ELEKTROTEHNIČKA ŠKOLA

Zagreb, Konavoska 2

ZAVRŠNI RAD

Ljepljive bilješke na VIDI-X platformi

Mentor:

Mario Tretinjak, struč. spec. ing. el.

Ime i prezime:

Josip Skledar, 4.D

U Zagrebu, 25. svibnja 2022.

Sadržaj

Popis slika

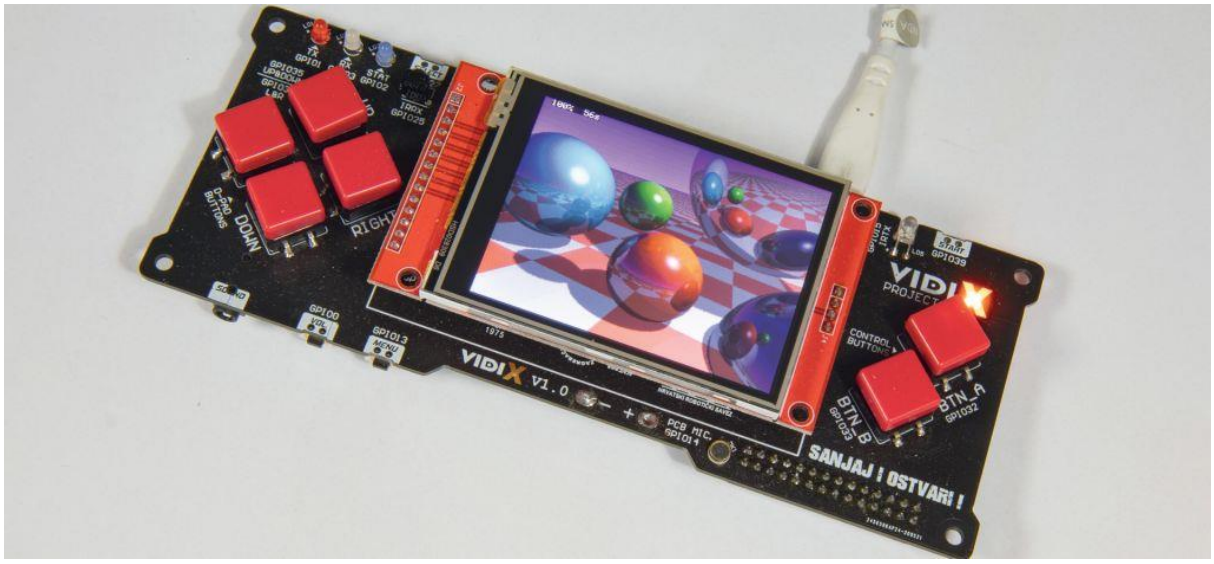
Uvod

Ovaj završni rad u digitalnom obliku imitira ljepljive bilješke koje zatrpavaju Vaš stol. Razvijen je za novo razvojno mikroračunalo VIDI-X. Zbog manjka literature za tu pločicu, počeci su bili teški, bio je problem napisati jedno slovo na ekranu. S vremenom sam učio. Najbitniji dio ovog rada je mogućnost mijenjanja sučelja bez gubitka onog što ste nacrtali na ekran. Uspio sam napraviti ono što sam htio i zadovoljan sam rezultatom.

