MicroPython za mikrokontrolere

Projekti za Thonny-IDE, uPyCraft-IDE i ESP32

Dr Günter Spanner

Agencija Eho
www.infoelektronika.net

• Sva prava zadržana. Nijedan deo ove knjige ne sme biti reprodukovan u bilo kom materijalnom obliku, uključujući fotokopiranje ili slučajno ili nenamerno smeštanje na bilo koji elektronski medijum sa ili uz pomoć bilo kog elektronskog sredstva, bez pismenog odobrenja nosioca autorskih prava osim u skladu sa odredbama zakona o autorskim pravima, dizajnu i patentima iz 1988. godine ili pod uslovima izdatim od Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London, England W1P 9HE. Prijave za pismene dozvole radi štampanja bilo kog dela ove publikacije upućuje se izdavaču ove knjige.

• Izjava: Autor i izdavač su uložili najveće napore da bi se obezbedila tačnost informacija sadržanih u ovoj knjizi. Autor i izdavač ne mogu da pretpostave neprijatnosti i ovom izjavom isključuju bilo kakvu odgovornost za bilo koju stranku koja bi imala gubitke ili štetu uzrokovanu greškama ili propustima u ovoj knjizi, bez obzira da li su greške ili propusti nastali usled nemara, nezgode ili bilo kog drugog razloga.

ISBN 978-86-80134-38-3

MicroPython za mikrokontrolere

Naslov originala: MicroPython for Microcontrollers
Autor: Günter Spanner

Prevod: Biljana Tešić

Izdaje i štampa: Agencija Eho, Niš
e-mail: redakcija@infoelektronika.net

Tiraž: 200

Godina izdanja: 2022

CIP - Каталогизација у публикацији
Библиотеке Матице српске, Нови Сад

004.43PYTHON

ШПАНЕР, Гинтер, 1965-
MicroPython za mikrokontrolore : projekti za Thonny-IDE, uPyCraft-IDE i ESP32 / Günter Spanner ; [prevod Biljana Tešić]. - Niš : Agencija Eho, 2022 (Niš : Agencija Eho). - 226 str. : ilustr. ; 24 cm

Tiraž 200.

ISBN 978-86-80134-38-3

a) Програмски језик "Python" -- Микроконтролори

COBISS.SR-ID 57696009

Poglavlje 1 - Uvod

Uvođenje ESP32 čipa kompanije Espressif Systems označava novu generaciju mikrokontrolera, koji obezbeđuje odlične performanse, Wi-Fi i Bluetooth funkcionalnost po ceni bez premca. Ove karakteristike su munjevito osvojile scenu proizvođača. Najrazličitije aplikacije i projekti u oblastima Internet stvari (IoT) i kućne automatizacije mogu se implementirati lako i ekonomično. ESP32 je lako programirati, kao i klasične Arduino ploče. Međutim, ako uporedimo sa drugim, ESP32, između ostalog, takođe nudi:

- veću fleš i SRAM memorija
- mnogo veću brzinu procesora
- integrisani Wi-Fi / WLAN
- Bluetooth funkcionalnost
- više GPIO pinova
- opsežnu funkcionalnost interfejsa
- analogno/digitalni pretvarač sa višom rezolucijom
- digitalno/analogni pretvarač
- bezbednosne funkcije i funkcije enkripcije

Iz ovih razloga, može se smatrati najperspektivnijim naslednikom Arduina. Izraz „ubica Arduino“ se često koristi u ovom kontekstu.

Ova knjiga predstavlja programiranje savremenih sistema sa jednim čipom (Systems on Chip – SoCs). Pored tehničke pozadine, fokus je na programskom jeziku Python, posebno u svojoj varijanti „MicroPython“. Osnovne veze između elektronike i elektrotehnike biće razmatrane samo u meri u kojoj je to neophodno za dizajn električnog kola i eksperimente.

„Hardver“ za početak ionako može biti veoma jednostavan. Na početku je potrebna samo kontrolna ploča i neke diode koje emituju svetlost, kao i odgovarajući serijski otpornici. Računar ili laptop potreban za programiranje čipa treba da bude dostupan u svakom domaćinstvu. Odgovarajuće programsko okruženje može se besplatno preuzeti sa interneta. Međutim, kada koristite programsko okruženje MicroPython, brzo se javljaju prvi problemi. Dobar uvod stoga može dovesti do odličnih performansi.

Python je doživeo ogroman uspon poslednjih godina. Različiti sistemi sa jednom pločom poput Raspberry Pi-a posebno su doprineli njegovoj popularnosti. Ali, upotreba Python-a preovladava i u drugim oblastima, kao što su veštačka inteligencija ili mašinsko učenje. Stoga je logičan izbor da se koristi Python ili varijanta MicroPython-a i za primenu u SoC-ovima.

Međutim, nećemo se ograničiti na puki uvod u programski jezik. U mnogim slučajevima, naučene veštine programiranja se primenjuju u praksi, zajedno sa električnim ko-

lima. Kompletno opisani projekti su pogodni za upotrebu u laboratorijama ili u svakodnevnom životu.

Osim edukativnog efekta, u prvom planu je i zadovoljstvo sklapanja kompletnih i korisnih uređaja. Korišćenjem laboratorijskih priključnih ploča, električna kola svih vrsta mogu se realizovati uz malo truda. Testiranje i isprobavanje aplikacija tako postaje obrazovno zadovoljstvo.

Zbog različitih primena, kao što su meteorološke stanice, digitalni voltmetri i generatori funkcija, predstavljeni projekti su takođe idealno pogodni za stažiranje ili studijske kurseve iz prirodnih nauka ili za časove nauke i tehnologije.

1.1 Python, C ili Arduino?

Za početnike, Arduino programsko okruženje je jedno od najlakših mesta za programiranje ESP32. Posle ovog interfejsa sledi Arduino verzija C, kao i C++. Ova dva programska jezika su godinama popularna u razvoju ugrađenih sistema. Arduino verzija jezika C je dodatno olakšala početak. Osim toga, jedna od najvećih tehnoloških zajednica na svetu razvila se za ovu svrhu. Zahvaljujući novim bibliotekama, softverskim ispravkama i podrškom za ploče, problemi bi se obično mogli brzo rešiti. Međutim, to što Arduino-C radi samo u svom predviđenom okruženju nije beznačajno. Posebno za razvoj opsežnih projekata nedostaju korisne i važne funkcije. Stoga je Arduino-C ostao uglavnom ograničen na hobi projekte i projekte za početnike.

MicroPython je relativno nov. Korisnička zajednica raste, a podržava se sve više platformi. MicroPython je u suštini „tanka“ verzija Pythona, jednog od najpopularnijih programskih jezika na svetu. Stoga se specifični problemi mogu rešavati i u drugim zajednicama, a ne samo u MicroPython zajednicama. U stvari, opšti Python forumi sve više doprinose rešavanju MicroPython problema.

Osim podrške zajednice, MicroPython takođe ima određene funkcije koje ga stavljaju daleko iznad klase Arduina. Jedna od ovih funkcija je takozvana REPL funkcija. REPL je skraćenica za „Read-Evaluate-Print Loop“. Ova funkcija omogućava da se programi i delovi koda brzo izvrše. Kompajliranje ili otpremanje nije potrebno. Na ovaj način se delovi koda mogu testirati brzo i efikasno tokom razvoja.

MicroPython sadrži veoma kompaktnu implementaciju Python interpretera. Potrebno je samo 256 KB fleš memorije i 16 KB RAM-a. Ipak, interpreter je dizajniran za maksimalnu kompatibilnost sa standardnim Python-om. Sintaksa i opseg jezika u velikoj meri odgovaraju Python verziji 3.4, tako da bi iskusni Python programeri trebalo odmah da se snađu. Osim toga, definisano je nekoliko jezičkih elemenata izvan standarda, koji održavaju niske zahteve za memorijom i povećavaju brzinu izvršavanja.

1.2 Zahtevi

Da biste uspešno koristili knjigu, potrebno je ispuniti sledeće uslove:

- Sigurno upravljanje operativnim sistemom Windows;
- Osnovno poznavanje bilo kog programskog jezika, kao što je C, Java ili slični;
- Podrazumeva se osnovno znanje iz oblasti elektronike, posebno iz oblasti „struja – napon – „otpor".

Za specijalizovano znanje u oblasti elektronike, pogledajte opsežnu tehničku literaturu, posebno od Elektora, u vidu knjiga, časopisa i kompleta.

Struktura hardvera je namerno jednostavna, jer fokus treba da bude na programiranju pomoću MicroPython-a. Ipak, potrebne su različite komponente i delovi. Objašnjenja se mogu naći u pojedinačnim poglavljima u kojima se komponente prvo koriste. Osim toga, u poslednjim odeljcima knjige objašnjene su neke osnovne komponente, kao što su otpornici ili diode koje emituju svetlost. Možete pogledati te odeljke ako imate nejasnoće u vezi sa pojedinačnim komponentama.

Dodatni zahtevi su

- računar ili laptop sa USB interfejsom,
- operativni sistem Windows 10,
- pristup internetu.

Takođe je korisno koristiti aktivno USB hub između računara i kontrolera. Ono ima prednost, jer garantuje određenu zaštitu za računar. Hub treba da ima svoje napajanje od 5 V pomoću zasebne jedinice za napajanje. Onda su računar ili laptop najbolje zaštićeni od kratkih spojeva iza hub-a, jer je malo verovatno da će kratki spoj da se „probije" do aktivnog hub-a na USB portu računara.

