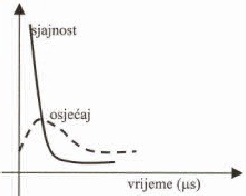
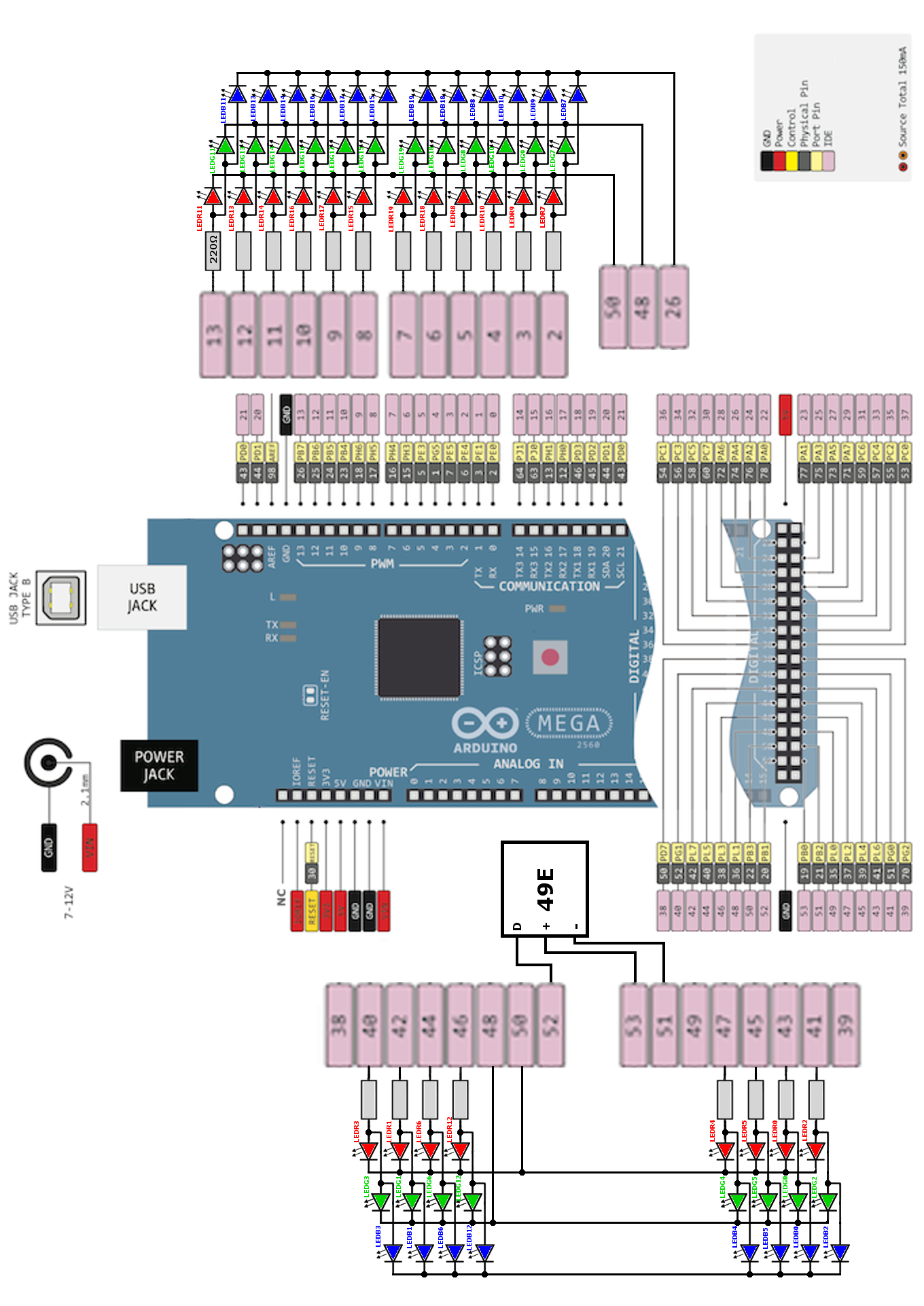
1. **UVOD**

