VIDI-X

Doc. dr. sc. Ana Sović Kržić
Zagreb, 2021.

Sadržaj

1. Instalacija

Instalacija

1. Preuzmite Arduino IDE, verziju 1.XX: https://www.arduino.cc/en/software
2. Pokrenite Arduino IDE
3. U izborniku Datoteka (File) → Preporučene vrijednosti (Preferences…) → dodajte URL: https://dl.espressif.com/dl/package_esp32_index.json
4. Dodajte ESP32: izbornik Alati → Pločica → Boards Manager → upišite ESP32 → kliknite na njega → Install
5. Preuzimite „ODROID-GO library“ sa stranice: https://github.com/hardkernel/ODROID-GO klikom na Code → Download ZIP
6. U Arduino IDE kliknite na Skica → Include Library → Add .zip library → ODROID-GO library

Provjera uspješnosti

1. Povežite VIDI-X na računalo i kliknite Datoteka → Primjeri → ODROID-GO → Applications → FlappyBird
2. Izbornik Alati → Pločica → ESP32 Arduino → ESP32 Wrover Module
3. Izbornik Alati → Port → port na koji je spojen VIDI-X
4. Prenesi ili Upload 

Mogući problemi i rješenja

1. ukoliko piše „in sketchbook“ pored naziva pločice: izbornik Alati → Pločica → ESP32 Arduino (in sketchbook) → instalacija nije provedena uspješno → ponoviti instalaciju ESP32
2. ukoliko ne čita biblioteku (library): provjeriti put do biblioteke, ne smije biti na One Drive → promjeniti lokaciju biblioteke

Prenošenje programa na VIDI X

1. Povežite VIDI-X na računalo koristeći USB
2. Izbornik Alati → Pločica → ESP32 Arduino → ESP32 Wrover Module
3. Izbornik Alati → Port → port na koji je spojen VIDI-X
4. Prenesi ili Upload 

Literatura

Detaljnije: https://vidi-x.org/radionice/vidi-project-x-91-arduino-ide/

Druge instalacije: https://vidi-x.org/vidi-x-razvojno-okruzenje/

2. Kako radi semafor

VIDI-X ima 3 LED-ice koje su kontrolnog karaktera kako bismo vizualno mogli provjeriti postoji li aktivnost na pinovima GPIO01 (crvena), GPIO02 (plava) i GPIO03 (bijela). Crvena i bijela LED induciraju serijsku komunikaciju i njih nije moguće programirati. GPIO02 (plava) LED je statusna i možemo ju programirati. Oznaka GPIO predstavlja ulaz / izlaz opće namjene (engl. general purpose input/output).

ZADATAK 1. Programirajte VIDI-X prema uputama. Testirajte. Što se dogodilo?

```
int LEDPin1 = 2;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  digitalWrite(LEDPin1, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  delay(500);
  digitalWrite(LEDPin1, LOW);
  delay(500);
}
```

RJEŠENJE. Ovaj program pali i gasi plavu LED na pola sekunde.

Svaki program se sastoji od sljedećih dijelova:

1. **definicija** konstanti i varijabli (npr. naredba `int LEDPin1 = 2;`)
2. **inicijalizacija** koja će se izvršiti samo jednom prilikom prvog pokretanja programa (sve naredbe koje se nalaze unutar vitičastih zagrada `void setup(){}`)
3. **glavni program** koji se neprestano izvršava (sve naredbe koje se nalaze unutar vitičastih zagrada `void loop(){}`)

Svaka naredba završava s točka-zarezom. VIDI-X razlikuje velika i mala slova u pojedinim naredbama. Bitno je napisati ispravno, inače će vratiti poruku o greški prilikom prebacivanja programa.

`LEDPin1` je naziv varijable u kojoj je pohranjen broj pina na koji je spojena LED. Kako je plava LED spojena na GPIO02, koristimo pin 2. Broj 2 je cijeli broj, pa je ova varijabla tipa integer. Naredba za definiciju pina na koji je spojena plava LED glasi: `int LEDPin1 = 2;`

Na početku programa potrebno je najaviti jesu li korišteni pinovi ulazni (`INPUT`) ili izlazni (`OUTPUT`) pomoću naredbe `pinMode`. LED je izlazni pin, pa to najavljujemo naredbom: `pinMode(LEDPin1, OUTPUT);`

Za paljenje i gašenje LED koristi se naredba `digitalWrite`. Naredbi je potrebno reći na kojem pinu je spojena LED i da li ju želimo upaliti (`HIGH`) ili ugasiti (`LOW`). Naredba za paljenje LED glasi: `digitalWrite(LEDPin1, HIGH);`

Naredba za gašenje LED glasi: `digitalWrite(LEDPin1, LOW);`

Naredba `delay` zaustavlja pozivanje sljedeće naredbe sve dok se prethodna naredba ne izvrši do kraja. Naredbi je potrebno dati informaciju koliko milisekundi se zaustavlja nastavak programa. Broj 500 predstavlja 500 ms, odnosno 0.5 sekundi. Ako želimo upaliti ili ugasiti LED na pola sekunde koristimo naredbu: `delay(500);`

Sve što je unutar `void loop()` se ponavlja beskonačno dugo, što znači da nakon paljenja LED na pola sekunde i onda gašenja LED na pola sekunde, obje radnje se ponavljaju.

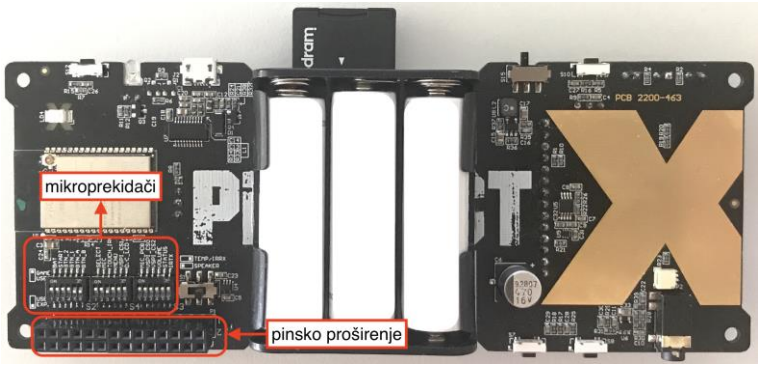
ZADATAK 2. Promijenite program tako da plava LED dioda svijetli svaku 1 sekundu.

RJEŠENJE. Kako bi palili i gasili plavu LED na 1 sekundu potrebno je promijeniti broj unutar naredbe `delay` na vrijednost 1000.

```
int LEDPin1 = 2;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  digitalWrite(LEDPin1, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
  delay(1000);
}
```

ZADATAK 3. Spojite dodatnu crvenu LED diodu na VIDI-X i programirajte ju da bude ugašena i upaljena po 1 sekundu.

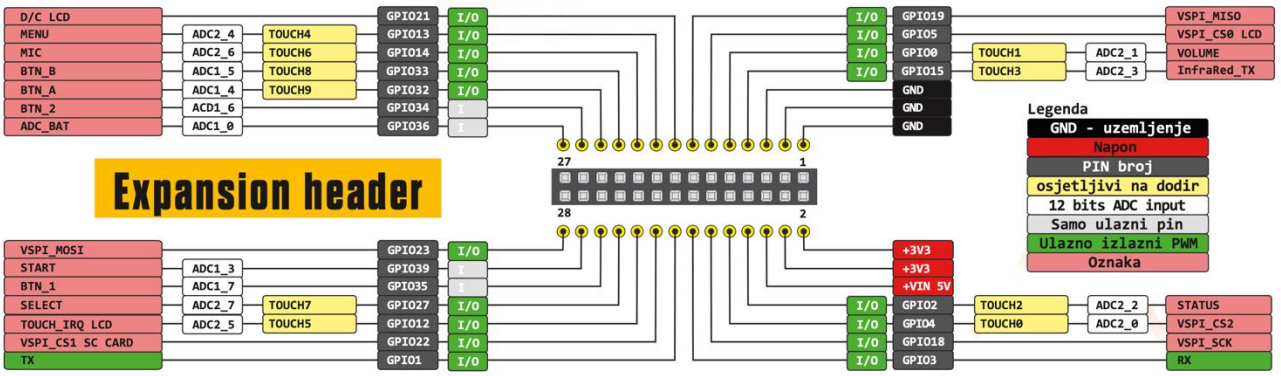
RJEŠENJE. Osim već ugrađene plave LED, na Vidi-X je moguće spojiti i dodatne LED koristeći pinove na pinskom proširenju (Expansion header). Kako bi bilo moguće koristiti ove pinove, potrebno je prebaciti određene mikroprekidače smještene iznad pinskog proširenja, sa stražnje strane pločice.



Pojedini pinovi imaju već unaprijed definirana svojstva:

- pinovi 1, 3 i 5 su uzemljenje (GND),
- pinovi 2 i 4 su napajanje od 3.3V,
- pin 6 je napajanje od 5V,
- pinovi 7, 8, 9, 10, 11, 15, 17, 18, 19, 20, 21, 22 i 23 se mogu koristiti kao ulazno izlazni pinovi,
- pinovi 24, 25, 26 i 27 su samo ulazni pinovi,
- pinovi 12, 13 i 28 se koriste pri radu s TFT ekranom,
- pinovi 14 i 16 su konstantno spojeni na GPIO3 (RX) i GPIO1 (TX) i njih nije moguće drugačije koristiti te za njih nema mikroprekidača.

Kada je pojedini mikroprekidač u gornjem položaju (Game use) moguće je koristiti senzore i aktuatore koji su ugrađeni na VIDI-X (npr. tipkala). Kada je pojedini mikroprekidač u donjem položaju (Use exp) moguće je koristiti pinsko proširenje te na njega spojiti dodatne senzore i aktuatore. Svaki mikroprekidač obilježava jedan točno određeni ulazno-izlazni ili ulazni pin. Na primjer mikroprekidač s oznakom BTN_A (mikroprekidač je na panelu S2, redni broj 5) kada je u gornjem položaju omogućuje korištenje tipke BTN_A. U donjem položaju ova tipka se više neće moći programirati i koristiti, ali ćemo moći spojiti dodatne elektroničke komponente na utor 23 pinskog proširenja. Pri programiranju tipke A, njezina oznaka je GPIO32 te se u programu kao oznaka pina koristi 32. Kada je mikroprekidač u donjem položaju ista oznaka GPIO32 će se koristiti pri programiranju spojene dodatne elektroničke komponente.
Mikroprekidač TOUCH_IRQ (panel S4, redni broj 3) mora biti u položaju Use exp. kako bi VIDI-X bilo moguće programirati i kako bi se novi kod mogao prebaciti na VIDI-X.

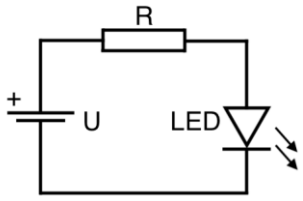


LED je kratica od Light Emitting Diode. LED emitira svjetlo kada kroz nju prolazi električna struja. Svaka LED ima dulju i kraću nožicu. Dulja nožica je pozitivna i naziva se anoda, dok je kraća nožica negativna i naziva se katoda. Struja teče od anode prema katodi te je bitno kako se LED spaja u strujni krug: katoda mora ići na GND (zemlju). Ukoliko se LED krivo okrene, neće svijetliti.

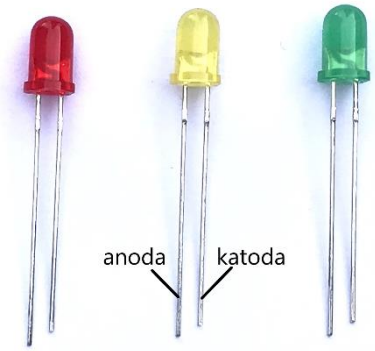
Boja LED ovisi o poluvodičkom materijalu od kojeg je napravljena (ne radi se o obojanoj plastici). S obzirom da se radi o različitim materijalima, različite boje LED će imati različite nominalne radne napone. Okvirne vrijednosti su u tablici. Nominalna radna struja LED je najčešće 20 mA.

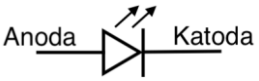
Crvena	1.7 V
Žuta	1.9-2.1 V
Zelena	1.9-2.2 V
Plava	2.7-3.2 V
Bijela	2.7-3.4 V

LED se nikada ne smije samostalno spajati u strujni krug, već uvijek mora imati odgovarajući otpornik. Koji otpornik treba upotrijebiti ovisi o vrsti i boji LED te naponu izvora, a računa se pomoću Ohmovog zakona.
VIDI-X radi na 3.3V i to će biti napon izvora našeg kruga u koji ćemo spojiti LED. Koristit ćemo crvenu LED koja ima radni napon 1.7V i radnu struju 20 mA = 0.02 A. Otpornik spajamo u seriju s LED.
Napon na otporniku je: $U_R = U_{VIDIX} - U_{LED} = 3.3 - 1.7 = 1.6\,V$.
Vrijednost otpora iznosi: $R = \frac{U_R}{I_R} = \frac{1.6V}{0.02A} = 80\ \Omega$.
Kako $80\ \Omega$ nije standardna vrijednost otpornika, uzima se najsličnija, malo veća vrijednost, npr. 100Ω ili 220Ω.

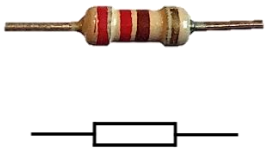


Crvenu LED ćemo spojiti na GPIO32. Mikroprekidač TOUCH_IRQ (panel S4, redni broj 3) i BTN_A (panel S2, redni broj 5) moraju biti u položaju Use exp. Žicu prema otporniku i LED spajamo na pinsko proširenje utor 23. Na taj način će LED biti spojena na GPIO32, te će se u programu kao oznaka pina koristi 32.

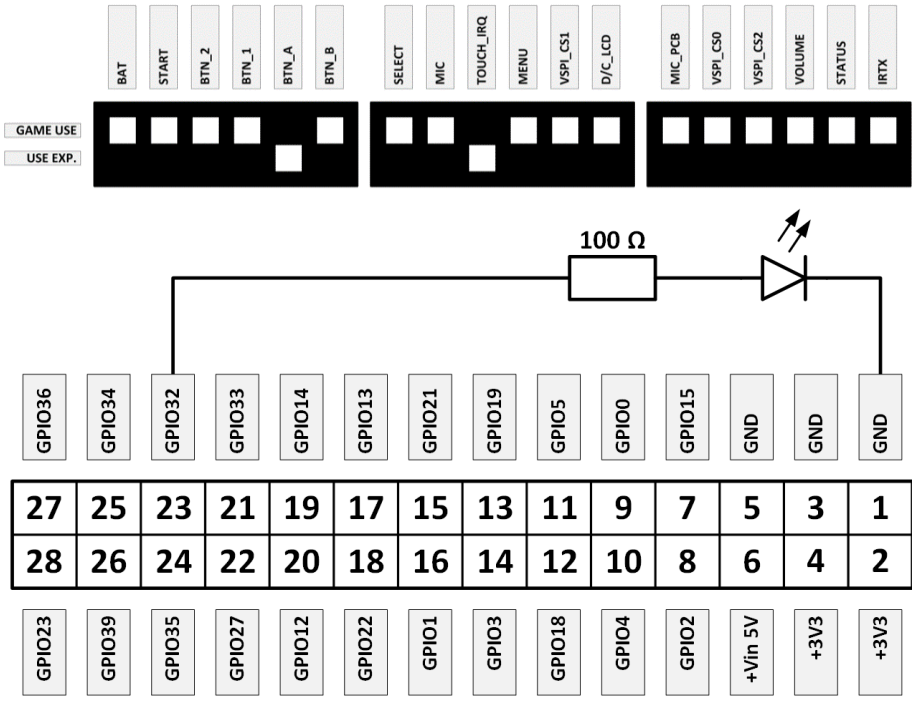




LED dioda



Otpornik 220 Ω
(boje: crvena, crvena, smeđa)
ili otpornik 100 Ω
(boje: smeđa, crna, smeđa)

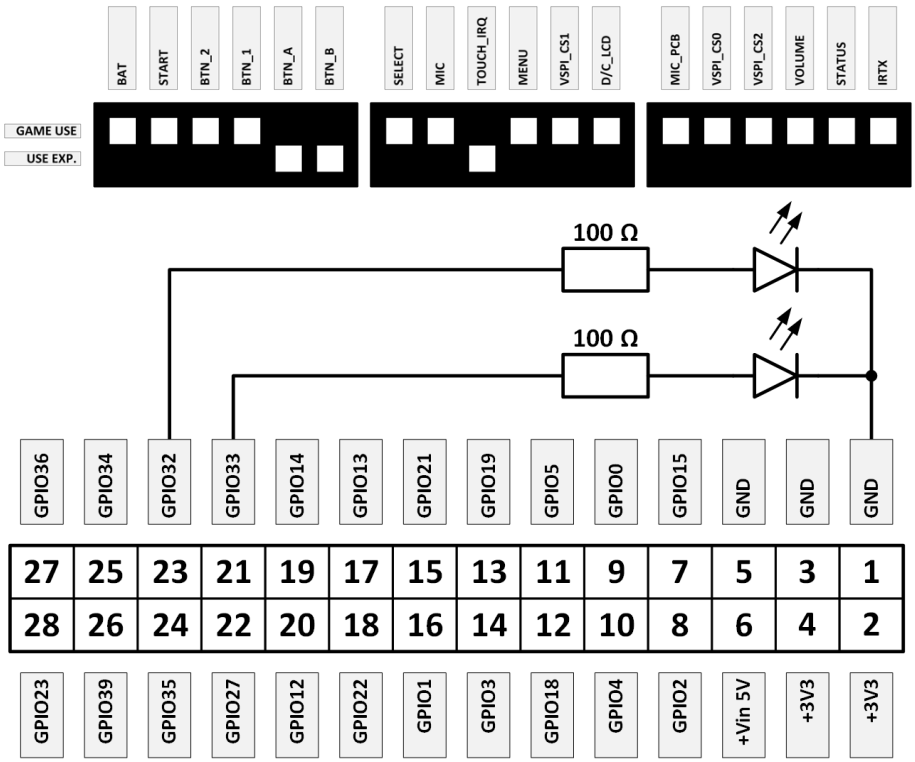


Program je identičan onom u prošlom zadatku, osim što je sada korišten drugi izlazni pin (32 umjesto 2).

```
int LEDPin1 = 32;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  digitalWrite(LEDPin1, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
  delay(1000);
}
```

ZADATAK 4. Spojite sada jednu žutu LED diodu, na pin GPIO33. Nemojte zaboraviti spojiti i pripadajući otpornik te prebaciti odgovarajući mikroprekidač. Prilikom spajanja elektroničkih elemenata, isključite VIDI-X. Napravite program tako da obje diode blinkaju istovremeno.

RJEŠENJE. Pri spajanju žute LED, moramo prebaciti mikroprekidač BTN_B u položaj Use exp. LED i pripadajući otpornik od 100Ω spajamo na 21. utor u pinskom proširenju, što je programski pin GPIO33.




```
int LEDPin1 = 32;
int LEDPin2 = 33;


void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  delay(1000);
}
```

ZADATAK 5. Promijenite program kako bi crvena i žuta LED dioda blinkale NAIZMJENIČNO.

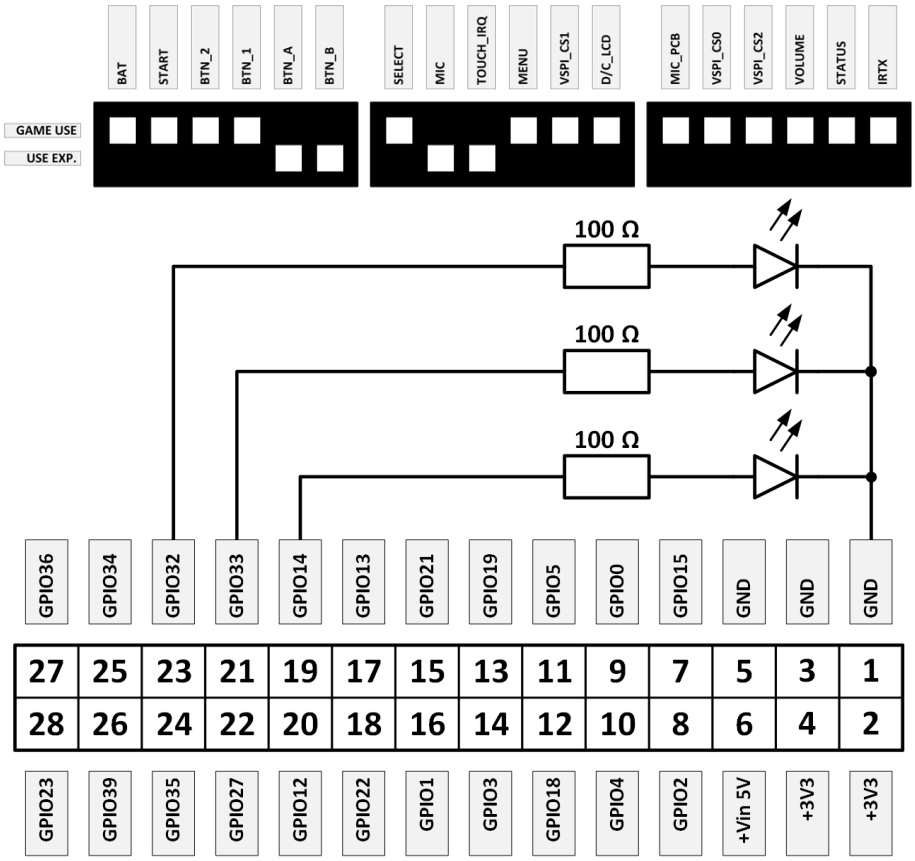
RJEŠENJE. Jedina razlika je što je sada kada je jedna LED upaljena, druga je ugašena i obratno.

```
int LEDPin1 = 32;
int LEDPin2 = 33;


void setup(){
  pinMode(LEDPin1, OUTPUT);
```

```
  pinMode(LEDPin2, OUTPUT);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
}
void loop(){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, LOW);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, HIGH);
  delay(1000);
}
```

ZADATAK 6. Dodajte jednu zelenu LED diodu na pin broj GPIO14. U seriju spojite jedan otpornik. Dodajte naredbe kako bi i zelena LED dioda blinkala istovremeno kad i crvena i žuta dioda.

RJEŠENJE. Pri spajanju zelene LED, moramo prebaciti mikroprekidač MIC u položaj Use exp. LED i pripadajući otpornik od 100Ω spajamo na 19. utor u pinskom proširenju, što je programski pin GPIO14.




```
int LEDPin1 = 32;
int LEDPin2 = 33;
int LEDPin3 = 14;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
```

```
  digitalWrite(LEDPin3, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
  delay(1000);
}
```

ZADATAK 7. Promijenite program tako da diode svijetle naizmjenično: svaka po 300 ms.

RJEŠENJE. Program glasi:

```
int LEDPin1 = 32;
int LEDPin2 = 33;
int LEDPin3 = 14;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
  delay(300);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, LOW);
  delay(300);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, HIGH);
  delay(300);
}
```

ZADATAK 8. Promijenite program tako da prvo svijetli crvena dioda 3 sekunde, zatim žuta i crvena zajedno 1 sekundu, pa samo zelena 3 sekunde. Za kraj svijetli samo žuta 1 sekundu. **Čestitamo – napravili ste semafor!**

RJEŠENJE. Program glasi:

```
int LEDPin1 = 32;
int LEDPin2 = 33;
int LEDPin3 = 14;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
}

void loop(){
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, LOW);
  delay(3000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, LOW);
  delay(1000);
  digitalWrite(LEDPin1, LOW);
```

```
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, HIGH);
  delay(3000);
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, LOW);
  delay(1000);
}
```

ZADATAK 9. Na početku neka sve LED diode svijetle. Redom isključujte jednu po jednu diodu, dok ostale ostaju svijetliti. Dobili ste trčeću tamu!