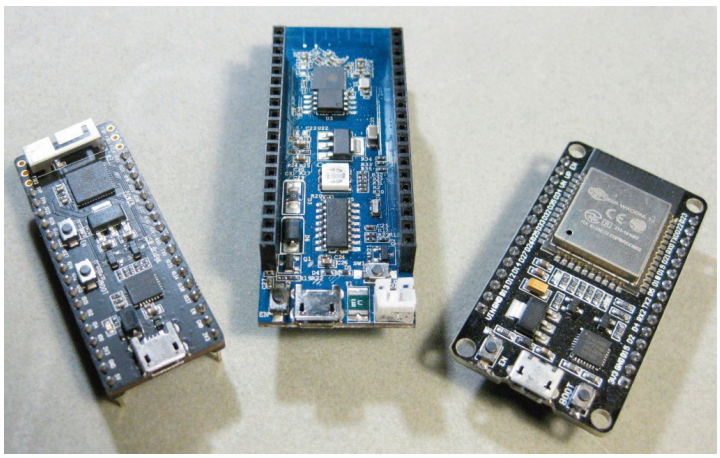
Poglavlje 2 – Raznolikost ESP ploča

ESP32 je moderan i izuzetno moćan mikrokontroler. Osim visoke frekvencije radnog takta i opsežnih unutrašnjih funkcionalnih jedinica, čip ima integrisan Wi-Fi i Bluetooth. Kontroler je razvila Espressif Systems, kineska kompanija sa sedištem u Šangaju, koja uživa sve veću popularnost. Funkcije performansi ESP-a daleko prevazilaze dobro poznate Arduino ploče u pogledu cene i performansi.

Pošto je ESP32 dostupan samo kao SMD čip, potreban je takozvani break-out board (BoB) ili razvojna ploča ukoliko se kontroler koristi u neprofesionalnom okruženju. U međuvremenu, na tržištu je dostupan niz različitih verzija kojima se praktično ne može upravljati. Ovo su najpoznatije verzije:

Board	Pins	Buttons	LiPo Charger
ESP32-PICO-KIT	34	EN and BOOT	no
JOY-iT NodeMCU	30	EN and BOOT	no
ESP32 DEV KIT DOIT	30/36	EN and BOOT	no
Adafruit HUZZAH32	28	RESET	yes
ESP32 Thing	40	EN and BOOT	yes
LOLIN32	40	EN and BOOT	no
Node-ESP-Board	38	EN and BOOT	no

Na sledećim slikama prikazane su samo tri različite verzije:



Slika 2.1 – ESP32 ploče (PICO-KIT, Node-ESP i NodeMCU)

Ove ploče imaju neophodne preduslove za rad ESP kontrolera na priključnoj ploči ili prototipskoj ploči bez lemljenja. Često se, osim kontrolera, na ploče montiraju i druge

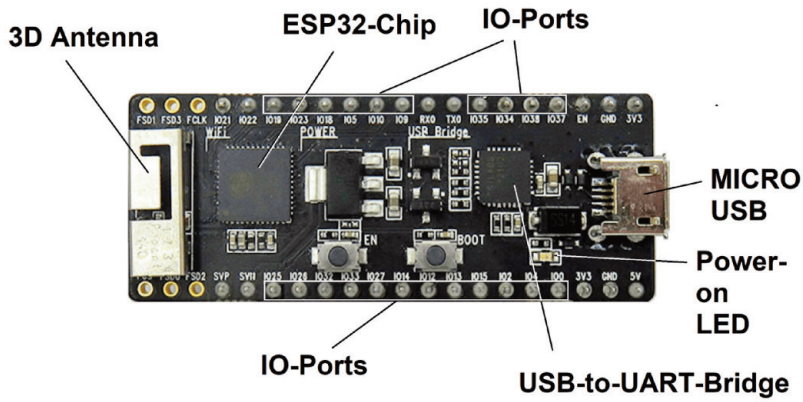
komponente, kao što su tasteri, punjač litijum-jonskih baterija ili razni LED-ovi koji su montirani na dostupnim pločama. To znači da se početni testovi i eksperimenti mogu izvoditi bez eksternih kola.

U sledećem odeljku sumirani su najvažniji podaci o ESP32. Pregled treba da obezbedi samo prvi utisak. Detaljnije objašnjenje pojedinačnih karakteristika i funkcija je dato u relevantnim odeljcima u knjizi.

Procesor:	160 / 240 MHz Tensilica LX6 mikroprocesor sa dva jezgra
Memorija:	520 KB SRAM / 16 MB fleš memorije
Napajanje:	2,2 V do 3,6 V
Potrošnja energije:	Standardno: pribl. 50 – 70 mA Wi-Fi režim: pribl. 80 - 170 mA Režim dubokog „spavanja“: pribl. 2,5 µA
Temperatura okoline:	od –40 °C do +125 °C
Ulazi/Izlazi (GPIO-ovi):	32 porta opšte namene sa PWM-om Funkcija i logika tajmera
Wi-Fi:	802.11 b/g/n/e/i
Mrežni propusni opseg:	135 MBit/s (pomoću UDP protokola)
Osetljivost prijemnika:	–98 dBm
Bluetooth:	V4.2 BR/EDR i BLE
Bluetooth funkcionalnost:	Classic i Bluetooth Low Energy (sa integrisanom antenom)
Senzori:	Holov senzor 10 kapacitivnih senzora osetljivh na dodir
Interfejsi:	3 UART-a sa kontrolom protoka pomoću 3 hardverska SPI interfejsa CAN bus 2.0 kontroler dva I2S i dva I2C interfejsa 18 analognih ulaza sa 12-bitnim analogno-digitalnim pretvaračima 2 analogna izlaza sa 10-bitnim digitalno-analognim pretvaračima Infracrveni (IR) (TX/RX) Motor PWM LED-PWM sa najviše 16 kanala Interfejs za eksternu SPI fleš memoriju sa najviše 16 MB Hardver za SD karticu
Bezbednost:	Wi-Fi: WFA, WPA/WPA2 i WAPI Secure Boot Flash Encryption Cryptographic Hardware Acceleration Elliptic Curve Cryptography (ECC) Random Number Generator (RNG)

ESP32-PICO-KIT se obično koristi za primere primene u ovoj knjizi. Alternativno se može koristiti ESP32 DEV KIT ili druga ploča. Korišćena varijanta se eksplicitno pominje u svakom slučaju. U principu, međutim, različite ploče su uglavnom kompatibilne. One

se uglavnom razlikuju po veličini i redosledu rasporeda pinova. Na slici 2.2. prikazan je PICO-KIT sa svojim funkcionalnim jedinicama i priključcima.



Slika 2.2 - ESP32 PICO-KIT ploča

Breakout ploče su najprikladnije kao eksperimentalne i razvojne ploče. Pomoću priključaka portova, elektronskih komponenti, kao što su LED-ovi, temperaturni senzori ili čak manji aktuatori, kao što su servo R/C modeli, mogu se direktno povezati. Pico Kit ploča ima sledeće karakteristike, između ostalih:

- ESP kontroler sa dva 32-bitna jezgra
- Brzi Wi-Fi i WLAN interfejs (do 150 MBit/s)
- ADC & DAC funkcionalnost
- Jedinica senzora osetljivog na dodir
- Host kontroler za SD/SDIO/MMC
- SDIO/SPI kontroler
- EMAC i PWM jedinica za kontrolu LED-ova i motora
- UART, SPI, I^2C i I^2S interfejsi
- Infracrveni daljinski upravljač
- GPIO interfejs
- Bluetooth/Bluetooth LE (4.2)
- USB-to-Serial-Chip za pristup pomoću USB interfejsa

2.1 Puštanje u rad i testiranje funkcije

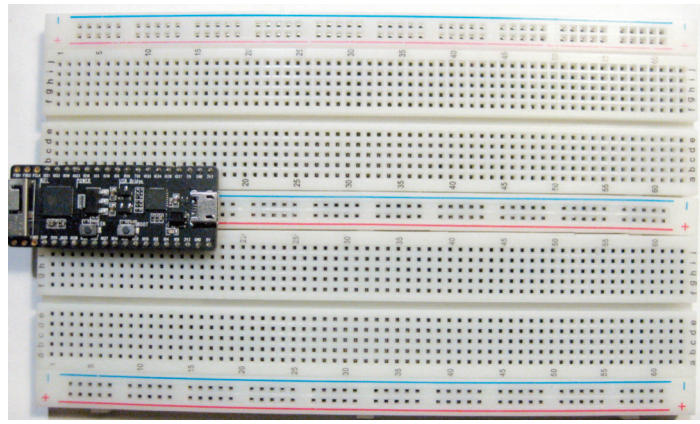
Čim ploča bude dostupna, treba je podvrgnuti početnom funkcionalnom testu. U tu svrhu, USB kabl je povezan sa računarom ili laptopom i mikro-USB priključkom na ploči. Ako postoji, USB hub bi već trebalo da bude povezan.

Na većini ploča LED će se uključiti kada se poveže na USB port sa napajanjem. Ako, suprotno očekivanjima, ovaj takozvani „power-on" LED ne zasvetli, USB vezu treba odmah prekinuti. Na ovaj način možete sprečiti mogući kratak spoj koji bi izazvao veliku štetu.

Za dalje rešavanje problema, korisni saveti su dati u odgovarajućem poglavlju na kraju knjige.

ESP ploče uvek treba da rade na priključnim pločama bez lemljenja (pogledajte sliku 2.3).

Međutim, ako radite bez prototipske ploče, uverite se da korišćena baza nije provodljiva, jer može doći do kratkog spoja između pinova. Osim same ESP ploče, to može, čak i da uništi USB port računara.



Slika 2.3 - ESP ploča priključena na prototipsku ploču

2.2 ESP32 na baterije

U većini slučajeva, ESP ploča se napaja strujom pomoću Micro-USB priključka. Pošto je kontroler već povezan sa računarom ili laptopom tokom kreiranja programa, nije potrebno dodatno napajanje.

Ako direktna razmena podataka sa računarom više nije potrebna, ploča se takođe može napajati pomoću USB napajanja. To bi trebalo da može da obezbedi struju od najmanje 1 A da bi se izbegli neželjeni padovi napona. Osim toga, postoji određena količina rezervnog napajanja za rad nekih LED-ova, displeja ili senzora.

Čak i bez USB veze, ploča može da šalje i prima podatke pomoću Wi-Fi-a i Bluetooth-a. Da bi se postigla potpuna nezavisnost od kablova za napajanje i prenos podataka, modul onda treba da se napaja samo iz (punjivih) baterija.

Neke ploče imaju konektor za litijum-jonsku (Li-Ion) bateriju za ovu svrhu (pogledajte sliku 2.4). Tamo se odgovarajuće ćelije mogu direktno povezati pomoću standardnog utikača. Unutrašnja regulacija napona onda osigurava da se kontroleri optimalno napajaju. Osim toga, povezana baterija se puni čim se ploča poveže na USB priključak pod naponom.

