor obstaja prostor tako za klasični Bluetooth z uporabo SPP, kot za BLE, glede na nadzor/nadzor zunanje naprave, s pametnim telefonom. Poglejmo si različne prednosti/slabosti.

Klasični Bluetooth v SPP načinu:

- *Razmeroma preprosto pisanje kode, saj ta protokol v bistvu samo posnema standardno serijsko UART povezavo.*
- *Dodajanje Bluetooth SPP povezave na osebni računalnik (za razvoj in odpravljanje napak) je preprosto z uporabo poceni USB-Bluetooth modula HC05.*
- *Zahteva več energije – to je v resnici samo premislek na zunanjem perifernem koncu povezave in le takrat, ko ga napaja majhna baterija, kot je recimo gumb baterija.*

Bluetooth Low Energy (BLE):

- *Precej težko je napisati kodo za zunanji MCU del povezave. To se zgodi, ker obstaja veliko vnaprej določenih plasti protokola, ki se jih je treba naučiti: Storitve, Karakteristike, Vrednosti itd. Vsi ti parametri so določeni z uporabo dolgih šestnajstiških nizov, ki si jih je težko zapomniti, zato ste dovzetni za napake.*
- *Če lahko obvladate zaplete v 1) zgoraj in če vaša naprava izvaja običajno funkcijo, boste morda lahko uporabili obstoječo komercialno aplikacijo na delu povezave s telefonom/tablico. To pomeni, da vam morda ne bo treba napisati lastne aplikacije po meri za telefon.*
- *Kot pove že ime, BLE porabi manj energije, zato je koristno, če vaša zunanja periferna naprava deluje na majhno baterijo, je nenadzorovana in mora delovati dolgo časa.*

V primeru digitalnega znaka le-ta črpa 5-10 vatov energije iz električnega omrežja (odvisno od nastavljenega nivoja svetilnosti). Edina uporabljena baterija je gumb celica, ki napaja samo čip RTC, tako da Bluetooth poraba energije ni problem - tudi na telefonu, saj se konfiguracija digitalnega znaka izvaja redko. Iz tega razloga sem za ta projekt izbral klasični Bluetooth način z uporabo SPP.

GUI-O gradniki

Projekt Digitalni prikazovalnik ne zahteva modernega uporabniškega vmesnika ali »nadzorne plošče«. Pravzaprav je začetna različica mojega projekta uporabljala zelo preprosto spletno stran, ki je vsebovala le nekaj besedilnih polj in gumbov. Ta različica projekta ima bolj moderno "nadzorno ploščo" kot spletna različica, vendar še vedno uporablja le majhen del pripomočkov, ki so na voljo v GUI-O. Slika 3 prikazuje nekaj GUI-O gradnikov – nekatere sem uporabil v tem projektu.

Priročnik za GUI-O razvijalce je zelo dobro napisan in ilustriran. Opisal bom samo nekaj osnov, povezanih z nastavitvijo vaše nadzorne plošče in branjem rezultatov, ki jih pošlje nazaj, ko uporabnik na nek način komunicira s to nadzorno ploščo.

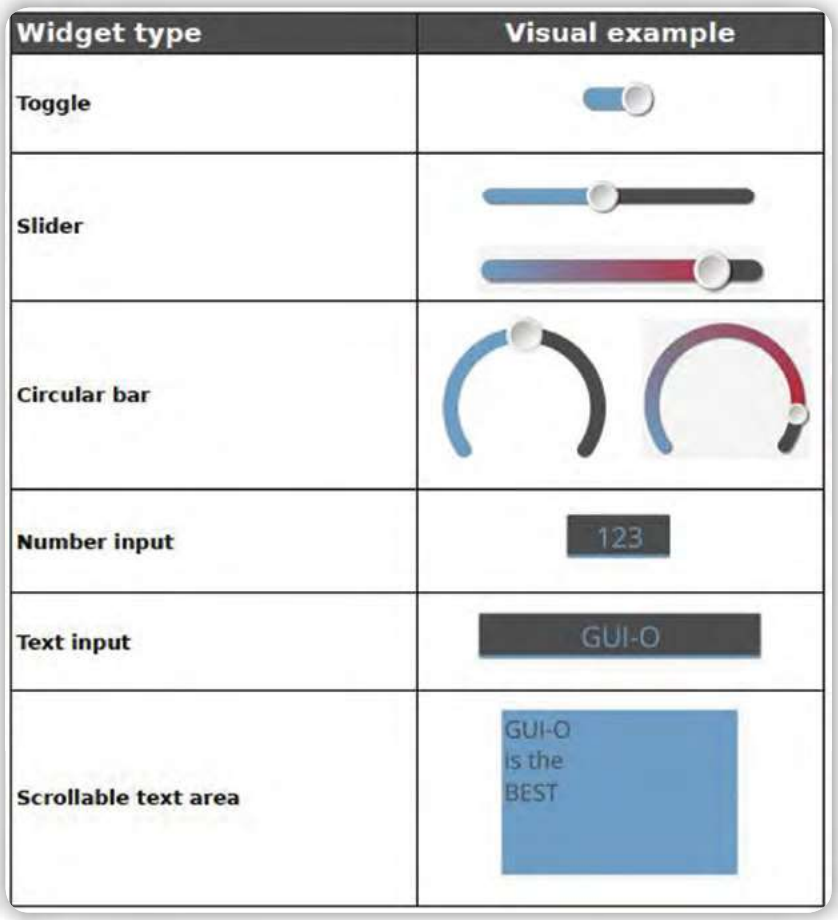
Ko zaženete GUI-O aplikacijo v telefonu, morate določiti povezavo z neko zunanjo napravo. O tem postopku bom razpravljal pozneje, a za zdaj je dovolj vedeti, da bo GUI aplikacija poslala naslednje sporočilo z zahtevo za inicializacijo prek Bluetooth SPP na ESP32:

```
@init
```

