

Mikroupravljački sustav za otkrivanje požara

Grgić, Matko

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:200:332237>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-11-27**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science and Information Technology Osijek](#)



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH
TEHNOLOGIJA OSIJEK**

**MIKROUPRAVLJAČKI SUSTAV ZA OTKRIVANJE
POŽARA**

Završni rad

Matko Grgić

Osijek, 2023.

SADRŽAJ

1. UVOD	1
1.1. Zadatak rada	1
2. PRIMJENJENE TEHNOLOGIJA I ALATI	2
2.1. ESPDUINO-32 – mikroupravljačko sklopovlje	2
2.2. MLX90641 – Infracrveni senzor topline sa 16x12 matricom	2
2.3. Periferni uređaji	3
2.4. Arduino IDE	4
2.5. Visual Studio Code.....	5
3. REALIZACIJA SUSTAVA	6
3.1. Načini detekcije požara	6
3.2. Infracrveno zračenje	6
3.3. Razumijevanje i spajanje senzora.....	7
3.4. Izrada biblioteke	8
3.5. Programsko rješenje sustava.....	10
4. EKSPERIMENT	17
5. ZAKLJUČAK	19
LITERATURA	20
SAŽETAK	22
ABSTRACT	23
ŽIVOTOPIS	24
PRILOZI	25

1. UVOD

U suvremenom društvu, zaštita ljudi, imovine i okoliša od požara predstavlja izazov od velike važnosti. Upravljanje rizicima vezanim uz požare zahtijeva sve sofisticiranija tehnološka rješenja, a mikroupravljački sustavi postaju ključni elementi u ostvarivanju visoke razine sigurnosti. Mikroupravljački sustavi su integrirani sklopovi temeljeni na mikrokontrolerima ili mikroprocesorima, sposobni za precizno izvođenje niza zadataka u realnom vremenu. Njihova primjena u domenama kao što su otkrivanje požara omogućuje brzu, pouzdanu i inteligentnu reakciju na potencijalno opasne situacije.

Ovaj završni rad usredotočuje se na analizu, dizajn i implementaciju mikroupravljačkog sustava namijenjenog otkrivanju požara. Kroz sustavan pristup, istraživački rad istražuje različite aspekte vezane uz detekciju požara, uključujući različite tipove senzora za detekciju topline i plamena, metode obrade signala te algoritme za donošenje odluka.

U nastavku rada, detaljno je opisan proces projektiranja sustava, od odabira komponenata do implementacije softverskih algoritama. Također, proučeni su rezultati eksperimenata koji demonstriraju učinkovitost i pouzdanost razvijenog mikroupravljačkog sustava u kontekstu otkrivanja požara. Kroz ovu analizu, rad doprinosi sveobuhvatnijem razumijevanju uloge mikroupravljačkih sustava u poboljšanju sustava zaštite od požara te potiče daljnja istraživanja i inovacije u ovom važnom području.

U prvom poglavlju rada predstavljeni su osnovni ciljevi i motivacija za izradu završnog rada na zadanu temu. Drugo poglavlje daje detaljan prikaz svih korištenih alata i tehnologija nužnih za izradu rada dok treće poglavlje pruža pogled u detalje izrade fizičkog modela rada kao i programske podrške za isti. Četvrto poglavlje je opis održanog eksperimenta te rezultati i zaključci izvedeni iz eksperimenta.

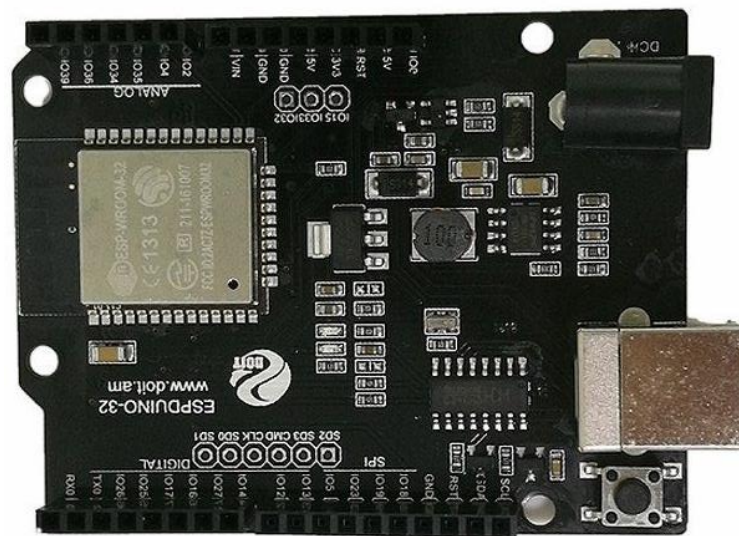
1.1. Zadatak rada

Zadatak rada je napraviti mikroupravljački sustav za otkrivanje požara. Glavni element rada je Melexis MLX90641 infracrveni senzor topline. Potrebno je razumjeti način njegovog djelovanja te ga povezati s mikroupravljačem ESPduino-32 tvrtke DOIT. Potrebno je osigurati i odgovarajuću programsku podršku te periferne uređaje. Na kraju sve zajedno povezati i omogućiti ispis na zaslon.

2. PRIMJENJENE TEHNOLOGIJA I ALATI

2.1. ESPDUINO-32 – mikroupravljačko sklopovlje

DOIT ESPduino-32 (Sl. 2.1.) je popularano mikroupravljačko sklopovlje koji se često koristi u različitim elektroničkim projektima. Njime upravlja Xtensa LX6 mikroprocesor, a uz njega na tiskanoj pločici se nalaze WiFi modul za bežični prijenos podataka i pristup internetu, Bluetooth 4.2 te modul za upravljanjem u stvarnom vremenu. Napaja se preko USB ulaza ili konektora istosmjerne struje na napon između 5 i 12V te strujom od 80mA [1].



Sl. 2.1. DOIT ESPduino-32 pločica s Xtensa LX6 mikrokontrolerom i perifernim uređajima [1]

2.2. MLX90641 – Infracrveni senzor topline sa 16x12 matricom

MLX90641 je industrijski standardiziran senzor topline s 16x12 matičnim prikazom, tj. 192 jedinstvena infracrvena senzora. Zapakiran je u TO-39 pakiranje s digitalnim sučeljem. Radi na 3.3V s jačinom struje od 12mA. Omogućava jednostavnu I2C komunikaciju brzine do 1MHz što ga čini pogodnim za jednostavne, ali i složenije elektroničke projekte. Neke od njegovih upotreba u stvarnom svijetu su: beskontaktno mjerenje temperature, mikrovalne pećnice, prepoznavanje pokreta, klima uređaji s automatskim paljenjem i sl. Senzor ima četiri pina – dva za I2C komunikaciju (SDA i SCL) te dva za spajanje samog uređaja u strujni krug [2].