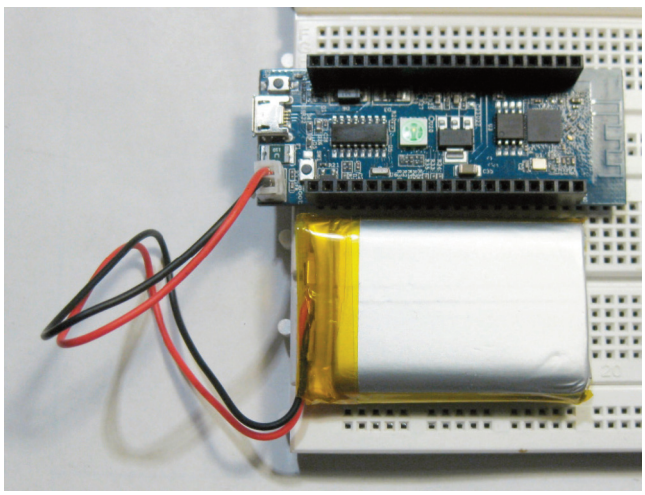
Za ovu primenu pogodne su ćelije sa kapacitetom od oko 1500 mAh ili više. Manje baterije ispod 300 mAh ne bi trebalo da se koriste, jer bi mogle da budu preopterećene integrisanim kontrolerom punjenja.

Pri uobičajenoj potrošnji energije od približno 50 mA, varijanta od 1.500 mAh može postići vreme rada od oko 30 sati, tj. nešto više od jednog dana. Kada koristite funkcije sleep u kontroleru, mogu se postići, čak i znatno duže vreme rada.

Jednoćelijske LiPo (litijum-polimerne) ili litijum-jonske baterije obezbeđuju dovoljno napajanja za ESP32. Međutim, njihov napon od 3,7 do 4,2 V, u zavisnosti od stanja napunjenosti, je previsok za ESP32. Zbog toga se reguliše pomoću unutrašnjeg modula.

Pošto je rad sa litijum-jonskim baterijama uvek povezan sa određenim stepenom opasnosti, ne treba izostaviti sledeće informacije:

- Litijum-jonske baterije osetljivo reaguju na neodgovarajuće struje ili napone punjenja. Pod određenim okolnostima postoji čak i rizik od požara ili eksplozije.
- Svaki korisnik je odgovoran za svoju konstrukciju i rad projekta.
- Izdavač i autor nisu odgovorni.



Slika 2.4 – Node ESP ploča u baterijskom režimu

Poglavlje 3 – Programsko i razvojno okruženje

Za razliku od situacije sa, recimo, Arduino sistemom, postoji nekoliko integrisanih razvojnih okruženja (IDE-a) za rad sa MicroPython-om. U principu, možete pisati programe u svim IDE-ovima i učitavati ih na kontroler. Ovo su trenutno dva najrasprostranjenija programska okruženja:

- µPyCraft
- Thonny

Oba imaju svoje specifične prednosti i nedostatke. Razlike su uglavnom u različitim procedurama za razvoj i upravljanje programskim kodom za aplikacione projekte.

Prva varijanta, koja se zove µPiCraft, nudi relativno jednostavan interfejs za razvoj MicroPython-a na ESP32 kontroleru. Radi sa jednostavnim grafičkim elementima i podseća na tekstualno orijentisane operativne sisteme. Rukovanje pojedinačnim funkcijama je lako razumeti, a korišćenje različitih menija je lako naučiti.

Thonny, s druge strane, ima potpuno grafički interfejs u Windows stilu. IDE je veoma popularan među proizvođačima, posebno zato što je dostupan u okviru Raspbian operativnog sistema na Raspberry Pi-u. Mnogi korisnici Raspberry Pi-a su stoga već dobro upoznali Thonny-a.

IDE-ovi predstavljaju najvažnije operativne sisteme, kao što su

- Windows PC
- Mac OS X
- Linux Ubuntu

i koji su besplatni na internetu.

Ako dođe do problema tokom instalacije ili korišćenja bilo kog sistema, druga verzija se može koristiti kao alternativni sistem za programiranje. Verzija koju treba izabrati zavisi naravno od ličnih sklonosti i navika korisnika.

3.1 Instaliranje uPyCraft IDE-a

Pre instaliranja uPyCraft IDE-a, najnovija verzija Python-a 3.7.X treba da bude instalirana na računaru koji koristite. Ako nije instalirana, instalacija se može obaviti prema sledećim uputstvima:

1. Preuzmite instalacionu datoteku sa Python stranice Download na adresi:
   www.python.org/downloads

2. Nakon operacije preuzimanja, datoteka koja se zove

   python-3.7.X.exe

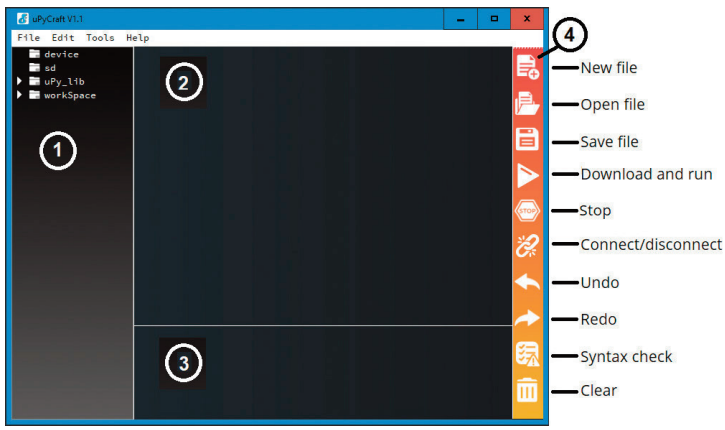
   treba da se nalazi na vašem računaru. Dvostruki klik na datoteku pokreće instalaciju.

3. Izaberite „Add Python 3.7 to PATH“ i kliknite na dugme „Install Now“.
4. Proces instalacije je završen nakon nekoliko sekundi i prikazuje se poruka „Setup was successful“. Zatim možete da zatvorite prozor.

Sada možete da preuzmete uPyCraft IDE za Windows

https://github.com/DFRobot/uPyCraft

kao datoteku koja se zove uPyCraft_V1.x.exe. Nakon što kliknete na ovu .exe datoteku, otvoriće se uPyCraft-IDE:



Slika 3.1 – uPyCraft-IDE

Nakon što se IDE instalira na računar, ESP32 firmver se može učitati na čip.

Trenutnu verziju MicroPython firmvera za ESP32 možete naći na adresi

http://micropython.org/download#esp32

Tamo skrolujete do odeljka „ESP32 modules“. Nakon što kliknete na link „Generic ESP32 module“, bićete preusmereni na stranicu za preuzimanje ESP32-BIN datoteke. To će izgledati ovako:

esp32-idf3-20191220-v1.12.bin

Sada možete da pokrenete uPyCraft-IDE. U okviru opcija

Tools -> Serial

Poglavlje 4 – Prvi koraci u programiranju

Python je već nekoliko godina jedan od najčešće korišćenih programskih jezika. Jedan od razloga za to je što je bio veoma jednostavno dizajniran i stoga ga je lako naučiti. Razvoj MicroPython-a čini programiranje sistema mikrokontrolera relativno jednostavnim i direktnim. Ovo čini programski jezik veoma pogodnim za početnike u svetu ugrađenih sistema.

Programeri MicroPython-a postavili su sebi cilj da programiranje digitalne elektronike učine što lakšim. Na ovaj način se mogu obratiti najvećem mogućem krugu korisnika. Python programi se mogu naći u oblasti hobija, kao i u obrazovnoj ili naučnoj upotrebi. Ali i profesionalni programeri sve više koriste Python. U IT industriji, brojni lideri na tržištu, kao što su Google ili Amazon, već duže vreme koriste Python za razvoj softvera.

Osim toga, besplatno dostupni moduli i biblioteke, kao što su MatPlotLib, NumPy, SciKit ili SciPy, pružaju široke mogućnosti – od analize naučnih podataka do mašinskog učenja i veštačke inteligencije.

MicroPython je razvijen kao tanka verzija Python-a 3. Pošto jezik mora da se interpretira, generalno je sporiji od kompajliranih sistema. MicroPython je dizajniran da radi što efikasnije na malim ugrađenim sistemima. Zbog toga se takođe može pokrenuti na mikrokontrolerima koji su mnogo sporiji i imaju mnogo manje memorije od tipičnih personalnih računara.

Nedostatak klasičnog Python programiranja je taj što je kontrole niskog nivoa veoma teško implementirati. Iz tog razloga, klasične Python varijante se koriste prilično sekundarno u programiranju koje se odnosi na hardver. Ovaj nedostatak je u velikoj meri eliminisan MicroPython-om. Na osnovu standarda, Micro verzija je takođe snažno zasnovana na Python-u 3 u svojoj sintaksi. Osim toga, postoje virtuelne mašine i srodne biblioteke.

Ako uporedimo dva najpopularnija programska jezika u okruženju mikrokontrolera, vidimo se da se Python češće preferira u odnosu na C/C++. Na rang listi najpopularnijih programskih jezika, Python je sve češće na prvom mestu. S druge strane, konkurent C/C++ sve više pripada nižem rangu. Razlog za ovaj razvoj je uglavnom zasnovan na sledećim prednostima Python-a:

- Python je veoma pogodan za početnike zbog svoje jednostavne jezičke strukture.
- Razni internet forumi podržavaju programera nudeći uputstva i primere kodova.
- Dostupne su velike biblioteke.

Početnici obično brzo pronađu rešenja za svoje probleme na forumima. U drugim jezicima ovaj oblik međusobne podrške nije toliko izražen.