RJEŠENJE. Program glasi:

```
int LEDPin1 = 32;
int LEDPin2 = 33;
int LEDPin3 = 14;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
}

void loop(){
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, LOW);
  delay(1000);
}
```

ZADATAK 10. Napravite „trčeću tamu“ koristeći 5 crvenih dioda.

RJEŠENJE. Za ovaj zadatak potrebno je spojiti još 2 LED. Možemo ih spojiti na GPIO13 i GPIO21 (17. i 15. utor u pinskom proširenju). Također, moramo prebaciti mikroprekidače D/C LCD i MENU u položaj Use exp.

```
int LEDPin1 = 32;
int LEDPin2 = 33;
int LEDPin3 = 14;
int LEDPin2 = 13;
int LEDPin3 = 21;

void setup(){
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
  pinMode(LEDPin3, OUTPUT);
  pinMode(LEDPin4, OUTPUT);
  pinMode(LEDPin5, OUTPUT);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
  digitalWrite(LEDPin4, HIGH);
  digitalWrite(LEDPin5, HIGH);
}

void loop(){
```

```
  digitalWrite(LEDPin1, LOW);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
  digitalWrite(LEDPin4, HIGH);
  digitalWrite(LEDPin5, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, LOW);
  digitalWrite(LEDPin3, HIGH);
  digitalWrite(LEDPin4, HIGH);
  digitalWrite(LEDPin5, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, LOW);
  digitalWrite(LEDPin4, HIGH);
  digitalWrite(LEDPin5, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
  digitalWrite(LEDPin4, LOW);
  digitalWrite(LEDPin5, HIGH);
  delay(1000);
  digitalWrite(LEDPin1, HIGH);
  digitalWrite(LEDPin2, HIGH);
  digitalWrite(LEDPin3, HIGH);
  digitalWrite(LEDPin4, HIGH);
  digitalWrite(LEDPin5, LOW);
  delay(1000);
}
```

ZADATAK 11. Animirajte 5 crvenih dioda tako da su početno sve isključene. Neka se prvo upale vanjske dvije na 600 ms. Nakon što se one ugase, upale se dvije diode pored njih, na 400 ms. Srednja dioda neka svijetli sama i to 500 ms.

ZADATAK 12. Napravite cvijet koji svijetli. Rasporedite 5 crvenih dioda u krug i jednu žutu diodu u sredinu. Neka latice cvijeta naizmjenično svijetle, svaka po 400 ms. Sredina cvijeta neka svijetli bez prestanka.

ZADATAK 13. Napravite cijelo križanje: jedan semafor za aute (crveno, žuto, zeleno), te jedan semafor za pješake (crveno, zeleno). Programirajte VIDI-X tako da diode svijetle kao pravo križanje.

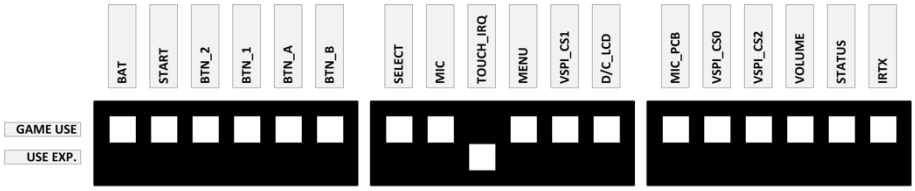
PROJEKT. Osmislite kako LED upotrijebiti za ukrašavanje Božićnog drvca.

LITERATURA.

https://electronicsclub.info/leds.htm#calculate
https://www.espressif.com/sites/default/files/documentation/esp32-wrover-b_datasheet_en.pdf

3. Morseov kod

VIDI-X ima 6 velikih crvenih tipki sa svoje prednje strane te 4 male tipke s gornje i donje strane. Tipke LEFT i RIGHT su spojene na GPIO34, UP i DOWN na GPIO35, tipka BTN_A je spojena na GPIO32, a BTN_B na GPIO33. Kada koristimo ove tipke, njihovi pripadajući mikroprekidači trebaju biti u položaju Game Use: BTN_A, BTN_B, BTN_1 (UP i DOWN) i BTN_2 (LEFT i RIGHT).
Tipka Select koristi pin GPIO27 i pripadajući mikroprekidač SELECT.
Tipka Vol koristi pin GPIO0 i pripadajući mikroprekidač VOLUME.
Tipka Menu koristi pin GPIO13 i pripadajući mikroprekidač MENU.
Tipka Start koristi pin GPIO39 i pripadajući mikroprekidač START.



ZADATAK 1. Prepišite program. Što program radi? Koje tipkalo se koristi u programu? Je li tipkalo ulazni ili izlazni element? Kojom naredbom je to definirano u programu?

```
int PinLed = 2;
int PinTipkalo = 32;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == LOW) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

RJEŠENJE. Pritiskom na tipku BTN_A, ugrađena plava LED svijetli. Kada se tipka otpusti, LED se gasi.
Koje tipkalo se koristi u programu? U programu se koristi tipka BTN_A, spojena na GPIO32.
Je li tipkalo ulazni ili izlazni element? Tipka je ulazni element, čeka od korisnika da ju pritisne i time nešto naredi računalu.
Kojom naredbom je to definirano u programu? `pinMode(PinTipkalo, INPUT_PULLUP);`

Za razliku od LED, tipka je ulazni element, te se pri njenoj definiciji mora to navesti pomoću `INPUT_PULLUP`.
Stanje tipka se provjerava pomoću naredbe `digitalRead`. Ovoj naredbi je potrebno reći koji pin treba provjeravati. Vrijednost koju očita s tog pina upisuje u varijablu `StanjeTipkala`:

```
StanjeTipkala = digitalRead(PinTipkalo);
```

`digitalRead` kao rezultat daje logičku vrijednost: kada je tipka pritisnuta `StanjeTipkala` iznosi `LOW`, a kada tipka nije pritisnuta iznosi `HIGH`.

Trenutno stanje varijable provjeravamo pomoću naredbe `if - else`. Uvjet koji provjeravamo mora biti zapisan u zagradi. Ako provjeravamo jednakost koriste se dva simbola jednako: ==. Jedan simbol = se koristi prilikom pridruživanja vrijednosti varijabli.

ZADATAK 2. Promijenite program tako da LED dioda svijetli kada je tipkalo otpušteno.

RJEŠENJE. Jedina razlika je što sada provjeravamo je li `StanjeTipkala == HIGH:`

```
int PinLed = 2;
int PinTipkalo = 32;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == HIGH) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

ZADATAK 3. Ponovite program iz prvog zadatka (neka plavi LED svijetli kada je tipkalo pritisnuto) koristeći tipke BTN_B, Select, Vol, Menu, Start. Svaku tipku testirajte odvojeno.

RJEŠENJE. Jedina razlika u programu je u broju pina tipkala (drugi redak programa). Za BTN_B pin je 33, za tipku Select je 27, Vol 0, Menu 13 i Start 39.

```
int PinLed = 2;
int PinTipkalo = 33; // ovdje treba promijeniti broj pina
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == LOW) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

ZADATAK 4. Ponovite program iz prvog zadatka (neka plavi LED svijetli kada je tipkalo pritisnuto) koristeći tipke LEFT i RIGHT.

RJEŠENJE. Tipke LEFT i RIGHT su spojene na isti pin GPIO34. Kako bismo otkrili koja od ove dvije tipke je pritisnuta, umjesto `digitalRead` treba koristiti `analogRead`. Ovoj naredbi kao ulazni podatak treba dati redni broj pina kojeg očitavamo. Rezultat koji naredba vraća je direktno očitanje senzora i može iznositi između 0 i 4095 (što je 2^{12} različitih vrijednosti). Kada je pritisnuta tipka LEFT, `analogRead` će vratiti 4095, a kada je pritisnuta tipka RIGHT `analogRead` će vratiti broj između 1930 i 1940.
Pri provjeri je li tipka pritisnuta i koja je tipka pritisnuta, imamo 3 stanja:

precizno očitanje	**izbor tipke**	**očitanje uz prag tolerancije**
`StanjeTipkala == 4095`	pritisnuta je tipka LEFT	`StanjeTipkala > 4000`
`StanjeTipkala > 1930 i StanjeTipkala < 1940`	pritisnuta je tipka RIGHT	`StanjeTipkala > 1800 i StanjeTipkala < 2000`
`StanjeTipkala == 0`	niti jedna tipka nije pritisnuta	`StanjeTipkala < 1000`

Kako bismo spriječili moguće oscilacije pri očitanju, u programu možemo koristiti toleranciju i provjeravati je li pojedina očitana vrijednost unutar nešto šireg područja, kao što je danu u trećem stupcu tablice.

Napomena: ukoliko program ne reagira ispravno na tipu RIGHT, napravite provjeru očitanja koristeći serijski monitor. Kako ga upotrijebiti pogledajte u poglavlju Temperatura.

Program za provjeru je li pritisnuta tipka LEFT glasi:

```
int PinLed = 2;
int PinTipkalo = 34;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = analogRead(PinTipkalo);
  if (StanjeTipkala > 4000) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

Program za provjeru je li pritisnuta tipka RIGHT glasi:

```
int PinLed = 2;
int PinTipkalo = 34;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = analogRead(PinTipkalo);
  if (StanjeTipkala > 1800 and StanjeTipkala < 2000) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

ZADATAK 5. Ponovite program iz prošlog zadatka (neka plavi LED svijetli kada je tipkalo pritisnuto) koristeći tipke UP i DOWN.

RJEŠENJE. Tipke UP i DOWN su također spojene na isti pin, ali GPIO35. Kada je pritisnuta tipka UP, `analogRead` će vratiti 4095, a kada je pritisnuta tipka DOWN `analogRead` će vratiti broj između 1930 i 1950. Pri provjeri je li tipka pritisnuta i koja je tipka pritisnuta, imamo 3 stanja:

precizno očitanje	**izbor tipke**	**očitanje uz prag tolerancije**
`StanjeTipkala == 4095`	pritisnuta je tipka UP	`StanjeTipkala > 4000`
`StanjeTipkala > 1930 i` `StanjeTipkala < 1940`	pritisnuta je tipka DOWN	`StanjeTipkala > 1800 i` `StanjeTipkala < 2000`
`StanjeTipkala == 0`	niti jedna tipka nije pritisnuta	`StanjeTipkala < 1000`

Program za provjeru je li pritisnuta tipka UP glasi:

```
int PinLed = 2;
int PinTipkalo = 35;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = analogRead(PinTipkalo);
  if (StanjeTipkala > 4000) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

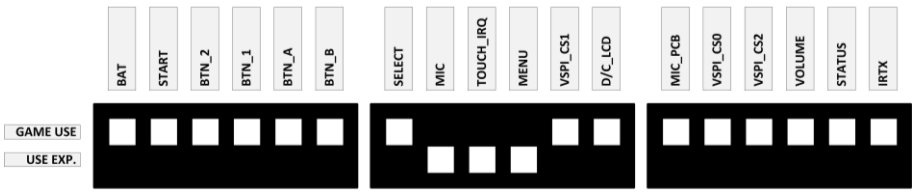
Program za provjeru je li pritisnuta tipka DOWN glasi:

```
int PinLed = 2;
int PinTipkalo = 35;
int StanjeTipkala;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed, LOW);
}

void loop() {
  StanjeTipkala = analogRead(PinTipkalo);
  if (StanjeTipkala > 1800 and StanjeTipkala < 2000) {
    digitalWrite(PinLed, HIGH);
  } else {
    digitalWrite(PinLed, LOW);
  }
}
```

Napomena: ukoliko program ne reagira ispravno na tipu DOWN, napravite provjeru očitanja koristeći serijski monitor. Kako ga upotrijebiti pogledajte u poglavlju Temperatura.

ZADATAK 6. Spojite 2 dodatne LED (npr. crvenu i zelenu) na VIDI-X. Nemojte zaboraviti spojiti pripadajući otpornik i prebaciti pripadajući mikroprekidač. Neka obje LED diode svijetle kada je tipkalo BTN_A pritisnuto. Kada je tipkalo otpušteno, neka LED diode ne svijetle.

RJEŠENJE. Pri spajanju dodatnih LED treba uzeti u obzir kako se sada koristi BTN_A te LED nije moguće spojiti na pripadajući 23. utor pinskog proširenja. Tipka je spojena na GPIO32 i pripadajući mikroprekidač BTN_A treba biti u položaju Game Use. Zato ćemo crvenu LED spojiti na GPIO14 (19. utor), zelenu na GPIO13 (17. utor) i prebaciti pripadajuće mikroprekidače MIC i MENU u položaj Use Exp.



Program sada glasi:

```
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena
int PinTipkalo = 32;
int StanjeTipkala;
```

```
void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == LOW) {
    digitalWrite(PinLed1, HIGH);
    digitalWrite(PinLed2, HIGH);
  } else {
    digitalWrite(PinLed1, LOW);
    digitalWrite(PinLed2, LOW);
  }
}
```

ZADATAK 7. Promijeni program tako da crvena LED dioda svijetli kada je tipkalo pritisnuto, a zelena LED dioda kada je tipkalo otpušteno.

RJEŠENJE. Program sada glasi:

```
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena
int PinTipkalo = 32;
int StanjeTipkala;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  pinMode(PinTipkalo, INPUT_PULLUP);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == LOW) {
    digitalWrite(PinLed1, LOW);
    digitalWrite(PinLed2, HIGH);
  } else {
    digitalWrite(PinLed1, HIGH);
    digitalWrite(PinLed2, LOW);
  }
}
```

ZADATAK 8. Napišite program koji će paliti LED diode kada je pritisnuto tipkalo BTN_A i gasiti LED diode kada je pritisnuto tipkalo BTN_B.

RJEŠENJE. U ovom programu kada pritisnemo tipku BTN_A i otpustimo ju, LED i dalje svijetle. Sve dok se ne pritisne tipka BTN_B. Program sada glasi:

```
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena
int PinTipkalo1 = 32;
int PinTipkalo2 = 33;
int StanjeTipkala1;
int StanjeTipkala2;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  pinMode(PinTipkalo1, INPUT_PULLUP);
  pinMode(PinTipkalo2, INPUT_PULLUP);
  digitalWrite(PinLed1, LOW);
```

```
  digitalWrite(PinLed2, LOW);
}

void loop() {
  StanjeTipkala1 = digitalRead(PinTipkalo1);
  StanjeTipkala2 = digitalRead(PinTipkalo2);
  if (StanjeTipkala1 == LOW) {
    digitalWrite(PinLed1, HIGH);
    digitalWrite(PinLed2, HIGH);
  }
  if (StanjeTipkala2 == LOW) {
    digitalWrite(PinLed1, LOW);
    digitalWrite(PinLed2, LOW);
  }
}
```

ZADATAK 9. Napišite program koji na pritisak tipkala BTN_A pali crvenu LED diodu i gasi zelenu LED diodu. Na pritisak tipkala BTN_B radi suprotno: gasi crvenu LED diodu i pali zelenu LED diodu.

RJEŠENJE. Program sada glasi:

```
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena
int PinTipkalo1 = 32; //BTN_A
int PinTipkalo2 = 33; //BTN_B
int StanjeTipkala1;
int StanjeTipkala2;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  pinMode(PinTipkalo1, INPUT_PULLUP);
  pinMode(PinTipkalo2, INPUT_PULLUP);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
}

void loop() {
  StanjeTipkala1 = digitalRead(PinTipkalo1);
  StanjeTipkala2 = digitalRead(PinTipkalo2);
  if (StanjeTipkala1 == LOW) {
    digitalWrite(PinLed1, LOW);
    digitalWrite(PinLed2, HIGH);
  }
  if (StanjeTipkala2 == LOW) {
    digitalWrite(PinLed1, HIGH);
    digitalWrite(PinLed2, LOW);
  }
}
```

ZADATAK 10. Napišite program koji će na svaki pritisak tipke BTN_A mijenjati stanje LED dioda: kada se prvi puta pritisne, diode će svijetliti. Kada se sljedeći puta pritisne, diode će se ugasiti.

RJEŠENJE. Kako bismo znali treba li pritiskom tipke upaliti ili ugasiti LED, moramo znati njihovo stanje prije pritiska tipke. Zato, nakon provjere je li tipkalo pritisnuto, potrebno je provjeriti je li LED dioda ugašena ili upaljena. To činimo dodatnom `if` naredbom. Nakon pritiska tipkala, potrebno je promijeniti stanje diode: ako je prije pritiska tipkala, LED bila ugašena, treba ju upaliti (`StanjeLED = HIGH;`) i obratno, ako je bila upaljena, treba ju ugasiti (`StanjeLED = LOW;`). Nakon toga se LED postavljaju na trenutnu vrijednost varijable `StanjeLED`.

Prije promjene stanja diode dodano je čekanje od 50 ms, kako bi VIDI-X stigao reagirati na pritisak tipke.

```
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena
int PinTipkalo1 = 32; //BTN_A
```

```
int StanjeTipkala1;
int StanjeLED = LOW;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  pinMode(PinTipkalo1, INPUT_PULLUP);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
 }

void loop() {
  StanjeTipkala1 = digitalRead(PinTipkalo1);
  if (StanjeTipkala1 == LOW) {
    if (StanjeLED == LOW) {
      StanjeLED = HIGH;
    } else {
      StanjeLED = LOW;
    }
    delay(50);
    digitalWrite(PinLed1, StanjeLED);
    digitalWrite(PinLed2, StanjeLED);
  }
}
```

PROJEKT. Morseov kod je izmislio Samuel Morse za šifriranje i prenošenje slova, brojeva i znakova koristeći odgovarajući niz udaraca, tonova ili svjetlosnih treptaja različitih dužina trajanja. Morseovom kodom se koriste spasilačke službe, radioamateri i vojnici. Npr. ostali ste zarobljeni na planini, po noći. Helikopter kruži oko planine i traži vas. Kako ćete mu signalizirati gdje se nalazite?

Oznaka za svako slovo se sastoji od određenog broja crtica i točkica. Kod prenošenja poruke vrijede pravila:

1. Točka je jedan kratak udarac, ton ili svjetlosni signal.
2. Crtica je traje 3 puta dulje od točke.
3. Razmak između pojedinih točaka ili crtica je trajanja 1 točke.
4. Pojedina slova se odvajaju tišinom u trajanju tri točke.

A	.-	**G**	--.	**O**	---	**0**	-----
B	-...	**H**	....	**P**	.--.	**1**	.----
C	-.-.	**I**	..	**R**	.-.	**2**	..---
Č	-.--.	**J**	.---	**S**	...	**3**	...--
Ć	-.-..	**K**	-.-	**Š**	---.-	**4**	....-
D	-..	**L**	.-..	**T**	-	**5**	.....
DŽ	--.-.	**LJ**	.---.	**U**	..-	**6**	-....
Đ	-..--	**M**	--	**V**	...-	**7**	--...
E	.	**N**	-.	**Z**	--..	**8**	---..
F	..-.	**NJ**	--.--	**Ž**	--..-	**9**	----.

Osmislite tajnu poruku koju ćete prenijeti prijatelju koristeći Morseov kod i svjetlost.

U ovom projektu ćemo prenijeti tajnu poruku koristeći tipkalo i LED. Kada je tipkalo pritisnuto LED svijetli, kada tipkalo nije pritisnuto LED ne svijetli. Pošaljite tajnu poruku prema tablici Morseovih kodova. Je li ju prijatelj uspio dekodirati?

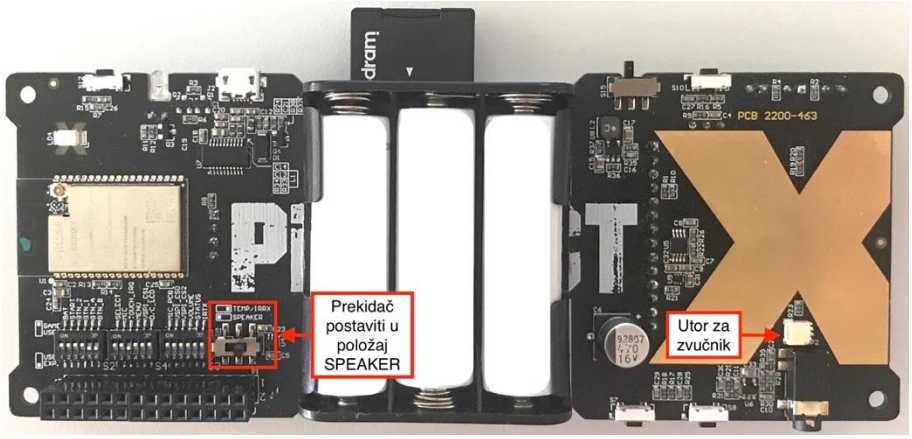
POSLANA PORUKA: _____

PORUKA PRIMLJENA: _____

4. Klavir

Zvučnik se spaja sa stražnje strane VIDI X na već ugrađeni utor za zvučnik. Zvučnik koristi pin GPIO25 i nema svoj mikroprekidač, ali Prekidač mora biti u položaju SPEAKER.

Zvučnik



ZADATAK 1. Programirajte zvučnik da proizvodi zvuk sirene.

RJEŠENJE. Zujalicom možemo upravljati koristeći pulsno-širinsku modulaciju, PWM (engl. pulse width modulation). PWM omogućuje dobivanje analognih signala koristeći digitalne. Digitalni signal može poprimiti vrijednosti LOW i HIGH (0V ili 3.3V). Kako svaka od tih vrijednosti traje neko vrijeme, digitalni signal zapravo ima oblik pravokutnika vrijednosti 0V ili 3.3V. Mijenjajući trajanje tih pravokutnika i razmaka između njih unutar jednog perioda mogu se proizvesti i vrijednosti napona između 0V i 3.3V, tj. analogne vrijednosti signala.

ESP32 na kojem se temelji VIDI-X ima 16 nezavisnih kanala (između 0 i 15) koji se mogu konfigurirati da generiraju PWM signale različitih svojstava. U programu prvo moramo definirati koji PWM kanal koristimo (0), postaviti frekvenciju PWM signala (10 kHz) te izabrati rezoluciju (12 bita, što znači da vrijednosti mogu biti između 0 i 4095):

```
ledcSetup(kanal, frekvencija, rezolucija);
```

Vezu između pina na koji je spojena zujalica i PWM kanala definiramo naredbom:

```
ledcAttachPin(PinZvucnik, kanal);
```

Konačno, zvuk proizvodimo pomoću naredbe:

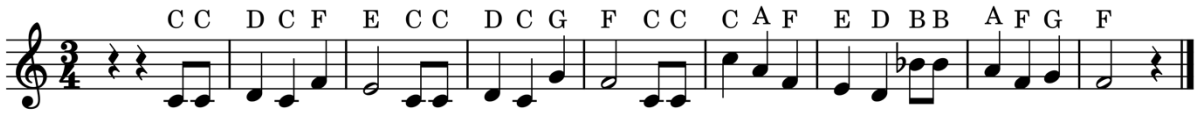
```
ledcWriteTone(kanal, frekvencija);
```

Svaka nota je definirana određenom frekvencijom. Nota C ima frekvenciju 262 Hz, nota E 330 Hz, a nota A 440 Hz. Kako bi lakše bilo svirati, na početku programa definiramo nazive svake note kako bismo kasnije mogli koristiti te nazive umjesto upisivanja točno određene frekvencije.

```
#define NOTA_C4 262
#define NOTA_D4 294
#define NOTA_E4 330
#define NOTA_F4 349
#define NOTA_G4 392
#define NOTA_A4 440
#define NOTA_B4 466
#define NOTA_H4 494
#define NOTA_C5 523

int PinZvucnik = 25;
```

```
void setup(){
 ledcSetup(0, 10000, 12);
 ledcAttachPin(PinZvucnik, 0);
}

void loop(){
  ledcWriteTone(0, NOTA_C4);
  delay(500);
  ledcWriteTone(0, NOTA_E4);
  delay(500);
  ledcWriteTone(0, 0);
  delay(500);
}
```

ZADATAK 2. Odsvirajte „Sretan rođendan“.