Tehnologija

VIDI-X

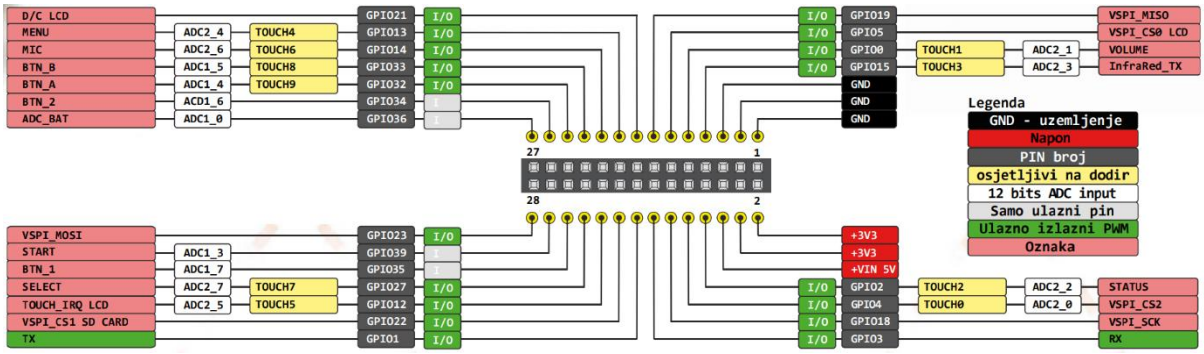
Tehnološka kuća VIDI, najpoznatija po časopisu VIDI, nedavno je pokrenula projekt VIDI-X. VIDI-X je mikroračunalo s ESP 32-bitnim procesorom. Na njemu se nalazi TFT LCD zaslon i brojni senzori. Zbog toga je pogodno za nastavu informatike u osnovnim i srednjim školama. Naša škola se prijavila za donaciju tih računala, a da bi dobila ta mikroračunala, trebao se napraviti video. Taj video je prošle godine izradila grupa od trojice učenika, u kojoj sam bio i ja. Neka od imena koja su podržala taj projekt su: Rimac automobili, FER, Gideon Brothers, Hrvatski robotički savez… VIDI tvrdi kako je VIDI Project X snažnija platforma od Arduina, ali principi su isti. Za programiranje ovog završnog rada koristio sam program Arduino IDE.



Slika 1 VIDI-X

Pinout shema/Dijagram izvoda

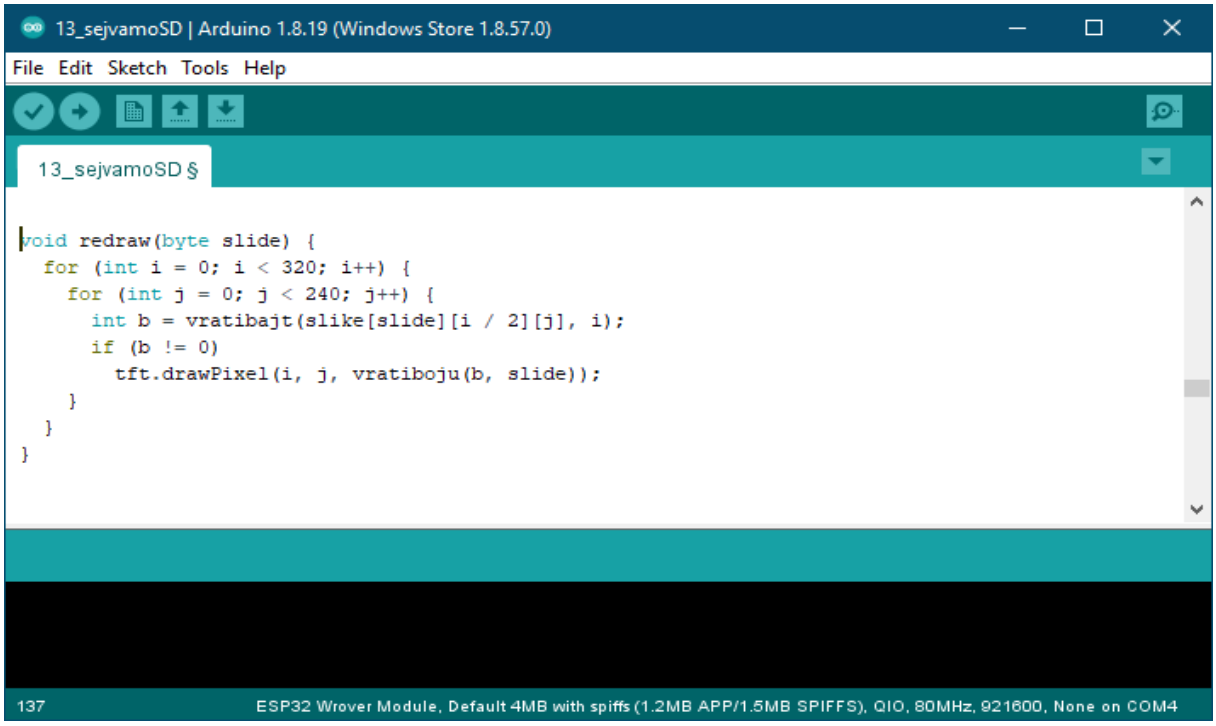
Za komunikaciju s ekranom koristimo pinove 5 (fizički pin 13, GPI05, VSPI_CS0 LCD, u kôdu *TFT_CS*) i 21 (fizički pin 15, GPI021, D/C LCD, u kôdu *TFT_DC*). Za touchscreen koristimo pin 4 (fizički pin 10, GPI04, VSPI_CS2, u kôdu *TS_CS).*