Vaš program mora spremljati dohodna Bluetooth sporočila in ko prejme to inicializacijsko sporočilo, se mora odzvati s pošiljanjem serije sporočil, da na zaslon postavi potrebne pripomočke. Ta postopek se začne z nekaj preprostimi ukazi za brisanje zaslona in nastavitev barve ozadja:

```
sendMsg("@cls\r\n");
sendMsg("@guis BGC:#FFFFFF\r\n");
```

Upoštevajte, da je sendMsg() funkcija, ki pošlje ukazno sporočilo nazaj v Android telefon/GUI-O aplikacijo prek Bluetooth SPP. Medtem ko se predpona »@« uporablja za številne interakcije z GUI-O aplikacijo, ko postavljate nove pripomočke na zaslon, te pripomočke določite z ukazom, ki se vedno začne s simbolom črte »|« . Seznam 1 je koda, ki sem jo uporabil za risanje nadzorne plošče za ta projekt.



Slika 3

Vsaka vrsta gradnika vsebuje ime iz dveh črk: LB je oznaka, TI je vnos besedila, SL je drsnik itd. Vsakemu gradniku je treba dodeliti edinstven ID uporabnika (UID) – ta UID se uporabi drugje v vaši kodi, če spremenite kateri koli atribut tega gradnika ali preberite njegovo stanje (v primeru uporabniško nastavljivih gradnikov, kot je na primer drsnik ali gumb). Vsi atributi gradnika, kot so velikost, barva, položaj itd. so določeni z označevalcem (z velikimi črkami), ki mu sledi »:« in nato vrednost tega atributa.

Eno posebnost teh ukaznih vrstic lahko vidite v naslednjem ukazu:

```
sendMsg("|LB UID:lb1 X:50 Y:10 FSZ:5 FFA:\"font1\"
TXT:\"ESP32 Digital Sign\"\r\n");
```

Upoštevajte, da je za TXT parametrom (ki določa besedilo, za katerega želite, da je prikazano v gradniku Label) prikazan »\« pred vsakim pojavom simbola dvojnega narekovaja. "\" je simbol ESCAPE. Potreben je, ker funkcija sendMsg pričakuje posredovanje spremenljivke String. Ker je spremenljivka String zaključena s simbolom dvojnega narekovaja, in če morate v ukazni niz GUI-O vdelati dvojni narekovaj, morate uporabiti simbol ESCAPE »\«, sicer bo funkcija sendMsg predvidevala, da se niz konča s prvim dvojnim narekovajem, ki ga najde znotraj ukaza.

Najboljši način za razumevanje GUI-O gradnikov je, da si ogledate mojo kodo v seznamu 1, si ogledate stran priročnika za razvijalce za vsak uporabljeni gradnik in pogledate tudi, kako je videti na zaslonu vašega telefona.

Kadar koli uporabnik komunicira z GUI-O aplikacijo, na primer s premikanjem drsnikov, se GUI-O odzove z:

- *ustrezno spreminjanje videza ikone na zaslonu*
- *pošiljanje sporočila, kot je @sl1 20.0\r\n*
- *kar pomeni, da se je slider1 premaknil na 20.0*

Vsi takšni odgovori se bodo začeli s simbolom »@«, ki mu bo sledil UID gradnika, ki smo se ga dotaknili. Vaš MCU bo moral spremljati dohodna sporočila iz pametnega telefona prek Bluetooth povezave za ta sporočila in jih razčleniti, da določi ime gradnika, ki smo se ga dotaknili, in kakšna je nova vrednost. V mojem programu je ta funkcija parseGuioMsg() funkcija. Na primer, ko uporabnik vnese besedilo v polje za vnos besedila ti1, bo GUI-O odgovoril:

```
@ti1 xxxxxxx\r\n
```

kjer je xxxxxx besedilo, ki je bilo vneseno v polje za vnos