RJEŠENJE. Primjer programa:

```
#define NOTA_C4 262
#define NOTA_D4 294
#define NOTA_E4 330
#define NOTA_F4 349
#define NOTA_G4 392
#define NOTA_A4 440
#define NOTA_B4 466
#define NOTA_H4 494
#define NOTA_C5 523

int PinZvucnik = 25;
int trajanje = 300;

void setup(){
 ledcSetup(0, 10000, 12);
 ledcAttachPin(PinZvucnik, 0);
}

void loop(){
  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_D4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_C4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_F4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_E4);
  delay(4*trajanje);

  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_D4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_C4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_G4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_F4);
  delay(4*trajanje);
```

```
  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_C4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_C5);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_A4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_F4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_E4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_D4);
  delay(2*trajanje);

  ledcWriteTone (0, NOTA_B4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_B4);
  delay(trajanje);
  ledcWriteTone (0, NOTA_A4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_F4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_G4);
  delay(2*trajanje);
  ledcWriteTone (0, NOTA_F4);
  delay(4*trajanje);

  ledcWriteTone(0, 0);
  delay(3000);
}
```

ZADATAK 3. Skladajte svoju melodiju!

ZADATAK 4. Napravite klavir: pritišćući različite tipke na VIDI-X napravite instrument. Pritiskom svake tipke neka se proizvede druga nota na zujalici.

RJEŠENJE. Svakoj noti možemo definirati jednu tipku:

nota	frekvencija	tipka	pin
NOTA_C4	262	LEFT	GPIO34
NOTA_D4	294	UP	GPIO35
NOTA_E4	330	RIGHT	GPIO34
NOTA_F4	349	DOWN	GPIO35
NOTA_G4	392	BTN_B	GPIO33
NOTA_A4	440	BTN_A	GPIO32
NOTA_B4	466	SELECT	GPIO27
NOTA_H4	494	VOLUME	GPIO0
NOTA_C5	523	START	GPIO39

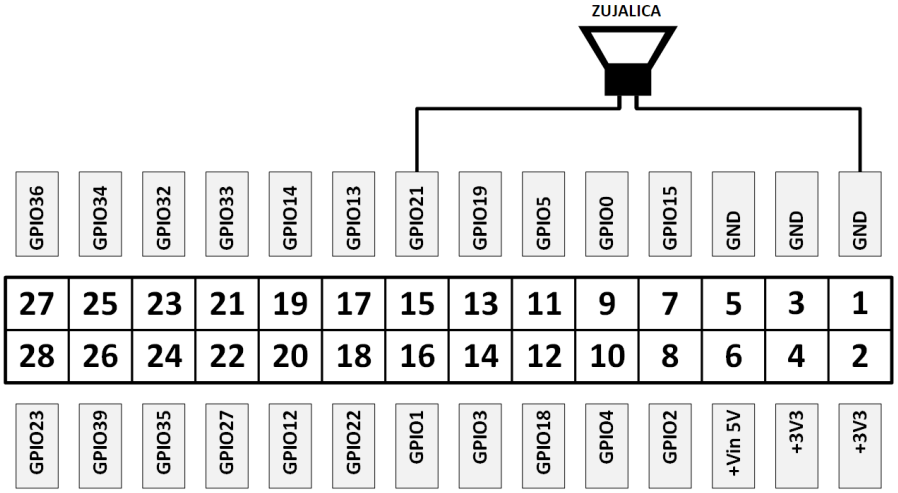
Svi pripadajući mikroprekidači trebaju biti u položaj Game Use: BTN_A, BTN_B, BTN_1, BTN_2, SELECT, VOLUME, START. Pri tome je mikroprekidač D/C_LCD u položaj Use Exp.
Program glasi:

```
#define NOTA_C4 262
#define NOTA_D4 294
#define NOTA_E4 330
#define NOTA_F4 349
#define NOTA_G4 392
#define NOTA_A4 440
#define NOTA_B4 466
#define NOTA_H4 494
#define NOTA_C5 523
```

```
int PinZvucnik = 25;
int PinTipkalo_C_E = 34;
int PinTipkalo_D_F = 35;
int PinTipkalo_G = 33;
int PinTipkalo_A = 32;
int PinTipkalo_B = 27;
int PinTipkalo_H = 0;
int PinTipkalo_C5 = 39;

void setup(){
 ledcSetup(0, 10000, 12);
 ledcAttachPin(PinZvucnik, 0);

 pinMode(PinTipkalo_C_E, INPUT_PULLUP);
 pinMode(PinTipkalo_D_F, INPUT_PULLUP);
 pinMode(PinTipkalo_G, INPUT_PULLUP);
 pinMode(PinTipkalo_A, INPUT_PULLUP);
 pinMode(PinTipkalo_B, INPUT_PULLUP);
 pinMode(PinTipkalo_H, INPUT_PULLUP);
 pinMode(PinTipkalo_C5, INPUT_PULLUP);
}

void loop(){
  if (analogRead(PinTipkalo_C_E) > 4000) {
    ledcWriteTone(0, NOTA_C4);
    delay(300);
  } else if (analogRead(PinTipkalo_C_E)>1900 and analogRead(PinTipkalo_C_E)<2000) {
    ledcWriteTone(0, NOTA_E4);
    delay(300);
  } else if (analogRead(PinTipkalo_D_F) > 4000) {
    ledcWriteTone(0, NOTA_D4);
    delay(300);
  } else if (analogRead(PinTipkalo_D_F)>1900 and analogRead(PinTipkalo_D_F)<2000) {
    ledcWriteTone(0, NOTA_F4);
    delay(300);
  } else if (digitalRead(PinTipkalo_G)== LOW) {
    ledcWriteTone(0, NOTA_G4);
    delay(300);
  } else if (digitalRead(PinTipkalo_A)== LOW) {
    ledcWriteTone(0, NOTA_A4);
    delay(300);
  } else if (digitalRead(PinTipkalo_B)== LOW) {
    ledcWriteTone(0, NOTA_B4);
    delay(300);
  } else if (digitalRead(PinTipkalo_H)== LOW) {
    ledcWriteTone(0, NOTA_H4);
    delay(300);
  } else if (digitalRead(PinTipkalo_C5)== LOW) {
    ledcWriteTone(0, NOTA_C5);
    delay(300);
  } else {
    ledcWriteTone(0, 0);
    delay(300);
  }
}
```

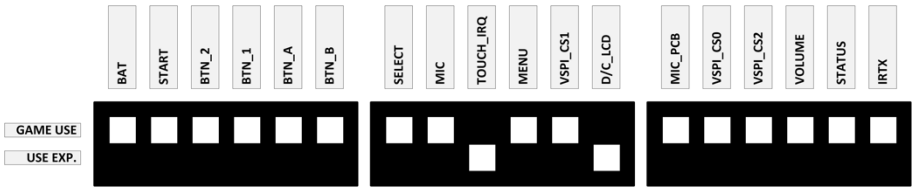
ZADATAK 5. Spojite zujalicu na VIDI-X prema shemi.

Piezo zujalica



RJEŠENJE. Piezoelektrični efekt je pojava stvaranja električnog naboja na površini kristala koji se deformira vanjskom silom. Jedna strana kristala se nabija pozitivno, druga negativno i kristal postaje električki polariziran, na neki način sličan kapacitetu. Moguća je i obrnuta primjena kada se pomoću električnog napona postiže mehanička deformacija materijala. Na principu piezoelektričnog efekta rade „tanki“ zvučnici, mikrofoni, digitalne vage, detektori pritiska, sonari za istraživanje podmorja, mehanički satovi, električna zvona i drugo. U ovoj vježbi koristimo piezo element, buzzer ili zujalicu PKM22EPP-40. Pojedini slovo u oznaci predstavlja: PK – piezoelektrična zvučna komponenta, M – zujalica, 22 – vanjski promjer u mm, E – upravljan vanjski (npr. pomoću signala s VIDI-X), PP – vrsta koja ima pin, 40 – frekvencija osciliranja (4 kHz). Ako se koristi mala zujalica, moguće ju je direktno spojiti na digitalni pin.

Zujalicu možemo spojiti na pinsko proširenje VIDI-X, npr. na GPIO21 (15. utor pinskog proširenja) i pri tome prebaciti mikroprekidač D/C_LCD u položaj Use Exp. Druga strana zujalice ide na GND (npr. utor 1 pinskog proširenja).



ZADATAK 6. Ponovite zadatke 1 – 4 koristeći ovako dodanu zujalicu.

RJEŠENJE. Programi su identični rješenjima zadataka 1. – 4. Jedina razlika je u pinu koji se koristi. U slučaju dodavanja zujalice na pinsko proširenje i njeno spajanje na GPIO21, u programima se mora koristiti:

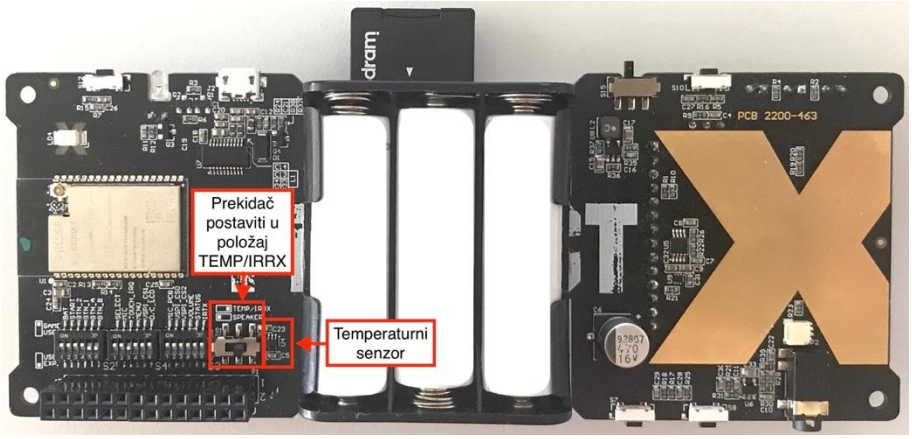
```
int PinPiezo = 21;
```

LITERATURA.

https://www.arduino.cc/documents/datasheets/PIEZO-PKM22EPPH4001-BO.pdf
https://physics.stackexchange.com/questions/502258/why-is-a-piezoelectric-material-a-capacitor
https://www.arduino.cc/en/Tutorial/Foundations/PWM

5. Temperatura

Na VIDI-X pločici se nalazi senzor temperature MCP9700AT koji ima raspon mjerenja od -40°C do 125°C uz točnost ±2°C. Nalazi se sa stražnje strane pločice i spojen je na GPIO26. Kako bi bilo moguće koristiti ovaj senzor potrebno je Prekidač postaviti u položaj TEMP/IRRX. Temperatura se očitava u 12 bitnoj rezoluciji, što znači da je moguće očitati 2^{12} = 4096 različitih vrijednosti.



ZADATAK 1. Očitajte vrijednost temperaturnog senzora. Očitanu vrijednost ispišite na zaslon računala.

RJEŠENJE. Senzor temperature je ulazni element te se definira pomoću `pinMode(PinTemp, INPUT)`. Kako želimo očitati prave izmjerene vrijednosti, koristi se analogno očitavanje: `temp = analogRead(PinTemp)`.

Za ispis na zaslon računala potrebno je omogućiti serijsku komunikaciju između VIDI-X i računala. Komunikacija započinje otvaranjem serijske veze i definiranjem brzine prijenosa podataka `Serial.begin(9600)`, gdje je 9600 brzina prijenosa u bitovima u sekundi (9600 bps). Sam ispis na ekran vrši se pomoću naredbe `Serial.print()` kojoj se zada varijabla ili tekst koji želimo ispisati na ekranu. Pomoću ove naredbe sav test se ispisuje u istom retku. Ako želimo nakon ispisa prijeći u novi red naredba je `Serial.println`.

Kako bismo vidjeli što se ispisalo na računalu, nakon prebacivanja programa, pritisnite oznaku za serijski monitor (Serial monitor). Primijetite da za vrijeme rada serijske veze (ispisa na serijski monitor), ugrađena crvena LED treperi.

```
int PinTemp = 26;
int temp;

void setup() {
  pinMode(PinTemp, INPUT);
  Serial.begin(9600);
}

void loop() {
  temp = analogRead(PinTemp);

  Serial.print(temp);

  delay(1000);
}
```

Koliko je očitanje senzora? Zagrijte temperaturni senzor (npr. puhnite topli zrak ili prstima). Koje se sada vrijednosti ispisuju?
Vrijednosti koje se ispisuju na Serijski monitor su sirova očitanja senzora, iznosa između 0 i 4095. Kako bi dobili vrijednosti u stupnjevima Celzijusa, ove iznose je potrebno preračunati.

ZADATAK 2. Preračunajte očitane vrijednosti temperaturnog senzora u stvarne vrijednosti temperature.

RJEŠENJE. Temperaturni senzor očitava vrijednosti između 0 i 4095 (ukupno 4096 različitih vrijednosti). Procesor ESP32 radi na 3.3V = 3300 mV, što znači da je očitanje 0 isto što i 0V, a očitanje 4095 isto što i 3. 3V = 3300 mV. Kako bismo linearno pretvorili vrijednosti očitanja u napone, koristimo naredbu `map`:
`tempV = map(temp, minTemp, maxTemp, minTempVolti, maxTempVolti);`
U našem slučaju: `tempV = map(temp, 0, 4095, 0, 3300);`
`tempV` je decimalni broj. Na početku programa ga treba deklarirati pomoću: `float tempV;`

Za kraj još pretvorimo napon u stupnjeve Celzijeve: `tempC = tempV / 10 - 40;`
Kako senzor očitava temperature od -40°, a ne od 0°, od izračunate vrijednosti moramo oduzeti 40, kako bismo sva očitanja ispravno postavili i na pozitivne i negativne vrijednosti temperature.
`tempC` je također decimalni broj, pa je deklariran kao `float tempC;`
Uz pomoć termometra možemo provjeriti jesu li očitanja ispravna. Ukoliko postoji razlika, potrebno je kalibrirati senzor. To možemo učiniti laganom promjenom broja 40, ovisno o pogreški.

```
int PinTemp = 26;
int temp;

float tempV;
float tempC;

void setup() {
  pinMode(PinTemp, INPUT);
  Serial.begin(9600);
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;

  Serial.println(tempC);

  delay(1000);
}
```

Ako nam je zima za ruke – pušemo u njih da se ugrijemo. Ako nam je juha vruća, pušemo u nju da ju ohladimo. Je li nam dah topli ili hladni? Kolika je temperatura puhanja ako smo blizu senzoru? Kolika je temperatura puhanja ako smo daleko od senzora? Testirajte koristeći gornji program i VIDI-X.

ZADATAK 3. Kada je temperatura veća od 21.5°C neka se plava LED upali, kada je temperatura manja od 21.5°C neka se plava LED ugasi.

RJEŠENJE. Program glasi:

```
int LEDPin1 = 2;
int PinTemp = 26;
int temp;

float tempV;
float tempC;

void setup() {
  pinMode(PinTemp, INPUT);
  Serial.begin(9600);
  pinMode(LEDPin1, OUTPUT);
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;
```

```
  Serial.println(tempC);

  if (tempC > 21.5) {
    digitalWrite(LEDPin1, HIGH);
  } else {
    digitalWrite(LEDPin1, LOW);
  }

  delay(1000);
}
```

Prag za paljenje i gašenje LED podesite u ovisnosti o temperaturi okoline u kojoj testirate.

ZADATAK 4. Dodajte jednu LED diodu. Kada je temperatura veća od 21.5° neka svijetli ugrađena plava LED, a kada je temperatura manja neka svijetli druga, dodana LED.

RJEŠENJE. Dodatnu LED ćemo spojiti na GPIO14 (19. utor) i prebaciti pripadajući mikroprekidač MIC u položaj Use Exp. Program glasi:

```
int LEDPin1 = 2;
int LEDPin2 = 14;
int PinTemp = 26;
int temp;

float tempV;
float tempC;

void setup() {
  pinMode(PinTemp, INPUT);
  Serial.begin(9600);
  pinMode(LEDPin1, OUTPUT);
  pinMode(LEDPin2, OUTPUT);
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;

  Serial.println(tempC);

  if (tempC > 21.5) {
    digitalWrite(LEDPin1, HIGH);
    digitalWrite(LEDPin2, LOW);
  } else {
    digitalWrite(LEDPin1, LOW);
    digitalWrite(LEDPin2, HIGH);
  }

  delay(1000);
}
```

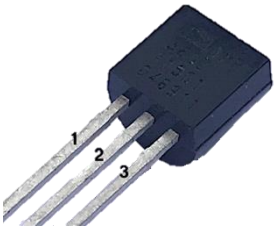
PROJEKT. Imate li temperaturu? Napravite uređaj koji će mjeriti temperaturu tijela pritiskom dva prsta oko temperaturnog senzora.

Ako je temperatura manja od 35 neka se upali PLAVA LED.
Ako je temperatura viša od 35 i manja od 37 neka se upali ZELENA LED.
Ako je temperatura viša od 37 i manja od 39 neka se upali ŽUTA LED.
Ako je temperatura viša od 39 neka se upali CRVENA LED.

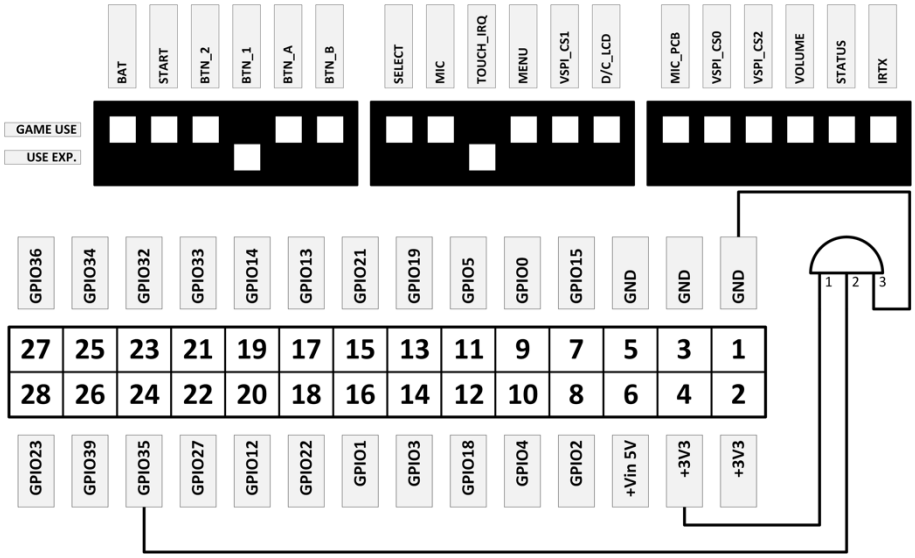
ZADATAK 5. Spojite dodatni temperaturni senzor na VIDI-X prema shemi. Na serijski monitor ispisujte očitanja ugrađenog i ovog dodatnog temperaturnog senzora. Poklapaju li se mjerenja?

RJEŠENJE. Kao dodatni temperaturni senzor koristimo TMP36. Ovaj senzor radi na niskim naponima (2.7V do 5.5V) te je idealan za spajanje na VIDI-X. Prvi pin se spaja na napajanje (3.3V, utor 4), drugi pin je signalni i spojit ćemo ga na utor 24, GPIO35, dok treći pin ide na GND (utor 1). Prvi pin se nalazi s lijeve strane ukoliko senzor gledate prema njegovoj ravnoj strani (prema slici). Mikroprekidač BTN_1 postavljamo u Use Exp. Temperaturni raspon ovog senzora je -40°C do 125°C uz točnost ±2°C i faktor skale 10mV/°C.

Moguće je koristiti i druge temperaturne senzore, ali prije provjerite raspored pinova, potrebno napajanje i temperaturni raspon.



Temperaturni senzor
pin 1 na 3.3V
pin 2 na npr. GPIO35
pin 3 na GND



Program je sličan kao u drugom zadatku, samo što sada ispisujemo i vrijednosti ugrađenog i ovog dodatnog senzora temperature.

```
int PinTemp = 26;  // ugrađeni senzor
int PinTemp2 = 35; // dodatni senzor
int temp;
int temp2;

float tempV;
float tempC;

float tempV2;
float tempC2;

void setup() {
  pinMode(PinTemp, INPUT);
  pinMode(PinTemp2, INPUT);

  Serial.begin(9600);
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;

  temp2 = analogRead(PinTemp2);
  tempV2 = map(temp2, 0, 4095, 0, 3300);
  tempC2 = tempV2 / 10 - 40;


  Serial.print("Ugradeni:");
  Serial.print("    ");
  Serial.print(temp);
  Serial.print("    ");
```

```
  Serial.print(tempV);
  Serial.print("   ");
  Serial.print(tempC);
  Serial.print("   Dodani:");
  Serial.print("   ");
  Serial.print(temp2);
  Serial.print("   ");
  Serial.print(tempV2);
  Serial.print("   ");
  Serial.println(tempC2);
  delay(1000);
}
```

Primjer ispisa dan je na slici. Očitanja oba senzora su slična.

```
08:34:57.017 -> Ugradeni:   761   613.00   21.30   Dodani:   762   614.00   21.40
08:34:58.001 -> Ugradeni:   762   614.00   21.40   Dodani:   763   615.00   21.50
08:34:59.008 -> Ugradeni:   761   613.00   21.30   Dodani:   761   613.00   21.30
08:35:00.006 -> Ugradeni:   763   615.00   21.50   Dodani:   761   613.00   21.30
08:35:00.996 -> Ugradeni:   759   612.00   21.20   Dodani:   759   612.00   21.20
```

LITERATURA.

https://www.analog.com/en/products/tmp36.html#product-overview
https://www.instructables.com/Simple-Thermometer-Using-TMP36-and-ESP32/
https://learn.adafruit.com/tmp36-temperature-sensor?view=all

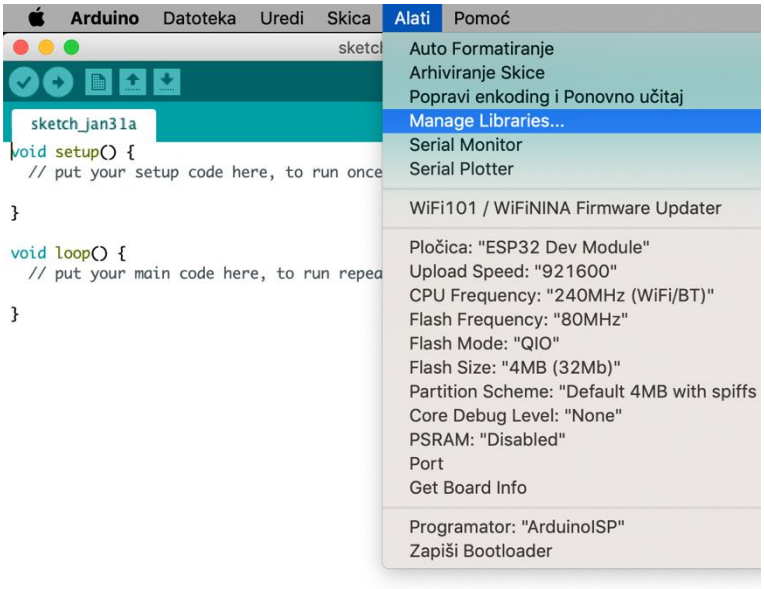
6. TFT ekran – ispis teksta

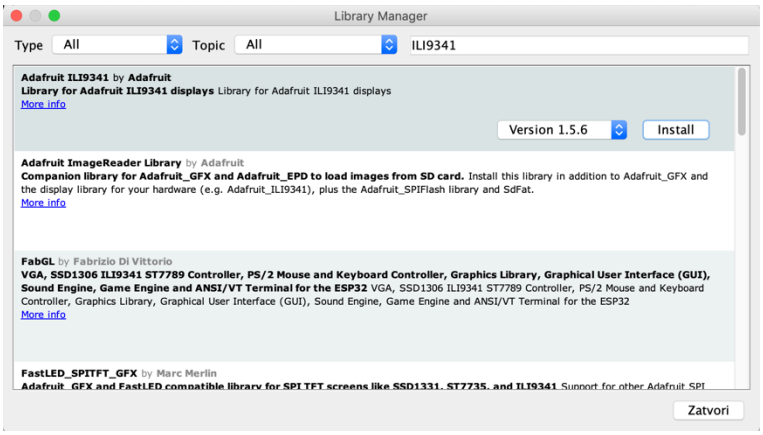
Ekran koji dolazi s VIDI-X je 2.8'' ILI9341 SPI TFT LCD ekran s rezolucijom 320x240 piksela i osjetljiv je na dodir. Sam ekran ima mnoštvo pinova, od kojih je dio namijenjen napajanju, dio upravljanju, dio slanju podataka koji će biti prikazani na ekranu, a dio za primanje informacija ukoliko se piše po ekranu.