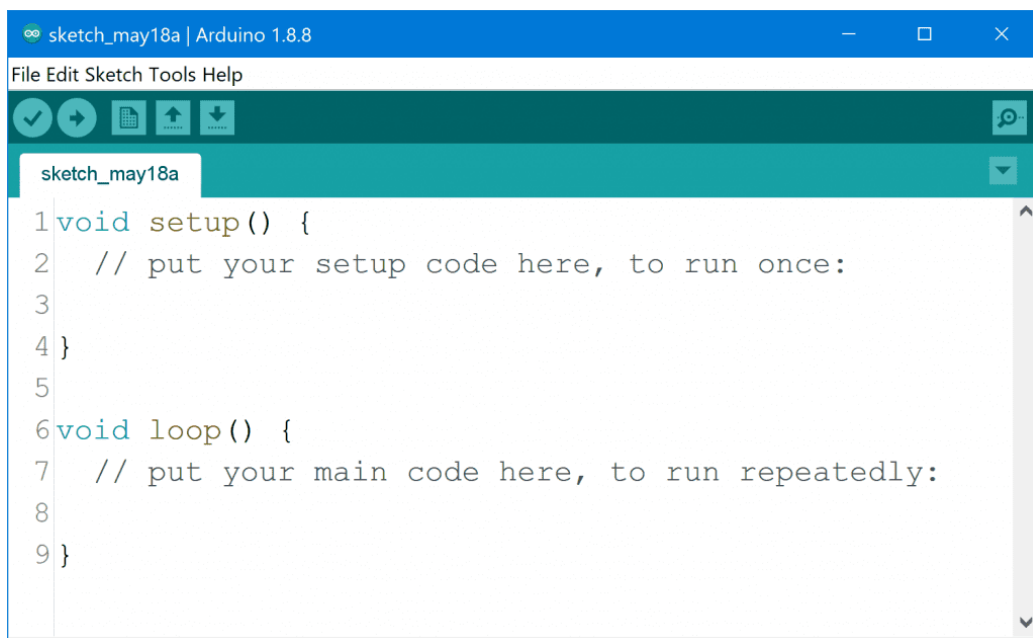
Sl. 2.2. MLX90641 senzor topline sa 16x12 matričnim prikazom [3]

2.3. Periferni uređaji

Niti jedan elektronički projekt nije moguć bez perifernih uređaja kao što su: otpornici, vodiči i diode. Za ovaj projekt korišteni su otpornici od 300 i 2200 Ω te LE dioda za označavanje alarmantnog stanja. Otpornici su nužni za održavanje sustava u stabilnom stanju – kada nema I2C komunikacije potrebno je pomoću pull-up otpornika dovesti logičku jedinicu na ulaze I2C komunikacije, a kad se događa prijenos podataka dolazi logička nula koja označava početak. Otpornici također služe za zaštitu ostalih perifernih uređaja, što je u ovom slučaju dioda, od pregrijavanja smanjujući struju koja prolazi kroz uređaj. Za realizaciju sustava korištena je i eksperimentalna pločica (*engl. breadboard*). Ova pločica idealna je za eksperimentalne sustave jer omogućuje spajanje komponenti na glavnu upravljačku jedinicu bez trajnog lemljenja komponenti. Sadrži velik broj „rupa“ (*engl. pins*) koji služe kao mjesto kontakta za vodiče. Redci pločice povezani su u seriju, pa se može gledati svaki redak kao zasebno mjesto kontakta. Ukratko, zbog svoje jednostavnosti eksperimentalne pločice često su korištenje za edukaciju o osnovama rada elektroničkih krugova, ali i za ozbiljnije eksperimentalne radove.

2.4. Arduino IDE

Arduino Integrated Development Environment (engl. *Arduino integrirano razvojno sučelje*) je program tvrtke Arduino poznat po svojim brojnim razvojnim pločicama. Ovo sučelje je programski dodatak tim, ali i mnogim drugim pločicama, koje omogućuje komunikaciju s hardverom, programiranje istog te njegovo testiranje. Sadrži uređivač teksta, prozor za izlaz, Serial monitor – korisni alat za provjeru rada koda preko fizičkog mikroupravljača i brojne druge mogućnosti. Budući da je sučelje kroz godine nadograđivano kako bi podržalo brojne uređaje, razvojni inženjeri su morali pisati svoje biblioteke kodova za svoje sklopovlje te je i gotove biblioteke moguće instalirati preko ugrađenog pretraživača. Dvije su glavne funkcije u kodu pisanog u Arduino IDE – `setup()` i `loop()`. `Setup()` (engl. *postavljanje*) je funkcija koja se izvodi samo jednom i poziva se na početku izvođenja programa. U njemu se govori koji izlazi se koristi i u koju svrhu, definiranje biblioteka, inicijalizacija varijabli i serijske komunikacije i sl. Nakon što su određeni početni uvjeti izvodi se `loop()` (engl. *ponavljanje*) funkcija. Kao što joj ime kaže, sav kod u njoj se ponavlja sve dok ne dođe do prekida programa. Na taj način moguće je čitati nove podatke koje senzor prikuplja, paliti i gasiti LED-icu i dr. Arduino programsko sučelje podržano je na svim većim platformama uključujući Windows, MacOS i Linux. [4]

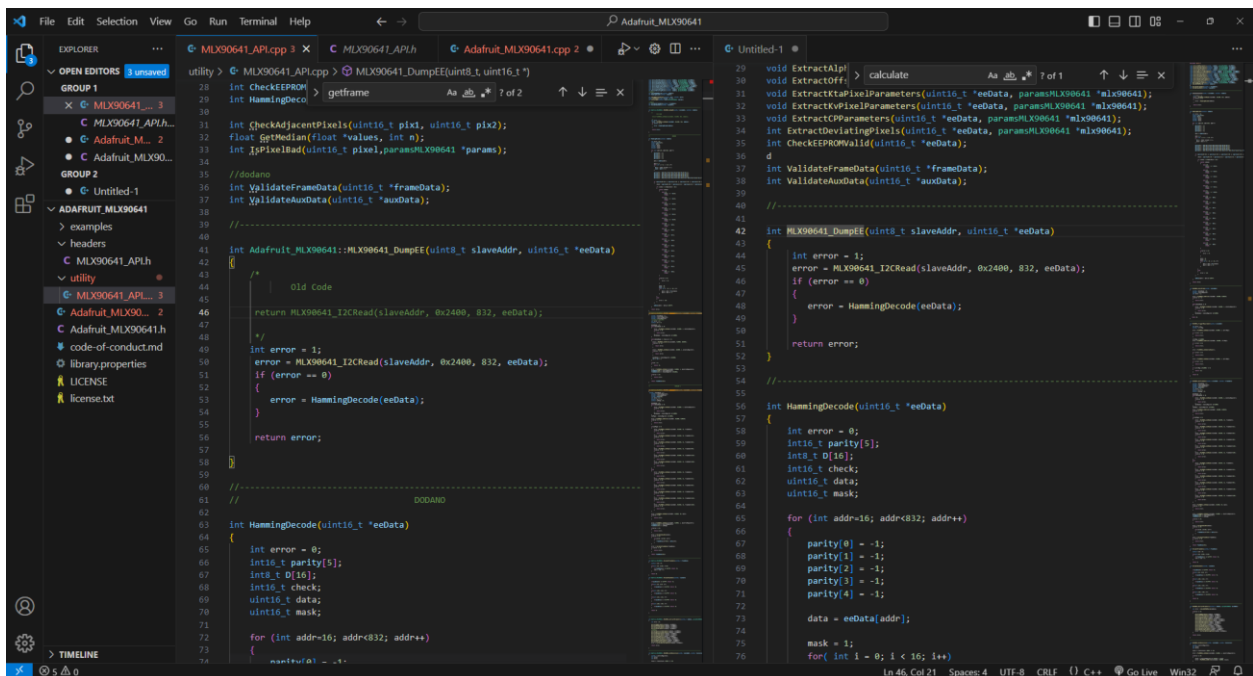


```
sketch_may18a | Arduino 1.8.8
File Edit Sketch Tools Help
sketch_may18a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

Sl. 2.4. Arduino integrirano razvojno sučelje [5]

2.5. Visual Studio Code

Visual Studio Code (VS Code) je široko korišteni alat tvrtke Microsoft koji služi za uređivanje teksta, odnosno koda. Uz svoju potporu za brojne programske i opisne jezike, jednostavan je i intuitivan, ali i vrlo brz što ga čini povoljnim za širok spektar korisnika. Sadrži i inteligentna svojstva koja omogućuju samostalno završavanje započetih kodova, ali i predlaže sljedeće korake u pisanju programa. Ono što ga odvaja od ostalih alata slične prirode je njegova tzv. trgovina dodatcima. Slično kao i Arduino, postoje mnoga proširenja uključena u program od strane zajednice programera, kao i samog Microsofta, koja olakšavaju pisanje, uređivanje i razumijevanje koda. Program je besplatan, redovno nadograđivan i pouzdan što ga izdvaja kao jednog od najboljih alata u svojoj kategoriji. [6]