Slika 2 VIDI-X pinout shema

Arduino IDE

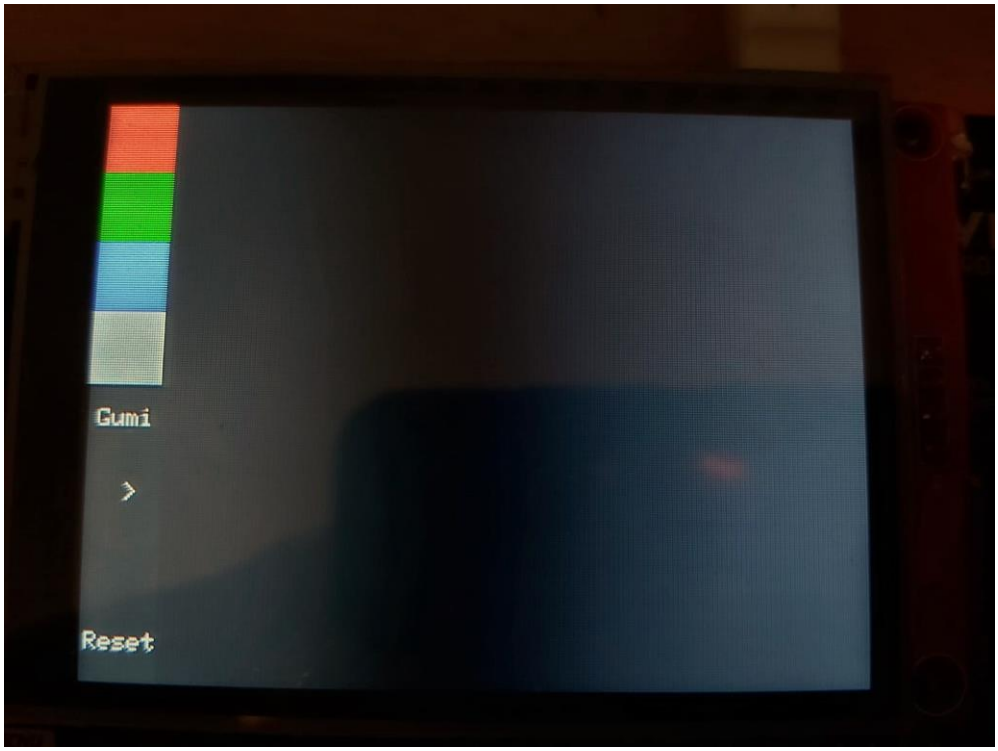
Arduino je platforma koja promiče hardware i software koji se lako koristi. Često sam je sretao u mom srednjoškolskom obrazovanju. Jedan od alata koji nam pomaže da pišemo kôd i prenesemo ga na Arduino platformu zove se Arduino IDE – integrirano razvojno okruženje. Program je vrlo jednostavan, ali moramo instalirati dodatnu podršku za našu pločicu, odnosno ESP32 procesor. Arduino platforma, kao i VIDI-X, koristi jezike C i C++. Nešto između C i C++. Prema mom iskustvu, sličan je C-u, ali je veliki plus to što dozvoljava upotrebu funkcija (objektno orijentirano programiranje), što C nema.