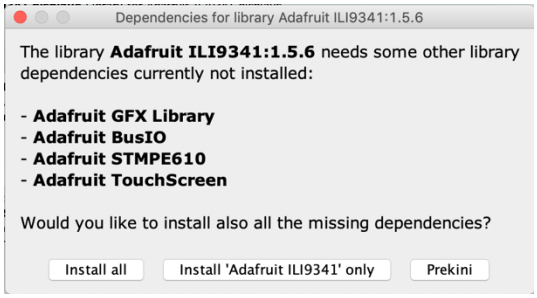
Pri korištenju ekrana za ispis bitno je da mikroprekidači D/C LCD i VSPI_CS0 budu u položaju Game Use. Pin D/C LCD služi za razlikovanje šalju li se naredbe ili podaci dok pin VSPI_CS0 ima ulogu izbora integriranog kruga. Za pristup D/C LCD koristimo GPIO21, a za VSPI_CS0 koristimo GPIO05. Pinovi VSPI_MOSI (GPIO23), VSPI_MISO (GPIO19) i VSPI_SCK (GPIO18) nemaju mikroprekidače već su interno povezani s pinovima ekrana. TOUCH_IRQ mora biti u položaju Use Exp kako bi bilo moguće prebaciti program u VIDI-X.

Ako se ekran koristi za čitanje podataka, koristi se pin GPIO04 te mikroprekidač VSPI_SC2 mora biti u položaju Game Use (više u poglavlju 8). Dodatno, uz ekran se nalazi i SD kartica, koja također koristi pinove VSPI_SCK, VSPI_MISO, i VSPI_MOSI interno, uz dodatak VSPI_SC2 koji mora biti u Game Use.

Za upravljanje ekranom koristi se ILI9341 driver, kojeg je potrebno dodatno preuzeti s Interneta i instalirati. Za preuzimanje i instalaciju potrebno je unutar Arduino IDE u izborniku *Alati* izabrati *Manage Libraries*. U pretraživač unesite naziv biblioteke ILI9341. Pritiskom na Install instalirajte najnoviju verziju. Svakako odaberite instaliranje svih dodatnih biblioteka koje ovaj driver zahtjeva (kliknite *Install all* u novom prozoru koji se otvori).







Instalirane biblioteke su:

- GFX je osnovna grafička biblioteka i omogućuje rad s pikselima, linijama, pravokutnicima, kružnicama, oblim pravokutnicima, trokutima, pisanim tekstom i rotacijama.
- BusIO biblioteka omogućuje korištenje UART, I2C i SPI sučelja.
- STMPE610 sadrži 4-žični kontroler ekrana osjetljivog na dodir i koristi se za proširenje ulazno-izlaznih portova kako bi se isti pinovi mogli koristiti i kao ulaz i kao izlaz.
- TouchScreen biblioteka omogućuje rad s rezistivnim ekranom osjetljivim na dodir.

U ovom poglavlju ćemo upoznati ispis teksta na ekran, a u narednima iscrtavanje slika i korištenje senzora dodira.

ZADATAK 1. Na sredini TFT ekrana ispišite svoje ime. Pozadina ekrana neka bude plave boje, a ime žute boje.

RJEŠENJE. Kako bi u program uključili funkcije koje su fizički napisane u nekoj drugoj datoteci, na početku programa je potrebno najaviti njihovo korištenje. Moguće je upotrijebiti dva načina:

```
#include <DatotekaIzBiblioteke.h> ili #include "LokalnaDatoteka.h"
```

Ako se koriste uglate zagrade, program će pretražiti staze biblioteka u potrazi za željenom datotekom. Ako se koriste navodnici, u potrazi za datotekom će se prvo pretražiti direktorij u kojem je spremljen sam program, a tek nakon toga staze biblioteka.

Pri ispisu teksta na TFT ekran, moramo uključiti biblioteke:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
```

Nakon toga je potrebno definirati pinove koje ekran koristi:

```
#define TFT_CS   5
#define TFT_DC  21
```

I stvoriti objekt ekrana `tft`. Pod tim imenom ćemo u nastavku programa prepoznavati naredbe i funkcije koje se odnose na rad s ekranom.

```
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

Inicijalizaciju naredbi vezanih uz ekran obavlja funkcija `tft.begin();`

Tekst se ispisuje naredbama

```
tft.print("tekst");
tft.println("tekst");
```

ovisno želimo li sljedeći ispis nastaviti tamo gdje je prethodni stao ili želimo ispis u novi red.

Cijeli ekran je moguće obojati u željenu boju naredbom:

```
tft.fillScreen(boja);
```

Tekst koji ispisujemo je moguće obojati pomoću naredbe:

```
tft.setTextColor(boja);
```

Standardno je pozadina svakog znaka prozirna i vidi se boja ekrana. Ako želimo obojati tekst i njegovu pozadinu koristimo naredbu:

```
tft.setTextColor(boja_teksta,boja_pozadine);
```

Boje koje je moguće koristiti su:

```
ILI9341_BLACK
ILI9341_WHITE
ILI9341_RED
ILI9341_GREEN
ILI9341_BLUE
ILI9341_YELLOW
```

Boju je moguće zadati i pomoću heksadekadskog broja. Svaka piksel ima 16 bita i to 5 bita za crvenu, 6 za zelenu i 5 za plavu boju. Ukupno je moguće napraviti 65.536 boja. Primjeri boja:

```
#define BLACK    0x0000
#define BLUE     0x001F
#define RED      0xF800
#define GREEN    0x07E0
#define CYAN     0x07FF
#define MAGENTA  0xF81F
#define YELLOW   0xFFE0
#define WHITE    0xFFFF
```

Tekst na ekran možemo ispisati u više smjerova, obilježenih brojevima 0, 1, 2 ili 3. Smjer se definira naredbom:

```
tft.setRotation(smjer);
```

Ono što je već prije napisano na ekranu će zadržati svoju orijentaciju i nakon korištenja ove naredbe. Ona se odnosi samo na ispise koje slijede nakon nje. Ova naredba se najčešće koristi samo jednom, unutar `void setup()`.

Položaj gdje će se tekst ispisati se definira naredbom:

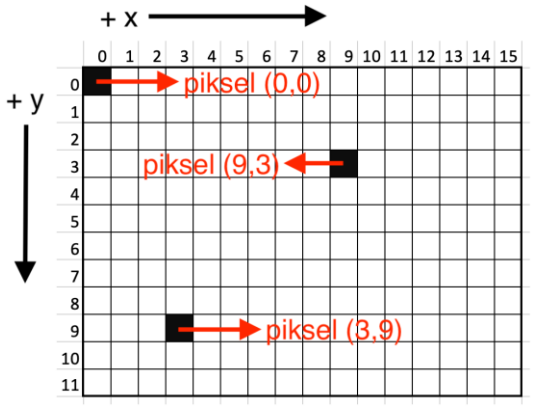
```
tft.setCursor(x, y);
```

Koordinate (0,0) predstavljaju piksel u gornjem lijevom uglu ekrana. Pri tome gornji lijevi ugao je određen rotacijom (smjerom) kako je ekran okrenut. Kako ekran ima 320x240 piksela, to su maksimalni položaji ispisa. Ukoliko se zada veći broj, tekst neće biti vidljiv na ekranu. Ukoliko ne stane cijeli tekst u jedan redak, njegov nastavak će biti na početku sljedećeg retka (pri tome je redak definiran veličinom slova). Ako se želi zadržati lokacija teksta koristi se naredba:

```
tft.setTextWrap(false);
```

a ako se želi vratiti originalni način pisanja:

```
tft.setTextWrap(true);
```



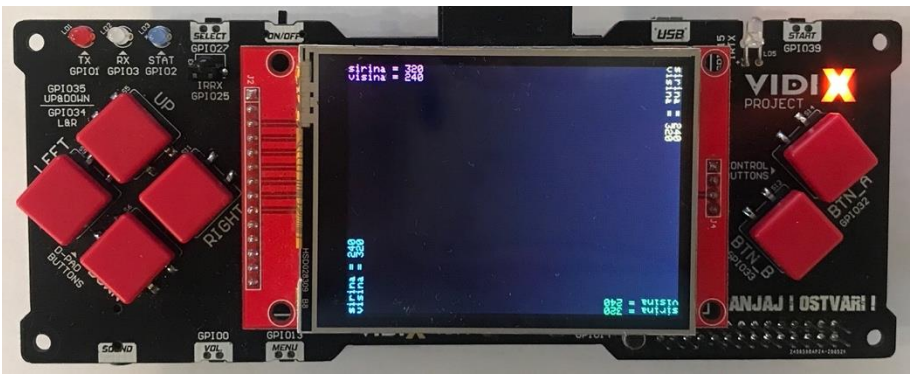
Promjena veličine teksta se definira naredbom:

```
tft.setTextSize(velicina);
```

Program za ispis imena žutom bojom na sredinu plavog ekrana glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                          // inicijalizacija zaslona

  tft.setRotation(3);                   // postavi orijentaciju
  tft.fillScreen(ILI9341_BLUE);         // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);     // boja teksta
  tft.setCursor(65, 105);               // polozaj pocetka ispisa tekst
  tft.setTextSize(4);                   // velicina teksta
  tft.print("VIDI - X");                // ispis teksta
}

void loop() {
}
```

ZADATAK 2. Napravite program koji će u gornji lijevi ugao ekrana napisati širinu i visinu ekrana u pikselima. Napravite ovaj ispis za sve 4 moguće rotacije ekrana. Svaki puta širinu i visinu ispišite u drugoj boji. Između ispisa pričekajte 1 sekundu.



RJEŠENJE. U ovom programu se osim naredbi iz prošlog zadatka koriste naredbe za dobivanje visine i širine ekrana:

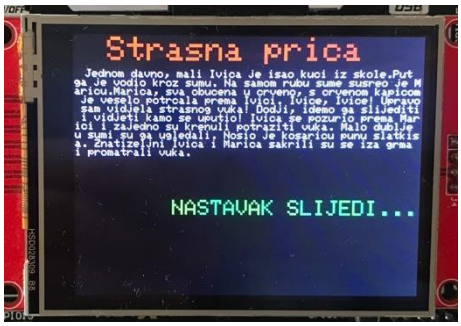
```
tft.height();
tft.width();
```

Kada se ekran koristi u različitim rotacijama, visina i širina će se mijenjati.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                      // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);    // boja zaslona
  tft.setTextSize(1);               // velicina teksta
}

void loop() {

  tft.setRotation(0);               // postavi orijentaciju
  tft.setTextColor(ILI9341_YELLOW); // boja teksta
  tft.setCursor(0, 0);              // polozaj pocetka ispisa tekst
  tft.print((String)"sirina = " + tft.width() + "\n" + "visina = " + tft.height());
  delay(1000);

  tft.setRotation(1);               // postavi orijentaciju
  tft.setTextColor(ILI9341_GREEN);  // boja teksta
  tft.setCursor(0, 0);              // polozaj pocetka ispisa tekst
  tft.print((String)"sirina = " + tft.width() + "\n" + "visina = " + tft.height());
  delay(1000);

  tft.setRotation(2);               // postavi orijentaciju
  tft.setTextColor(ILI9341_CYAN);   // boja teksta
  tft.setCursor(0, 0);              // polozaj pocetka ispisa tekst
  tft.print((String)"sirina = " + tft.width() + "\n" + "visina = " + tft.height());
  delay(1000);

  tft.setRotation(3);               // postavi orijentaciju
  tft.setTextColor(ILI9341_MAGENTA); // boja teksta
  tft.setCursor(0, 0);              // polozaj pocetka ispisa tekst
  tft.print((String)"sirina = " + tft.width() + "\n" + "visina = " + tft.height());
  delay(1000);

}
```

ZADATAK 3. Napravite program koji će na ekran napisati tekst kratke priče. Naslov priče treba biti na vrhu ekrana, u sredini retka, većim slovima, ostatak teksta manjom veličinom slova, kroz cijeli ekran.

RJEŠENJE. U ovom programu koristimo naredbe iz prijašnjih zadataka , kombinirajući veću količinu teksta. Tekst koji ne stane na ekran se automatski nastavlja prikazivati u sljedećem retku.



Ako u samom kodu želimo razlomiti tekst u više redaka kako bi ga bilo lakše čitati, na željenom mjestu je potrebno zatvoriti navodnike, prijeći u novi redak te ponovno otvoriti navodnike.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                        // inicijalizacija zaslona

  tft.setRotation(3);                 // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setTextColor(ILI9341_RED);      // boja teksta
  tft.setCursor(30, 1);               // polozaj pocetka ispisa tekst
  tft.setTextSize(3);                 // velicina teksta
  tft.print("Strasna prica");         // ispis teksta

  tft.setTextColor(ILI9341_WHITE);    // boja teksta
  tft.setCursor(10, 30);              // polozaj pocetka ispisa tekst
  tft.setTextSize(1);                 // velicina teksta
  tft.print("Jednom davno, mali Ivica je isao kuci iz skole."
     "Put ga je vodio kroz sumu. Na samom rubu sume susreo je Maricu."
     "Marica, sva obucena u crveno, s crvenom kapicom je veselo potrcala "
     "prema Ivici. Ivice, Ivice! Upravo sam vidjela strasnog vuka! Dodji, "
     "idemo ga slijediti i vidjeti kamo se uputio! "
     "Ivica se pozurio prema Marici i zajedno su krenuli potraziti vuka. "
     "Malo dublje u sumi su ga ugledali. Nosio je kosaricu punu slatkisa. "
     "Znatizeljni Ivica i Marica sakrili su se iza grma i promatrali vuka. ");

  tft.setTextColor(ILI9341_GREEN);    // boja teksta
  tft.setCursor(90, 150);             // polozaj pocetka ispisa tekst
  tft.setTextSize(2);                 // velicina teksta
  tft.print("NASTAVAK SLIJEDI...");
}

void loop() {
}
```

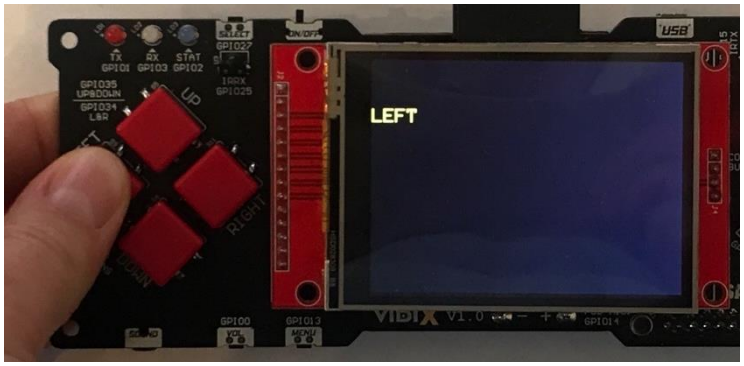
ZADATAK 4. Napravite program koji će na ekran ispisivati trenutnu temperaturu.

RJEŠENJE. Naredbe za očitavanje i preračunavanje temperatura su identične onima iz cjeline 4, zadatak 2. Pripazite da je je prekidač za senzor temperature u ispravnom položaju.
Kako u ovom zadatku nećemo mijenjati rotaciju ekrana, boju i veličinu teksta, pripadajuće naredbe postavljano unutar `void setup()`. Naredbe za ispis temperature su unutar `void loop()`. Naredba za boju ekrana se nalazi prije svakog ispisa temperature. Ona služi za brisanje teksta koji je prije pisao na ekranu kako ne bi došlo do pisanja po istom mjestu jedno preko drugoga.

Ista temperatura se osim na TFT ekran ispisuje i na serijski monitor.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int PinTemp = 26;
int temp;

float tempV;
float tempC;

void setup() {
  pinMode(PinTemp, INPUT);

  Serial.begin(9600);                 // ispis i na Serijski monitor

  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);   // boja teksta
  tft.setTextSize(2);                 // velicina teksta
  tft.setRotation(3);                 // postavi orijentaciju
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;

  Serial.println(tempC);               // ispis i na Serijski monitor

  tft.fillScreen(ILI9341_BLACK);      // boja zaslona - brisanje sto je pisalo prije
  tft.setCursor(0, 50);                // polozaj pocetka ispisa tekst
  tft.println((String)"Temperatura zraka: ");
  tft.setCursor(10, 70);               // polozaj pocetka ispisa tekst
  tft.println(tempC);                  // ispis temperature

  delay(1000);
}
```

ZADATAK 5. Napravite program koji će na ekran ispisivati koje tipkalo je pritisnuto.



RJEŠENJE. U ovom programu ćemo koristiti sve tipke i ekran, čime ćemo upotrijebiti gotovo sve ulazno-izlazne pinove. Uz mikroprekidače za ekran i svi pripadajući mikroprekidači za tipke moraju biti u položaju Game Use, kako je navedeno u tablici. Tekst na ekranu ostaje napisan pola sekunde nakon čega je ekran crne boje do sljedećeg ispisa.

tipka	pin	mikroprekidač
LEFT	GPIO34	BTN_2
UP	GPIO35	BTN_1
RIGHT	GPIO34	BTN_2
DOWN	GPIO35	BTN_1
BTN_B	GPIO33	BTN_B
BTN_A	GPIO32	BTN_A
SELECT	GPIO27	SELECT
VOLUME	GPIO0	VOLUME
START	GPIO39	START
MENU	GPIO13	MENU

Program glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int PinTipkalo_L_R = 34;
int PinTipkalo_U_D = 35;
int PinTipkalo_B = 33;
int PinTipkalo_A = 32;
int PinTipkalo_SEL = 27;
int PinTipkalo_VOL = 0;
int PinTipkalo_ST = 39;
int PinTipkalo_MEN = 13;


void setup(){
  pinMode(PinTipkalo_L_R, INPUT_PULLUP);
  pinMode(PinTipkalo_U_D, INPUT_PULLUP);
  pinMode(PinTipkalo_B, INPUT_PULLUP);
  pinMode(PinTipkalo_A, INPUT_PULLUP);
  pinMode(PinTipkalo_SEL, INPUT_PULLUP);
  pinMode(PinTipkalo_VOL, INPUT_PULLUP);
  pinMode(PinTipkalo_ST, INPUT_PULLUP);
  pinMode(PinTipkalo_MEN, INPUT_PULLUP);

  tft.begin();                          // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);     // boja teksta
  tft.setTextSize(2);                   // velicina teksta
  tft.setRotation(3);                   // postavi orijentaciju
}

void loop(){

  tft.fillScreen(ILI9341_BLACK);       // boja zaslona - brisanje sto je pisalo prije
  tft.setCursor(0, 50);                // polozaj pocetka ispisa tekst

  if (analogRead(PinTipkalo_L_R) > 4000) {
    tft.println("LEFT");                    // ispis tipke
  } else if (analogRead(PinTipkalo_L_R)>1900 and analogRead(PinTipkalo_L_R)<2000) {
    tft.println("RIGHT");                   // ispis tipke
  } else if (analogRead(PinTipkalo_U_D) > 4000) {
    tft.println("UP");                      // ispis tipke
  } else if (analogRead(PinTipkalo_U_D)>1900 and analogRead(PinTipkalo_U_D)<2000) {
```

```
    tft.println("DOWN");                    // ispis tipke
  } else if (digitalRead(PinTipkalo_B)== LOW) {
    tft.println("B");                       // ispis tipke
  } else if (digitalRead(PinTipkalo_A)== LOW) {
    tft.println("A");                       // ispis tipke
  } else if (digitalRead(PinTipkalo_SEL)== LOW) {
    tft.println("SELECT");                  // ispis tipke
  } else if (digitalRead(PinTipkalo_VOL)== LOW) {
    tft.println("VOLUME");                  // ispis tipke
  } else if (digitalRead(PinTipkalo_ST)== LOW) {
    tft.println("START");                   // ispis tipke
  } else if (digitalRead(PinTipkalo_MEN)== LOW) {
    tft.println("MENU");
  }
  delay(500);
}
```

LITERATURA.

http://educ8s.tv/arduino-2-8-ili9341-tutorial/
Hardver: https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf
GFX biblioteka: https://learn.adafruit.com/adafruit-gfx-graphics-library/overview

7. TFT ekran – slike

U ovom poglavlju ćemo upoznati rad sa slikama i njihovo iscrtavanje na je 2.8'' ILI9341 SPI TFT LCD ekranu. Za to su nam potrebne biblioteke koje smo instalirali u prošlom poglavlju GFX i ILI9341. Pri korištenju ekrana za ispis bitno je da mikroprekidači D/C LCD i VSPI_CS0 budu u položaju Game Use. Za pristup D/C LCD koristimo GPIO21, a za VSPI_CS0 koristimo GPIO05. Pinovi VSPI_MOSI (GPIO23), VSPI_MISO (GPIO19) i VSPI_SCK (GPIO18) nemaju mikroprekidače već su interno povezani s pinovima ekrana. TOUCH_IRQ je i dalje u položaju Use Exp kako bi bilo moguće prebaciti program u VIDI-X.

ZADATAK 1. Na sredini TFT ekrana obojite jedan piksel žutom bojom.

RJEŠENJE. Nakon definiranja biblioteka koje se koriste i inicijalizacije ekrana, potrebno je zadati njegovu rotaciju i pozadinu obojiti u crno.

Piksel se prikazuje pomoću naredbe:

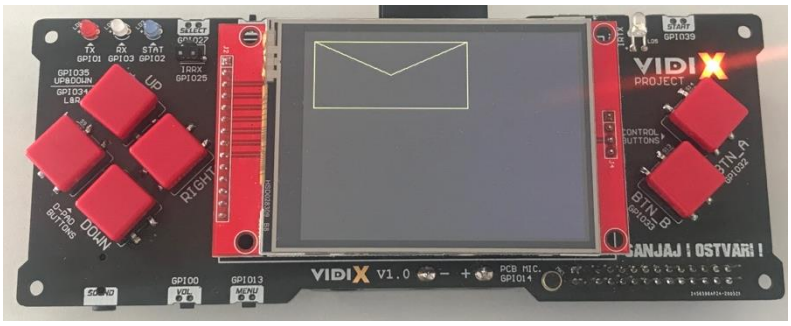
```
tft.drawPixel(x,y,boja);
```

Definicije koordinata i boja su identične kao u prošlom poglavlju.

Program glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                          // inicijalizacija zaslona

  tft.setRotation(3);                   // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona

  tft.drawPixel(159,119,ILI9341_YELLOW);  // ctanje piksela
}

void loop() {
}
```

ZADATAK 2. Nacrtajte zatvorenu kovertu na TFT ekranu.



RJEŠENJE. Za crtanje linija koristi se naredba:

```
tft.drawLine(x0, y0, x1, y1, boja);
```

gdje su x0 i y0 koordinate početka linije, a x1, y1 koordinate zadnjeg piksela u liniji.

Za crtanje vertikalnih linija postoji pojednostavljena naredba:

```
tft.drawFastVLine(x0, y0, duljina, boja);
```

gdje je osim koordinata početka linije x0 i y0 zadana i njena duljina u pikselima.