Rotirajući LED displej služi za proizvodnju statične dvodimenzionalne slike koristeći samo tri reda vertikalno poredanih crvenih, zelenih i plavih LED dioda. Ovaj uređaj proizvodi sliku koja mu je zadata u programskom kodu tako što u tačno određenim trenucima pali odgovarajuće LED diode. To je omogućeno elektromotorom koji svojom rotacijom omogućava diodama da iskorištavaju osobine oka i principe koji se takođe koriste u televiziji i općenito pri reprodukciji slike u elektronici. Najznačajnija pojava za Rotirajući LED displej je perzistencija (tromost) oka. Perzistencija oka je produženje osjećaja svjetline po prestanku pobude. To znači da ako se svjetlina naglo poveća ili smanji, potrebno je da prođe izvjesno vrijeme da se u mozgu stvori utisak odgovarajuće svjetline.

Na sljedećoj slici je dat dijagram zavisnosti osjećaja svjetlosti o svjetlini izvora:



**2. ELEKTRONSKA SHEMA**

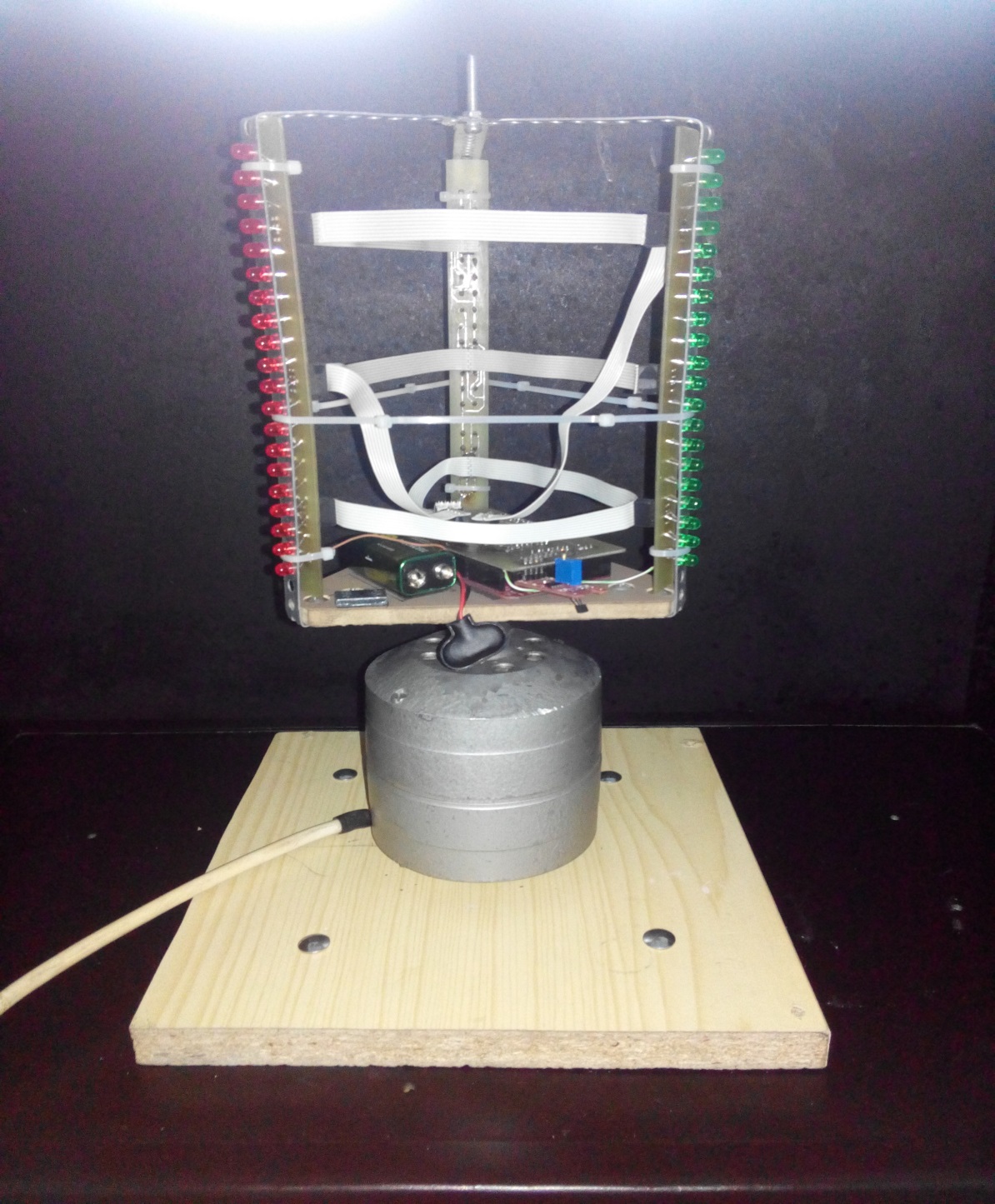
**3. SPECIFIKACIJA MATERIJALA**

|  |  |  |
| --- | --- | --- |
| **R. br.** | **Materijal** | **Količina** |
| 1. | LED dioda 5mm crvena | 20 kom |
| 2. | LED dioda 5mm zelena | 20 kom |
| 3. | LED dioda 5mm plava | 20 kom |
| 4. | Otpornik 220 Ω | 20 kom |
| 5. | Igličasti konektor muški | 129 iglica |
| 6 | Flat kabal 8 žica |  |
| 7. | Konektor za flat kabal ženski |  |
| 8. | Štampana ploča | 20cm x 15cm |
| 9. | Metalna traka sa rupama | 12mm x 1m |
| 10. | Kalaj za letovanje |  |
| 11. | Arduino Mega 2560 | 1 kom |
| 12. | Motor 50W 0.35A/220V/50Hz 1400 ob/min | 1 kom |
| 13. | Hall effect senzor 49E | 1 kom |

**4. PRINCIP RADA**

Nosač ramova sa LED diodama je pričvršćen na motor koji se okreće brzinom 1400 ob/min, te se napaja iz gradske mreže (220V/50Hz). Na jednom ramu se nalaze crvene, na drugom zelene, a na trećem ramu plave LED diode. LED diode su raspoređene po vertikali. Svaka grupa LED dioda se nalazi na svojoj štampanoj ploči preko koje ostvaruje spoj sa mikrokontrolerom. Sa štampanom pločom mikrokontrolera, pomoću igličastih konektora, povezana je dopunska štampana ploča koja je dalje povezana sa pojedinačnim štampanim pločama LED dioda. Dopunska štampana ploča je povezana sa štampanim pločama LED dioda pomoću flat kabla i odgovarajućih konektora.