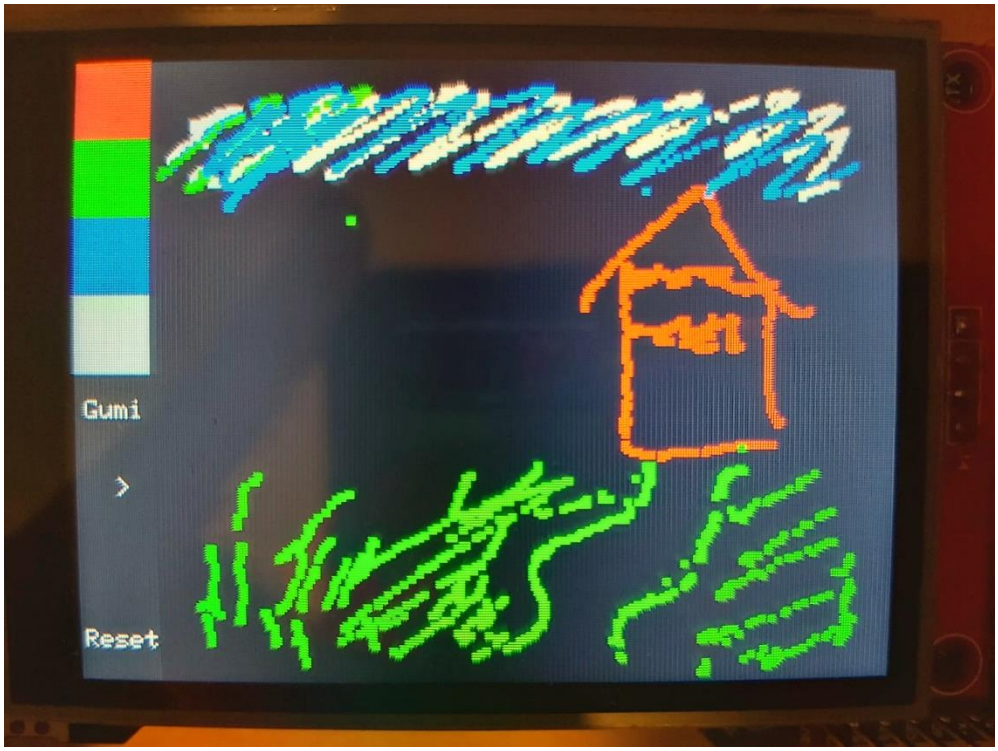
Slika 3 Arduino IDE

Funkcionalnost i sučelje

Cilj mog rada je da se može koristiti jednako lako kao papirić na stolu. Kada ga upalite, ne postoji poruka dobrodošlice niti izbornik, dočeka Vas prostor po kojem možete crtati. S lijeve strane možete izabrati boje kojima želite crtati i gumicu kojom možete brisati.



Slika 4 Crno sučelje



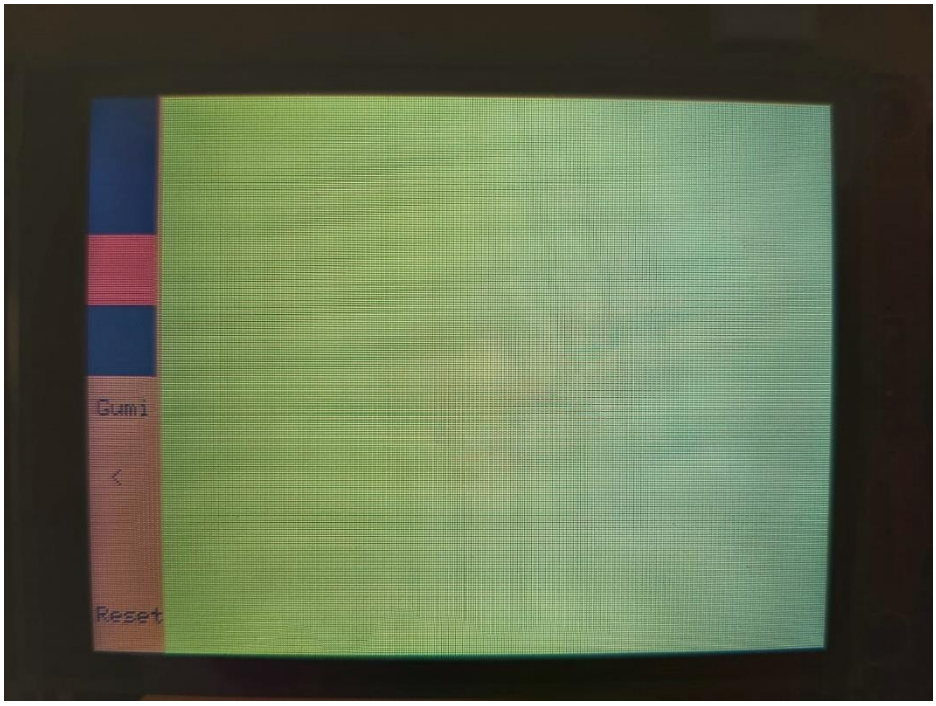
Slika 5 Crtanje na crnom sučelju

Reset gumb

Dolje lijevo nalazi se *Reset* gumb. Možete pretpostaviti da pritiskom na njega brišete sadržaj koji ste napisali ili nacrtali, ali ima jednu interesantnu funkciju. Kako bi spriječili posljedice slučajnih dodira, sadržaj briše samo ako držite taj gumb jednu sekundu.