```
28 int CheckEEPROM
29 int HammingDeco
30
31 int (checkAdjacentPixels(uint16_t pix1, uint16_t pix2);
32 float getMedian(float *values, int n);
33 int (getPixelIAD(uint16_t pixel, params_t *params);
34
35 //dodano
36 int (validateFrameData(uint16_t *frameData);
37 int (validateAuxData(uint16_t *auxData);
38
39 //-----
40
41 int Adafruit_MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
42 {
43     // Old Code
44     return MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
45 }
46
47 //
48
49 int error = 1;
50 error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
51 if (error == 0)
52 {
53     error = HammingDecode(eeData);
54 }
55
56 return error;
57
58 //-----
59
60 //-----
61 //-----
62
63 int HammingDecode(uint16_t *eeData)
64 {
65     int error = 0;
66     int16_t parity[5];
67     int8_t D[16];
68     int16_t check;
69     uint16_t data;
70     uint16_t mask;
71
72     for (int addr=16; addr<832; addr++)
73     {
74         parity[0] = -1;
75         parity[1] = -1;
76         parity[2] = -1;
77         parity[3] = -1;
78         parity[4] = -1;
79         data = eeData[addr];
80         mask = 1;
81         for (int i = 0; i < 16; i++)
82         {
83             calculate
84             void ExtractOff;
85             void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
86             void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
87             int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
88             int CheckEEPROMValid(uint16_t *eeData);
89             d
90
91             int ValidateFrameData(uint16_t *frameData);
92             int ValidateAuxData(uint16_t *auxData);
93
94             //-----
95             //-----
96             int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
97             {
98                 int error = 1;
99                 error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
100                 if (error == 0)
101                 {
102                     error = HammingDecode(eeData);
103                 }
104                 return error;
105             }
106
107             int HammingDecode(uint16_t *eeData)
108             {
109                 int error = 0;
110                 int16_t parity[5];
111                 int8_t D[16];
112                 int16_t check;
113                 uint16_t data;
114                 uint16_t mask;
115
116                 for (int addr=16; addr<832; addr++)
117                 {
118                     parity[0] = -1;
119                     parity[1] = -1;
120                     parity[2] = -1;
121                     parity[3] = -1;
122                     parity[4] = -1;
123                     data = eeData[addr];
124                     mask = 1;
125                     for (int i = 0; i < 16; i++)
126                     {
127                         calculate
128                         void ExtractOff;
129                         void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
130                         void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
131                         int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
132                         int CheckEEPROMValid(uint16_t *eeData);
133                         d
134
135                         int ValidateFrameData(uint16_t *frameData);
136                         int ValidateAuxData(uint16_t *auxData);
137
138                         //-----
139                         //-----
140                         int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
141                         {
142                             int error = 1;
143                             error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
144                             if (error == 0)
145                             {
146                                 error = HammingDecode(eeData);
147                             }
148                             return error;
149                         }
150
151                         int HammingDecode(uint16_t *eeData)
152                         {
153                             int error = 0;
154                             int16_t parity[5];
155                             int8_t D[16];
156                             int16_t check;
157                             uint16_t data;
158                             uint16_t mask;
159
160                             for (int addr=16; addr<832; addr++)
161                             {
162                                 parity[0] = -1;
163                                 parity[1] = -1;
164                                 parity[2] = -1;
165                                 parity[3] = -1;
166                                 parity[4] = -1;
167                                 data = eeData[addr];
168                                 mask = 1;
169                                 for (int i = 0; i < 16; i++)
170                                 {
171                                     calculate
172                                     void ExtractOff;
173                                     void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
174                                     void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
175                                     int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
176                                     int CheckEEPROMValid(uint16_t *eeData);
177                                     d
178
179                                     int ValidateFrameData(uint16_t *frameData);
180                                     int ValidateAuxData(uint16_t *auxData);
181
182                                     //-----
183                                     //-----
184                                     int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
185                                     {
186                                         int error = 1;
187                                         error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
188                                         if (error == 0)
189                                         {
190                                             error = HammingDecode(eeData);
191                                         }
192                                         return error;
193                                     }
194
195                                     int HammingDecode(uint16_t *eeData)
196                                     {
197                                         int error = 0;
198                                         int16_t parity[5];
199                                         int8_t D[16];
200                                         int16_t check;
201                                         uint16_t data;
202                                         uint16_t mask;
203
204                                         for (int addr=16; addr<832; addr++)
205                                         {
206                                             parity[0] = -1;
207                                             parity[1] = -1;
208                                             parity[2] = -1;
209                                             parity[3] = -1;
210                                             parity[4] = -1;
211                                             data = eeData[addr];
212                                             mask = 1;
213                                             for (int i = 0; i < 16; i++)
214                                             {
215                                                 calculate
216                                                 void ExtractOff;
217                                                 void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
218                                                 void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
219                                                 int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
220                                                 int CheckEEPROMValid(uint16_t *eeData);
221                                                 d
222
223                                                 int ValidateFrameData(uint16_t *frameData);
224                                                 int ValidateAuxData(uint16_t *auxData);
225
226                                                 //-----
227                                                 //-----
228                                                 int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
229                                                 {
230                                                     int error = 1;
231                                                     error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
232                                                     if (error == 0)
233                                                     {
234                                                         error = HammingDecode(eeData);
235                                                     }
236                                                     return error;
237                                                 }
238
239                                                 int HammingDecode(uint16_t *eeData)
240                                                 {
241                                                     int error = 0;
242                                                     int16_t parity[5];
243                                                     int8_t D[16];
244                                                     int16_t check;
245                                                     uint16_t data;
246                                                     uint16_t mask;
247
248                                                     for (int addr=16; addr<832; addr++)
249                                                     {
250                                                         parity[0] = -1;
251                                                         parity[1] = -1;
252                                                         parity[2] = -1;
253                                                         parity[3] = -1;
254                                                         parity[4] = -1;
255                                                         data = eeData[addr];
256                                                         mask = 1;
257                                                         for (int i = 0; i < 16; i++)
258                                                         {
259                                                             calculate
260                                                             void ExtractOff;
261                                                             void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
262                                                             void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
263                                                             int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
264                                                             int CheckEEPROMValid(uint16_t *eeData);
265                                                             d
266
267                                                             int ValidateFrameData(uint16_t *frameData);
268                                                             int ValidateAuxData(uint16_t *auxData);
269
270                                                             //-----
271                                                             //-----
272                                                             int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
273                                                             {
274                                                                 int error = 1;
275                                                                 error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
276                                                                 if (error == 0)
277                                                                 {
278                                                                     error = HammingDecode(eeData);
279                                                                 }
280                                                                 return error;
281                                                             }
282
283                                                             int HammingDecode(uint16_t *eeData)
284                                                             {
285                                                                 int error = 0;
286                                                                 int16_t parity[5];
287                                                                 int8_t D[16];
288                                                                 int16_t check;
289                                                                 uint16_t data;
290                                                                 uint16_t mask;
291
292                                                                 for (int addr=16; addr<832; addr++)
293                                                                 {
294                                                                     parity[0] = -1;
295                                                                     parity[1] = -1;
296                                                                     parity[2] = -1;
297                                                                     parity[3] = -1;
298                                                                     parity[4] = -1;
299                                                                     data = eeData[addr];
300                                                                     mask = 1;
301                                                                     for (int i = 0; i < 16; i++)
302                                                                     {
303                                                                         calculate
304                                                                         void ExtractOff;
305                                                                         void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
306                                                                         void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
307                                                                         int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
308                                                                         int CheckEEPROMValid(uint16_t *eeData);
309                                                                         d
310
311                                                                         int ValidateFrameData(uint16_t *frameData);
312                                                                         int ValidateAuxData(uint16_t *auxData);
313
314                                                                         //-----
315                                                                         //-----
316                                                                         int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
317                                                                         {
318                                                                             int error = 1;
319                                                                             error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
320                                                                             if (error == 0)
321                                                                             {
322                                                                                 error = HammingDecode(eeData);
323                                                                             }
324                                                                             return error;
325                                                                         }
326
327                                                                         int HammingDecode(uint16_t *eeData)
328                                                                         {
329                                                                             int error = 0;
330                                                                             int16_t parity[5];
331                                                                             int8_t D[16];
332                                                                             int16_t check;
333                                                                             uint16_t data;
334                                                                             uint16_t mask;
335
336                                                                             for (int addr=16; addr<832; addr++)
337                                                                             {
338                                                                                 parity[0] = -1;
339                                                                                 parity[1] = -1;
340                                                                                 parity[2] = -1;
341                                                                                 parity[3] = -1;
342                                                                                 parity[4] = -1;
343                                                                                 data = eeData[addr];
344                                                                                 mask = 1;
345                                                                                 for (int i = 0; i < 16; i++)
346                                                                                 {
347                                                                                     calculate
348                                                                                     void ExtractOff;
349                                                                                     void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
350                                                                                     void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
351                                                                                     int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
352                                                                                     int CheckEEPROMValid(uint16_t *eeData);
353                                                                                     d
354
355                                                                                     int ValidateFrameData(uint16_t *frameData);