Mikrokontroler je Arduino Mega 2560 koji radi na principu mikroprocesora ATMega 2560. Arduino Mega 2560 nam omogućava da ga pomoću računara programiramo, programom napisanim u programskom jeziku C.



**4.1 Upravljanje LED diodama**

Rotirajući LED displej radi slično LCD monitoru. Razlika je u tome što LCD monitor ispisuje horizontalne linije frekvencijom 60Hz, a rotirajući LED displej ispisuje vertikalne linije. U mikrokontroler je programski unesena matrica koja sadrži podatke o boji svakog „piksela“ slike. Boja piksela se dobija aditivnim miješanjem određenih količina svjetlosti primarnih boja. Primarne boje su crvena, zelena i plava (RGB). Matrica je u vidu dvodimenzionalnog niza, tipa unsigned long (32-bitni pozitivni cijeli broj). Podaci o boji su pohranjeni u heksadecimalnom zapisu, iz kojeg se kasnije izdvajaju vrijednosti pojedinih R, G i B komponenata.

Npr. žuta boja zapisana u heksadecimalnom obliku je: FFFF00.

Svjetlina dioda se upravlja pomoću PWM (Impulsno širinske modulacije) signala. Naime, ne kontroliše se intenzitet svijetljenja diode, nego se mijenja odnos vremena kada je dioda upaljena i vremena kada je ugašena:

Pošto je taj signal jako velike frekvencije, taj odnos ljudsko oko interpretira kao svjetlinu, zbog perzistencije oka.

**4.2 Određivanje pozicije sistema**

Na dopunskoj štampanoj ploči se nalazi Hall efekt senzor koji generiše impuls svaki put kada naiđe na magnetno polje. Pored uređaja se nalazi stalni magnet. Svaki put kada Hall efekt senzor prođe pored tog magneta, pošalje impuls mikrokontroleru. Mikrokontroler mjeri vrijeme između susjednih impulsa, te na osnovu toga mjeri frekvenciju okretanja.