Mijenjanje sučelja

Iznad *Reset* gumba nalazi se jedna strelica. Pritiskom na tu strelicu mijenja se *slajd*, mijenja se sučelje. Ovo sučelje imitira boju prave ljepljive bilješke, a boje kojima možete pisati više nisu *programerska* crvena, zelena i plava, već sve boje klasičnih kemijskih olovaka: crna, plava, crvena i siva. Isto tako, ako se želite vratiti na crno sučelje, pritisnut ćete strelicu. Vaš sadržaj **neće** se izbrisati prilikom promjene.



Slika 6 Žuto sučelje

Izrada završnog rada

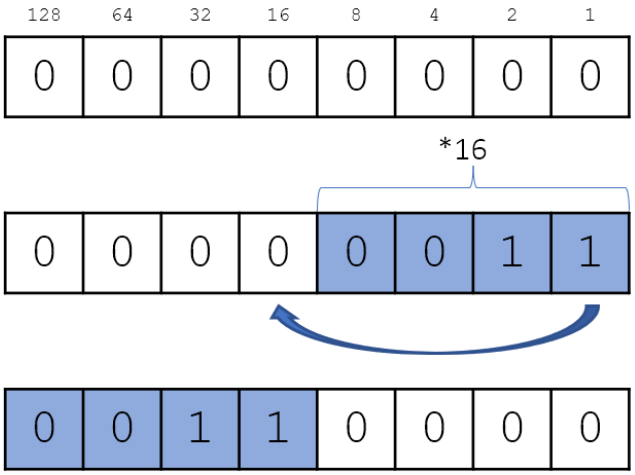
Problematika

Veliki problem i najteži dio posla bio je natjerati mikroračunalo da pamti što korisnik crta po ekranu. Gotovo je nemoguće povući sliku s ekrana u memoriju. Taj problem riješio sam tako što mikroračunalo pamti čim Vi dotaknete ekran. Informaciju šalje ekranu i u isto vrijeme je pamti u dvodimenzionalno polje koje kasnije koristi za rekonstrukciju Vaše slike.

VIDI-X ima malo memorije u koju mogu spremiti slike s ekrana. Postojalo je nekoliko rješenja:

1. Ne spremati. Najlakše je bilo napraviti mali pravokutnik po kojem se može pisati i brisati, tako sam i počeo. Nisam to htio tako ostaviti, to je bilo prelagano, premalo posla za završni rad.
2. Spremati na SD karticu. Ovo rješenje mi se svidjelo, ali sam ipak htio da mi sve stane na pločicu.
3. Spremati u memoriju. Bez ikakve modifikacije, spremanje jedne slike od 320x240 piksela (rezolucija ekrana) dok se za boju koriste 2 bajta preveliko je za unutarnju memoriju ovog mikroračunala, a ja sam htio strpati dvije slike.

Odabrao sam treće rješenje. Već sam naveo da za pohranu boje ovo mikroračunalo koristi 2 bajta. Ja koristim samo 4 boje u svakoj slici, plus gumica, a to je 5 vrijednosti. Zaključujem da mi za pohranu boje zapravo trebaju samo 3 bita, ali zbog (relativne) jednostavnosti koristim 4. Bajt dijelim na dva djela, tako da se u jednom bajtu pohranjuju dva piksela i tako smanjujem potrošnju memorije 4 puta. Prije zapisa i rekonstrukcije slike, pretvaram boju u jednu od vrijednost od 1 do 5. Ako se radi o „lijevom“ pikselu, vrijednost boje množim sa 16 (2^4) kako bi ju pomaknuo za 4 vrijednosna mjesta i spremio u „lijevi“ dio bajta. Ovo sve je potrebno zato što Arduino i C++ ne podržavaju tip podatka od 4 bita (*nibble*).