Na isti način je definirana i vodoravna linija:

```
tft.drawFastHLine(x0, y0, duljina, boja);
```

Program za crtanje zatvorene koverte glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                        // inicijalizacija zaslona

  tft.setRotation(3);                 // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona

  tft.drawLine(10, 10, 100, 50, ILI9341_YELLOW);   // kosa linija
  tft.drawLine(100, 50, 190, 10, ILI9341_YELLOW);  // kosa linija
  tft.drawFastVLine(190, 10, 80, ILI9341_YELLOW);  // vertikalna linija prema dolje
  tft.drawFastHLine(190, 90, -180, ILI9341_YELLOW);// vodoravna linija prema lijevo
  tft.drawFastVLine(10, 90, -80, ILI9341_YELLOW);  // vertikalna linija prema gore
  tft.drawFastHLine(10, 10, 180, ILI9341_YELLOW);  // vodoravna linija prema desno
}

void loop() {
}
```

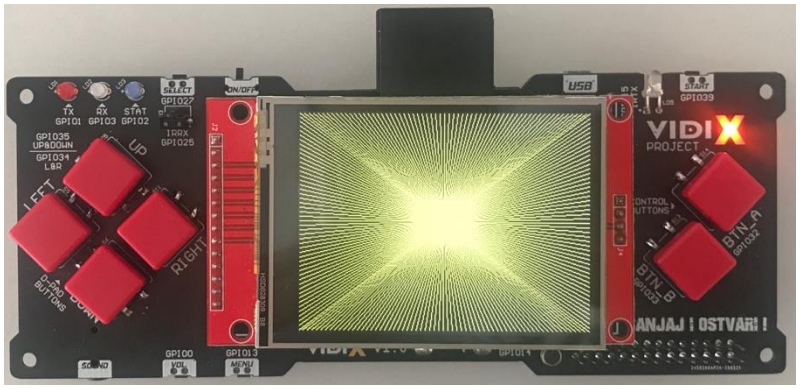
ZADATAK 3. Nacrtajte sunce na sredini TFT ekrana.

RJEŠENJE. Za crtanje sunca koristimo linije iz prošlog zadatka.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
```

```
  tft.begin();                          // inicijalizacija zaslona

  tft.setRotation(3);                   // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona

  tft.drawLine(0, 0, 319, 239, ILI9341_YELLOW);
  tft.drawLine(319, 0, 0, 239, ILI9341_YELLOW);
  tft.drawFastVLine(160, 0, 239, ILI9341_YELLOW);
  tft.drawFastHLine(0, 120, 319, ILI9341_YELLOW);
  tft.drawLine(80, 0, 240, 239, ILI9341_YELLOW);
  tft.drawLine(240, 0, 80, 239, ILI9341_YELLOW);
  tft.drawLine(0, 60, 319, 180, ILI9341_YELLOW);
  tft.drawLine(0, 180, 319, 60, ILI9341_YELLOW);
}

void loop() {
}
```

ZADATAK 4. Nacrtajte linije koje postepeno ispunjavaju ekran.



RJEŠENJE. Kako bi linijama popunili cijeli ekran koristit ćemo petlju:

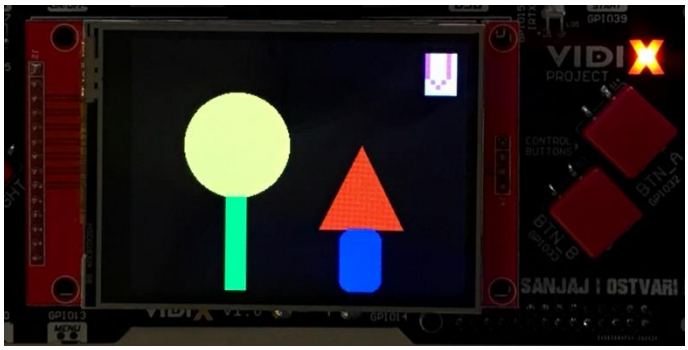
```
for (int brojac = 0; brojac <= granica; brojac = brojac + korak) {
  naredbe;
}
```

Petlji se zadaju početna i konačna vrijednost `brojača` te `korak` za koliko se povećava taj `brojač` u svakoj iteraciji. For petlja će služiti za automatsko računanje početne i završne točke svake linije. Koristimo dvije petlje, jednom za povezivanje gornje i donje strane ekrana, a drugi puta za povezivanje desne i lijeve strane ekrana.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                          // inicijalizacija zaslona

  tft.setRotation(3);                   // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona

  for (int i = 0; i <= 319; i = i + 5) {
    tft.drawLine(i, 0, 319-i, 239, ILI9341_YELLOW);
    delay(100);
  }
```

```
  for (int i = 0; i <= 239; i = i + 5) {
    tft.drawLine(319, i, 0, 239-i, ILI9341_YELLOW);
    delay(100);
  }
}

void loop() {
}
```

ZADATAK 5. Na TFT ekranu nacrtajte sliku što sličniju prikazanoj.



RJEŠENJE. Osim točke i linije, na ekran se mogu crtati i geometrijski likovi. Svakoga od njih je moguće prikazati sa ili bez ispune. Pravokutnik je definiran koordinatama gornjeg lijevog ugla (`x0, y0`), duljinama stranica u pikselima i bojom (ruba ili ispune):

```
tft.drawRect(x0, y0, širina, visina, boja);
tft.fillRect(x0, y0, širina, visina, boja);
```

Kružnica je zadana koordinatama svog središta, duljinom radijusa u pikselima i bojom:

```
tft.drawCircle(x0, y0, radijus, boja);
tft.fillCircle(x0, y0, radijus, boja);
```

Pravokutnik s oblim vrhovima je zadan koordinatama gornjeg lijevog ugla (`x0, y0`), duljinama stranica u pikselima, radijusom ugla (preporučljivo je da radijus bude neparan broj) i bojom:

```
tft.drawRoundRect(x0, y0, širina, visina, radijus, boja);
tft.fillRoundRect (x0, y0, širina, visina, radijus, boja);
```

Trokut je zadan koordinatama sva tri vrha (`x0, y0`), (`x1, y1`), (`x2, y2`) i bojom:

```
tft.drawTriangle(x0, y0, x1, y1, x2, y2, boja);
tft.fillTriangle(x0, y0, x1, y1, x2, y2, boja);
```

Ako se želi nacrtati jedan točno određeni znak, potrebno je zadati koordinate gornjeg lijevog ugla, upisati koji znak se želi prikazati, boja, boja pozadine i veličina. Znak je originalno velik 5x8 piksela, a pomoću parametra veličina ta dimenzija se može proporcionalno uvećati.

```
tft.drawChar(x0, y0, znak, boja, boja pozadine, veličina);
```

Kako bismo nacrtali zadanu sliku, program glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                      // inicijalizacija zaslona
```

```
  tft.setRotation(3);                // postavi orijentaciju
  tft.fillScreen(ILI9341_BLACK);     // boja zaslona

  tft.drawCircle(100, 100, 50, ILI9341_YELLOW);   // kruznica
  tft.fillCircle(100, 100, 50, ILI9341_YELLOW);   // ispuna kruznice

  tft.drawRect(90, 150, 20, 100, ILI9341_GREEN);  // pravokutnik
  tft.fillRect(90, 150, 20, 100, ILI9341_GREEN);  // ispuna pravokutnika

  tft.drawTriangle(220, 100, 180, 180, 260, 180, ILI9341_RED);  // trokut
  tft.fillTriangle(220, 100, 180, 180, 260, 180, ILI9341_RED);  // ispuna trokuta

  tft.drawRoundRect(200, 180, 40, 60, 11, ILI9341_BLUE);  // obli pravokutnik
  tft.fillRoundRect (200, 180, 40, 60, 11, ILI9341_BLUE); // ispuna

  tft.drawChar(280, 10, 'V', ILI9341_MAGENTA, ILI9341_WHITE, 5); // znak V
}

void loop() {
}
```

ZADATAK 6. Na TFT ekranu nacrtajte olimpijske krugove.

RJEŠENJE. Program glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup() {
  tft.begin();                       // inicijalizacija zaslona

  tft.setRotation(3);                // postavi orijentaciju
  tft.fillScreen(ILI9341_WHITE);     // boja zaslona

  tft.drawCircle(50, 100, 30, ILI9341_BLUE);
  tft.drawCircle(120, 100, 30, ILI9341_BLACK);
  tft.drawCircle(190, 100, 30, ILI9341_RED);
  tft.drawCircle(85, 130, 30, ILI9341_YELLOW);
  tft.drawCircle(155, 130, 30, ILI9341_GREEN);
}

void loop() {
}
```

ZADATAK 6. Napravite program koji će na ekran crtati kako se mijenja temperatura u vremenu.

RJEŠENJE. Dio programa za očitavanje temperature je isti kao u 5. zadatku prošlog poglavlja. Očitanu vrijednost ćemo ispisati u gornjem lijevom uglu ekrana, na koordinate `(10,70)`. Nova vrijednost će se ispisati svakim novim očitanjem. Kako se pri tome ne bi obrisao cijeli ekran, već samo prethodna vrijednost koristi se naredba

```
tft.fillRect(9, 69, 60, 16, ILI9341_BLACK);
```

koja će obojati pravokutnik točno na mjestu gdje se ispisuju očitane vrijednosti temperature i nigdje drugdje. Početne koordinate su `(9,69)`, širina po x-osi je `60` piksela i po y-osi `16` piksela.

Dodatno, u ovom poglavlju koristimo naredbe koje će crtati piksele u ovisnosti o trenutnoj vrijednosti temperature. X koordinata svakog piksela je zadana trenutkom u kojem je temperatura očitana. Upravo taj trenutak je zapisan u varijabli `j`. Svakim očitanjem vrijednosti temperature, varijabla y se poveća za 1: `j++`. Kada takvim povećavanje postane veća do najveće moguće vrijednosti `x`, tj. 240 i izađe iz dimenzija ekrana, ponovno se postavlja u nulu kako bi crtanje ponovno počelo na početku ekrana.

```
j++;
if (j > tft.width() + 1) {
   j = 0;
}
```

Pri crtanju novog piksela, treba obrisati onaj piksel koji je bio nacrtan na istoj `x` koordinati. Umjesto brisanja cijelog ekrana, obrisat ćemo samo pripadajuću vertikalnu liniju, tj. obojati ju u crno:

```
tft.drawFastVLine(j, 0, TFT.height(), ILI9341_BLACK);
```

Piksele crtamo pomoću naredbe:

```
tft.drawPixel(j, map(temp, 700, 800, 239, 119), ILI9341_RED);
```

Kako `map` pretvara cijele brojeve u cijele brojeve, koristit ćemo direktna očitanja senzora spremljena u varijabli `temp` i pretvarati ih u položaje na ekranu. Te vrijednosti crtamo na `y`-os. Kako je visina ekrana 240 piksela, a koordinate piksela mogu biti između 0 i 239, vrijednost temperature pretvaramo na brojeve u ovom rasponu. Prirodno je da je viša temperatura prikazana iznad niže temperature, zato su ciljane vrijednosti u naredbi `map` zadane obratnim redom: `239, 119`.

Jedan piksel je jako mali, pa kako bismo ga povećali, crtamo ga dva puta, jedan ispod drugog:

```
tft.drawPixel(j, map(temp, 700, 800, 238, 118), ILI9341_RED);
```

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int PinTemp = 26;
int temp;

float tempV;
float tempC;

int j = 0;

void setup() {
  pinMode(PinTemp, INPUT);

  Serial.begin(9600);                 // ispis i na Serijski monitor

  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);   // boja teksta
  tft.setTextSize(2);                 // velicina teksta
  tft.setRotation(3);                 // postavi orijentaciju
}

void loop() {
  temp = analogRead(PinTemp);
  tempV = map(temp, 0, 4095, 0, 3300);
  tempC = tempV / 10 - 40;

  Serial.println(tempC);               // ispis i na Serijski monitor

  tft.fillRect(9, 69, 60, 16, ILI9341_BLACK); // ispuna pravokutnika
                                              // umjesto brisanja ekrana
  tft.setCursor(10, 70);               // polozaj pocetka ispisa tekst
```

```
  tft.println(tempC);                  // ispis temperature

  // Varijabla j treba za kretanje po X-osi ekrana i crtanje piksela
  j++;
  if (j > tft.width() + 1) {  // Kada j izade iz ekrana
     j = 0;                   // postavi ga na nulu
  }

  // brisemo stari piksel kako bi mogli nacrtati novi na njegovom mjestu
  tft.drawFastVLine(j, 0, tft.height(), ILI9341_BLACK);

  // crtanje vrijednosti temperature
  tft.drawPixel(j, map(temp, 700, 800, 239, 119), ILI9341_RED);
  tft.drawPixel(j, map(temp, 700, 800, 238, 118), ILI9341_RED);


  delay(300);
}
```

LITERATURA.

GFX biblioteka: https://learn.adafruit.com/adafruit-gfx-graphics-library/overview

naprednije: https://vidi-x.org/radionice/vidi-project-x-70-vidi-x-mikroracunalo-u-ulozi-termostata-centralnog-grijanja/

8. Uređaj za crtanje

U ovom poglavlju ćemo upoznati korištenje senzora dodira ugrađenog u 2.8'' ILI9341 SPI TFT LCD ekran. Kada se ekran koristi za čitanje podataka, mikroprekidač VSPI_SC2 mora biti u položaju Game Use te se pripadajući utor pinskog proširenja ne može koristiti u druge svrhe. TOUCH_IRQ je i dalje u položaju Use Exp kako bi bilo moguće prebaciti program u VIDI-X. S obzirom da se koriste isti pinovi za čitanje i pisanje (GPIO18, GPIO19 i GPIO23), za vrijeme dok se čita, ispis na ekran se stavlja na čekanje. Ovi pinovi su povezani unutar VIDI-X.

Kako bi mogli koristiti ekran kao senzor dodira, moramo omogućiti komunikaciju sa SPI uređajima. SPI (engl. serial peripheral interface) je kratica za serijsko periferalno sučelje koje predstavlja protokol za sinkronizirani prijenos podataka pri komunikaciji s jednim ili više uređaja ili mikrokontrolera. Jedan uređaj je uvijek glavni, (master, obično mikrokontroler), a drugi je sluga (slave, dodatni uređaj). Da bi bilo moguće uspostaviti ovakvu vezu potrebna su bar tri pina:

- MISO (master in, slave out) – žica za slanje podataka sa sluge na mastera,
- MOSI (master out, slave in) – žica za slanje podataka sa mastera na slugu (na periferiju),
- SCK (serial clock) – vremenski pulsevi koje generira master i koji sinkroniziraju prijenos podataka.

Pri komunikaciji VIDI-X i TFT ekrana, neovisno radi li se o prikazu slike na ekranu, korištenju ekrana kao senzora dodira ili korištenju SD kartice, za SPI komunikaciju se koriste:

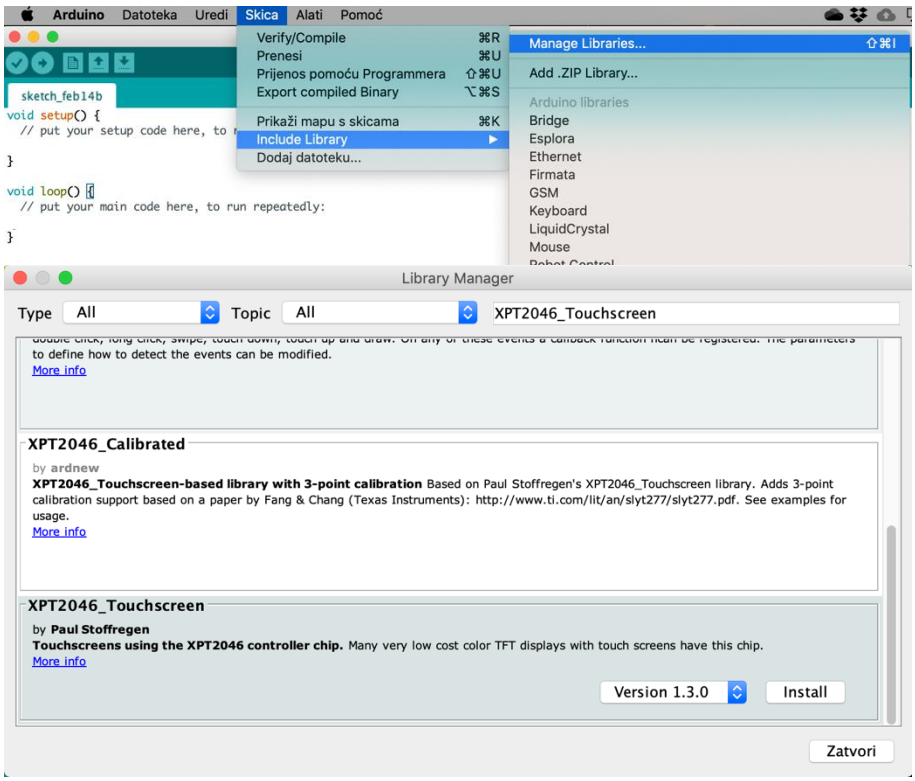
- MISO – GPIO19,
- MOSI – GPIO23,
- SCK – GPIO18.

SPI komunikacija se omogućuje uključenjem biblioteke koja dolazi s Arduino IDE i nije ju potrebno dodatno preuzimati s interneta:

```
#include <SPI.h>
```

Jedna od biblioteka koja omogućuje čitanje vrijednosti s ekrana kada se koristi kao senzor dodira je biblioteka XPT2046_Touchscreen. Nju je potrebno preuzeti s interneta odabirom Skica, Include Library, pa Manage Libraries unutar Arduino IDE sučelja. U novootvorenom prozoru se upiše XPT2046_Touchscreen, pronađe se među ponuđenima i pritisne Install. Za uključivanje ove biblioteke u program koristi se naredba:

```
#include <XPT2046_Touchscreen.h>
```



ZADATAK 1. Na serijskom monitoru ispišite je li TFT ekran pritisnut ili nije.

RJEŠENJE. Pin kojeg je potrebno definirati pri detekciji pritiska TFT ekrana je GPIO4 kojeg moramo definirati na samom početku programa:

```
#define TS_CS    4
```

Zatim stvaramo objekt ekrana `ts`. Pod tim imenom ćemo u nastavku programa prepoznavati naredbe i funkcije koje se odnose na rad s ekranom kao senzorom dodira.

```
XPT2046_Touchscreen ts(TS_CS);
```

Inicijalizaciju naredbi vezanih uz ekran kao senzor dodira obavlja funkcija `ts.begin();`
Potrebno je postaviti i orijentaciju ekrana u odnosu na korisnika `ts.setRotation(rotacija);`
Pri tome `rotacija` može biti broj između 0 i 3.

Očitanje je li ekran dodirnut ili nije, dobiva se naredbom:

```
ts.touched();
```

Rezultat očitanja može biti `1` (ako je ekran pritisnut) ili `0` (ako ekran nije pritisnut). Vrijednost očitanja se sprema u varijablu `pritisnut` i ovisno iznosi li ta vrijednost 0 ili 1 na serijski monitor se ispisuje poruka.

```
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

#define TS_CS   4 // deklaracija pina za senzor dodira

// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;                // varijabla za detekciju pritiska

void setup() {
  Serial.begin(38400);        // pokretanje serijskog monitora

  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {                      // ako je pritisnut
    Serial.println("Pritisnut je.");    // ispis na Serijski monitor
  } else {
    Serial.println("Nije pritisnut.");  // ispis na Serijski monitor
  }

  delay(500);
}
```

ZADATAK 2. Na serijskom monitoru ispisujte koordinate gdje je ekran pritisnut i jačinu pritiska.

RJEŠENJE. Očitanje koordinate gdje je ekran pritisnut i koliko je jako pritisnut je moguće dobiti na dva načina. Prvi način je pomoću očitanja objekta `TS_Point` koristeći naredbu `getPoint()`:

```
TS_Point p = ts.getPoint();
```

Vrijednosti koordinata i pritiska su spremljene u objekt `p`. Vrijednost koordinate `x` je moguće dohvatiti pomoću `p.x`, vrijednost koordinate `y` pomoću `p.y`, a vrijednost količine pritiska pomoću `p.z`. Upravo ove vrijednosti ispisujemo na serijski monitor.

```
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

#define TS_CS   4 // deklaracija pina za senzor dodira
```

```
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

void setup() {
  Serial.begin(38400);         // pokretanje serijskog monitora

  ts.begin();                  // inicijalizacija dodira
  ts.setRotation(1);           // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  TS_Point p = ts.getPoint();  // ocitavanje gdje je ekran pritisnut
  Serial.print("x = ");
  Serial.print(p.x);            // x koordinata
  Serial.print(", y = ");
  Serial.print(p.y);            // y koordinata
  Serial.print(", z = ");
  Serial.println(p.z);          // z koordinata = kolicina pritiska

  delay(500);
}
```

Prilikom testiranja primijetite kako `p.z` iznosi 0 ukoliko ekran nije pritisnut. Što je jači pritisak ta vrijednost raste, može iznositi između 0 i 4095, a najčešće iznosi oko `2000` uz normalan pritisak ekrana.
Koordinate `p.x` i `p.y` poprimaju vrijednost između `0` i `4095`, no vjerojatnije ćete prilikom rada dobivati vrijednosti između `300` i `3900` i ti brojevi se mogu razlikovati od uređaja do uređaja. Kako ih povezati s pravim vrijednostima piksela ekrana bit će objašnjeno uskoro. Primijetite i kako `p.x` i `p.y` zadržavaju svoje posljednje vrijednosti i kada je prestao dodir ekrana.

ZADATAK 3. Na serijskom monitoru ispisujte koordinate gdje je ekran pritisnut i jačinu pritiska ali samo ukoliko je ekran u tom trenu pritisnut.

RJEŠENJE. Program je kombinacija prethodna dva: ako je detektiran pritisak, provjerava se koja koordinata je pritisnuta i to se ispisuje na serijski monitor.

```
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

#define TS_CS   4 // deklaracija pina za senzor dodira

// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;                 // varijabla za detekciju pritiska

void setup() {
  Serial.begin(38400);         // pokretanje serijskog monitora

  ts.begin();                  // inicijalizacija dodira
  ts.setRotation(1);           // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();       // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();  // ocitavanje gdje je ekran pritisnut
    Serial.print("x = ");
    Serial.print(p.x);            // x koordinata
    Serial.print(", y = ");
    Serial.print(p.y);            // y koordinata
    Serial.print(", z = ");
    Serial.println(p.z);          // z koordinata = kolicina pritiska
  }
  delay(500);
}
```

ZADATAK 4. Kalibrirajte ekran. Neka gornji lijevi položaj ima koordinate 0,0, a donji desni položaj 320, 240.

RJEŠENJE. Prvi korak je pokrenuti program iz prethodnog zadatka i otkriti najmanju i najveću vrijednost koju poprimaju `p.x` i `p.y`. Minimalne vrijednosti možemo dobiti pritiskom uz sam gornji lijevi vrh ekrana, a maksimalne vrijednosti pritiskom uz sam donji desni vrh ekrana. Dobivene vrijednosti mogu biti:

min_x = 370
min_y = 210
max_x = 3900
max_y = 3800

Samo pretvaranje vrijednosti se računa pomoću naredbe `map` na isti način kako se naredba koristila i pri preračunavanju temperatura. Naredbi se zadaju trenutno očitanje, najmanja i najveća vrijednost očitanja (`min_x, max_x` ili `min_y, max_y`) te najmanja i najveća ciljana vrijednost. Najmanja ciljana vrijednost je u slučaju obje koordinate jednaka 0, a najveća ciljana vrijednost je 319 u slučaju `x` koordinate i 239 u slučaju `y` koordinate.

```
x_piksel = map(p.x, min_x, max_x, 0, 319);
y_piksel = map(p.y, min_y, max_y, 0, 239);
```

Na ovaj način će iznosi koordinata pritiska biti jednake rednom broju piksela kada ćemo nešto pisati ili crtati na ekran.

```
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

#define TS_CS   4 // deklaracija pina za senzor dodira

// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

void setup() {
  Serial.begin(38400);      // pokretanje serijskog monitora

  ts.begin();               // inicijalizacija dodira
  ts.setRotation(1);        // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();  // ocitavanje gdje je ekran pritisnut
    Serial.print("x = ");
    Serial.print(p.x);           // x koordinata
    Serial.print(", y = ");
    Serial.print(p.y);           // y koordinata
    Serial.print(", z = ");
    Serial.println(p.z);         // z koordinata = kolicina pritiska

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    Serial.print("x_piksel = ");
    Serial.print(x_piksel);             // x koordinata u vrijednostima piksela
    Serial.print(", y_piksel = ");
```

```
    Serial.println(y_piksel);           // y koordinata u vrijednostima piksela
  }
  delay(500);
}
```

ZADATAK 5. Na TFT ekranu ispišite poruku „Pritisnut je.“ ukoliko je ekran bilo gdje pritisnut, odnosno „Nije pritisnut.“ ukoliko ekran nije pritisnut.