Na osnovu frekvencije okretanja, u programu se određuje trenutni ugao sistema. Ram sa crvenim diodama se uzima kao referentni, tj. trenutni ugao tog rama je ujedno i ugao sistema. On se računa po sljedećoj formuli:

Mikrokontroler nakon prijema impulsa od induktivnog senzora, takođe resetuje trenutni ugao (izjednači ga sa nulom) da bi se neutralisale blage nepreciznosti mjerenja vremena. Kada se ne bi resetovao taj ugao, onda bi se slika blago rotirala, a to je nepoželjna pojava.

**4.3 Ispisivanje linija**

Mikrokontroler je programiran tako da u određenom trenutku iz matrice slike uzima podatke o pikselima u vertikalnom nizu koji odgovara trenutnom uglu. U svakom trenutku se mjeri ugao i pali odgovarajuća LED dioda, određenom svjetlinom. Naizmjenično se pale crvene, zelene i plave diode, u tom redoslijedu. U jednom trenutku mogu da svijetle samo diode jedne boje.

**5. PROGRAMSKI KOD**

Programski kod napisan je u programskom jeziku C. Pojašnjenja određenih dijelova programa data su u vidu komentara.

**NAPOMENA**: Ovaj program direktno manipuliše registre mikroprocesora. Smije se samo koristiti na uređaju Arduino Mega 2560!

#include <math.h>

#define RED 50

#define GREEN 48

#define BLUE 26

#define MATRIX\_HEIGHT 20

#define MATRIX\_WIDTH 19

#define cbi(sfr, bit) (\_SFR\_BYTE(sfr) &= ~\_BV(bit))

#define sbi(sfr, bit) (\_SFR\_BYTE(sfr) |= \_BV(bit))

double angle\_r = 0, frequency = .5;

uint8\_t index = 0, i = 0;//Pomoćne varijable

uint16\_t deltaTime = 0;

unsigned long lastTrigger = 0;//Trenutak kada je zadnji put senzor prošao pored magneta

unsigned long currentTime = 0, lastTime = 0;//Varijable koje služe za mjerenje trajanja funkcije loop

bool prevRead = 0, red = 0, green = 0, digRead = 0;

unsigned long matrix[MATRIX\_HEIGHT][MATRIX\_WIDTH] = { //Matrica sa podacima o boji

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }

};

volatile uint8\_t \*out;

uint8\_t red\_matrix[MATRIX\_HEIGHT][MATRIX\_WIDTH],

green\_matrix[MATRIX\_HEIGHT][MATRIX\_WIDTH],//Matrice sa pojedinačnim podacima o svjetlini

blue\_matrix[MATRIX\_HEIGHT][MATRIX\_WIDTH];

const double \_30 = PI / 6, \_150 = 5 \* \_30, \_3\_2PI = 3.0 / (2.0 \* PI), //Konstante

\_270 = 3 \* PI / 2, full\_circle = 2 \* PI / 1000000, \_360 = 2 \* PI;

uint8\_t nOverflows = 0, timer\_ticks = 0, pwm\_resolution = 128;

const uint8\_t pins[MATRIX\_HEIGHT] = {43, 42, 41, 40, 47, 45, 44, 2, 5, 3, 4, 13, 46, 12, 11, 8, 10, 9, 6, 7};//Tabela //pinova

void DigitalWrite(uint8\_t pin, bool val) //Ubrzana verzija funkcije digitalWrite

{

out = portOutputRegister(digitalPinToPort(pin));

if (val) \*out |= digitalPinToBitMask(pin);

else \*out &= ~digitalPinToBitMask(pin);

}

bool DigitalRead(uint8\_t pin) //Ubrzana verzija funkcije digitalRead

{

return \*portInputRegister(digitalPinToPort(pin)) & digitalPinToBitMask(pin);

}

void AnalogWrite(uint8\_t pin, int val) //Ubrzana verzija funkcije analogWrite

{

if (val == 0)

digitalWrite(pin, LOW);

else if (val == 255)

digitalWrite(pin, HIGH);

else

{

switch(digitalPinToTimer(pin))