U jeziku C se programiranje vrši pomoću kontrolnih registara, pokazivača i drugih struktura i instrukcija koje je često teško razumeti. Firmver za ciljni kontroler se mora programirati, kompajlirati i na kraju preneti na kontroler pomoću uređaja za programiranje. MicroPython integriše sve ove korake. Jednostavnim klikom miša, korisnici mogu da kontrolišu hardver niskog nivoa, kao što su LED-ovi, displeji ili motori. Dobijanje analognih vrednosti napona ili rad sa SD karticama postaje „dečja igra“ kada se koriste odgovarajuće biblioteke. Integrisano čišćenje memorije i proces dinamičke alokacije omogućavaju efikasno upravljanje memorijom u Python-u. To znači da skoro ne morate da pribegavate pokazivačima ili sličnim konstrukcijama, u koje je početnicima obično teško da proniknu.

Često zagonetni C-simboli, kao što su x++, <<, >> itd., kao i deklaracija složene promenljive, predstavljaju prepreku za početnike koju ne treba potcenjivati. Python je poznat po svojoj jednostavnosti i odličnoj čitljivosti koda.

Pošto je MicroPython razvijen kao „lakša verzija“ za aplikacije mikrokontrolera, nisu podržane sve biblioteke i funkcije standardnog Python-a. Ipak, možete lako da pređete na mikro verziju ako ste već upoznali Python. Samo nekoliko sintaksičkih struktura ili uputstava nije dostupno ili primenljivo u MicroPython-u.

Python se interpretira. To znači da originalni programski kod obrađuje direktno ciljni procesor. Kompajliranje stoga nije neophodno. Python stoga nudi mogućnost da se program koji je jednom napisan izvrši na velikom broju sistema. Za tu svrhu mora biti instaliran samo odgovarajući interpreter. Jedna od najvećih prednosti Python koda je njegova sveobuhvatna kompatibilnost. Python programi se mogu izvršavati na klasičnim računarima sa operativnim sistemima Windows, MacOS ili Linux, ali i na malim sistemima sa jednom pločom, kao što su Raspberry Pi ili uporedivi mikrosistemi. Naročito upotreba na „RPi-u“ (nemački: „Raspi“) takođe je doprinela sve većoj popularnosti Pythona.

Zahvaljujući novim moćnim kontrolerima, kao što je ESP32, sada je čak postalo moguće efikasno i praktično koristiti Python i u ovoj oblasti. Čvrsto integrisane konstrukcije obezbeđuju da se programi mogu lako razvijati i na malim i na velikim računarima. Python je stoga odlično skalabilan. Moguća enkapsulacija podataka i programskog koda u jasne, upotrebljive module, odnosno objekte, čini Python objektno-orijentisanim programskim jezikom. C++ se danas uglavnom koristi posebno u hardverski orijentisanom programiranju. Klasične Python varijante do sada nisu bile pogodne za ovu svrhu. Zahvaljujući MicroPython-u, taj jaz je sada zatvoren.

C++ uključuje klijentske aplikacije, kao i moćne serverske aplikacije, drajvere uređaja i ugrađene komponente drajvera. Oblast primene je od sistemskog softvera do programiranja aplikacija. Pošto je Python relativno nov programski jezik u odnosu na C jezik, on još uvek nije našao univerzalnu primenu u svim oblastima informacionih tehnologija. Međutim, vidi se da Python sve više uzima maha praktično u svim oblastima.

Glavni nedostatak Python-a je svakako njegova relativno mala brzina obrade. Tu kompajlirani jezici, npr. C, mogu jasno pokazati svoje prednosti. Brze kontrolne petlje ili si-

stemi u realnom vremenu, kontrole vozila i bezbednosni upiti mogu se realizovati mnogo lakše i bezbednije u C jeziku.

Međutim, pošto ove oblasti primene jedva da igraju ulogu za neprofesionalne korisnike, nedostatak brzine jedva da je značajan.

Python je takođe dobio poseban značaj u veoma aktuelnoj oblasti veštačke inteligencije (AI – artificial intelligence). Zahvaljujući opsežnim bibliotekama, kao što su NumPi, SciPi itd. i distribucijama, kao što je Anaconda, Python je postao najpopularniji programski jezik. Sva vrata su stoga otvorena za iskusnog korisnika Python-a. Od programiranja hardverskog kontrolera do AI aplikacija – uz Python nema ograničenja za intuiciju i kreativnost.

U ovom poglavlju će biti kompajlirane osnove MicroPython-a. Trebalo bi da imate dostupno funkcionalno programsko okruženje, jer su komande i uputstva uvek prikazani praktičnim primerima. Oni se zatim mogu odmah testirati direktno na ciljnom hardveru, tj. na ESP32. Ovo nije samo teorijski kurs programiranja, već stečeno znanje se može odmah primeniti u praksi.

4.1 Nikada bez: komentara

Komentari sa objašnjenjima su važni u svakom programskom jeziku. Jedino pomoću njih možete da mesecima ili godinama kasnije saznate šta radi određeni odeljak programa, a da ne morate da se udubljujete u stare detalje iznova i iznova.

Nije potrebno komentarisati svaku programsku liniju pojedinačno. Iskusni koderi bi trebalo da budu u stanju da razumeju pojedinačne instrukcije bez komentara. Samo u slučaju posebnih konstrukcija ili neobičnih ili inovativnih linija koda preporučuje se komentar u jednom redu. Međutim, za potprograme ili čitave logičke programske odeljke, kratko objašnjenje kako oni funkcionišu ne bi trebalo da izostane.

Jednostavni komentari će biti predstavljeni znakom #. Počinju znakom # i završavaju se krajem reda:

```
>>> print(“hello ESP32”) # this is a comment
hello ESP32
```

Komentari u više redova takođe mogu biti označeni dvostruko (“) trostrukim navodnikom (“”“). Isti niz znakova zatim završava komentar:

```
“””
prvi red komentara
drugi red komentara
“””
```

U praksi, to može izgledati ovako:

```
'''
This is a multi-line comment. Prints "hello world".
'''
print("hello world")
```

Poglavlje 5 - Kontroler u praktičnoj upotrebi

Sada, kada ste upoznati sa nekim važnim osnovama programiranja, vreme je za prve praktične primene. Pored izlaza na konzolu, posebno će se koristiti LED-ovi kao indikatori statusa portova. To već omogućava realizaciju nekih zanimljivih aplikacija.

Za ove primere mogu se koristiti ili integrisani LED-ovi pojedinačnih ploča ili eksterni LED-ovi. LED-ovi na ploči su obično povezani na port 02 – na primer, kao na NodeMCU ploči. Node-ESP ploča čak ima LED od tri boje povezan na portove 00, 02 i 04. Pico komplet nema integrisani LED. U tom slučaju morate da pređete na eksterne LED-ove (pogledajte slike 4.6 ili 4.7.) Ne zaboravite serijski otpornik.

5.1 Trepćući LED kao simulator alarmnog sistema

Trepćući LED je već implementiran u odeljku 4.4. Za razliku od stalno osvetljenih displeja, trepćuća svetla privlače pažnju. Treptanje stoga često skreće pažnju na izuzetak ili opasnu situaciju. Na primer, stalno uključeni LED na zamrzivaču obično ukazuje da jedinica radi ispravno. Ako LED počne da treperi, to je obično indikacija da unutrašnja temperatura više ne odgovara zadatoj vrednosti. Isto važi i za uređaje na baterije. Uključeni LED označava punu bateriju. Ako LED počne da treperi, preostala samo mala količina energije.

Za alarmne sisteme, trepćući LED označava da je uređaj aktiviran. LED u trepćućem režimu se stoga može koristiti kao simulator alarmnog sistema. Varijacije takvih uređaja se zapravo koriste u praksi. Mogu se, na primer, ugraditi u vozilo. Ne mogu se razlikovati od pravog, opremljenog alarmnog sistema i odvratiće barem neke oportunističke lopove od provale. Odgovarajući program se može naći u programskom paketu koji prati knjigu i može se preuzeti sa sajta Elektora ili Info Elektronike,

```
# Alarmsimulator.py

from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.value(1)
  sleep(.1)
  led.value(0)
  sleep(5)
```

Jedina razlika u odnosu na program koji treperi je u tome što su vrednosti u instrukciji sleep() promenjene na 0.1 ili 5. Ovo pretvara uobičajeno treptanje u kratko treptanje. Čim se novi program učita, LED nakratko treperi svakih pet sekundi i simulator alarmnog sistema je spreman za upotrebu.

5.2 Korisno u hitnim slučajevima: automatski SOS signal

Još jedna zanimljiva aplikacija je automatski SOS signal. Ovo čak može spasiti živote u vanrednim situacijama. Da bi emitovao poznati SOS signal, LED mora prvo da treperi tri kratka treptaja, zatim tri duga treptaja i na kraju ponovo tri kratka. Ova sekvenca signala treba da se stalno ponavlja.

Kod za simulator alarmnog sistema može se lako promeniti tako da LED šalje željeni signal:

```
# SOS.py

from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:

  # "S"
  led.value(1)
  sleep(.1)
  led.value(0)
  sleep(.5)
  led.value(1)
  sleep(.1)
  led.value(0)
  sleep(.5)
  led.value(1)
  sleep(.1)
  led.value(0)
  sleep(1)

  # "O"
  led.value(1)
  sleep(.4)
  led.value(0)
  sleep(.5)
  led.value(1)
  sleep(.4)
  led.value(0)
  sleep(.5)
  led.value(1)
  sleep(.4)
  led.value(0)
  sleep(1)
```

```
# "S"
led.value(1)
sleep(.1)
led.value(0)
sleep(.5)
led.value(1)
sleep(.1)
led.value(0)
sleep(.5)
led.value(1)
sleep(.1)
led.value(0)
sleep(1)

sleep(2)
```

Ovo svakako nije najelegantniji metod generisanja SOS signala. Međutim, ovaj program pokazuje da čak i bez opsežnog poznavanja programskog jezika Python, možete da realizujete svoje korisne projekte. U odeljku 6.4 prikazano je kako se takav program može mnogo efikasnije primeniti.

Ipak, program ispunjava svoju svrhu. Ako se koristi posebno svetli LED, električno kolo može biti veoma korisno u situacijama, kao što nevolje na moru ili u planinama.