Slika 7 Spremanje boje u polovicu bajta

Kôd

```
//potrebne datoteke za zaslon, grafike i touchscreen
#include <Adafruit_ILI9341.h>
#include <Adafruit_GFX.h>
#include <XPT2046_Touchscreen.h>
#include <SPI.h>

#define TFT_CS 5    //pinovi za zaslon
#define TFT_DC 21
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

#define TS_CS 4     //pinovi za touchscreen
XPT2046_Touchscreen ts(TS_CS);

int x_piksel, y_piksel; //koordinate dodira
uint16_t boja;          //boja za crtanje po zaslonu
byte boja_bajt = 1;     //boja za zapis u memoriju
byte promjer = 3;       //debljina kista
byte slide = 0;         //sučelje

byte slike [2][160][240]; //pohrana nacrtanog

void setup() {
  tft.begin();            //inicijalizacija zaslona
  tft.setRotation(3);     //orijentacija zaslona
  ts.begin();             //inicijalizacija touchscreena
  ts.setRotation(1);      //orijentacija touchscreena
  suceljecrno();          //početno crno sučelje
}

void loop() {
  if (ts.touched()) {     //ako je touchscreen dodirnut
    TS_Point p = ts.getPoint();             //točka dodira
    x_piksel = map(p.x, 370, 3900, 0, 319); //pretvaranje touchscreen koordinata u
    y_piksel = map(p.y, 210, 3800, 0, 239); //koordinate za crtanje po zaslonu

    if (x_piksel < 33) {       //ako je u lijevom djelu ekrana - izbornik
      promjer = 3;             //resetira debljinu kista
      if (y_piksel < 31) {     //odabir crvene/crne boje (ovisi o crnom/žutom
sučelju)
        boja_bajt = 1;
      } else if (y_piksel < 61) { //odabir zelene/plave boje
        boja_bajt = 2;
      } else if (y_piksel < 91) { //odabir plave/crvene boje
        boja_bajt = 3;
      } else if (y_piksel < 121) {//odabir bijele/sive boje
        boja_bajt = 4;
      } else if (y_piksel < 151) {//odabir gumice
        promjer = 18;             //gumica je deblja od kista
        boja_bajt = 0;
      } else if (y_piksel > 210) {//resetiranje sučelja
        delay(1000);              //sigurnosna funkcija
        if (ts.touched()) {       //da nebi bilo slučajnih dodira
          suceljecrno();
          resetiraj(slide);
        }
```

```
        delay(250);
      } else if (y_piksel < 181) {
        if (slide == 0) {    //ako je crno sčelje
          suceljebijelo();  //mijenja sučelje u žuto
          slide = 1;
        } else {             //ako je žuto sučelje
          suceljecrno();     //mijenja sučelje u crno
          slide = 0;
        }
        redraw(slide);       //ispis vašeg crteža pohranjenog u memoriji
        boja_bajt = 1;       //resetira kist
        delay(250);          //štiti od rapidnih promjena kod dužih dodira
      } else if (y_piksel > 210) {
        delay(1000);         //sigurnosna funkcija
        if (ts.touched()) { //da nebi bilo slučajnih dodira
          if (slide == 0)
            suceljecrno();
          else
            suceljebijelo();
          resetiraj(slide);
        }
      }
    }

    if (x_piksel > 32) {
      tft.fillRect(x_piksel, y_piksel, promjer, promjer, vratiboju(boja_bajt,
slide));
      zapisi(x_piksel, y_piksel, boja_bajt, slide);
    }
  }
  delay(2);
}
```