{

#if defined(TCCR0) && defined(COM00) && !defined(\_\_AVR\_ATmega8\_\_)

case TIMER0A:

sbi(TCCR0, COM00); OCR0 = val; break;

#endif

#if defined(TCCR0A) && defined(COM0A1)

case TIMER0A: sbi(TCCR0A, COM0A1); OCR0A = val; break;

#endif

#if defined(TCCR0A) && defined(COM0B1)

case TIMER0B: sbi(TCCR0A, COM0B1); OCR0B = val; break;

#endif

#if defined(TCCR1A) && defined(COM1A1)

case TIMER1A: sbi(TCCR1A, COM1A1); OCR1A = val; break;

#endif

#if defined(TCCR1A) && defined(COM1B1)

case TIMER1B: sbi(TCCR1A, COM1B1); OCR1B = val; break;

#endif

#if defined(TCCR1A) && defined(COM1C1)

case TIMER1C: sbi(TCCR1A, COM1C1); OCR1C = val; break;

#endif

#if defined(TCCR2) && defined(COM21)

case TIMER2: sbi(TCCR2, COM21); OCR2 = val; break;

#endif

#if defined(TCCR2A) && defined(COM2A1)

case TIMER2A: sbi(TCCR2A, COM2A1); OCR2A = val; break;

#endif

#if defined(TCCR2A) && defined(COM2B1)

case TIMER2B: sbi(TCCR2A, COM2B1); OCR2B = val; break;

#endif

#if defined(TCCR3A) && defined(COM3A1)

case TIMER3A: sbi(TCCR3A, COM3A1); OCR3A = val; break;

#endif

#if defined(TCCR3A) && defined(COM3B1)

case TIMER3B: sbi(TCCR3A, COM3B1); OCR3B = val; break;

#endif

#if defined(TCCR3A) && defined(COM3C1)

case TIMER3C: sbi(TCCR3A, COM3C1); OCR3C = val; break;

#endif

#if defined(TCCR4A)

case TIMER4A: sbi(TCCR4A, COM4A1);

#if defined(COM4A0)

cbi(TCCR4A, COM4A0);

#endif

OCR4A = val; break;

#endif

#if defined(TCCR4A) && defined(COM4B1)

case TIMER4B: sbi(TCCR4A, COM4B1); OCR4B = val; break;

#endif

#if defined(TCCR4A) && defined(COM4C1)

case TIMER4C: sbi(TCCR4A, COM4C1); OCR4C = val; break;

#endif

#if defined(TCCR4C) && defined(COM4D1)

case TIMER4D: sbi(TCCR4C, COM4D1);

#if defined(COM4D0)

cbi(TCCR4C, COM4D0);

#endif

OCR4D = val; break;

#endif

#if defined(TCCR5A) && defined(COM5A1)

case TIMER5A: sbi(TCCR5A, COM5A1); OCR5A = val; break;

#endif

#if defined(TCCR5A) && defined(COM5B1)

case TIMER5B: sbi(TCCR5A, COM5B1); OCR5B = val; break;

#endif

#if defined(TCCR5A) && defined(COM5C1)

case TIMER5C: sbi(TCCR5A, COM5C1); OCR5C = val; break;

#endif

}

}

}

void \_AnalogWrite(uint8\_t pin, uint8\_t val) //Funkcija koja simulira PWM

{

DigitalWrite(pin, timer\_ticks + nOverflows <= val \* pwm\_resolution / 256);

}

void setup()

{

for (int i = 0; i <= 53; i++)

{

pinMode(i, OUTPUT);

digitalWrite(i, LOW);

}

pinMode(52, INPUT); //Pripremanje izlaza za upotrebu

digitalWrite(53, HIGH);

pinMode(49, INPUT);

for (i = 0; i < MATRIX\_HEIGHT; i++)

for (uint8\_t j = 0; j < MATRIX\_WIDTH; j++)

red\_matrix[i][j] =(matrix[i][j] >> 16) & 0xff;

for (i = 0; i < MATRIX\_HEIGHT; i++)

for (uint8\_t j = 0; j < MATRIX\_WIDTH; j++) //Izdvajanje podataka o pojedinim bojama

green\_matrix[i][j] = (matrix[i][j] >> 8) & 0xff;

for (i = 0; i < MATRIX\_HEIGHT; i++)

for (int j = 0; j < MATRIX\_WIDTH; j++)

blue\_matrix[i][j] = (uint8\_t)((matrix[i][j] & 0xff) \* 0.2);//Plave diode su jače pa im smanjujemo svjetlinu