356                                                                                     int ValidateAuxData(uint16_t *auxData);
357
358                                                                                     //-----
359                                                                                     //-----
360                                                                                     int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
361                                                                                     {
362                                                                                         int error = 1;
363                                                                                         error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
364                                                                                         if (error == 0)
365                                                                                         {
366                                                                                             error = HammingDecode(eeData);
367                                                                                         }
368                                                                                         return error;
369                                                                                     }
370
371                                                                                     int HammingDecode(uint16_t *eeData)
372                                                                                     {
373                                                                                         int error = 0;
374                                                                                         int16_t parity[5];
375                                                                                         int8_t D[16];
376                                                                                         int16_t check;
377                                                                                         uint16_t data;
378                                                                                         uint16_t mask;
379
380                                                                                         for (int addr=16; addr<832; addr++)
381                                                                                         {
382                                                                                             parity[0] = -1;
383                                                                                             parity[1] = -1;
384                                                                                             parity[2] = -1;
385                                                                                             parity[3] = -1;
386                                                                                             parity[4] = -1;
387                                                                                             data = eeData[addr];
388                                                                                             mask = 1;
389                                                                                             for (int i = 0; i < 16; i++)
390                                                                                             {
391                                                                                                 calculate
392                                                                                                 void ExtractOff;
393                                                                                                 void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
394                                                                                                 void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
395                                                                                                 int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
396                                                                                                 int CheckEEPROMValid(uint16_t *eeData);
397                                                                                                 d
398
399                                                                                                 int ValidateFrameData(uint16_t *frameData);
400                                                                                                 int ValidateAuxData(uint16_t *auxData);
401
402                                                                                                 //-----
403                                                                                                 //-----
404                                                                                                 int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
405                                                                                                 {
406                                                                                                     int error = 1;
407                                                                                                     error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
408                                                                                                     if (error == 0)
409                                                                                                     {
410                                                                                                         error = HammingDecode(eeData);
411                                                                                                     }
412                                                                                                     return error;
413                                                                                                 }
414
415                                                                                                 int HammingDecode(uint16_t *eeData)
416                                                                                                 {
417                                                                                                     int error = 0;
418                                                                                                     int16_t parity[5];
419                                                                                                     int8_t D[16];
420                                                                                                     int16_t check;
421                                                                                                     uint16_t data;
422                                                                                                     uint16_t mask;
423
424                                                                                                     for (int addr=16; addr<832; addr++)
425                                                                                                     {
426                                                                                                         parity[0] = -1;
427                                                                                                         parity[1] = -1;
428                                                                                                         parity[2] = -1;
429                                                                                                         parity[3] = -1;
430                                                                                                         parity[4] = -1;
431                                                                                                         data = eeData[addr];
432                                                                                                         mask = 1;
433                                                                                                         for (int i = 0; i < 16; i++)
434                                                                                                         {
435                                                                                                             calculate
436                                                                                                             void ExtractOff;
437                                                                                                             void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
438                                                                                                             void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
439                                                                                                             int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
440                                                                                                             int CheckEEPROMValid(uint16_t *eeData);
441                                                                                                             d
442
443                                                                                                             int ValidateFrameData(uint16_t *frameData);
444                                                                                                             int ValidateAuxData(uint16_t *auxData);
445
446                                                                                                             //-----
447                                                                                                             //-----
448                                                                                                             int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
449                                                                                                             {
450                                                                                                                 int error = 1;
451                                                                                                                 error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
452                                                                                                                 if (error == 0)
453                                                                                                                 {
454                                                                                                                     error = HammingDecode(eeData);
455                                                                                                                 }
456                                                                                                                 return error;
457                                                                                                             }
458
459                                                                                                             int HammingDecode(uint16_t *eeData)
460                                                                                                             {
461                                                                                                                 int error = 0;
462                                                                                                                 int16_t parity[5];
463                                                                                                                 int8_t D[16];
464                                                                                                                 int16_t check;
465                                                                                                                 uint16_t data;
466                                                                                                                 uint16_t mask;
467
468                                                                                                                 for (int addr=16; addr<832; addr++)
469                                                                                                                 {
470                                                                                                                     parity[0] = -1;
471                                                                                                                     parity[1] = -1;
472                                                                                                                     parity[2] = -1;
473                                                                                                                     parity[3] = -1;
474                                                                                                                     parity[4] = -1;
475                                                                                                                     data = eeData[addr];
476                                                                                                                     mask = 1;
477                                                                                                                     for (int i = 0; i < 16; i++)
478                                                                                                                     {
479                                                                                                                         calculate
480                                                                                                                         void ExtractOff;
481                                                                                                                         void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
482                                                                                                                         void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
483                                                                                                                         int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
484                                                                                                                         int CheckEEPROMValid(uint16_t *eeData);
485                                                                                                                         d
486                                                                                                                         int ValidateFrameData(uint16_t *frameData);
487                                                                                                                         int ValidateAuxData(uint16_t *auxData);
488
489                                                                                                                         //-----
490                                                                                                                         //-----
491                                                                                                                         int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
492                                                                                                                         {
493                                                                                                                             int error = 1;
494                                                                                                                             error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
495                                                                                                                             if (error == 0)
496                                                                                                                             {
497                                                                                                                                 error = HammingDecode(eeData);
498                                                                                                                             }
499                                                                                                                             return error;
500                                                                                                                         }
501
502                                                                                                                         int HammingDecode(uint16_t *eeData)
503                                                                                                                         {
504                                                                                                                             int error = 0;
505                                                                                                                             int16_t parity[5];
506                                                                                                                             int8_t D[16];
507                                                                                                                             int16_t check;
508                                                                                                                             uint16_t data;
509                                                                                                                             uint16_t mask;
510
511                                                                                                                             for (int addr=16; addr<832; addr++)
512                                                                                                                             {
513                                                                                                                                     parity[0] = -1;
514                                                                                                                                     parity[1] = -1;
515                                                                                                                                     parity[2] = -1;
516                                                                                                                                     parity[3] = -1;
