Princip setup - loop

Na začetku je pripravljena cpp (C++) datoteka, kjer imamo funkciji oz. metodi setup() in loop(),

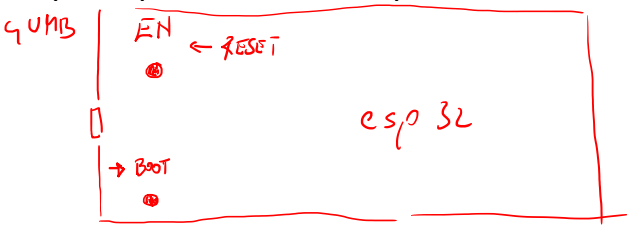
```
#include <Arduino.h>

void setup() {
  // put your setup code here, to run once:
  // tu zapišemo kodo za nastavitve, koda bo izvedena le enkrat:
}

void loop() {
  // put your main code here, to run repeatedly:
  // tu zapišemo kodo, ki se ponavljajoče izvaja:
}
```

#include <Arduino.h>
Vključimo knjižnico oz. “header” datoteko Arduino.h, ki je glavna datoteka za Arduino SDK (“Software Development Kit”) oz. programski razvojni kit.

1

setup()

- V funkciji setup() zapišemo kodo, ki se izvede le enkrat, na začetku izvedbe programa. Npr. tu lahko določimo ali bo določena nožica delovala kot vhod ali izhod, inicializiramo spremenljivke itd.
- Funkcija setup se požene ob priklopu modula na napetost ali ob pritisku na gumb reset.



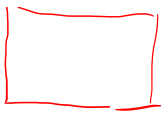
2

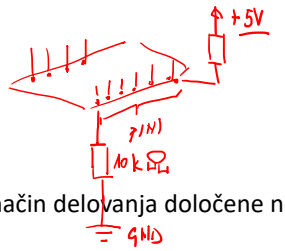
loop()

- Za tem, ko se izvede funkcija setup(), ki inicializira in nastavi začetne vrednosti se prične izvajati funkcija loop().
- Izvajanje funkcije loop() ali slovensko zanke, je ponavljajoče.
- Karkoli zapišemo znotraj telesa funkcije loop() se bo neprestano izvajalo in nam omogočilo upravljanje z modulom, t.j. strojno opremo.
- V tovrstni zanki npr. preverjamo ali je gumb na modulu pritisnjen ali ne ter v primeru, da je pritisnjen prižgemo LED diodo sicer jo ugasnemo.

3

pinMode()





- S pomočjo funkcije pinMode() določimo kakšen naj bo način delovanja določene nožice, t.j. ali naj določena nožica deluje kot vhod ali izhod.
- Pri določenih mikrokontrolerjih je možno potegniti potencial nožice navzdol ali navgor (INPUT_PULLUP ali INPUT_PULLDOWN).

Sintaksa

- pinMode(pin, mode)

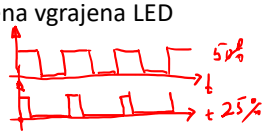
NAPETOSTI: • 5V • 3,3V

Parametra:

- pin: nožica na modulu za katero želimo nastaviti način delovanja
- mode: INPUT, OUTPUT, INPUT_PULLUP, INPUT_PULLDOWN (odvisno od strojne opreme)

- **Primer** (nožica 2 bo delovala kot izhod - na nožici 2 je na modulu esp32 priključena vgrajena LED dioda):
- `pinMode (2, OUTPUT);`

PWM PULSE WIDTH MODULATION



4

digitalWrite()

- S pomočjo funkcije digitalWrite() zapišemo visoko (HIGH) ali nizko (LOW) vrednost na digitalno nožico.
- Če smo nožico konfigurirali kot izhod (OUTPUT) s funkcijo pinMode() bo napetost nožice pri HIGH nastavljena na eno od naslednjih vrednosti: 5V (ali 3.3V, odvisno od naprave) oz., pri
- LOW bo napetost nastavljena na 0V (ozemljitev - ground GND)
- Če smo nožico konfigurirali kot vhodno (INPUT) bo digitalWrite() omogočil (HIGH) ali onemogočil (LOW) interni poteg navzgor (internal pullup) na vhodni nožici. Običajno na nožici, ki je konfigurirana kot vhodna izvedemo poteg navzgor INPUT_PULLUP ali poteg navzdol INPUT_PULLDOWN
- V primeru, da ne nastavimo pinMode() kot izhod (OUTPUT) in na nožico priključimo LED ob tem pa izvedemo digitalWrite(HIGH) lahko LED le brli. Če ne nastavimo pinMode() bo digitalWrite() omogočil notranji upor za poteg navzgor (pull-up), ki bo deloval kot velik upornik, ki bo omejil tok.

Sintaksa

- digitalWrite(pin, value)

Parametri

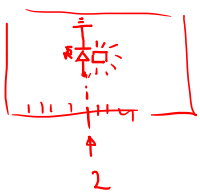
- pin: the Arduino pin number.
- value: HIGH ali LOW.