}

void loop()

{

if ((currentTime = 4 \* TCNT0) > lastTime)

deltaTime = (uint16\_t)(currentTime - lastTime);//Mjerenje vremena funkcije loop

lastTime = currentTime;

angle\_r += full\_circle \* frequency \* deltaTime; //Određivanje ugla formulom α = 2πft

if (angle\_r >= \_360)

angle\_r = 0;

red = \_30 < angle\_r && angle\_r < \_150;//Određivanje da li se pale crvene LED

green = \_150 < angle\_r && angle\_r < \_270;//Određivanje da li se pale zelene LED

if (!red)

DigitalWrite(RED, HIGH);//Da ne bi svijetlile crvene, na katodu se dovodi +5V(inverzna polarizacija)

if (!green)

DigitalWrite(GREEN, HIGH);//Da ne bi svijetlile zelene, na katodu se dovodi +5V(inverzna polarizacija)

if (red || green)

DigitalWrite(BLUE, HIGH);//Da ne bi svijetlile plave, na katodu se dovodi +5V(inverzna polarizacija)

if (timer\_ticks > TCNT0)

nOverflows = nOverflows < 255 ? nOverflows + 256 : 0;//Mjerenje vremena za simulirane PWM izlaze

timer\_ticks = TCNT0;

if (red) //Ispisivanje crvenih komponenti vertikalnih linija

{

index = (uint8\_t)((MATRIX\_WIDTH - 1) \* (1.25 - \_3\_2PI \* angle\_r));

for (i = 0; i < 5; i++)

\_AnalogWrite(pins[i], red\_matrix[i][index]);//Paljenje dioda na simuliranim PWM izlazima

for (i = 5; i < MATRIX\_HEIGHT; i++)

AnalogWrite(pins[i], red\_matrix[i][index]);//Paljenje dioda na PWM izlazima

DigitalWrite(RED, LOW);

}

else if (green) //Ispisivanje zelenih komponenti vertikalnih linija

{

index = (uint8\_t)((MATRIX\_WIDTH - 1) \* (2.25 - \_3\_2PI \* angle\_r));

for (i = 0; i < 5; i++)

\_AnalogWrite(pins[i], green\_matrix[i][index]);//Paljenje dioda na simuliranim PWM izlazima

for (i = 5; i < MATRIX\_HEIGHT; i++)

AnalogWrite(pins[i], green\_matrix[i][index]);//Paljenje dioda na PWM izlazima

DigitalWrite(GREEN, LOW); //Dovođenje katode na niži potencijal

}

else //Ispisivanje plavih komponenti vertikalnih linija

{

index = (uint8\_t)((MATRIX\_WIDTH - 1) \* (3.25 - \_3\_2PI \* angle\_r));

for (i = 0; i < 5; i++)

\_AnalogWrite(pins[i], blue\_matrix[i][index]);//Paljenje dioda na simuliranim PWM izlazima

for (i = 5; i < MATRIX\_HEIGHT; i++)

AnalogWrite(pins[i], blue\_matrix[i][index]);//Paljenje dioda na PWM izlazima

DigitalWrite(BLUE, LOW);

}

if (timer\_ticks + nOverflows > pwm\_resolution)

nOverflows = 0;

digRead = DigitalRead(52);//Očitanje signala sa Hall efekt senzora

if (prevRead == 0 && digRead) //Pokreće se pri uzlaznoj ivici impulsa

{

if ((currentTime = micros()) > lastTrigger)

{

angle\_r = 0;//Resetovanje ugla

frequency = 1000000.0 / (currentTime - lastTrigger); //Mjerenje frekvencije

lastTrigger = currentTime;

prevRead = 1;

}

}

else if (!digRead)

prevRead = 0;

}