517                                                                                                                                     parity[4] = -1;
518                                                                                                                                     data = eeData[addr];
519                                                                                                                                     mask = 1;
520                                                                                                                                     for (int i = 0; i < 16; i++)
521                                                                                                                                     {
522                                                                                                                                         calculate
523                                                                                                                                         void ExtractOff;
524                                                                                                                                         void ExtractTAPixelParameters(uint16_t *eeData, params_t *paramsMLX90641);
525                                                                                                                                         void ExtractC9Parameters(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
526                                                                                                                                         int ExtractDeviatingPixels(uint16_t *eeData, paramsMLX90641 *paramsMLX90641);
527                                                                                                                                         int CheckEEPROMValid(uint16_t *eeData);
528                                                                                                                                         d
529
530                                                                                                                                         int ValidateFrameData(uint16_t *frameData);
531                                                                                                                                         int ValidateAuxData(uint16_t *auxData);
532
533                                                                                                                                         //-----
534                                                                                                                                         //-----
535                                                                                                                                         int MLX90641_DumpEE(uint8_t slaveAddr, uint16_t *eeData)
536                                                                                                                                         {
537                                                                                                                                             int error = 1;
538                                                                                                                                             error = MLX90641_I2CRead(slaveAddr, 0x2400, 832, eeData);
539                                                                                                                                             if (error == 0)
540                                                                                                                                             {
541                                                                                                                                                 error = HammingDecode(eeData);
542                                                                                                                                             }
543                                                                                                                                             return error;
544                                                                                                                                         }
545
546                                                                                                                                         int HammingDecode(uint16_t *eeData)
547                                                                                                                                         {
548                                                                                                                                             int error = 0;
549                                                                                                                                             int16_t parity[5];
550                                                                                                                                             int8_t D[16];
551                                                                                                                                             int16_t check;
552                                                                                                                                             uint16_t data;
553                                                                                                                                             uint16_t mask;
554
555                                                                                                                                             for (int addr=16; addr<832; addr++)
556                                                                                                                                             {
557                                                                                                                                                 parity[0] = -1;
558                                                                                                                                                 parity[1] = -1;
559                                                                                                                                                 parity[2] = -1;
560                                                                                                                                                 parity[3] = -1;
561                                                                                                                                                 parity[4] = -1;
562                                                                                                                                                 data = eeData[addr];
563                                                                                                                                                 mask = 1;
564                                                                                                                                                 for (int i = 0; i < 16; i++)
565                                                                                                                                                 {
566                                                                                                             
```

Sl. 2.5. Visual Studio Code uređivač koda

3. REALIZACIJA SUSTAVA

3.1. Načini detekcije požara

Postoji puno načina za detekciju požara, ali četiri su najčešća: ionizacijsko-fotoelektrični, fotoelektrični, ionizacijski i toplinski. Dok su toplinski senzori aktivirani povećanom temperaturom, ostali pale alarmantno stanje detekcijom dima [7]. Za ovaj rad korišten je toplinski senzor koji pomoću infracrvenog zračenja daje matrični prikaz temperatura svoje okoline.

3.2. Infracrveno zračenje

Radi lakšeg razumijevanja rada senzora potrebno je upoznati se s osnovama infracrvenog zračenja. Infracrveno zračenje vrsta je elektromagnetskih valova valne duljine između 0.75 μm i 1 mm. Zasluge za otkriće ove vrste zračenja upisuju se Fredericku Williamu Herschelu, engleskom astronomu, graditelju teleskopa i glazbeniku njemačkog podrijetla [8].

Promatrajući disperziju sunčeve svjetlosti na optičkoj prizmi primijetio je kako se više temperature nalaze na području crvenog dijela vidljivog spektra. Ta se vrsta zračenja ne može vidjeti, kao ni veći dio spektra elektromagnetskih valova, ali može ga se osjetiti na koži kao osjećaj topline. Infracrveno zračenje nastaje kao posljedica sudaranja čestica prilikom gibanja u prostoru, pri vibracijama kristalne rešetke čvrstih tijela te rotaciji kemijskih veza atomskih skupina u molekulama organskih tvari i plinova na svim temperaturama višim od apsolutne nule na kojoj svako gibanje staje [9]. Infracrveno zračenje, stoga, može se pojednostaviti i kao prikaz temperature nekog tijela ili plina.



Sl. 3.1. Toplinska (termalna) fotografija dviju osoba [10]

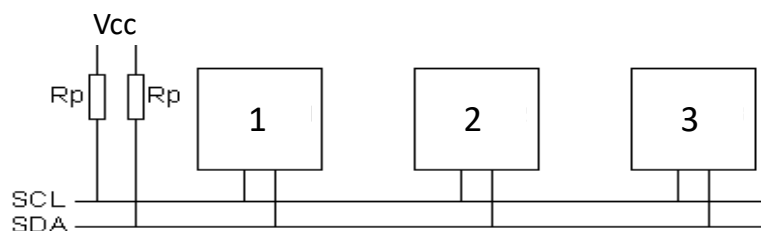
Na sl. 3.1. prikazana je toplinska fotografija. Koristeći srednje i duže vrijednosti valne duljine u infracrvenom spektru senzor „osjeti“ razliku topline. Tako je osoba, koja daje više topline od npr.

betonskog zida, prikazana svjetlijim (toplijim) bojama. Ova tehnologija često se koristi u sigurnosnim sustavima koji osiguravaju područje bez svjetla, primjerice vanjske prostore noću te područje s preprekama (šuma, polja s visokim raslinjem).

3.3. Razumijevanje i spajanje senzora

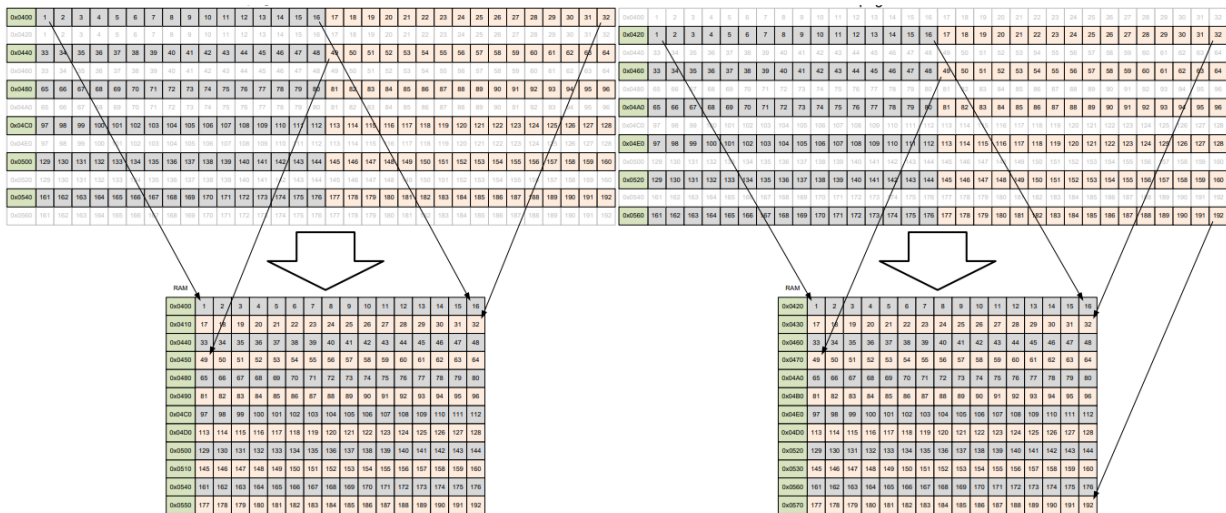
Prvi korak u izradi ovog rada je dobro se upoznati s glavnom komponentom. MLX90641 infracrveni senzor sa 16x12 matričnim prikazom okoline odabran je zbog svog jednostavnog izgleda i mogućnosti lakog rukovanja. Nalazi se u industrijski standardiziranom TO39 pakiranju – kućište napravljeno u obliku metalne kape, hermetički zatvoreno kako bi se mikročip zaštitio od okolnih utjecaja kao što su vlaga i zagađenje [11]. MLX90641 sadrži ambijentalni temperaturni senzor koji očitava vrijednosti temperature okoline spremajući vrijednosti u integriranu RAM memoriju, a kojoj pristupamo preko I2C protokola. Senzor ima 4 pina (nožice) preko kojih ga se napaja i komunicira s mikroupravljačkim sučeljem. Dvije nožice služe za napajanje dok druge dvije omogućavaju komunikaciju preko I2C protokola [2].

I2C protokol omogućuje mikrokontroleru komunikaciju s više različitih uređaja preko samo dvije žice – SDA i SCL. SCL (Serial Clock Line) stvara signal takta. Pri velikim brzinama koje se ostvaruju u svijetu elektronike svaka milisekunda je bitna. SCL sinkronizira, tj. ujednačuje, prijenos svih podataka u I2C kanalu kako bi signal istovremeno dolazio do svakog uređaja spojenog u taj kanal. SDA (Serial Data Line) služi za prijenos podatka od mikrokontrolera do uređaja i obratno – moguća je komunikacija u oba smjera, ali samo u jednom istovremeno. Uređaj koji šalje zahtjev nazivamo master, a onaj koji mu je podređen slave. Master šalje adresu uređaja s kojim želi komunicirati, uređaj koji ima tu adresu šalje/prima podatke dok ostali uređaji ostaju u stanju u kojem su prethodno bili. Za pravilnu I2C komunikaciju potrebno je na SCL i SDA linije postaviti pull-up otpornike. Oni spriječavaju da se u kanalu pojavi neodređeno stanje ukoliko se komunikacija trenutno ne odvija tako što spajaju linije s logičkom jedinicom („visokom“ naponskom razinom). Potrebno je razumjeti da prilikom spajanja elektroničkih potrebno je točno odrediti koje stanje se nalazi na izlazu (logička jedinica ili nula) kako bi se izbjegle pogreške [12].



Sl. 3.2. I2C komunikacija preko SCL i SDA linije i pripadajućim pull-up otpornicima [13]

MLX90641 senzoru potrebno je dovesti napon iznosa 3-3.6V i struju 10-14mA. I2C komunikacija radi na signalu taktu do 1MHz. Tvornička I2C adresa je 0x33 i potrebno ju je zapamtiti kako bi mogli programirati senzor. Sadrži 192 infracrvena senzora, tj. 192 vrijednosti temperature koje se prikazuju u matrici 16x12. Svaki piksel identificiran je svojim redom i stupcem kao Pix(i,j). Senzor sadrži i dvije stranice (*engl. subpage*) s kojih naizmjenično čita podatke.



Sl. 3.3. Stranice podataka unutar MLX90641 senzora [2]

Na Sl.3.3. prikazano je naizmjenično osvježavanje podataka po redcima – budući da se radi s 32 bitnim adresama potrebno je 32x6 prikaz transformirati u 16x12 koji je puno lakši za razumjeti i vizualizirati.

3.4. Izrada biblioteke

Biblioteka predstavlja kolekciju unaprijed napisanih kodova. Njihova svrha je ponuditi korisniku široku lepezu funkcija i mogućnosti koje može koristiti bez da ih osobno piše. Često se radi o kompliciranim funkcijama koje komuniciraju s operacijskim sustavom ili harverom, a razumijevanje istih nije nužno za daljnji rad programa. Kao što je ranije rečeno, Arduino IDE podržava biblioteke za velik broj senzora koji imaju sebi specifičan način rada. Često korišteni senzori i mikroupravljači imaju bolju podršku i unaprijed izrađene biblioteke od strane programera koji se time bave [14]. MLX90641 jedan je od onih koji nemaju svoju vlastitu biblioteku te je potrebno izraditi je iznova.

Tvrtka Melexis bavi se proizvodnjom elektroničkih komponenti među kojima je i MLX90641, ali i brojni drugi. Jedan od njihovih senzora, MLX90640, vrlo je sličan MLX90641

senzoru koji koristimo u projektu, a za njega postoje već unaprijed definirane biblioteke u Arduino IDE. U ovom slučaju, one su poslužile kao baza za izradu nove biblioteke.

Prvo je potrebno pomno proučiti razlike između dva senzora. Imaju gotovo istu namjenu, oba su matricni infracrveni senzori topline, ali različitih rezolucija – MLX90640 sadrži matricu dimenzija 32x24. Zbog toga je prvo potrebno ući u svaku datoteku biblioteke i promijeniti dimenzije, uvjete petlji, adrese i sl. Za to je korišten VS Code uređivač koda koji može otvoriti cijele mape i omogućiti vrlo pregledan prikaz datoteka u njima. Zatim, potrebno je dodati Hammingov dekoder koji je nepotreban za rad MLX90640 senzora, ali iznimno potreban za rad MLX90641 senzora. Hammingov kod koristi se kao alat za pronalaženje i ispravljanje grešaka prilikom prijenosa podataka. Zahtjeva dodavanje paritetnih bitova čija se vrijednost određuje prema već postojećim bitovima informacije koja se prenosi. U ovom slučaju koristimo Hamming(15,11) kodiranje gdje od 11 podatkovnih bita dobivamo 4 paritetna bita. Peti paritetni bit dobiva se kao kombinacija svih podatkovnih bitova i svih paritetnih bitova dobivenim Hamming kodiranjem.

$$P0 = D0 + D1 + D3 + D4 + D6 + D8 + D10$$

$$P1 = D0 + D2 + D3 + D5 + D6 + D9 + D10$$

$$P2 = D1 + D2 + D3 + D7 + D8 + D9 + D10$$

$$P3 = D4 + D5 + D6 + D7 + D8 + D9 + D10$$

$$P4 = D0 + D1 + D2 + D3 + D4 + D5 + D6 + D7 + D8 + D9 + D10 + P0 + P1 + P2 + P3$$

Formule 3.1. Računanje vrijednosti paritetnih bitova [2]

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
P4	P3	P2	P1	P0	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Sl. 3.4. Raspored podatkovnih i paritetnih bitova dobivenih Hamming kodiranjem te način kodiranja [2]

Izostavljanjem Hammingovog kodiranja, odnosno dekodiranja podataka, dobivaju se nasumične pozitivne i negativne vrijednosti, a često dolazi i do podataka koji nisu brojevi (NaN *engl. Not a Number*).

Nadalje, MLX90640 sadrži dva moda rada. Učitavanje podataka moguće je na jednak način kao i kod MLX90641 – čitanje svih vrijednosti iz „parnih“ heksadecimalnih adresa u jednom signalu takta, te čitanje svih vrijednosti iz „neparnih“ adresa u drugom taktu. To je

takozvani TV mod ili interleave mod rada. MLX90640 može raditi i na način da u svakoj stranici osvježava istovremeno sve adrese, ali i ne sve podatke u njima u obliku šahovnice (šah mod – engl. Chess mode).

0x0400	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	...
0x0420	34	67	100	133	166	199	232	265	298	331	364	397	430	463	496	529	...
0x0440	66	131	196	261	326	391	456	521	586	651	716	781	846	911	976	1041	...
0x0460	98	196	294	392	490	588	686	784	882	980	1078	1176	1274	1372	1470	1568	...
0x0480	129	257	385	513	641	769	897	1025	1153	1281	1409	1537	1665	1793	1921	2049	...
0x04A0	162	323	484	645	806	967	1128	1289	1450	1611	1772	1933	2094	2255	2416	2577	...
0x04C0	193	386	579	772	965	1158	1351	1544	1737	1930	2123	2316	2509	2702	2895	3088	...
0x04E0	225	449	673	897	1121	1345	1569	1793	2017	2241	2465	2689	2913	3137	3361	3585	...
0x0500	257	513	769	1025	1281	1537	1793	2049	2305	2561	2817	3073	3329	3585	3841	4097	...
0x0520	290	580	870	1160	1450	1740	2030	2320	2610	2900	3190	3480	3770	4060	4350	4640	...
0x0540	322	643	964	1285	1606	1927	2248	2569	2890	3211	3532	3853	4174	4495	4816	5137	...
0x0560	354	707	1060	1413	1766	2119	2472	2825	3178	3531	3884	4237	4590	4943	5296	5649	...
0x0580	385	770	1140	1510	1880	2250	2620	2990	3360	3730	4100	4470	4840	5210	5580	5950	...
0x05A0	418	835	1250	1665	2080	2495	2910	3325	3740	4155	4570	4985	5400	5815	6230	6645	...
0x05C0	449	897	1345	1990	2635	3280	3925	4570	5215	5860	6505	7150	7795	8440	9085	9730	...
0x05E0	482	963	1439	2178	2917	3656	4395	5134	5873	6612	7351	8090	8829	9568	10307	11046	...
0x0600	513	1025	1537	2305	3172	3939	4706	5473	6240	7007	7774	8541	9308	10075	10842	11609	...
0x0620	546	1090	1606	2413	3280	4047	4814	5581	6348	7115	7882	8649	9416	10183	10950	11717	...
0x0640	577	1153	1679	2526	3403	4180	4957	5734	6511	7288	8065	8842	9619	10396	11173	11950	...
0x0660	610	1216	1742	2655	3542	4329	5116	5903	6690	7477	8264	9051	9838	10625	11412	12199	...
0x0680	641	1279	1804	2717	3614	4401	5188	5975	6762	7549	8336	9123	9910	10697	11484	12271	...
0x06A0	674	1341	1866	2779	3686	4473	5260	6047	6834	7621	8408	9195	9982	10769	11556	12343	...
0x06C0	705	1403	1928	2841	3748	4535	5322	6109	6896	7683	8470	9257	10044	10831	11618	12405	...
0x06E0	738	1465	1990	2903	3810	4597	5384	6171	6958	7745	8532	9319	10106	10893	11680	12467	...

Sl. 3.5. Šah način rada senzora MLX90640 [15]

U oba moda rada potrebni su potpuni podaci (podaci s obje stranice) kako bi se mogla izračunati potpuna vrijednost svakog piksela matrice. Kako bi prilagodili kod svojem senzoru briše se sav kod koji u sebi sadrži šahovski način rada.

Na taj način kompletira se biblioteka prilagođena MLX90641 senzoru i njegovim specifikacijama. Ona na kraju sadrži: C++ datoteku s pomoćnim funkcijama koje se koriste u Arduino IDE programskom sučelju, MLX90641_API C++ datoteku koja sadrži funkcije za komunikaciju s hardverom samog senzora, u njemu se preko adresa dohvaćaju i obrađuju podaci, MLX90641_API.h datoteka koja u sebi sadrži strukturu s parametrima koji se popunjavaju u API C++ datoteci te druga .h datoteka s definicijom svih funkcija iz C++ datoteka, definira frekvencije osvježavanja samog senzora i okolnosti njegovog rada i klasu koja se popunjena podacima vraća u Arduino programsko sučelje kako bi se mogli vidjeti rezultati. Biblioteke je potrebno raditi u točno određenom formatu kako bi Arduino IDE prihvatio biblioteku kao važeću.

3.5. Programsko rješenje sustava

Nakon završene biblioteke potrebno je napisati kod u Arduino IDE sučelju. U njemu su korištene unaprijed definirane funkcije iz biblioteke. Stoga ju je potrebno uključiti u naš projekt:

```
#include <Adafruit_MLX90641.h>
```

Programski kod 3.1. Uključivanje biblioteke u program

Nadalje, potrebno je definirati senzor kao varijablu. U biblioteci je definirana klasa **Adafruit_MLX90641** i u njoj se nalaze sve funkcije koje se koriste s istom klasom. Korišten je operator izravnog odabira člana ili popularnije točkasti (engl. *dot*) operator kako bi se preko varijable pristupalo funkcijama. Za spremanje podataka prikupljenih sa senzora potrebno je polje decimalnih vrijednosti od 192 mjesta. Kasnije se iz tog polja slaže matrični prikaz 16x12 koji je mnogo jednostavniji za čitanje i razumijevanje od jednodimenzionalnog polja.

```
Adafruit_MLX90641 mlx;  
float frame[12*16]; // buffer for full frame of temperatures
```

Programski kod 3.2. Deklariranje klase varijablom i polja za spremanje temperatura

MLX90641 pomoću svoja 192 infracrvena senzora omogućava određivanje temperature iz okoline, ali češća je praksa prikazivanje podataka bojama ili simbolima, nego brojčano. U ovom slučaju programski je omogućen prikaz pomoću ASCII simbola. Svaka vrijednost temperature koja pripada određenom intervalu zamijenjena je i prikazana odgovarajućim simbolom. Vrlo visoke temperature koje daje plamen postavljaju sustav u alarmantno stanje koje naizmjenično pali i gasi LE diodu označavajući uzbunu. Kako bi mikroupravljač mogao komunicirati s LE diodom potrebno je unaprijed definirati pin preko kojeg se komunicira. Također, korišten je i brojač visokih temperatura kao osigurač. Moguće je zbog nepravilnosti ili greške da vrijednost skoči (engl. *spike value*) i prebaci sustav u alarmantno stanje. Brojač broji koliko se visokih temperatura nalazi u polju te, ako taj broj prelazi unaprijed određenu graničnu vrijednost, postavlja sustav u alarmantno stanje.