5

digitalWrite() (nad.)

Primer:

- digitalWrite(2, HIGH);
- digitalWrite(2, LOW);



6

delay()

- Funkcija zakasni program za specificiran čas v milisekundah, ki ga podamo kot parameter, npr. delay(1000) bo zakasnil izvedbo programa za 1000ms oz. 1s.

Sintaksa

- delay(ms)

Parameter

- ms: število milisekund trajanja pavze. Dovoljeni podatkovni tip: “unsigned long”

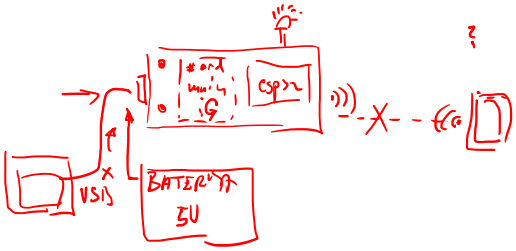
Primer

- delay(250);

7

#include <WiFi.h>

- Knjižnica omogoči priklop modula na internet. Modul lahko deluje kot strežnik, ki sprejema vhodne povezave ali kot klient, ki vzpostavlja izhodne povezave. Knjižnica podpira zaščito (Personal encryption).



8

Serial.begin()

- S pomočjo funkcije Serial.begin() določimo hitrost prenosa prek serijske komunikacije v bitih na sekkundo (baudih). Če želimo v Serijskem monitrju spremljati sporočila moramo nastaviti pravilno vrednost hitrosti prenosa. Opcijski drugi argument je namenjen konfiguraciji podatkovnega, paritetnega in stop bita. Prednastavljena vrednost je 8 podatkovnih bitov, brez paritete in en stop bit.

Sintaksa

- Serial.begin(speed)
- Serial.begin(speed, config)

esp32 >>>> Arduino UNO

Parametri

- Serial: objekt serial port
- speed: v bitih na sekundo (baud) - tip long
- config: nastavimo podatkovni, partetni in stop bit

Primer:

- Serial.begin(115200);

9600 ↳ baud

9

Serial.println()

- Funkcija izpiše podatke na serijska vrata v človeku berljivem ASCII besedilu, ki mu sledi konec vrstice - "carriage return" (ASCII 13 ali "\r") in znak za novo vrstico (ASCII 10 ali '\n').

Sintaksa

- Serial.println(val)
- Serial.println(val, format)

Parametri

- Serial: objekt serial port
- val: vrednost, ki jo želimo izpisati - katerikoli podatkovni tip
- format: določimo številsko osnovo za integralne podatkovne tipe (char , signed char , unsigned char -8 bit. short int , signed short int, unsigned short int -16 bit) ali število decimalnih mest (za tipe s plavajočo vejico)

Vrnjena vrednost

- size_t: println() vrne število bytov, ki so bili zapisani. Branje je opcijsko.

Primer

- Serial.println("Povezovanje");

10

WiFi.begin()

- Inicializacija omrežnih nastavitev WiFi knjižnice in določitev trenutnega statusa.

Sintaksa

- WiFi.begin();
- WiFi.begin(ssid);
- WiFi.begin(ssid, pass);
- WiFi.begin(ssid, keyIndex, key);

Parametri

- ssid: the SSID (Service Set Identifier) je ime WiFi omrežja, na katerega se želimo priklopiti.
- keyIndex: WEP encryption omrežje ima lahko 4 različne ključe. Tu določimo, kateri ključ bomo uporabili.
- key: heksadecimalni niz, ki ga uporabimo kot varnostno kodo pri WEP enkriptiranih omrežjih.
- pass: WPA enkriptirana omrežja uporabljajo gelso v obliki niza z namenom zagotovitve varnosti.

Vrnjena vrednost

- WL_CONNECTED ko smo priključeni na omrežje
- WL_IDLE_STATUS ko nismo priključeni na omrežje, modul pa je priključen na napajanje

Primer

- WiFi.begin(ssid, password);

11

WiFi.status()

Opis

- Vrne status povezave

Sintaksa

- WiFi.status();

Parametri

- brez

Funkcija vrne

- WL_CONNECTED: assigned when connected to a WiFi network;
- WL_NO_SHIELD: assigned when no WiFi shield is present;
- WL_IDLE_STATUS: it is a temporary status assigned when WiFi.begin() is called and remains active until the number of attempts expires (resulting in WL_CONNECT_FAILED) or a connection is established (resulting in WL_CONNECTED);
- WL_NO_SSID_AVAIL: assigned when no SSID are available;
- WL_SCAN_COMPLETED: assigned when the scan networks is completed;
- WL_CONNECT_FAILED: assigned when the connection fails for all the attempts;
- WL_CONNECTION_LOST: assigned when the connection is lost;
- WL_DISCONNECTED: assigned when disconnected from a network;

12