Poglavlje 6 - Programske strukture

Ni jedan programski jezik ne može da funkcioniše bez kontrolnih struktura. MicroPython obezbeđuje instrukcije grananja i petlje za ovu svrhu. Primer generisanja SOS signala je već pokazao da se mnoge identične ili slične sekvence moraju često ponavljati u programiranju. Petlje nude mnogo elegantniji metod za ovu svrhu, umesto da jednostavno ponavljaju instrukcije u kodu.

Ako se moraju doneti odluke, neophodne su instrukcija grananja. Ovo omogućava programu da pravilno reaguje na različite situacije.

Za razliku od drugih programskih jezika, MicroPython ima veoma izuzetnu funkcionalnost kada su u pitanju petlje. To će biti objašnjeno u narednim odeljcima.

6.1 Uslovi i petlje

U grani, uslov se definiše u programu pomoću ključnih reči „if“ i „else“. U zavisnosti od toga da li je ovaj uslov true ili false, program će se nastaviti u različitim tačkama. Dakle, program vraća

```
Temperature = 2

if temperature < 3:
  print("Risk of frost" )
else:
  print("No danger of frost")
```

to će rezultirati izlazom

```
>>> Risk of frost
```

Ako je podešeno da je Temperature = 17, izlaz će biti

```
>>> No danger of frost
```

Rezultat posle ključne reči „if“ vraća Bulov izraz. Ovo može biti ili true ili false. Ako je izraz true, iskazi se izvršavaju direktno nakon if linije. Ovi iskazi moraju biti uvučeni da bi bilo jasno koji iskazi pripadaju if bloku. Imajte na umu da se Bulovi izrazi u if liniji moraju završavati dvotačkom.

Iskazi „else“ se izvršavaju samo ako je if upit netačan.

Petlje se koriste za ponavljanje instrukcija. Pomoću petlji, blokovi koda se mogu izvršiti nekoliko puta. Izvršenje se nastavlja sve dok se ne ispuni zadati uslov. U MicroPython-u su dostupne dve vrste petlji:

- petlje while
- petle for

Ako želite da se u konzoli ispišu brojevi od 1 do 10, možete da koristite sledeću petlju while:

```
number=1
while number<=10:
   print(number)
   number=number+1
```

Kod koji treba ponoviti je ponovo označen uvlačenjem. Izvršava se sve dok je vrednost promenljive „number“ manja ili jednaka (<=) 10. U svakoj petlji se prikazuje trenutni broj, a zatim se vrednost broja povećava za 1.

Zadatak se takođe može završiti pomoću petlje „for“:

```
number = 1
for number in range(1, 10):
      print(number)
```

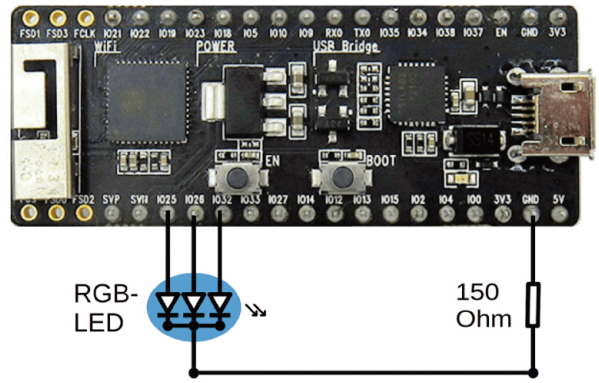
Petlja for se izvršava sve dok je vrednost u promenljivoj brojača u opsegu od 1 do 10. Funkcija range() dodeljuje odgovarajući opseg vrednosti promenljivoj. Petlja for nastavlja da se izvršava sve dok je promenljiva brojača manja od navedene konačne vrednosti. Stoga se broj 9 ispisuje kao najveća vrednost.

For petlje se stoga koriste kada blok koda treba da se ponovi sa unapred određenim brojem prolaza. Dok petlje, s druge strane, preuzimaju deo koda dok se ne ispuni određeni uslov.

Baš kao uslovni iskazi, Bulovi izrazi u petljama for i while moraju da se završavaju dvotačkom.

6.2 Navigaciona svetla i osvetljenje aerodroma

Do sada je ESP ploča kontrolisala samo jedan LED. Ali, pomoću petlji, nekoliko LED-ova sa zanimljivim obrascima može se kontrolisati bez ikakvih problema. U ovom odeljku ćemo na početku koristiti 5 LED-ova. Oni bi trebalo da se uključuju jedan za drugim. Takva svetla se mogu videti, na primer, na kraju autoputa ili na gradilištima puteva. Ovde je nekoliko sijalica spojeno tako da stvaraju utisak da se svetleća tačka kreće duž signalnog lanca. I na aerodromskim pistama takva „navigaciona svetla“ često služe kao signal za približavanje aviona. Električna šema ove aplikacije izgleda ovako:



Slika 6.3 – Električna šema za kontrolu LED-a u više boja

6.4 SOS kompaktni stil

Sada znamo programske strukture, pa SOS aplikaciju možemo programirati elegantnije i kompaktnije:

```
# SOS_compact.py
from machine import Pin
from time import sleep
led=Pin(2,Pin.OUT)
while True:
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.4)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(2)
```

Ovde postaje jasna prednost petlje. Instrukcije koje treba ponoviti treba samo jednom navesti. Petlja osigurava da se one ponavljaju u skladu sa zahtevima.

Poglavlje 7 - Generisanje analognog signala

Klasični I/O portovi mikrokontrolera mogu prihvatiti samo digitalne vrednosti napona. ESP32 kontroler radi na 3.3 V i samim tim samo dva napona mogu da budu prisutna:

LOW: 0 V
HIGH: 3.3 V

Nasuprot tome, stvarni svet nije digitalno konstruisan. Temperature se ne menjaju naglo. Tokom dana, osvetljenost u prostoriji bez veštačkog osvetljenja se stalno povećava i smanjuje. Dakle, u prirodi i svakodnevnom životu uvek postoje kontinuirane, odnosno analogne, međuvrednosti. U mnogim aplikacijama, ovo ponašanje takođe treba oponašati pomoću tehničkih sredstava. Na primer, mnogo je prijatnije da vas probudi muzika koja polako postaje sve glasnija nego iznenadni ton alarma. Svetlost koja se postepeno pojačava je prijatnija za oko od lampe koja se pali u jednom jarkom bljesku.

Dostupne su različite metodi za generisanje analognih napona. Dva najvažnija metoda su:

- modulisanje širine impulsa (PWM - pulsewidth modulation)
- digitalno-analogni pretvarač (DAC - Digital to Analogue Converter).

U ovom poglavlju je prikazano kako se oba metoda mogu koristiti u MicroPython-u. Zatim su objašnjene različite aplikacije.

7.1 Modulisanje širine impulsa

Ako LED-ovi ne samo da se uključuju i isključuju, već i preuzimaju klizne vrednosti osvetljenosti, može se koristiti metoda modulaisanja širine impulsa (PWM). Ovom metodom moguće je generisati kvazianalogni signal na zapravo čisto digitalnom izlazu. Ovo se postiže brzim prebacivanjem porta između LOW i HIGH. Kontroliše se pomoću dva parametra:

- frekvencija uključivanja/isključivanja
- radni ciklus

Radni ciklus je definisan dužinom visokog nivoa u odnosu na dužinom celog perioda signala. Maksimalni napon signala se postiže kada je port stalno uključen (radni ciklus = 100%). Ako je port kontinuirano na LOW nivou (radni ciklus = 0%), izlaz je minimalni nivo signala od 0 V. Na sledećoj slici je prikazana kriva signala koja se javlja u ovom slučaju.

Na strani hardvera, LED se ponovo povezuje na port 25 (pogledajte sliku 4.7). Da biste aktivirali port koji podržava PWM, prvo morate da uvezete vremenski modul :

```
from time import sleep
```

Onda možete da kreirate PWM objekat koji se zove „LED“:

```
LED = machine.Pin(25)
pwmLED = machine.PWM(LED)
```

Ovaj objekat zahteva dva parametra:

pin na koji je povezan

osnovnu frekvencija PWM signala

Za osnovnu frekvenciju može se koristiti vrednost između 0 i 78125. Frekvencija od 5000 Hz je savršeno adekvatna za kontrolu LED-a:

```
pwmLED.freq(5000)
```

PWM radi pomoću rezolucije od 10 bitova, pa su moguće vrednosti 2^10 = 1024. PWM promenljiva stoga može imati vrednost između 0 i 1023. Ovde 1023 odgovara radnom ciklusu od 100% (puna osvetljenost), a 0 odgovara radnom ciklusu od 0% (tamni LED).

Metoda duty() se koristi za podešavanje odnosa uključeno/isključeno. Parametar

```
timedelay = 0.005
```

kontroliše brzinu rada. Ako LED treba da polako postane svetliji, a zatim ponovo tamniji, potreban je sledeći kod:

```
# LED_fader.py

import machine
from time import sleep

LED = machine.Pin(25)

# create the PWM object
pwmLED = machine.PWM(LED)

timedelay = 0.005

# set frequency and duty cycle between 0 (all off) and 1023 (all on)
# 512 -> 50% duty cycle
```

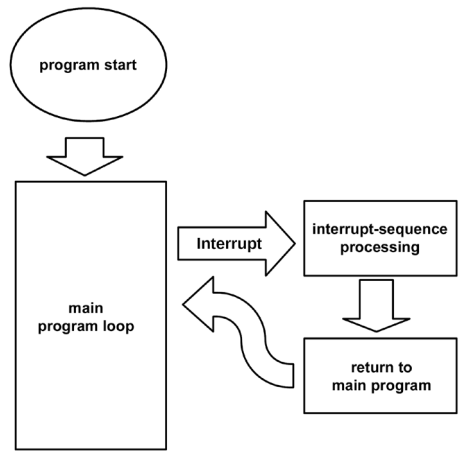
Poglavlje 8 – Prekidi i tajmeri

U okruženju mikrokontrolera, privremeni prekidi pokrenutog programa se nazivaju prekidi. Koriste se često za izvršavanje vremenski kritičnog procesa. Tipične primene su alarmne poruke ili sigurnosno isključivanje komponenti i delova mašine. Ako se, na primer, generator elektronski nadgleda, na signal povećanja temperature sa senzora mora se odmah reagovati. Vremenska kašnjenja zbog trenutno velikog opterećenja procesora mogu u ovom slučaju imati katastrofalne posledice i stoga nisu tolerantna.