RJEŠENJE. Zadatak je sličan prvom zadatku, samo što se sada poruka o pritisku ispisuje na TFT ekranu. U program je potrebno uključiti i biblioteke za ispis na ekran i za senzor dodira ekrana.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;                // varijabla za detekciju pritiska

void setup() {
  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setRotation(3);                 // postavi orijentaciju

  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    tft.fillScreen(ILI9341_BLACK);
    tft.setCursor(10, 70);              // polozaj pocetka ispisa tekst
    tft.println("Pritisnut je.");       // ispis na ekran
  } else {
    tft.fillScreen(ILI9341_BLACK);
    tft.setCursor(10, 70);              // polozaj pocetka ispisa tekst
    tft.println ("Nije pritisnut.");    // ispis na ekran
  }

  delay(500);
}
```

ZADATAK 6. Na TFT ekranu ispišite koordinate gdje se dogodio pritisak i iznos tog pritiska.

RJEŠENJE. Zadatak je sličan četvrtom zadatku, samo što se sada koordinate ispisuju na TFT ekran. U program je potrebno uključiti i biblioteke za ispis na ekran i za senzor dodira ekrana.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

```
// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

void setup() {
  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setRotation(3);                 // postavi orijentaciju

  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();         // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    tft.fillScreen(ILI9341_BLACK);      // brisanje ekrana
    tft.setCursor(10, 70);              // polozaj pocetka ispisa tekst
    tft.println((String)"x = " + x_piksel + ", y = " + y_piksel + ", z = " + p.z);
  }
  delay(500);
}
```

ZADATAK 7. Na mjestu gdje je pritisnut ekran nacrtajte krug. Kada prst dotakne drugi dio ekrana, prethodni krug se briše.

RJEŠENJE. Jedina razlika u odnosu na prošli zadatak je da se sada crta žuti krug na mjestu pritiska umjesto ispisivanja koordinata pritiska.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima
```

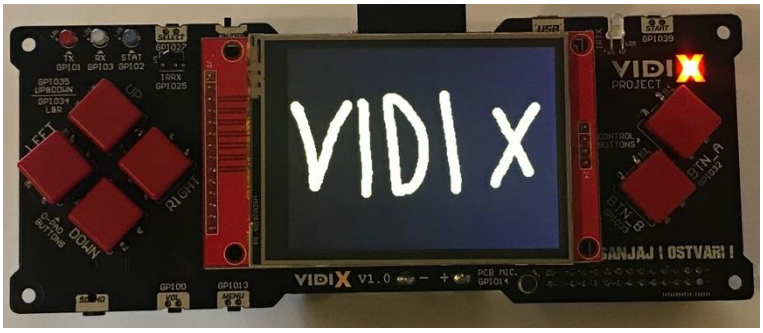
```
void setup() {
  tft.begin();                          // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona
  tft.setRotation(3);                   // postavi orijentaciju

  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();               // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();           // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    tft.fillScreen(ILI9341_BLACK);        // brisanje ekrana
    tft.fillCircle(x_piksel, y_piksel, 30, ILI9341_YELLOW);
  }
  delay(10);
}
```

ZADATAK 8. Crtajte prstom. Kako prst mičete po ekranu tako neka ostaje tanki žuti trag.

RJEŠENJE. Jedina razlika u odnosu na prošli zadatak je da se ne briše ekran između svaka dva pritiska prstom i radijus kruga je nešto manji. Možete li napisati svoje ime?

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

void setup() {
  tft.begin();                          // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona
  tft.setRotation(3);                   // postavi orijentaciju

  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();               // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();           // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
```

```
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    tft.fillCircle(x_piksel, y_piksel, 5, ILI9341_YELLOW);
  }
  delay(10);
}
```



ZADATAK 9. Crtajte prstom. Kako prst mičete po ekranu tako neka ostaje tanki trag. Pritiskom na tipku BTN_A boja kojom se crta neka bude žuta, a pritiskom na BTN_B boja neka bude crvena.

RJEŠENJE. U program moramo dodati naredbe za očitavanje jesu li tipkala pritisnuta i koje od njih je pritisnuto. Mikroprekidači BTN_A i BTN_B moraju biti u Game Use položaju. Nakon provjere koja je tipka pritisnuta, u varijablu boja se pohranjuje naziv boje koja će se koristiti.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int PinTipkalo_A = 32;      // tipkalo A
int PinTipkalo_B = 33;      // tipkalo B

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

uint16_t boja;

void setup() {
  pinMode(PinTipkalo_A, INPUT_PULLUP);
  pinMode(PinTipkalo_B, INPUT_PULLUP);

  tft.begin();                      // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);    // boja zaslona
  tft.setRotation(3);               // postavi orijentaciju

  ts.begin();               // inicijalizacija dodira
  ts.setRotation(1);        // postavi orijentaciju ekrana za detekciju dodira
}
```

```
void loop() {
  // provjera koje tipkalo je pritisnuto, kako se bi se odabrala boja
  if (digitalRead(PinTipkalo_A) == LOW) {
    boja = ILI9341_YELLOW;
  } else if (digitalRead(PinTipkalo_B) == LOW) {
    boja = ILI9341_RED;
  }

  pritisnut = ts.touched();              // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();          // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    tft.fillCircle(x_piksel, y_piksel, 5, boja);
  }
  delay(10);
}
```

Ako želimo pri crtanju koristiti veći raspon boja, boje je moguće zadavati i pomoću količina svake od komponenata: crvene (red), zelene (green) i plave (blue). Svaka od ovih boja može poprimiti cjelobrojne vrijednosti između 0 i 255. Naredba za pretvaranje ovih vrijednosti u oblik koji prihvaća TFT ekran je:

```
tft.color565(boja_crvena, boja_zelena, boja_plava)
```

Isti ovaj program, ali koristeći ručno definiranje boja glasi:

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int PinTipkalo_A = 32;      // tipkalo A
int PinTipkalo_B = 33;      // tipkalo B

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

int bojaR;
int bojaG;
int bojaB;

void setup() {
  pinMode(PinTipkalo_A, INPUT_PULLUP);
  pinMode(PinTipkalo_B, INPUT_PULLUP);

  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setRotation(3);                 // postavi orijentaciju
```

```
  ts.begin();                 // inicijalizacija dodira
  ts.setRotation(1);          // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  // provjera koje tipkalo je pritisnuto, kako se bi se odabrala boja
  if (digitalRead(PinTipkalo_A) == LOW) {
    bojaR = 255;
    bojaG = 255;
    bojaB = 0;
  } else if (digitalRead(PinTipkalo_B) == LOW) {
    bojaR = 255;
    bojaG = 0;
    bojaB = 0;
  }

  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();         // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    tft.fillCircle(x_piksel, y_piksel, 5, tft.color565(bojaR, bojaG, bojaB));
  }
  delay(10);
}
```

ZADATAK 10. Crtajte prstom. Kako prst mičete po ekranu tako neka ostaje tanki trag. S lijeve strane ekrana neka se nalazi paleta boja. Pritiskom na željenu boju odabirete boju kojom ćete crtati po ekranu.

RJEŠENJE. Unutar `void setup()` crtamo četiri kvadrata koji će predstavljati paletu boja. Svaki će biti duljine stranice 60 piksela. Unutar `void loop()` provjeravamo je li ekran pritisnut na prostoru ta četiri kvadrata. I ukoliko je, definira se boja s kojom će se crtati. Ukoliko se ekran pritisne desno od 60-tog piksela po x-osi, to je prostor za crtanje i tamo će se prikazati slika u izabranoj boji.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

uint16_t boja;

void setup() {
  tft.begin();                      // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);    // boja zaslona
  tft.setRotation(3);                       // postavi orijentaciju
```

```
  tft.fillRect(0, 0, 60, 60, ILI9341_YELLOW);  // crtanje izbornika boja
  tft.fillRect(0, 60, 60, 60, ILI9341_GREEN);
  tft.fillRect(0, 120, 60, 60, ILI9341_BLUE);
  tft.fillRect(0, 180, 60, 60, ILI9341_RED);

  ts.begin();               // inicijalizacija dodira
  ts.setRotation(1);        // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();         // ocitavanje gdje je ekran pritisnut

    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    if (x_piksel<61) {                  // provjera gdje je pritisnut ekran,
      if (y_piksel<61) {                // ovisno o tome se izabire boja
        boja = ILI9341_YELLOW;          // s kojom se ce crtati
      } else if (y_piksel<121) {
        boja = ILI9341_GREEN;
      } else if (y_piksel<181) {
        boja = ILI9341_BLUE;
      } else if (y_piksel<239) {
        boja = ILI9341_RED;
      }
    }

    // ako je ekran pritisnut desno od 60-tog piksela - to je prostor za crtanje
    if (x_piksel>60) {
      tft.fillCircle(x_piksel, y_piksel, 5, boja);
    }

  }
  delay(10);
}
```

ZADATAK 11. U program iz prošlog zadatka dodajte da se ekran briše pritiskom na tipkalo START.

RJEŠENJE. Potrebno je dodati naredbe za definiciju pina tipkala START (GPIO39) i kako se radi o ulaznom pinu. U `void loop()` se provjerava je li tipkalo pritisnuto i ako je, crta se crni pravokutnik u prostoru desno od `x = 60`, kako se izbornik boja ne bi obrisao. Provjera je li tipkalo pritisnuto se odvija neovisno o provjeri postoji li pritisak na ekran.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21
// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

```
// deklaracija pina za senzor dodira
#define TS_CS   4
// stvaranje objekta za senzor dodira koji će se zvati ts
XPT2046_Touchscreen ts(TS_CS);

int PinTipkalo = 39;        // tipkalo START

int pritisnut;              // varijabla za detekciju pritiska

int min_x = 370;            // minimalne i maksimalne vrijednosti koordinata dodira
int min_y = 210;
int max_x = 3900;
int max_y = 3800;

int x_piksel, y_piksel;     // varijable gdje se biti koordinate dodira u pikselima

uint16_t boja;

void setup() {
  pinMode(PinTipkalo, INPUT_PULLUP);

  tft.begin();                        // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);      // boja zaslona
  tft.setRotation(3);                           // postavi orijentaciju

  tft.fillRect(0, 0, 60, 60, ILI9341_YELLOW);  // crtanje izbornika boja
  tft.fillRect(0, 60, 60, 60, ILI9341_GREEN);
  tft.fillRect(0, 120, 60, 60, ILI9341_BLUE);
  tft.fillRect(0, 180, 60, 60, ILI9341_RED);

  ts.begin();               // inicijalizacija dodira
  ts.setRotation(1);        // postavi orijentaciju ekrana za detekciju dodira
}

void loop() {
  pritisnut = ts.touched();             // provjeri je li ekran pritisnut
  if (pritisnut) {
    TS_Point p = ts.getPoint();         // ocitavanje gdje je ekran pritisnut
    x_piksel = map(p.x, min_x, max_x, 0, 319);  // pretvaranje ocitanja u piksele
    y_piksel = map(p.y, min_y, max_y, 0, 239);

    if (x_piksel<61) {                  // provjera gdje je pritisnut ekran,
      if (y_piksel<61) {                // ovisno o tome se izabire boja
        boja = ILI9341_YELLOW;          // s kojom se ce crtati
      } else if (y_piksel<121) {
        boja = ILI9341_GREEN;
      } else if (y_piksel<181) {
        boja = ILI9341_BLUE;
      } else if (y_piksel<239) {
        boja = ILI9341_RED;
      }
    }

    // ako je ekran pritisnut desno od 60-tog piksela - to je prostor za crtanje
    if (x_piksel>60) {
      tft.fillCircle(x_piksel, y_piksel, 5, boja);
    }
  }

  // brisanje ekrana ako je pritisnuta tipka START
  if (digitalRead(PinTipkalo) == LOW) {
    tft.fillRect(60, 0, 260, 240, ILI9341_BLACK);
  }
  delay(10);
}
```

LITERATURA.

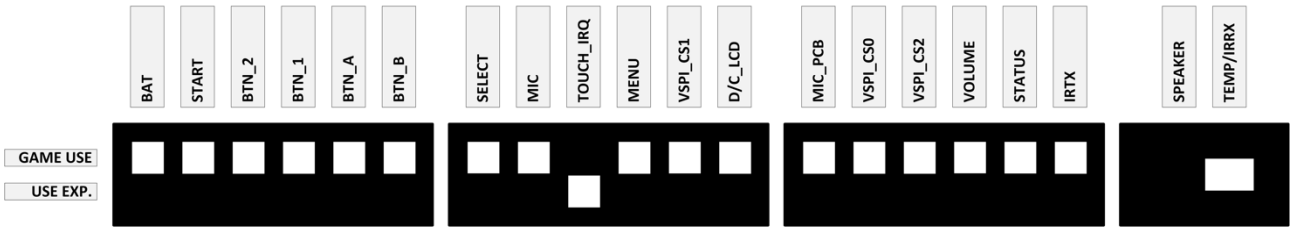
https://www.arduino.cc/en/reference/SPI
https://github.com/PaulStoffregen/XPT2046_Touchscreen

9. Infracrveni predajnik i prijamnik

Infracrveno zračenje se u spektru nalazi na frekvencijama nižima od vidljivog spektra (300 GHz – 400 THz), odnosno valnim duljinama većima od vidljivog spektra (750 nm – 1 mm) te ovu svjetlost ne možemo vidjeti i ne može proći kroz zidove i prepreke. Ipak, možemo ju koristiti za komunikaciju ukoliko su predajnik i prijamnik na direktnoj vidljivosti.

Infracrveni predajnik (IRTX) izgleda kao obična LED, no ne odašilje vidljivu svjetlost, već infracrvenu (npr. prema televizoru ili drugom VIDI-X uređaju). Spojen je na GPIO15. Pri korištenju infracrvenog predajnika, mikroprekidač IRTX mora biti u položaju Game Use (pri tome je 7. utor pinskog proširenja zauzet ovim infracrvenim predajnikom i ne može se koristiti u druge svrhe).

Infracrveni prijamnik (IRRX) je zapravo fotodioda koja pretvara infracrveno svjetlo u električni signal (npr. s daljinskog upravljača ili drugog VIDI-X uređaja). Spojen na GPIO25. Infracrveni prijemnik nema pripadajući mikroprekidač, ali Prekidač mora biti u položaju TEMP/IRRX.



Pomoću IRTX i IRRX možemo kontrolirati robote, udaljene senzore, pratiti otkucaje srca, upravljati televizorima, klima uređajima i mnogim drugim uređajima u pametnoj kući.

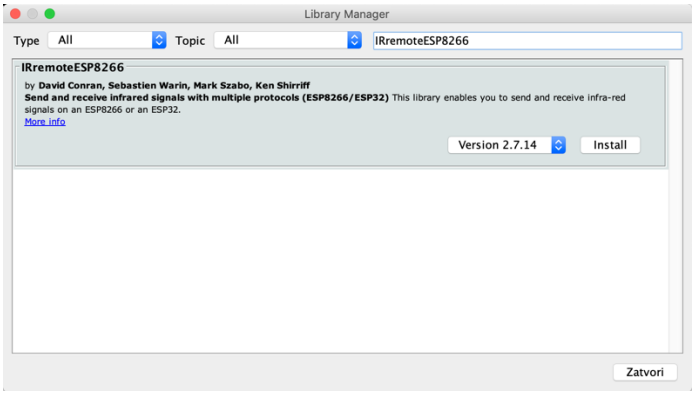
Sunce, žarulje i drugi uređaji koji griju emitiraju infracrveno zračenje te se ono nalazi svuda oko nas. Ukoliko želimo koristiti točno određene zrake koje emitira IRTX, moramo ih modulirati. Modulirani signal se sastoji od niza IR svjetlosnih pulseva na visokoj frekvenciji koja je rijetka u prirodnim signalima te ih zbog toga IRRX može dekodirati i pretvoriti u niz nula i jedinica koje VIDI-X može prepoznati. Način na koji se signal kodira i dekodira definiran je protokolima: Sony, NEC, RC5,… Glavna razlika među njima je u duljini trajanja pulseva i vremenu između pulseva. Upravo ova vremena definiraju radi li se o nuli i jedinici.

Svaki puta kada se pritisne tipka na nekom daljinskom upravljaču, generira se jedinstveni heksadekadski kod (koji se zapravo sastoji od nula i jedinica). Taj kod se kodira (modulira) i šalje putem IRTX prema drugom uređaju i IRRX koji će ga dekodirati. Kako bi taj uređaj znao dekodirati primljeni signal, mora poznavati koji kod pripada kojoj tipki daljinskog upravljača. Svaki proizvođač, svaki daljinski i svaka njegova tipka imaju svoje jedinstvene kodove.

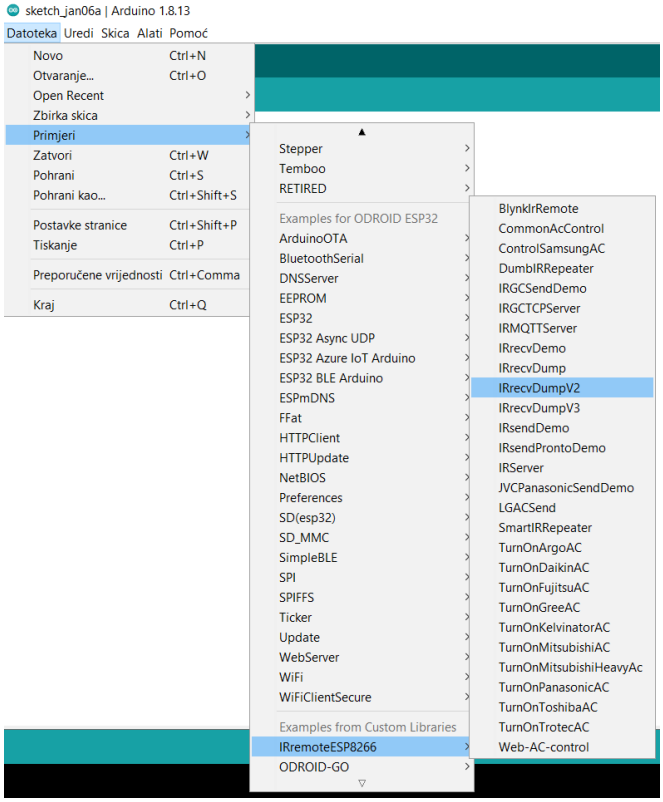
Za ovu vježbu potreban je obični daljinski upravljač, od bilo kojeg televizor ili linije.

ZADATAK 1. Koristeći infracrveni prijamnik očitajte kodove i osnovne informacije nekoliko tipki daljinskog upravljača. Neka se kodovi ispišu na serijskom monitoru.

RJEŠENJE. Naredbe za infracrvenu komunikaciju nisu sastavni dio Arduino IDE te ih je potrebno dodatno preuzeti s Interneta i instalirati. Koristit ćemo IRremoteESP8266 biblioteku koju su razvili David Conran, Sebastien Warin, Mark Szabo i Ken Shirriff.
Za preuzimanje i instalaciju potrebno je unutar Arduino IDE u izborniku *Skica* izabrati *Include Library*, pa *Manage Libraries*. U pretraživač unesite naziv biblioteke IRremoteESP8266. Pritiskom na Install instalirajte najnoviju verziju.



Najbrži način dobivanja kodova daljinskog upravljača je pomoću demo programa IRrecvDumpV2 koji dolazi s ovom bibliotekom. Nalazi se u izborniku Datoteka, Primjeri, IRremoteESP8266.



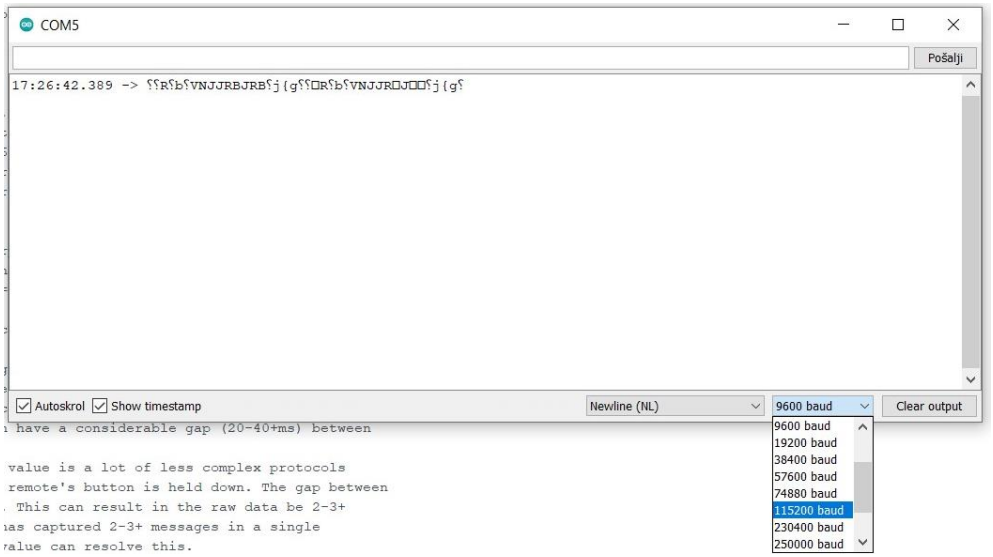
Kod ima detaljne komentare za svaku od korištenih varijabli i naredbi. Prije pokretanja programa potrebno je definirati pin na koji je spojen infracrveni prijemnik. Definicija pina je prva konstanta na samom početku programa nakon najave svih korištenih biblioteka. Pin koji koristimo je GPIO25, zato naredba treba glasiti:

```
const uint16_t kRecvPin = 25;
```

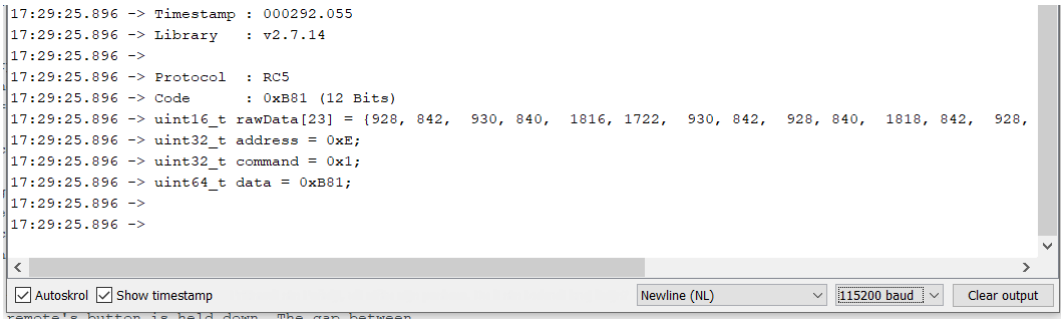
Nakon prebacivanja programa u VIDI-X potrebno je otvoriti serijski monitor. Ukoliko se prikažu čudni simboli, brzina prijenosa podataka nije odgovarajuća. Kako je u demo programu zadana s:

```
const uint32_t kBaudRate = 115200;
```

istu brzinu treba odabrati i unutar serijskog monitora.



Na slici su prikazani podaci dobiveni pritiskom na tipku 1 korištenog daljinskog upravljača. Među prikazanim podacima su između ostalih naziv korištenog protokola (ovdje RC5) i 12-bitni kod pritisnute tipke: 0xB81. Ovaj kod je zapisan u heksadekadskom obliku.