Funkcije za pohranu i ponovno crtanje podataka

```
void zapisi(int x, int y, byte boja, byte br) {
  int kraj = 3;
  if (boja == 0) { //gumica
    boja = 0;
    kraj = 18;
  }

  for (int i = 0; i < kraj; i++) {
    for (int j = 0; j < kraj; j++) {
      int x_temp = (x + i) / 2;
      if ((x + i) % 2) {                          //LIJEVI pixel, npr. x=1 x=3
        //briše prva 4 b
        slike[br][x_temp][y + j] -= (slike[br][x_temp][y + j] / 16) * 16;
        slike[br][x_temp][y + j] += boja * 16;  //sprema boju u "lijevi" dio bajta
      } else {                                    //DESNI pixel, npr. x=2 x=4
        //briše zadnja 4 b
        slike[br][x_temp][y + j] -= slike[br][x_temp][y + j] % 16;
        slike[br][x_temp][y + j] += boja;       //sprema boju u "desni" dio bajta
      }
    }
  }
}

void redraw(byte slide) {       //rekonstrukcija slike
  for (int i = 0; i < 320; i++) {
    for (int j = 0; j < 240; j++) {
      int b = vratibajt(slike[slide][i / 2][j], i);
      if (b != 0)
        tft.drawPixel(i, j, vratiboju(b, slide));
    }
  }
}

void resetiraj(byte slide) {    //reset gumb
  for (int i = 0; i < 160; i++)
    for (int j = 0; j < 240; j++)
      slike[slide][i][j] = 0;
}

uint16_t vratiboju(byte b, byte slide) {
  uint16_t kolor;
  if (slide == 0) {             //boje za crno sučelje
    switch (b) {
      case 1:
        return ILI9341_RED;
      case 2:
        return ILI9341_GREEN;
      case 3:
        return ILI9341_BLUE;
      case 4:
        return ILI9341_WHITE;
      case 0:
        return 0x0001;
    }
  } else {
```

```
    switch (b) {                //boje za žuto sučelje
      case 1:
        return 0x0002;
      case 2:
        return ILI9341_NAVY;
      case 3:
        return ILI9341_RED;
      case 4:
        return ILI9341_DARKGREY;
      case 0:
        return 0xFFA8;
    }
  }
}

byte vratibajt(byte b, int x) {
  if (x % 2)           //ako je piksel neparan
    return b / 16;     //vraća "lijevi" dio bajta, lijevi piksel
  return b % 16;       //vraća "desni" dio bajta, desni piksel
}

void suceljecrno() {  //crta crno sučelje
  tft.fillScreen(0x0001);
  tft.fillRect(0, 0, 32, 240, ILI9341_DARKGREY);
  tft.fillRect(0, 0, 30, 30, ILI9341_RED);
  tft.fillRect(0, 30, 30, 30, ILI9341_GREEN);
  tft.fillRect(0, 60, 30, 30, ILI9341_BLUE);
  tft.fillRect(0, 90, 30, 30, ILI9341_WHITE);
  tft.setTextSize(1);
  tft.setTextColor(ILI9341_WHITE);
  tft.setCursor(4, 130);
  tft.print("Gumi");
  tft.setCursor(16, 160);
  tft.print(">");
  tft.setCursor(2, 220);
  tft.print("Reset");
}

void suceljebijelo() {  //crta žuto sučelje
  tft.fillScreen(0xFFA8);   //post-it boja
  tft.fillRect(0, 0, 32, 240, 0xFE88);
  tft.fillRect(0, 0, 30, 30, ILI9341_BLACK);
  tft.fillRect(0, 30, 30, 30, ILI9341_NAVY);
  tft.fillRect(0, 60, 30, 30, ILI9341_RED);
  tft.fillRect(0, 90, 30, 30, ILI9341_DARKGREY);
  tft.setTextSize(1);
  tft.setTextColor(ILI9341_BLACK);
  tft.setCursor(4, 130);
  tft.print("Gumi");
  tft.setCursor(9, 160);
  tft.print("<");
  tft.setCursor(2, 220);
  tft.print("Reset");
}
```

Zaključak

Ne volim baš Arduino, ovakva mikroračunala. Smatram da nisu savršena i većinu puta problem nije u mojim rukama, nego problem ima veze s mikroračunalom. Tako je bilo i ovaj put. Jako dugo sam znao gledati u ekran samo kako bih shvatio da problem nije moj kôd, već npr. zastarjela biblioteka. Često se oglašavaju kao jednostavan način i uvod u programiranje za početnike, ali kad se dogode ovakve stvari, vjerujem da nije motivirajuće. Zbog toga mi je drago što ih nisam susreo u osnovnoj školi. Ovaj rad nije promijenio moj stav prema mikroračunalima, štoviše, sad još više stojim iza njega, ali mi je pružao lijep izazov.

Literatura

1. VIDI-X stranica: https://vidi-x.org/
2. Arduino stranica: https://www.arduino.cc/