Prekidi su takođe potrebni za upotrebu tajmera. Ovde, hardverske jedinice u kontroleru postavljaju određene vremenske intervale. Ako su istekli, pokreće se prekid. Ovo omogućava postizanje veoma preciznog vremena, nezavisno od opterećenja glavnog jezgra procesora. Tipične primene su satovi i kontrole vremena svih vrsta. Preciznost tajmera na kraju zavisi samo od tačnosti kristala procesora. Pošto se preciznost u opsegu od 1/1000 može postići bez problema, tajmeri se koriste u mnogim praktičnim aplikacijama.

8.1 Tražena smetnja: prekidi

Prekidi se koriste kada je potrebno brzo reagovati na nepredvidive ili neuobičajene događaje. Ako je promena statusa otkrivena na pinu prekida, posebna hardverska jedinica u kontroleru pokreće događaj prekida. Stoga nije neophodno stalno pratiti odgovarajući pin putem rutine upita. Nakon što je prekid pokrenut, može se pozvati unapred definisana funkcija koja reaguje na događaj na odgovarajući način.



Slika 8.1 - Prekidi

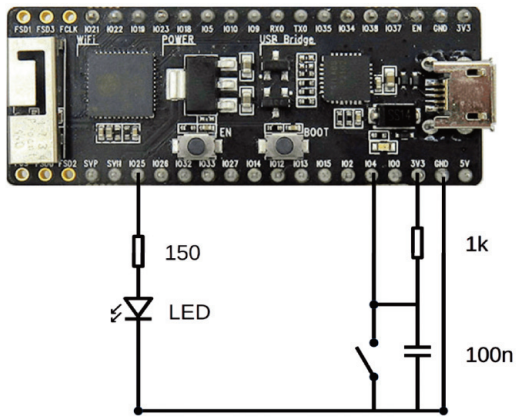
Dakle, rutine prekida rade kvazi-paralelno sa glavnim programom. Ali, ovo nije stvarna paralelna obrada. Procesor umesto toga obrađuje glavni program i rutinu prekida jednu za drugim.

Stoga, rutine prekida treba da budu što kraće, odnosno njihova obrada treba da traje što je moguće kraće. Unutar rutine prekida ne bi trebalo da se vrše proračuni koji oduzimaju mnogo vremena, niti da se obrađuju opsežne programske petlje itd.

8.2 Automatsko noćno svetlo

U ovom odeljku je objašnjeno kako da konfigurišete prekide u MicroPython-u. Na primer, taster se očitava. Ovaj taster bi trebalo da uključi LED na određeno vreme. Nakon toga, LED će se automatski isključiti. Na primer, ovo kolo može da se primenjuje kao automatsko osvetljenje kupatila ili stepeništa. Ali, takva kola se takođe koriste za unutrašnje osvetljenje u vozilima. Sa svetlo belim LED-om, ESP ploča se takođe može koristiti kao automatska noćna lampa. Na slici 8.2 prikazana je električna šema.

Više informacija o hardverskim tajmerima za ESP32 potražite u sledećem poglavlju.



Slika 8.2 – Električna šema automatskog noćnog svetla

Za korišćenje prekida, modul machine se takođe prvo ponovo uvozi, a zatim se koristi za pristup funkcijama hardvera:

```
from machine import Pin
```

Osim toga, potrebne su metode sleep, jer je kasnije u programu potreban period čekanja:

```
from time import sleep
```

Promenljiva se koristi za ispitivanje da li je taster pritisnut. Ova promenljiva je globalno definisana, jer se menja u funkciji obrade prekida. Dovoljno je ako može da preuzme Bulove vrednosti True ili False. Inicijalizacija se vrši pomoću vrednosti False:

```
key_pressed = False
```

U slučaju prekida treba pokrenuti sledeću rutinu:

```
def interrupt_handler(pin):
  global key_pressed
  key_pressed = True
```

Ova funkcija se poziva svaki put kada se pritisne taster. Funkcija interrupt_handle ima ulazni parametar (pin) u kojem se prosleđuje objekat klase Pin ako dođe do prekida. Parametar utvrđuje na kom pinu je prekinut pokrenut. Ako se koristi samo jedan pin prekida, ova informacija nije potrebna. Međutim, ako više prekida može pokrenuti istu funkciju obrade prekida, može biti od značaja koji GPIO je pokrenuo prekid.

U primeru interrupt_handler menja promenljivu key_pressed u True. Pošto vreme obrade funkcije prekida treba da bude što je moguće kraće, funkcije kao što su sleep() ili print() treba izbegavati, ako je moguće. Instrukcije koje treba izvršiti kada dođe do prekida treba da budu u glavnom programu.

Da bi se promenljiva koristila i unutar funkcije i u celom kodu, mora biti deklarisana kao „globalna". U suprotnom, uključivanje i isključivanje LED-a ne bi funkcionisalo, jer bi promena promenljive imala efekat samo unutar funkcije, ali ne i u glavnom programu.

Kada napišemo,

```
led = Pin(25, Pin.OUT)
button = Pin(4, Pin.IN)
```

pinovi za LED i dugme su definisani. Instrukcija

```
button.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt)
```

izaziva pokretanje prekida svaki put kada se pritisne taster. Parametar „trigger" može imati sledeće vrednosti:

- IRQ_RISNING: okidač na rastućoj ivici
- IRQ_FALLING: okidač na padajućoj ivici

Ako se prekid aktivira, kontroler prelazi na obradu prekida i promenljiva key_pressed je postavljena na vrednost True.

Nakon povratka u glavni program, prikazuje se poruka „Interrupt detected!". Osim toga, prijavljuje se na kom pinu je prekinut. LED se zatim uključuje na određeni period, a zatim se ponovo isključivanje. Konačno, key_pressed je ponovo postavljeno na vrednost False i kontroler je dostupan za sledeći prekid. Kompletan program izgleda ovako:

Poglavlje 9 - Korišćenje senzora

Ocenjivanje senzora i elektronskih mernih transformatora je centralni zadatak tehnologije mikrokontrolera. Ovo poglavlje se stoga detaljnije bavimo upotrebom MicroPythona u tehnologiji merenja i senzora. ESP32 ima osamnaest analognih ulaza, pa je idealno opremljen za merne aplikacije.

Osim evaluacije senzorskih signala, senzorska tehnologija uključuje i izlaz i obradu izmerenih vrednosti i obradu signala svih vrsta. Posebno, ono što se naziva „kondicioniranje signala“ igra izuzetnu ulogu. Metodi kao što su

- konverzija signala
- linearizacija
- pojačanje signala
- filtriranje

su klasični zadaci akvizicije podataka zasnovani na mikrokontroleru. Bez ovih metoda mnogi zadaci i procedure u tehnici, primenjenoj fizici ili medicini ne bi bili izvodljivi.

Pomoću MicroPython-a, izmerene vrednosti mogu biti prikazane direktno u konzoli. Thonny čak ima i poseban ploter koji omogućava grafički prikaz vrednosti. Alternativno, takođe je moguće prikazati vrednosti na displeju, tako da ESP32 može da radi i u samostalnom režimu. Drugi metod je detaljnije opisan u sledećem poglavlju.

9.1 Akvizicija mernih i senzorskih vrednosti

Čak i pomoću digitalnih I/O pinova, već je moguće očitati izmerene vrednosti. Dakle, binarni nivoi napona se mogu očitati 0 V ili 3.3 V „izmerene vrednosti“ sa svakog GPIO ulaza ESP kontrolera. Ovo je često dovoljno za jednostavne zadatke merenja ili praćenja.

Međutim, prave analogne, tj. kontinuirane, vrednosti se ne mogu meriti pomoću digitalnih portova. Kao što je već napomenuto u odeljku o digitalno-analognoj konverziji, priroda je pretežno analogna. Zbog toga su razvijene posebne metode i procedure za akviziciju signala u analognom domenu. U ESP32 su za ovu svrhu dostupni takozvani analogno/digitalni pretvarači (ADC-ovi). Oni su, da tako kažemo, pandan DAC-ovima i omogućavaju simultanu akviziciju nekoliko analognih kanala.

ADC-ovi su postali sastavni deo savremene akvizicije signala. U svim oblastima tehnologije, oni prate odgovarajuću funkciju komponenti i sistema. U audio i komunikacionim uređajima, kao što su pametni telefoni ili tableti, oni konvertuju signale mikrofona u visokokvalitetne digitalne tokove podataka. Temperatura, intenzitet svetlosti ili pritisak su samo neki primeri mnogih parametara koje ADC-ovi očitavaju i procenjuju u vozilima ili zgradama.

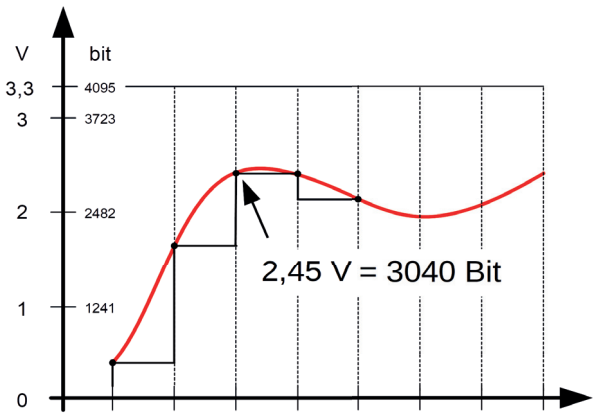
Konvertori se mogu okarakterisati pomoću dva važna parametra:

- rezolucija u bitovima
- brzina konverzije

Preciznost kojom se analogni signal može digitalizovati određena je prvom vrednošću. Vreme konverzije je odlučujuće za brzinu rada. Ono određuje maksimalnu frekvenciju kojom se signal može kvantovati. Ovo vreme konverzije zavisi uglavnom od korišćenog metoda konverzije.