ZADATAK 2. Koristeći infracrveni prijamnik očitajte samo heksadekadskom kodove nekoliko tipki daljinskog upravljača. Neka se kodovi ispišu na serijskom monitoru.

RJEŠENJE. Ako želimo koristiti programski koristiti očitane kodove za upravljanje aktuatorima na VIDI-X, moramo pristupiti željenom kodu pojedine tipke, odnosno dobivenu vrijednost pospremiti u varijablu.

Na početku programa moramo najaviti korištenje biblioteka za rad s IR protokolima:

```
#include <IRremoteESP8266.h>
#include <IRac.h>
```

Zatim slijedi definicija svih potrebnih konstanti, varijabli i klasa. Pojedina značenja dana su u samom kodu. Zanimljiva varijabla koju treba definirati je `kTimeout`. Ona predstavlja vrijeme za hvatanje poruke i mjeri se u ms. Veći broj omogućuje hvatanje dulje poruke. No, ukoliko je poruka kraća, za vrijeme jednog pritiska željene tipke postoji mogućnost da se više puta pročita ista poruka. Obično daljinski za klima uređaje imaju dulje poruke, pa i ova varijabla treba imati veću vrijednost. Maksimalni iznos je 130 ms nakon kojeg se gotovo sve poruke ponavljaju. Pokrenite program više puta i nađite optimalnu vrijednost ove varijable za vaš daljinski upravljač.

Naredba koja omogućuje primanje infracrvenog signala glasi:

```
irrecv.enableIRIn();
```

Za provjeru je li kod primljen koristi se `irrecv.decode(&rezultat)`. Ova naredba vraća `True`, ako je kôd primljen.

Primljeni signal (kôd) je pohranjen u `rezultat`. Ako nas zanima samo kôd pojedine tipke, on se može očitati pomoću: `rezultat.value`.

Za ispis na serijski monitor ne možemo koristiti standardni `Serial.print()` jer ne barata heksadekadskim brojevima. Za ispis heksadekadske vrijednosti koristimo naredbu:

```
serialPrintUint64(varijabla, HEX);
```

Ako želimo ispisati više primljenih informacija, mogu se koristiti i naredbe:

```
Serial.println(resultToSourceCode(&rezultat));
Serial.println(resultToHumanReadableBasic(&rezultat));
```

Na kraju `loop` funkcije dodaje se naredba za resetiranje prijemnika i priprema ga se za očitavanje sljedeće vrijednosti:

```
irrecv.resume();
```

```
#include <IRremoteESP8266.h>
#include <IRac.h>

const uint16_t kRecvPin = 25;        // pin IRRX
const uint32_t kBaudRate = 115200;  // brzina prijenosa podataka prema serijskom monitoru
const uint16_t kCaptureBufferSize = 1024; // veličina buffera koji hvata kôd
const uint8_t kTimeout = 100;        // vrijeme za hvatanje poruke
uint64_t kod;  // varijabla u koju ćemo zapisati pročitanu vrijednost pritisnute tipke

IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);  // kreiranje instance klase,
// tj. kreiranje objekta irrecv klase IRrecv
decode_results rezultat; // kreiranje objekta results klase decode_results - dekodirani
// rezultati će se pohraniti u „results“

void setup() {
  Serial.begin(kBaudRate);        // pokretanje serijske veze
  irrecv.enableIRIn();            // omogućuje primanje infracrvenog signala
}

void loop() {
  if (irrecv.decode(&results)) { // provjera da li je IR kôd primljen
    kod = rezultat.value;         // pročitana vrijednost kôda pritisnute tipke
    serialPrintUint64(kod, HEX); // ispis na serijski monitor heksadekadske vrijednosti
    irrecv.resume();              // očitaj sljedeću vrijednost
  }
}
```

Ako se pojedina tipka daljinskog upravljača drži predugo pritisnuta, doći će do ponovnog očitavanja prethodne tipke i moguće je da će se umjesto ispravnog kôda ispisati: `0xFFFFFFFF`.
Isto tako, moguće je da jedna tipka ima više različitih kôdova. Pri hvatanju kôdova, svaku tipku pritisnite nekoliko puta kako biste se uvjerili da ste detektirali sve moguće kôdove.

ZADATAK 3. Prepoznajte koja tipka daljinskog je pritisnuta. Na serijski monitor ispišite naziv tipke.

RJEŠENJE. Nakon što detektiramo sve kodove za svaku tipku i zapamtimo ih, ispis koja je tipka pritisnuta dobivamo jednostavnom usporedbom vrijednosti. Ono što jedino treba imati na umu je da su kodovi heksadekadski i da svaki započinje s `0x`. Primjer programa za ispisa broja 1 i broja 2 za daljinski upravljač iz prvog zadatka glasi:

```
#include <IRremoteESP8266.h>
#include <IRac.h>

const uint16_t kRecvPin = 25;
const uint32_t kBaudRate = 115200;
const uint16_t kCaptureBufferSize = 1024;
const uint8_t kTimeout = 100;
uint64_t kod;
```

```
IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);
decode_results rezultat;

void setup() {
  Serial.begin(kBaudRate);
  irrecv.enableIRIn();
}

void loop() {
  if (irrecv.decode(&rezultat)) {
    kod = rezultat.value;
    if (kod == 0x381 or kod == 0xB81){
      Serial.println("1");
    }
    if (kod == 0x382 or kod == 0xB82){
      Serial.println("2");
    }
    irrecv.resume();
  }
}
```

Upotpunite program kako bi ispisao vrijednosti više tipki s vašeg daljinskog. Ovdje napisane vrijednosti zamijenite s vrijednostima koje je očitao vaš VIDI-X za vaš daljinski.

Kako u ovom programu imamo mnogo uvjeta koje želimo provjeriti, umjesto `if` naredbe možemo upotrijebiti `switch-case`. Ova naredba uspoređuje vrijednosti varijable s više različitih vrijednosti definiranima u svakom `case` slučaju. Kada se pronađe koji slučaj je zadovoljen, izvode se pripadajuće naredbe. Svaki slučaj završava s `break` kako bi se nakon izvršavanja naredbi definiranih za taj slučaj prekinulo daljnje testiranje preostalih slučajeva. Posljednji slučaj `default` će se izvršiti ukoliko niti jedan od slučajeva nije zadovoljen. Sintaksa naredbe je:

```
switch (varijabla) {
    case vrijednost1:
          // naredbe1
    break;
    case vrijednost2:
          // naredbe2
    break;
    default:
          // naredbe3
    break;
}
```

Program za detekciju tipki s daljinskog upravljača glasi:

```
#include <IRremoteESP8266.h>
#include <IRac.h>

const uint16_t kRecvPin = 25;
const uint32_t kBaudRate = 115200;
const uint16_t kCaptureBufferSize = 1024;
const uint8_t kTimeout = 100;

uint64_t kod;
IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);
decode_results rezultat;

void setup() {
  Serial.begin(kBaudRate);
  irrecv.enableIRIn();
}

void loop() {
  if (irrecv.decode(&rezultat)) {
    kod = rezultat.value;

    switch (kod) {
        case 0x381:
```

```
            Serial.println("1");
            break;
        case 0xB81:
            Serial.println("1");
            break;
        case 0x382:
            Serial.println("2");
            break;
        case 0xB82:
            Serial.println("2");
            break;
        default:
            Serial.println("Ne znam.");
            break;
    }
    irrecv.resume();
  }
}
```

ZADATAK 4. Napravite program koji će upravljati LED koristeći daljinski upravljač: kada se na daljinskom pritisne tipka 1, neka se upali crvena LED, kada se pritisne tipka 2, neka se upali zelena LED.

RJEŠENJE. Crvenu LED ćemo spojiti na GPIO14 (19. utor), zelenu na GPIO13 (17. utor) i prebaciti pripadajuće mikroprekidače MIC i MENU u položaj Use Exp.
Sam program je vrlo sličan onima iz prethodnog zadatka, samo što se umjesto ispisa na serijski monitor uključuju LED. Zadatak je moguće riješiti pomoću `if` naredbi:

```
#include <IRremoteESP8266.h>
#include <IRac.h>

const uint16_t kRecvPin = 25;
const uint16_t kCaptureBufferSize = 1024;
const uint8_t kTimeout = 100;
uint64_t kod;
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena

IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);
decode_results rezultat;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
  irrecv.enableIRIn();
}

void loop() {
  if (irrecv.decode(&rezultat)) {
    kod = rezultat.value;
    if (kod == 0x381 or kod == 0xB81){
        digitalWrite(PinLed1, LOW);
        digitalWrite(PinLed2, HIGH);
    }
    if (kod == 0x382 or kod == 0xB82){
        digitalWrite(PinLed1, HIGH);
        digitalWrite(PinLed2, LOW);
    }
    irrecv.resume();
  }
}
```

ili pomoću `switch` naredbe:

```
#include <IRremoteESP8266.h>
#include <IRac.h>
```

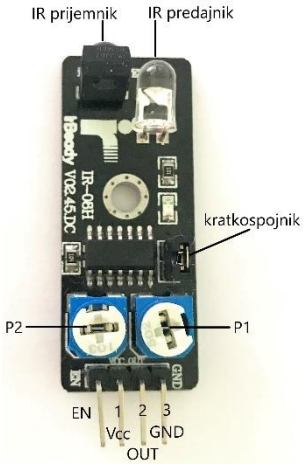
```
const uint16_t kRecvPin = 25;
const uint16_t kCaptureBufferSize = 1024;
const uint8_t kTimeout = 100;
uint64_t kod;
int PinLed1 = 13; // zelena
int PinLed2 = 14; // crvena

IRrecv irrecv(kRecvPin, kCaptureBufferSize, kTimeout, true);
decode_results rezultat;

void setup() {
  pinMode(PinLed1, OUTPUT);
  pinMode(PinLed2, OUTPUT);
  digitalWrite(PinLed1, LOW);
  digitalWrite(PinLed2, LOW);
  irrecv.enableIRIn();
}

void loop() {
  if (irrecv.decode(&rezultat)) {
    kod = rezultat.value;

    switch (kod) {
        case 0x381:
            digitalWrite(PinLed1, LOW);
            digitalWrite(PinLed2, HIGH);
            break;
        case 0xB81:
            digitalWrite(PinLed1, LOW);
            digitalWrite(PinLed2, HIGH);
            break;
        case 0x382:
            digitalWrite(PinLed1, HIGH);
            digitalWrite(PinLed2, LOW);
            break;
        case 0xB82:
            digitalWrite(PinLed1, HIGH);
            digitalWrite(PinLed2, LOW);
            break;
        default:
            digitalWrite(PinLed1, LOW);
            digitalWrite(PinLed2, LOW);
            break;
    }
    irrecv.resume();
  }
}
```

PROJEKT. Na TFT ispišite koja tipka daljinskog upravljača je pritisnuta.

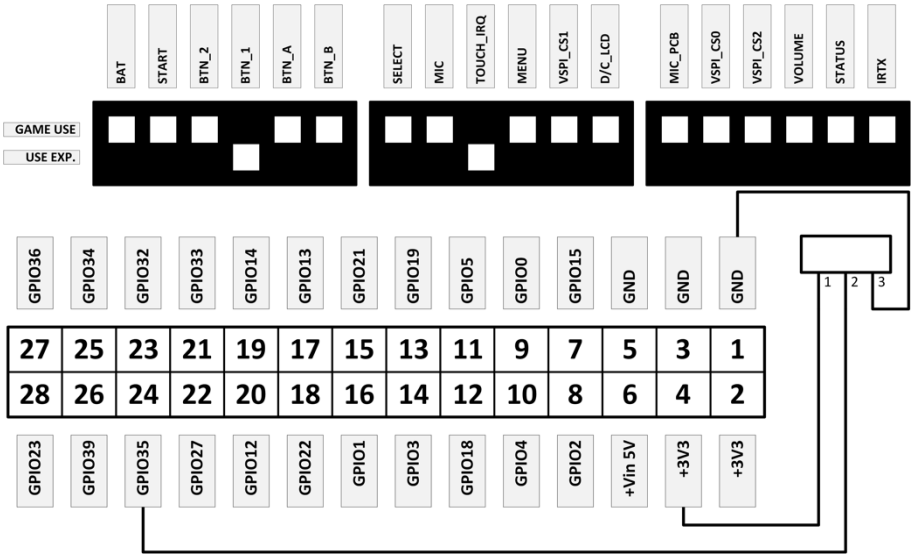
ZADATAK 5. Na VIDI-X spojite dodatni IR prijemnik/predajnik. Pomoću njega očitajte nalazi li se ispred senzora neka prepreka. Neka ugrađena LED zasvijetli kada je prepreka u blizini.

RJEŠENJE. Infracrveni predajnik i prijemnik se mogu koristiti kao detektori prepreke. Predajnik šalje IR zraku. Ukoliko se ispred senzora nalazi prepreka, zraka će se odbiti te će ju prijamnik detektirati. Kada nema prepreke ili je prepreka crna i upija zrake, neće doći do refleksije i senzor neće detektirati prepreku. Iz tog razloga je ovaj senzor moguće koristiti i kao detektor crne / bijele boje.
Senzor može imati 3 ili 4 pina: uzemljenje (GND), napajanje (Vcc, 3.3V do 5V), signal (OUT) te EN (enable, samo ukoliko senzor ima 4 pina). Redoslijed pinova može varirati u ovisnosti o proizvođaču senzora, pa pažljivo provjerite prije spajanja. GND spajamo na GND (utor 1), napajanje na +3.3V (utor 4), a signal na GPIO35 (utor 24). EN u ovom primjeru nećemo koristiti, no u tom slučaju kratkospojnik (jumper) mora biti utaknut.
Senzor ima i dva potenciometra. Moguće ih je zakrenuti koristeći plosnati odvijač. Pomoću potenciometra P1 zadajemo osjetljivost senzora, tj. određujemo udaljenost na koju želimo da senzor reagira. Potenciometar P2

u principu ne bi trebalo mijenjati. On kontrolira frekvenciju IR signala, što bi unaprijed trebalo biti podešeno. No, ovaj potenciometar je moguće koristiti ukoliko dolazi do interferencije s drugim IR predajnicima.



Infracrveni senzor
pin EN nije potrebno spojiti
pin VCC na +3V3
pin OUT na npr. GPIO35
pin GND na GND



Sam program je gotovo identičan programu koji detektira pritisak tipkala: direktno čitamo stanje na pinu GPIO35 i provjeravamo je li HIGH (nema prepreke) ili LOW (ima prepreke). Na temelju očitanja palimo ili gasimo LED i ispisujemo poruku na serijskom monitoru.

```
int PinLed = 2;
int PinIR = 35;
int StanjeIR;

void setup() {
  pinMode(PinLed, OUTPUT);
  pinMode(PinIR, INPUT);
  Serial.begin(9600);
}

void loop() {
  StanjeIR = digitalRead(PinIR);
  if (StanjeIR == LOW) {
    Serial.println("Prepreka");
    digitalWrite(PinLed, HIGH);
  } else {
    Serial.println("Nema prepreke");
    digitalWrite(PinLed, LOW);
  }
  delay(200);
}
```

Testirajte približavajući prepreku senzoru. Testirajte s bijelom i crnom preprekom. Crnu prepreku senzor ne bi trebao detektirati. Zakrećite potenciometar P1 i izmjerite udaljenost od prepreke na kojoj ju senzor detektira.

ZADATAK 6. Na serijskom monitoru ispišite očitanja IR senzora udaljenosti kada približavate ili udaljavate prepreku.

RJEŠENJE. Shema spajanja je ista kao u prethodnom zadatku. Jedina razlika je da sada vrijednosti senzora očitavamo kao analogne vrijednosti i takve ispisujemo na serijski monitor.

```
int PinIR = 35;
int StanjeIR;

void setup() {
  pinMode(PinIR, INPUT);
  Serial.begin(9600);
}

void loop() {
  StanjeIR = analogRead(PinIR);
  Serial.println(StanjeIR);
  delay(200);
}
```

Približavajte i udaljavajte bijelu prepreku od senzora. Na kojoj udaljenosti ju senzor primjećuje i koju vrijednost tada ispiše na serijski monitor? Ponovite pokus sa crnom preprekom.
Kada je ispred senzora bijela prepreka, vrijednosti se mogu spustiti na 500 ili manje, ovisno o refleksiji IR zrake. Ukoliko je prepreka crna, senzor ju ne bi trebao detektirati i cijelo vrijeme bi trebao ispisivati 4095 (maksimalnu vrijednost). Istu vrijednost ispisuje i kada ispred senzora uopće nema prepreke.
Ovakav senzor se često koristi na robotima, za slijeđenje crne linije pri autonomnom kretanju u prostoru.

PROJEKT. Koristeći infracrveni predajnik upravljajte televizorom. Upute kako ovo postići možete vidjeti u VIDI-X projektu broj 96 „Univerzalni daljinski upravljač“.

LITERATURA.

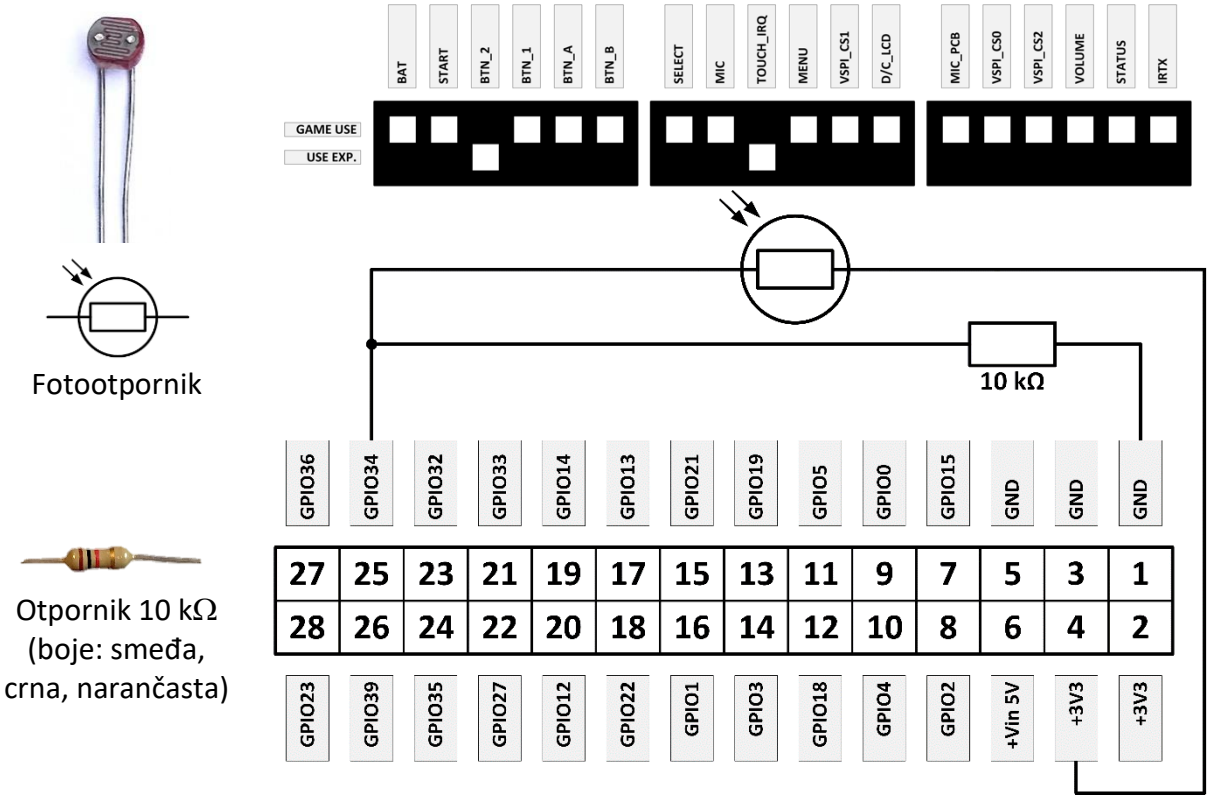
https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/
https://vidi-x.org/radionice/vidi-project-x-96-univerzalni-daljinski-upravljac/
https://github.com/crankyoldgit/IRremoteESP8266/wiki/Troubleshooting-Guide
https://osoyoo.com/2017/07/24/arduino-lesson-obstacle-avoidance-sensor/

10.Svjetlosno sviranje

Fotootpornik je otpornik koji je osjetljiv na svjetlo. Njegov otpor je manji ako je veći intenzitet ulazne svjetlosti (oko 1 kΩ), odnosno njegov otpor je veći kada se nalazi u mraku (do 200 kΩ). Obje njegove nožice su jednako dugačke i nebitno je na koju stranu se okrene prilikom spajanja. U ovoj cjelini se koristi 5mm CdS fotootpornik. VIDI X čita njegove vrijednosti kao broj između 0-4095.

ZADATAK 1. Spojite fotootpornik na VIDI X.

RJEŠENJE. Jednu nožicu fotootpornika spajamo na 3.3V (utor 4), a njegovu drugu stranu na utor 25, GPIO34. Pri tome mikroprekidač BTN_2 mora biti u položaju Use exp. Ista nožica fotootpornika, osim što je spojena na VIDI X, mora preko otpornika 10 kΩ biti spojena na uzemljenje (GND, utor 1).



ZADATAK 2. Očitanja fotootpornika ispisujte na serijski monitor. Koja je najniža vrijednost koju senzor može postići? Je li tada mrak ili svjetlo? Koja je najviša vrijednost koju senzor može postići? Je li tada mrak ili svjetlo?

RJEŠENJE. Fotootpornik je ulazni element te ga unutar `void setup()` definiramo kao `INPUT_PULLUP`. S obzirom da želimo očitati analogne vrijednosti između 0 i 4095, očitanje senzora očitava pomoću naredbe `svjetlo = analogRead(PinFoto);`

Koja je najniža vrijednost koju senzor može postići? Je li tada mrak ili svjetlo? Senzor postiže najnižu vrijednost ukoliko je oko njega tama. Tamu možemo napraviti tako da senzor sakrijemo u šaku ili u neku kutiju. U teoretskom slučaju ta minimalna vrijednost će iznositi 0, no u praktičnim situacijama će iznositi npr. oko 100. Koja je najviša vrijednost koju senzor može postići? Je li tada mrak ili svjetlo? Najviše vrijednosti senzor postiže na maksimalnom svjetlu. Tada su mu očitanja 4095.

```
int PinFoto = 34;
int svjetlo;

void setup() {
  pinMode(PinFoto,INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  svjetlo = analogRead(PinFoto);
  Serial.println(svjetlo);
  delay(100);
}
```

ZADATAK 3. Očitanja fotootpornika ispisujte na TFT ekran.

RJEŠENJE. Program iz prošlog zadatka povezujemo s programom za ispis na TFT ekran (npr. zadatak 5 cjeline 5).