```
// uncomment *one* of the below  
//#define PRINT_TEMPERATURES  
#define PRINT_ASCIIART  
const int pinToControl = 26;  
int count = 0;
```

Programski kod 3.3. Definicija načina rada, definiranje komunikacijskog pina za LE diodu te deklaracija i inicijalizacija brojača

```

xx. .x+. .xx--#%**
%x. .x+--xx-+#X**
xx. -+x--x+-*##**
%x-*xx--+-xx-*
x+--xx. -x+. -xx.*
xx--%x*-x+--x+--
%x*-x+**x+**+**
+**xx*-xx--x+-*
+x**xx**x+*x+**
xx**xx-*xx**x+*-
x+*xx*-+x+*xx--
+*+*+x*-xx**x+--

```

Sl. 3.6. Ispis termalne slike prilikom PRINT_ASCIIART načina rada

```

26.2, 26.3, 26.3, 26.4, 26.3, 26.4, 26.3, 26.2, 25.9, 26.0, 26.0, 26.1, 26.0, 25.8, 25.7, 25.8,
26.2, 26.3, 26.5, 26.6, 27.0, 27.1, 26.5, 26.3, 26.1, 26.0, 25.9, 26.0, 26.1, 26.0, 26.0, 26.1,
26.3, 26.6, 27.0, 28.4, 29.8, 29.8, 28.6, 27.1, 26.6, 26.3, 26.1, 26.0, 26.2, 26.0, 26.2, 26.1,
26.1, 26.2, 26.4, 27.7, 28.8, 30.0, 30.0, 29.1, 28.6, 27.2, 26.2, 26.3, 26.3, 26.0, 26.1, 26.2,
26.2, 26.4, 26.8, 28.3, 29.7, 31.0, 31.1, 30.9, 30.0, 29.3, 28.6, 27.2, 26.4, 26.1, 26.1, 26.2,
26.0, 26.2, 26.3, 26.8, 28.6, 30.0, 31.2, 31.4, 31.3, 29.7, 28.6, 26.9, 26.5, 26.2, 26.0, 26.1,
25.8, 26.0, 26.2, 26.9, 28.2, 30.1, 31.2, 31.4, 31.5, 31.2, 31.1, 29.9, 27.3, 26.3, 26.2, 26.3,
25.8, 25.7, 26.0, 26.3, 27.3, 28.4, 30.1, 31.4, 31.4, 31.4, 31.4, 31.2, 28.5, 26.8, 26.1, 26.2,
25.7, 25.7, 26.3, 27.3, 28.3, 28.7, 30.4, 31.2, 31.5, 31.6, 31.5, 31.4, 29.9, 27.0, 26.3, 26.1,
25.8, 26.0, 26.3, 27.0, 27.1, 27.9, 28.1, 30.1, 31.4, 31.4, 31.5, 31.6, 31.4, 31.1, 26.0, 27.3,
26.2, 25.8, 26.2, 26.0, 26.5, 27.0, 27.9, 28.7, 30.0, 31.8, 31.9, 32.0, 32.0, 32.1, 31.9, 31.8,
25.8, 25.9, 25.8, 26.1, 26.1, 26.0, 27.1, 27.6, 28.9, 30.2, 31.8, 31.7, 32.1, 31.8, 32.0, 31.9,

```

Sl. 3.7. Ispis brožanih vrijednosti temperatura prilikom PRINT_TEMPERATURES načinu rada

Nakon svih potrebnih deklaracija kreće startup() funkcija Arduino programa. U njoj je prvo definiran pin LE diode alarmantnog stanja kao izlaz – na njega se signal šalje, tj. ne očekuju se podaci natrag prema mikroupravljaču s tog pina. Zatim se pokreće serijska komunikaciju. Pomoću nje i Arduino Serial Monitor-a prikazuju se temperature, odnosno ASCII simboli na zaslonu. Bitna stavka kod serijske komunikacije je baud rate – količina informacija koja prolazi kroz komunikacijski kanal. U kontekstu serijske komunikacije „115200 baud“ znači da kroz kanal serijske komunikacije prolazi maksimalno 115.200 bita po sekundi. Na brzinama većim od 76.800 bit/s potrebno je obratiti pažnju i na dužinu kabla za programiranje – veća brzina zahtjeva manji put u istom vremenu prijenosa podataka. [16] Nadalje, nakon uspostavljanja serijske komunikacije ispisuju se osnovni podaci o senzoru i postavljaju inicijalne vrijednosti, odabire se „interleaved“ način rada podržan od strane senzora i rezolucija ulaza. Rezolucija označava broj bitova koji se šalje prema i od mikroupravljača. Tako primjerice tvornička vrijednost od 18 bita rezolucije

označava 2^{18} bajta podataka u toku. Potrebno je postaviti i stopu osvježavanja (engl. *refresh rate*), tj. koliko puta u sekundi senzor prikuplja nove podatke. U slučaju MLX90641 senzora vrijednosti stope osvježavanja kreću se od 2^{-1} (0,5) Hz do 2^6 (64) Hz.