Na slici 9.1 prikazana je digitalizacija analogne krive napona sa ADC-om. Pretvarač skenira analognu krivu napona u određenim vremenskim momentima i konvertuje utvrđenu vrednost u digitalnu veličinu. ADC u ESP32 čipu ima rezoluciju od 12 bitova. Kada se normalizuje na 3,3 V, napon od 2,45 V, na primer, proizvodi:

$$Q = 2.45\ V / 3.3\ V * 4095 = 3040 \text{ bitova}$$



Slika 9.1 – Digitalizacija analognih vrednosti

Široko polje primene digitalne merne tehnologije dovelo je do razvoja različitih ADC procesa. Neke oblasti primene zahtevaju najveću moguću preciznost merenja. Ostali procesi se oslanjaju na najbržu moguću konverziju. Pošto se obe funkcije ne mogu konvertovati u isto vreme, razvijeno je nekoliko principa ADC konverzije koji se jasno razlikuju po preciznosti i brzini. To je sumirano u sledećoj tabeli:

Merna procedura	Primena	Svojstva
Paralelni pretvarač	brzi digitalni osciloskopi, upravljački inženjering, istraživanje	veoma brz, složen, velika potrošnja energije
Sukcesivna aproksimacija (SAR - Successive approximation)	procedura za većinu internih AD pretvarača mikrokontrolera, standardna metoda	brz, visoka preciznost, složen
Single-slope ili dual-slope	multimetar	niska cena, dobra linearnost, spor
Delta-Sigma	precizna merenja, audio tehnologija	jeftin, spor

Kao dobar kompromis, sukcesivna aproksimacija se uglavnom koristi u tehnologiji mikrokontrolera. Dva pretvarača u ESP32 su takođe zasnovana na ovoj metodi. Analogni multiplekseri obezbeđuju da ESP čip može da snimi 18 analognih kanala brzinom od nekoliko hiljada merenja u sekundi. Ostvarljiva brzina konverzije zavisi, između ostalog, od izabrane stvarne rezolucije ADC-a. Dodatne informacije možete pronaći u sledećim odeljcima.

9.2 Precizno snimanje napona: voltmetar „uradi sam“

U prvom primeru aplikacije, biće prikazana evaluacija analogno-digitalnih pretvarača i u okviru Micro-Python-a. Za tu svrhu, analogni napon koji generiše potenciometar se snima i prikazuje u konzoli. Pinovi 25–34 treba da se koriste kao univerzalni ADC ulazi. Ostalim ADC ulazima su dodeljene važne dvostruke funkcije, tako da ih treba koristiti samo ako su drugi kanali već zauzeti.

U sledećem primeru se koristi analogni ulazni pin 34. U sledećem programu prikazan je najlakši način na koji možete da očitate ADC ESP32:

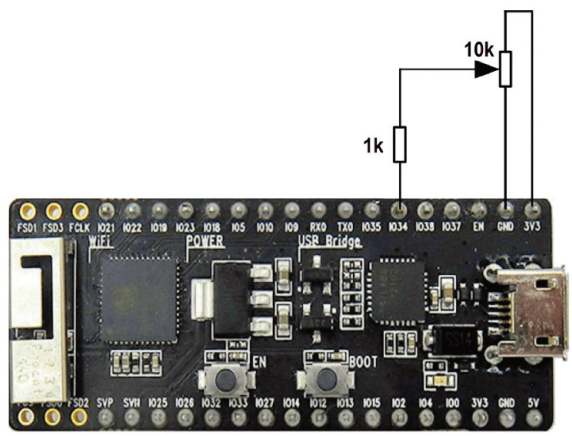
```
# ADC_atten_11db.py

from machine import ADC, Pin
from time import sleep

adc = ADC(Pin(34))     # create ADC object on ADC pin
adc.atten(ADC.ATTN_11DB)

while True:
  print(adc.read())    # read value, 0-1024
  sleep(1)
```

Na strani hardvera, samo potenciometar od 10 kilooma mora biti povezan na pin 34:



Slika 9.2 - Merenje napona pomoću potenciometra

Poglavlje 10 – Tehnologija prikaza i ekrani male veličine

Prikazivanje tekstova i vrednosti u konzoli je savršeno adekvatno za mnoge testne svrhe i jednostavne aplikacije. Međutim, ova varijanta ima veliki nedostatak, jer je uvek potreban računar ili barem laptop. Ako je sistem u stalnom radu danima ili nedeljama, rad ovih uređaja će trošiti nepotrebnu količinu energije. Štaviše, nije uvek poželjno ili praktično koristiti zaseban računar za svaku aplikaciju mikrokontrolera.

Kada je reč samo o prikazu vremena ili izmerenih podataka sa klimatskih senzora, mnogo je bolje da povežete svoj mali displej na kontroler. Najčešće korišćeni tip je SSD1306. Ovaj displej ima rezoluciju od 128 x 64 piksela i veličinu od samo 0.96 inča (nešto manje od 2.5 cm).

Kao standard, MicroPython dolazi sa drajverom za ovu verziju displeja. Ovaj drajver je već učitan prilikom otpremanja sistema datoteka na ESP32. Drajver omogućava prikaz tekstualnih i numeričkih podataka, ali i jednostavne grafike. Displej SSD1306 je opremljen internom RAM memorijom i svojim oscilatorom. Zbog toga može da radi bez ikakvih spoljnih komponenti. Osim toga, ima kontrolu osvetljenosti sa 256 podesivih nivoa.

Ovo su glavne funkcije SSD1306:

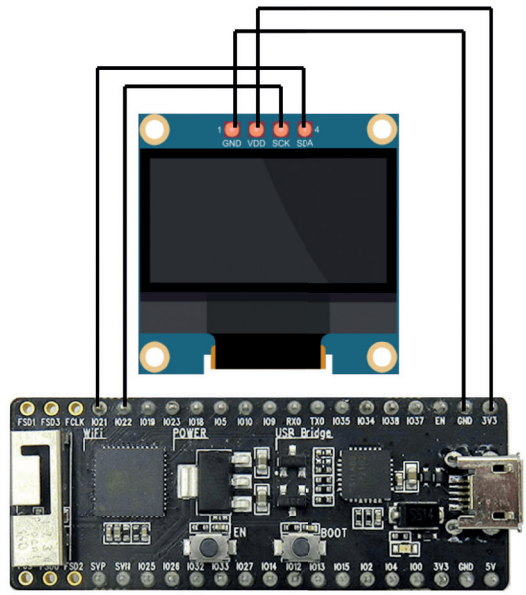
- Rezolucija: 128 x 64 matrica
- Napajanje: 1.65 V do 3.3 V
- Opseg radne temperature: –40 °C do +85 °C
- Integrisani 128 x 64-bitni SRAM bafer prikaza
- Funkcija neprekidnog pomeranja u horizontalnom i vertikalnom pravcu
- Oscilator na čipu

OLED displejima nije potrebno pozadinsko osvetljenje, jer svaki pojedinačni piksel može da emituje svetlost. Zbog tog razloga, ova varijanta je i dalje laka za čitanje, čak i pod nepovoljnim uslovima osvetljenja. Osim toga, kontrast je znatno veći u poređenju sa ekranima sa tečnim kristalima (LCD-ovima). Pošto piksel troši energiju samo kada se uključi, OLED displeji su veoma energetski efikasni.

Najjednostavnije verzije SSD1306 ploča imaju samo četiri pina. Ovo je dovoljno za kontrolu displeja pomoću I^2C magistrale. Druge verzije imaju pinove za resetovanje ili dodatni SPI interfejs. Za većinu aplikacija, međutim, dovoljan je jednostavan dizajn. U sledećoj tabeli prikazane su sve potrebne veze.

OLED-Pin	ESP32
VDD	3V3
GND	GND
SCK	GPIO 22
SDA	GPIO 21

„Električna šema“ izgleda ovako:



Slika 10.1 - SSD1306 displej na ESP32

Sledeći skript generiše tekstualnu poruku i jednostavan grafički element u obliku okvira na displeju:

```
# SSD1306_DEMO.py

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))

# I2C Pin assignment
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width, oled_height, i2c)
lin_hight = 9
col_width = 8
```

Poglavlje 11 – LED matrice i veliki displeji

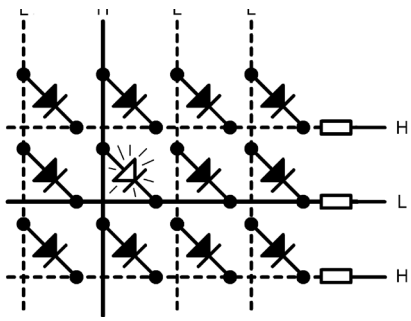
Nakon što smo u prethodnom poglavlju detaljno opisali upotrebu OLED displeja, u sledećim odeljcima ćemo se fokusirati na kontrolu još jednog veoma popularnog ekrana, koristeći MicroPython. Pomoću takozvanih matričnih displeja, takođe, možete prikazati brojeve, slova, simbole i grafiku.

Glavna razlika u odnosu na OLED displej je dostižna osvetljenost i veličina ekrana. LED matrice mogu biti veličine do nekoliko metara. Veoma visok nivo osvetljenosti se takođe može postići pomoću modernih LED-ova. Ova varijanta displeja se stoga često koristi za displeje velike površine ili reklamne table na prometnim trgovima, železničkim stanicama ili aerodromima i u javnim autobusima i vozovima. U svetskim berzanskim halama aktuelni finansijski podaci se pojavljuju na takozvanim „tikerima uživo".

Tačkaste matrice su posebno popularne u Aziji, jer se mogu koristiti i za predstavljanje znakova Dalekog istoka, kao što je prikazano na sledećoj slici:

Slika 11.1 - LED tačkaste matrice su veoma popularne u Aziji

Široko se koriste LED matrice sa 5 x 7 = 35 LED-ova. Ali, dostupni su i razni drugi dizajni. Često možete da vidite i dizajni sa 8 x 8 LED-ova.