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int PinFoto = 34;
int svjetlo;

void setup() {
  pinMode(PinFoto, INPUT_PULLUP);

  tft.begin();                          // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);     // boja teksta
  tft.setTextSize(2);                   // velicina teksta
  tft.setRotation(3);                   // postavi orijentaciju
}

void loop() {
  temp = analogRead(PinFoto);

  tft.fillScreen(ILI9341_BLACK);       // boja zaslona - brisanje sto je pisalo prije
  tft.setCursor(0, 50);                // polozaj pocetka ispisa tekst
  tft.println((String)"Svjetlo: " + svjetlo);

  delay(1000);
}
```

ZADATAK 4. Napravite program koji će na ekran crtati kako se mijenja očitano svjetlo u vremenu.

RJEŠENJE. Program je sličan zadatku 7, cjeline 6 s jednom bitnom razlikom: sada iscrtavamo promjenu svjetla u vremenu umjesto promjene temperature u vremenu.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

```
int PinFoto = 34;
int svjetlo;

int j = 0;

void setup() {
  pinMode(PinFoto,INPUT_PULLUP);

  Serial.begin(9600);                // ispis i na Serijski monitor

  tft.begin();                       // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);     // boja zaslona
  tft.setTextColor(ILI9341_YELLOW);  // boja teksta
  tft.setTextSize(2);                // velicina teksta
  tft.setRotation(3);                // postavi orijentaciju
}

void loop() {
  svjetlo = analogRead(PinFoto);

  Serial.println(svjetlo);              // ispis i na Serijski monitor

  tft.fillRect(9, 69, 60, 16, ILI9341_BLACK); // ispuna pravokutnika
                                              // umjesto brisanja ekrana
  tft.setCursor(10, 70);              // polozaj pocetka ispisa tekst
  tft.println(svjetlo);               // ispis kolicine svjetla

  // Varijabla j treba za kretanje po X-osi ekrana i crtanje piksela
  j++;
  if (j > tft.width() + 1) {  // Kada j izade iz ekrana
     j = 0;                   // postavi ga na nulu
  }

  // brisemo stari piksel kako bi mogli nacrtati novi na njegovom mjestu
  tft.drawFastVLine(j, 0, tft.height(), ILI9341_BLACK);

  // crtanje vrijednosti ocitanja svjetla
  tft.drawPixel(j, map(svjetlo, 0, 4095, 239, 119), ILI9341_RED);
  tft.drawPixel(j, map(svjetlo, 0, 4095, 238, 118), ILI9341_RED);

  delay(300);
}
```

ZADATAK 5. Napravite pametnu lampu: kada fotootpornik očitava mnogo svjetla, neka TFT ekran bude potpuno crn. Kada fotootpornik očitava malo svjetla, neka TFT ekran bude bijeli.

RJEŠENJE. Kako očitanja fotootpornika mogu biti između 0 i 4095, granica za crni ili bijeli ekran je postavljena na sredini na 2047. Kada je vrijednost očitanja veća od 2047 ekran je crni, a kada je vrijednost očitanja manja, ekran je bijeli.

```
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>

// ILI9341 TFT LCD deklaracija pinova
#define TFT_CS   5
#define TFT_DC  21

// stvaranje objekta ekrana koji će se zvati tft
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

int PinFoto = 34;
int svjetlo;

void setup() {
  pinMode(PinFoto,INPUT_PULLUP);

  Serial.begin(9600);                // ispis i na Serijski monitor
```

```
  tft.begin();                          // inicijalizacija zaslona
  tft.fillScreen(ILI9341_BLACK);        // boja zaslona
  tft.setRotation(3);                   // postavi orijentaciju
}

void loop() {
  svjetlo = analogRead(PinFoto);

  Serial.println(svjetlo);              // ispis i na Serijski monitor

  if (svjetlo > 2047) {
    tft.fillScreen(ILI9341_BLACK);      // boja zaslona crna
  } else {
    tft.fillScreen(ILI9341_WHITE);      // boja zaslona bijela
  }

  delay(300);
}
```

ZADATAK 6. Na VIDI X spojite zvučnik. Napišite program koji će svirati različite note (frekvencije) ovisno koliko ruku približite ili udaljite od fotootpornika. Fotootpornik će biti „daljinski upravljač“ za sviranje.

RJEŠENJE. U programu moramo definirati pinove za fotootpornik i zvučnik. Nakon analognog očitavanja fotootpornika, vrijednosti koje je očitao moramo pretvoriti u frekvencije koje ljudi čuju: npr. između 50 Hz i 4000 Hz, pomoću naredbe `map()`. Mijenjanjem ovih brojeva možemo dobiti instrument koji svira samo duboke ili samo visoke tonove. Testirajte.

```
int PinFoto = 34;
int svjetlo;

int PinZvucnik = 25;
int nota;

void setup() {
  pinMode(PinFoto, INPUT_PULLUP);

  pinMode(PinZvucnik, OUTPUT);
  ledcSetup(0, 10000, 12);
  ledcAttachPin(PinZvucnik, 0);
}

void loop() {
  svjetlo = analogRead(PinFoto);

  nota = map(svjetlo, 0, 4095, 50, 4000);

  ledcWriteTone(0, nota);

  delay(10);
}
```

Koja je najniža, a koja najviša frekvencija koju možeš čuti? Ljudi čuju frekvencije između 20 i 20000 Hz. Raspon frekvencija se s godinama smanjuje.
Koje frekvencije moraš izabrati kako bi tvoj instrument svirao samo duboke tonove? Primjer za sviranje dubokih tonova bi bio: `nota = map(svjetlo, 0, 4095, 50, 500);`
Koje frekvencije moraš izabrati kako bi tvoj instrument svirao samo visoke tonove? Primjer za sviranje visokih tonova bi bio: `nota = map(svjetlo, 0, 4095, 500, 4000);`

ZADATAK 7. Neka instrument počne raditi nakon pritiska tipkala START. Kada se START ponovno pritisne, neka se instrument ugasi.

RJEŠENJE. Tipkalo START koristi GPIO36, a mikroprekidač START mora biti u položaju Game Use. Kako bismo znali treba li pritiskom tipke START pokrenuti ili ugasiti instrument, moramo znati je li instrument prije toga svirao ili šutio (slično zadatku 10, cjeline 2). Zato nakon provjere je li tipkalo pritisnuto `if (StanjeTipkala`

`== LOW)` provjeravamo u kakvom je stanju bilo prije (`if (ProsloStanje == 0)`). S obzirom da smo upravo pritisnuli tipkalo, to stanje se sada mijenja. U ovisnosti o njegovoj novoj vrijednosti pokrećemo ili gasimo instrument. Poslije promjene stanja dodano je čekanje od 500 ms, kako bi VIDI-X stigao reagirati na pritisak tipke.

```
int PinFoto = 34;
int svjetlo;

int PinZvucnik = 25;
int nota;

int PinTipkalo = 39;    // tipkalo START
int StanjeTipkala;
int ProsloStanje = 1;

void setup() {
  pinMode(PinFoto,INPUT_PULLUP);

  pinMode(PinZvucnik, OUTPUT);
  ledcSetup(0, 10000, 12);
  ledcAttachPin(PinZvucnik, 0);

  pinMode(PinTipkalo, INPUT_PULLUP);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);
  if (StanjeTipkala == LOW) {
    if (ProsloStanje == 0) {
          ProsloStanje = 1;
        } else {
          ProsloStanje = 0;
        }
    delay(500);
  }

  if (ProsloStanje == 0) {
    svjetlo = analogRead(PinFoto);
    nota = map(svjetlo, 0, 4095, 50, 4000);
    ledcWriteTone(0, nota);
  }

  if (ProsloStanje == 1) {
    ledcWriteTone(0, 0);
  }

  delay(10);
}
```

ZADATAK 8. Kalibrirajte fotootpornik.

RJEŠENJE. Kako bi si olakšali posao i kako ne bismo trebali svaki puta čitati na serijskom monitoru vrijednosti osvjetljenja, moguće je napraviti automatsku kalibraciju fotootpornika. Kalibracija traje 5 sekundi i u tom vremenu treba osvijetliti fotootpornik svim mogućim osvjetljenjima kakva nas mogu okružiti za vrijeme sviranja. Kalibracija se radi unutar `void setup()`. Naredba za mjerenje vremena u milisekundama je

```
millis();
```

Ova naredba vraća koliko je vremena prošlo od kada je VIDI-X započeo izvoditi trenutni program.
Početno se najniža vrijednost koju senzor može očitati postavlja na najveću moguću vrijednost `int senzorDonja = 4095;` Ova vrijednost se uspoređuje sa svakim novim očitanjem i ako je očitanje manje od prethodnog, najniža vrijednost će postati upravo to očitanje `senzorDonja = svjetlo;`
Za najvišu vrijednost postupak je sličan, samo se ta vrijednost početno postavlja na 0 i svakim očitanjem koje je veće, najviša vrijednost poprima veći iznos `senzorGornja = svjetlo;`
Nakon kalibracije, u , program je sličan kao u prethodnim zadacima, uz korištenje minimalne i maksimalne vrijednosti očitanja: `senzorDonja, senzorGornja`.

```
int senzorDonja = 4095;
int senzorGornja = 0;

int PinFoto = 34;
int svjetlo;

int PinZvucnik = 25;
int nota;

int PinLed = 2;

void setup() {
  pinMode(PinFoto,INPUT_PULLUP);

  pinMode(PinZvucnik, OUTPUT);
  ledcSetup(0, 10000, 12);
  ledcAttachPin(PinZvucnik, 0);

  pinMode(PinLed, OUTPUT);
  digitalWrite(PinLed, HIGH);

  while (millis() < 5000) {
    svjetlo = analogRead(PinFoto);
    if (svjetlo > senzorGornja) {
      senzorGornja = svjetlo;
    }
    if (svjetlo < senzorDonja) {
      senzorDonja = svjetlo;
    }
  }
  digitalWrite(PinLed, LOW);
}

void loop() {
  svjetlo = analogRead(PinFoto);
  nota = map(svjetlo, senzorDonja, senzorGornja, 50, 4000);

  ledcWriteTone(0, nota);

  delay(10);
}
```

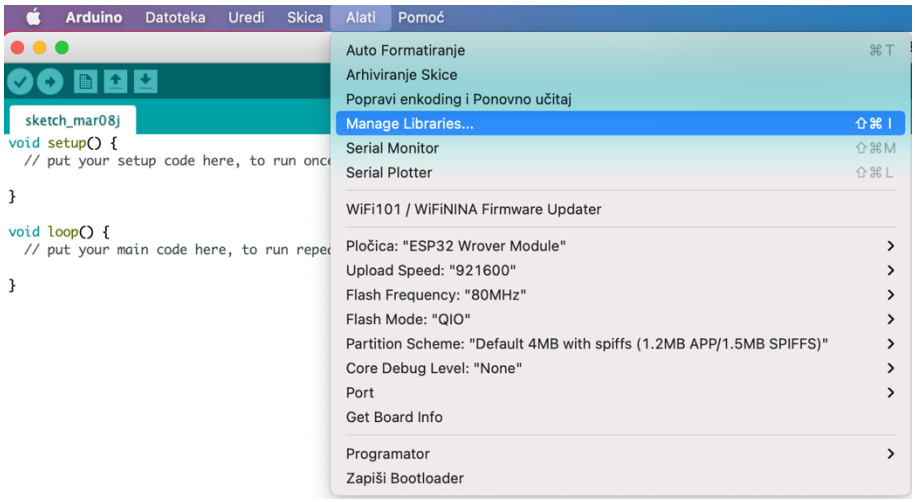
PROJEKT. Neka svatko podesi drugačije frekvencije rada svojih instrumenata. Skladajte zajedničku pjesmu!

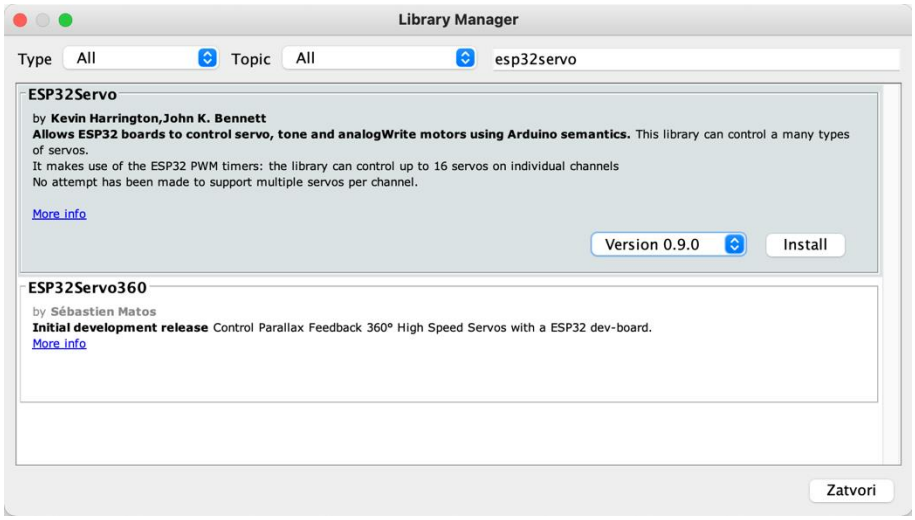
11.Čovječe, ne ljuti se

Servomotori su istosmjerni motori sa zupčanicima koji omogućuju precizno pozicioniranje osovine motora. Većina servomotora može zakrenuti svoju osovinu za 180°, no ima i onih koji se mogu zakrenuti do 270° ili čak 360°. U ovoj cjelini se koristi motor SM-S2309S koji može napraviti 180°. Osim po kutu koji osovina može napraviti, servomotori se fizički razlikuju po veličini i građi zupčanika (plastični ili metalni). Mogu se upravljati analogno putem PWM signala ili digitalno I/O signalima. Većina serva se napaja s 5V-6V, stoga se servo može direktno spojiti na VIDI X bez dodatne baterije. Ipak, ukoliko se želi spojiti više servomotora, treba dodati vanjsko napajanje.

Kako bismo mogli upravljati servo motorima na VIDI X potrebno je preuzeti biblioteku naredbi ESP32Servo s interneta odabirom Alati, pa Manage Libraries, unutar Arduino IDE sučelja. U novootvorenom prozoru se upiše ESP32Servo, pronađe se među ponuđenima i pritisne Install. Za uključivanje ove biblioteke u program koristi se naredba:

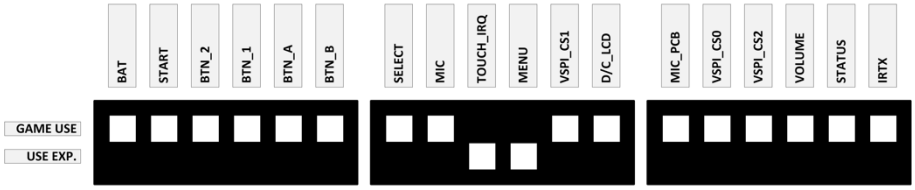
```
#include <ESP32Servo.h>
```

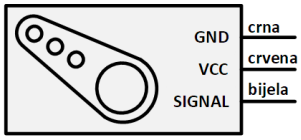




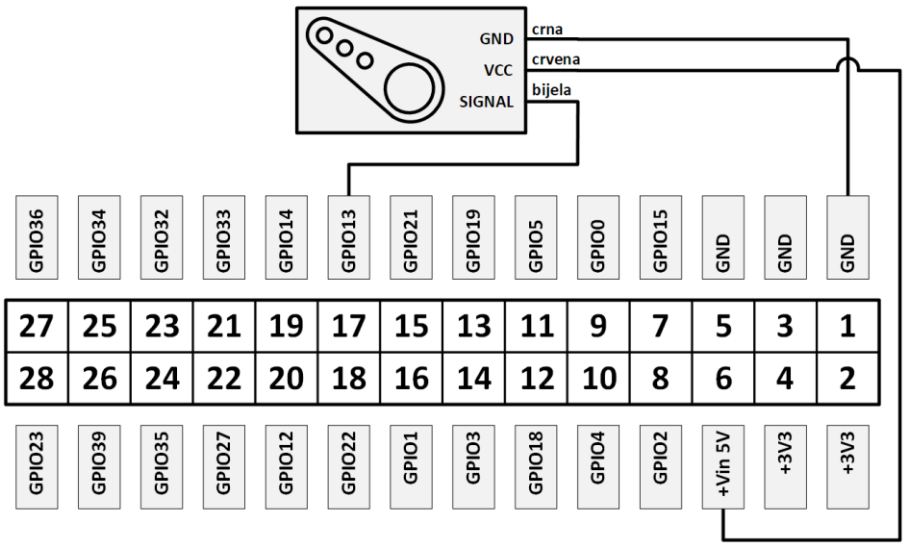
ZADATAK 1. Spojite servomotor na VIDI X.

RJEŠENJE. Servomotor ima tri žice, koje ovisno o proizvođaču mogu biti različitih boja. Najčešće crna (ili smeđa) žica predstavlja zemlju i spaja se na GND (utor 1 na pinskom proširenju). Crvena žica je napajanje i spaja se na 5V (utor 6). Bijela žica (ili žuta) je signalna i u ovoj cjelini ćemo ju spojiti na pin GPIO13 (utor 17, mikroprekidač MENU u položaju Use Exp).





Servomotor
crna žica – na GND
crvena žica – na 5V
bijela žica – na GPIO13



ZADATAK 2. Napišite program koji će kazaljku servomotora postaviti na točno zadani kut.

RJEŠENJE. Na početku programa uključujemo biblioteku:

```
#include <ESP32Servo.h>
```

Zatim stvaramo objekt `MojServo`. Pod tim imenom ćemo u nastavku programa prepoznavati naredbe i funkcije koje se odnose na rad sa servomotorom.

```
Servo MojServo;
```

Inicijalizaciju naredbi vezanih uz servomotor obavlja funkcija

```
MojServo.attach(PinServo);
```

Postavljanje servomotora na željeni kut vrši se naredbom

```
MojServo.write(kut);
```

Servomotor može poprimiti kutove između 0° i 180°. Vrijednosti manje od 0 se automatski postavljaju na 0°, vrijednosti između 181 i 499 na 180°. Vrijednosti veće od 500 se tretiraju kao milisekunde širine pulseva.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

void setup() {
  MojServo.attach(PinServo);

  kut = 90;
  MojServo.write(kut);
}

void loop() {
}
```

ZADATAK 3. Promijenite program tako da kazaljka pokazuje kut 0 – početni položaj servo motora.

RJEŠENJE. Kut 0 je početni položaj servo motora. Kako bi lakše pratili kamo i koliko se motor zakrenuo na njega je moguće staviti kazaljku (obično dolazi s motorom). Položaj kazaljke na servo motoru najbolje je namjestiti dok je motor u položaju 0. Tada ju možemo poravnati u smjeru samog motora, okomito na motor ili u proizvoljnom smjeru ovisno o primjeni.

```
#include <ESP32Servo.h>

int PinServo = 13;

Servo MojServo;

int kut;

void setup() {
  MojServo.attach(PinServo);

  kut = 0;
  MojServo.write(kut);
}

void loop() {
}
```

ZADATAK 4. Napišite program koji će naizmjence pomicati kazaljku na 0°, pa na 90°.

RJEŠENJE. Okretanje kazaljke servo motora sada prebacujemo u `void loop()`. Kut postavljamo na 0°, pričekamo 1 sekundu, pa ga postavljamo na 90° i opet pričekamo 1 sekundu.

```
#include <ESP32Servo.h>

int PinServo = 13;
Servo MojServo;
int kut;

void setup() {
  MojServo.attach(PinServo);
}

void loop() {
  kut = 0;
  MojServo.write(kut);
  delay(1000);

  kut = 90;
  MojServo.write(kut);
  delay(1000);
}
```

ZADATAK 5. Napišite program koji će pomicati kazaljku od 179° prema 0°, svaku milisekundu za 1 stupanj.

RJEŠENJE. Kut početno postavimo na 180° unutar `void setup()`. Svakim prolazom `void loop()` kut smanjujemo za 1 i pomičemo servo na taj novi položaj. Pričekamo 100ms prije ponavljanja postupka.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

void setup() {
  MojServo.attach(PinServo);
  kut = 180;
}
```

```
void loop() {
  kut = kut-1;
  MojServo.write(kut);
  delay(100);
}
```

ZADATAK 6. Napišite program koji će pomicati kazaljku od 179° prema 0°, svaku milisekundu za 1 stupanj. Kada dođe do 0° neka krene iz početka, od 179°.

RJEŠENJE. Dodatak u odnosu na prošli zadatak je da sada kut, kada postigne vrijednost 0° vraćamo na 180°. Na taj način će `void loop()` ponovno brojati od 179 prema 0.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

void setup() {
  MojServo.attach(PinServo);
  kut = 180;
}

void loop() {
  kut = kut-1;
  MojServo.write(kut);
  delay(100);

  if (kut==0) {
    kut = 180;
  }
}
```

ZADATAK 7. Napišite program koji će pomicati kazaljku od 179° prema 0°, svaku milisekundu za 5 stupnjeva. Kada dođe do 0° neka krene iz početka, od 179°. Nakon pomicanja očitajte položaj serva i ispišite na serijski monitor.

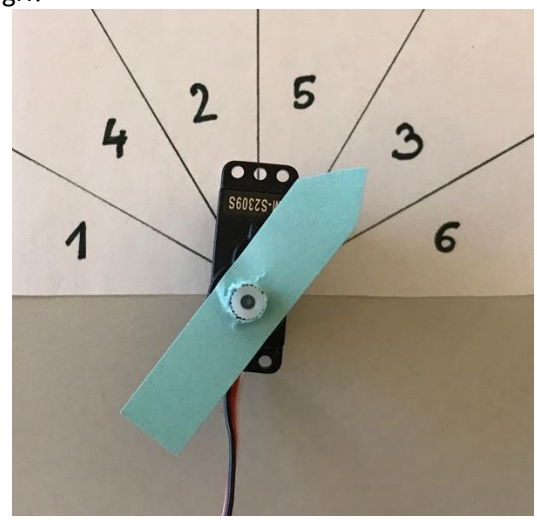
RJEŠENJE. Za očitavanje trenutne vrijednosti kuta u kojem se servo nalazi koristi se naredba

```
MojServo.read()
```

Kao rezultat vraća vrijednost kuta u stupnjevima između 0 i 180.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

void setup() {
  MojServo.attach(PinServo);
  kut = 180;

  Serial.begin(9600);
}

void loop() {
  kut = kut-1;
  MojServo.write(kut);
  Serial.println(MojServo.read());
  delay(100);

  if (kut==0) {
    kut = 180;
  }
}
```

ZADATAK 8. Napravite uređaj koji će izabirati broj u igri Čovječe, ne ljuti se. Izrežite iz kartona jedan polukrug i podijelite ga u 6 kružnih isječaka. U svaki isječak upišite jedan broj između 1 i 6. Kazaljka servomotora se mora brzo pomicati i pokazivati na sve brojeve. Kada tipkalo START pritisnuto, kazaljka servo se pomiče. Kada se tipkalo otpusti, kazaljka se zaustavi iznad jednog broja. Ono što će kazaljka pokazati – to je broj koji treba upotrijebiti u igri!



RJEŠENJE. Tipkalo START koristi pin GPIO39. Kada je tipkalo pritisnuto izvršava se isti kod kao u prethodnom zadatku. Kada se tipkalo otpusti, servo se zaustavlja u svom trenutnom položaju.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

int PinTipkalo = 39;
int StanjeTipkala;

void setup() {
  MojServo.attach(PinServo);
  kut = 180;

  pinMode(PinTipkalo, INPUT_PULLUP);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);

  if (StanjeTipkala == LOW) {
    kut = kut-1;
    MojServo.write(kut);
    delay(10);
    if (kut==0) {
      kut = 180;
    }
  } else {
    MojServo.write(kut);
  }
}
```

ZADATAK 9. Zakomplicirajte zaustavljanje servo motora: kada se tipkalo START otpusti, neka se kazaljka pomakne za 15° prije nego se zaustavi na konačnom broju.

RJEŠENJE. Jedina promjena je u položaju serva kada se tipkalo otpusti. Tada će položaj serva biti `kut-15`.

```
#include <ESP32Servo.h>

Servo MojServo;

int PinServo = 13;
int kut;

int PinTipkalo = 39;
int StanjeTipkala;

void setup() {
  MojServo.attach(PinServo);
  kut = 180;

  pinMode(PinTipkalo, INPUT_PULLUP);
}

void loop() {
  StanjeTipkala = digitalRead(PinTipkalo);

  if (StanjeTipkala == LOW) {
    kut = kut-1;
    MojServo.write(kut);
    delay(10);
    if (kut==0) {
      kut = 180;
    }
  } else {
    MojServo.write(kut-15);
  }
}
```

PROJEKT. Osmislite gdje bi se još mogao upotrijebiti servo motor na način kao u posljednjem zadatku.

LITERATURA.

https://dronebotworkshop.com/esp32-servo/
https://github.com/RoboticsBrno/ServoESP32/blob/master/src/Servo.h#L73
https://vidi-x.org/radionice/vidi-project-x-97-upravljanje-servo-motorom/