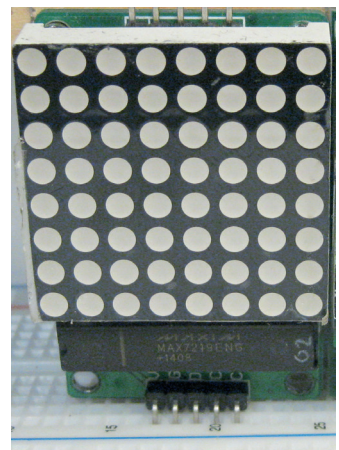
Slika 11.2 - Princip matrične kontrole

Direktno upravljanje tačkastim matricama 8 x 8 zahtevalo bi 65 linija, pošto bi 64 pinske veze i zajednička katoda morali da budu povezani. Sa matricom, potrebno je znatno manje veza. Ovde je 8 LED-ova povezano u jednu kolonu i jedan red. Dakle, potrebno je samo 8 + 8 = 16 veza. Na prethodnoj slici pokazan je princip za matricu 3 x 4. Umesto 13 veza potrebno je samo 7 veza za individualnu kontrolu LED-ova.

Na primer, za kontrolu drugog LED-a u drugoj liniji, sve linije veze osim druge moraju biti postavljene na HIGH. Druga veza, međutim, mora biti na GND-potencijalu. U slučaju kolona, HIGH potencijal može biti prisutan samo u drugoj koloni.

Direktna kontrola tačkastih matrica zahteva veliki deo resursa kontrolera. Ako vrednosti senzora takođe treba da se snimaju ili aktuatori treba da se kontrolišu, čak i moćni ESP32 kontroler brzo dostiže svoje granice. Kontrola većih displeja sa sto ili više LED-ova takođe brzo postaje problem, s jedne strane zbog potrebne računarske snage, a s druge zbog ograničenog broja dostupnih pinova na kontroleru.

Tu mogu pomoći isplativi drajveri za ekran, npr. MAX7219. Ovi uređaji imaju kompatibilan interfejs sa SPI-om i tako mogu da kontrolišu ekrane sa do 8 x 8 = 64 matričnih elemenata pomoću samo tri digitalna pina. Drajveri su dostupni kao kompletni moduli. Na sledećoj slici je prikazan primer:



Slika 11.3 - LED matrični modul

Povezivanje modula drajvera sa ESP32 je veoma jednostavno. Samo tri SPI pina moraju biti povezana na kontrolnu ploču:

- MAX7219_data (DIN) Port D02
- MAX7219_load (CS) Port D05
- MAX7219_clock (CLK)→ Port D04
- MAX7219_GND ESP31 GND

Zbog relativno velike potrošnje energije, preporučuje se da se napajanje obezbedi eksterno. Ovaj metod omogućava podešavanje skoro bilo koje veličine ekrana. Kontroler se više jedva učitava, pošto se pomoću SPI magistrale moraju slati samo pojedinačne komande. Osim toga, dovoljno pinova ostaje slobodno za rad spoljnih senzora ili drugih perifernih uređaja.

Sada ništa ne stoji na putu realizaciji displeja velikog formata, na primer, u reklamne svrhe ili na sportskim događajima.

Link za odgovarajuću Python biblioteku za kontrolu Maxim IC-a može se naći u paketu za preuzimanje. Nakon što je datoteka drajvera „Max7219.py“ preuzeta u kontroler, dostupne su sledeće instrukcije:

```
spi = SPI(1, baudrate=10000000, polarity=1, phase=0,
sck=Pin(CLK), mosi=
Pin(DIN))
ss = Pin(CS, Pin.OUT)
```

za gornju dodelu pinova, postavite

- CLK = 4
- DIN = 2
- CS = 5

Zatim možete da kreirate „objekat“ prikaza:

```
display = max7219.Matrix8x8(spi, ss, MN)
```

pri čemu se za MN mora uneti broj upotrebljenih matričnih elemenata. Nakon toga, tekstovi i grafike se mogu dizajnirati pomoću sledećih komandi:

display.pixel(x,y,1)	postavlja piksel na koordinatu x,y
display.pixel(x,y,0)	briše piksel na koordinati x,y
display.hline(x,y,l,1)	horizontalna linija sa x,y - dužina l
display.vline(x,y,l,1)	vertikalna linija sa x,y - dužina l
display.line(a,b,c,d,1)	linija od a,b do c,d display.rect(a,b,c,d,1) pravougaonik sa uglovima a, b, c, d
display.text('text',x,y,1)	tekst na poziciji x,y display.scroll(x,0) skrolovanje po x pikselima
display.show()	ažuriranje prikaza

Ovo vam omogućava da kreirate reklame koje su efikasne i lako čitljive, čak i sa udaljenosti od nekoliko metara.

11.1 LED matrica u akciji

Sledeći kod pokazuje primer ekrana sa šest matričnih elemenata 8 x 8:

```
# LED_matrix_test.py

import max7219
from machine import Pin, SPI

spi = SPI(1, baudrate=10000000, polarity=1, pha
mosi=Pin(2))
ss = Pin(5, Pin.OUT)
```

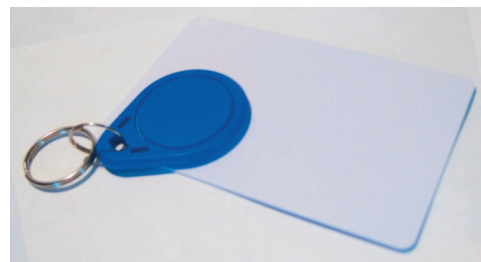
Poglavlje 13 - RFID i bežični prenos podataka

RFID je skraćenica za radiofrekventnu identifikaciju. Uređaji za čitanje sa ovom funkcijom mogu bežično da čitaju podatke sa pasivnih transpondera koji se nazivaju RFID tagovi, preko vazduha, odnosno putem radija. Transponder (predajnik / davalac odgovora) može da bude označen posebnim kodom koji ga čini jasno prepoznatljivim. MicroPython obezbeđuje specijalnu biblioteku koja omogućava da se ovi kodovi čitaju bežično (pogledajte paket za preuzimanje).

Ovo omogućava da se ESP32 koristi za implementaciju sistema zaključavanja ili sličnih projekata u kojima osoba mora da se identifikuje sa predajnikom.

Transponderi su dostupni u velikom broju oblika i varijanti. Na sledećim slikama prikazani su prvo modul prijemnika, a zatim dva različita RFID TAG-a. Jedan od dva taga je u obliku priveska za ključeve, a drugi se sastoji od plastične kartice u formatu kreditne kartice.

Slika 13.1 - RFID modul



Slika 13.2 - RFID kartica i privezak za ključeve

Transponderi mogu apsorbovati energiju iz elektromagnetnog polja koje generiše predajnik. To znači da predajnik integrisan u transponder može da radi bez svog izvora energije. Klasični RFID TAG stoga ne zahteva ugrađene baterije ili akumulatore i trajno je

spreman za upotrebu. Transponder šalje nazad svoj integrisani kod na fiksnoj frekvenciji. Ovaj kod se zatim prima i dalje obrađuje.

RFID tagovi se takođe mogu napisati pomoću novih kodova. Međutim, ovo je veoma dugotrajno, zbog čega se postupak ovde ne nastavlja. Većina aplikacija se takođe može izvršiti pomoću kodova koji su već prisutni u tagovima.

13.1 Čitanje kartica i čipova

Pomoću biblioteke mfrc522.py može se pročitati broj „Jedinstvenog identifikatora“ (UID-Unique Identifier), koji sadrži pojedinačne identifikacione podatke RFID TAG-a. Python program za ovo je sledeći:

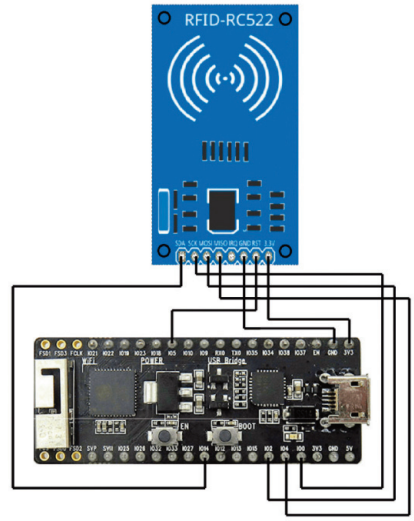
```
# RFID_test.py
import mfrc522

# RFID RX pinning
sck = 0
mosi= 2
miso= 4
rst = 5
cs  =14    # SDA on RFID-RC522 boards

def do_read():
    rdr = mfrc522.MFRC522(sck, mosi, miso, rst, cs)
    print("Place card before reader")
    try:
        while True:
            (stat, tag_type) = rdr.request(rdr.REQIDL)
            if stat == rdr.OK:
                (stat, raw_uid) = rdr.anticoll()
                if stat == rdr.OK:
                    print("Card detected")
                    print("type: 0x%02x" % tag_type)
                    print(raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3])
                    print("")
    except KeyboardInterrupt:
        print("Bye")

while True:
    do_read()
```

Ovde su prvo definisani pinovi za povezivanje RFID modula na ESP32. Na sledećoj slici je prikazan dijagram povezivanja:



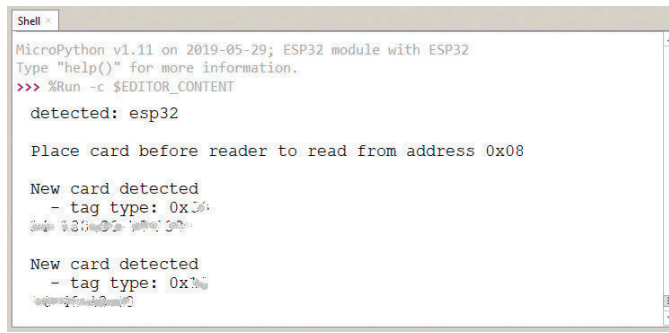
Slika 13.3 - RFID modul na ESP32

U sledećoj tabeli su navedene potrebne veze:

RFID modul	ESP32
SCK	I/O 00
MOSI	I/O 02
MISO	I/O 04
RST	I/O 05
CS (SDA)	I/O 14
RST	nije povezano

Rutina koja se zove do_read() onda očitava TAG-ove.

Nakon pokretanja programa, kartica ili drugi TAG se mora staviti u domet RFID modula. Izlaz u sledećem obliku bi se tada trebalo pojaviti u konzoli:



Slika 13.4 - ID brojevi različitih RFID tagova