```
void setup() {  
  pinMode(pinToControl, OUTPUT);  
  Serial.begin(115200);  
  delay(100);  
}
```

Programski kod 3.4. Inicijalizacija serijske komunikacije

```
Serial.print("Serial number: ");  
Serial.print(mlx.serialNumber[0], HEX);  
Serial.print(mlx.serialNumber[1], HEX);  
Serial.println(mlx.serialNumber[2], HEX);  
  
mlx.setMode(MLX90641_INTERLEAVED);  
//mlx.setMode(MLX90641_CHESS);  
Serial.print("Current mode: Interleave ");  
/* if (mlx.getMode() == MLX90641_CHESS) {  
  | Serial.println("Chess");  
  } else {  
  | Serial.println("Interleave");  
  */  
}
```

Programski kod 3.5. Ispis osnovnih podataka senzora i postavljanje radnom načina

```
mlx.setResolution(MLX90641_ADC_18BIT);  
Serial.print("Current resolution: ");  
mlx90641_resolution_t res = mlx.getResolution();  
switch (res) {  
  | case MLX90641_ADC_16BIT: Serial.println("16 bit"); break;  
  | case MLX90641_ADC_17BIT: Serial.println("17 bit"); break;  
  | case MLX90641_ADC_18BIT: Serial.println("18 bit"); break;  
  | case MLX90641_ADC_19BIT: Serial.println("19 bit"); break;  
}
```

Programski kod 3.6. Postavljanje i ispisivanje korištene rezolucije

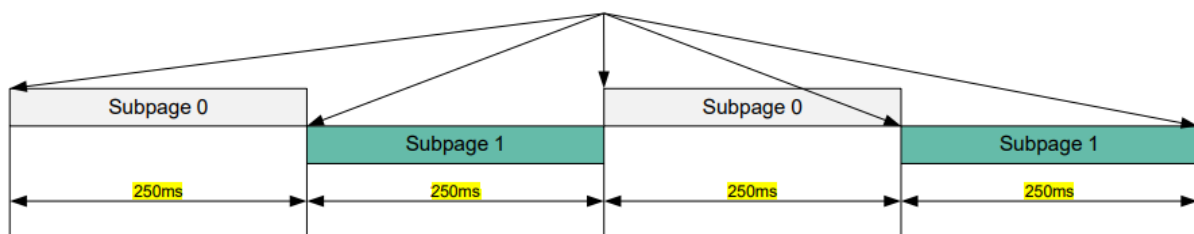

```

mlx.setRefreshRate(MLX90641_64_HZ);
Serial.print("Current frame rate: ");
mlx90641_refreshrate_t rate = mlx.getRefreshRate();
switch (rate) {
  case MLX90641_0_5_HZ: Serial.println("0.5 Hz"); break;
  case MLX90641_1_HZ: Serial.println("1 Hz"); break;
  case MLX90641_2_HZ: Serial.println("2 Hz"); break;
  case MLX90641_4_HZ: Serial.println("4 Hz"); break;
  case MLX90641_8_HZ: Serial.println("8 Hz"); break;
  case MLX90641_16_HZ: Serial.println("16 Hz"); break;
  case MLX90641_32_HZ: Serial.println("32 Hz"); break;
  case MLX90641_64_HZ: Serial.println("64 Hz"); break;
}

```

Programski kod 3.7. Postavljanje stope osvježavanja senzora te ispis

Nakon postavljanja početnih uvjeta, kreće radni dio programa, tj. loop() funkcija koja iznova izvodi isti programski kod i svakim prolaskom kroz kod dobiva nove podatke od senzora. U svrhu pravilnog dohvaćanja podataka sa senzora učitavaju se obje stranice podataka sa senzora nad kojima se vrše kalkulacije. Željena stopa osvježavanja je 64 Hz što znači da je za osvježavanje jedne stranice potrebno $\frac{1}{64}$ sekundi. Taj broj pomnožen s dvije stranice daje približno 32 ms – iznos čekanja prije ponovnog dohvaćanja podataka s MLX90641 senzora.



Sl. 3.8. Osvježavanje stranica prilikom 4 Hz stope osvježavanja [2]

```

void loop() {
  count = 0;
  delay(32);
  if (mlx.getFrame(frame) != 0) {
    Serial.println("Failed");
    return;
  }
}

```

Programski kod 3.8. Postavljanje čekanja u funkciji loop() i dohvaćanje podataka

Korištenjem `.getFrame()` funkcije definirane unutar biblioteke dohvaćaju se podaci sa senzora i spremaju u lokalnu varijablu `frame`. Dopisana je i pripadajuća poruka u slučaju neuspješnog dohvaćanja podataka. U programskom kodu 3.8. spomenuta `count` varijabla koristi se za izbjegavanje pokretanja alarmantnog stanja prilikom pojave stršćih vrijednosti.

U drugom dijelu `loop()` funkcije formatira se ispis podataka. Pomoću dvije petlje jednodimenzionalno polje sa 192 mjesta ispisuje se u formatu dvodimenzionalnog polja sa 16 stupaca i 12 redaka. Varijabla `float t` predstavlja očitane temperaturu pojedinog piksela.

```
for (uint8_t h=0; h<12; h++) {  
  for (uint8_t w=0; w<16; w++) {  
    float t = frame[h*16 + w];
```

Programski kod 3.9. Formatiranje ispisa

Zadnji dio radne funkcije grana se pomoću direktiva. One predstavljaju uputu predprocesoru koji dio koda se uvažava, odnosno zanemaruje prilikom prevođenja programa. U ovom slučaju, provjerava se ranije definiran način rada – `PRINT_ASCIIART` ili `PRINT_TEMPERATURES`. Ispisivanje temperatura je jednostavnije budući da je to ono što senzor i daje. Koristi se funkcija `Serial.print()` koja preko Serial Monitora ispisuje brojčane vrijednosti na zaslon.

```
#ifdef PRINT_TEMPERATURES  
  Serial.print(t, 1);  
  Serial.print(", ");  
#endif  
#ifdef PRINT_ASCIIART  
  char c = '&';  
  if (t < 15) c = ' '  
  else if (t < 20) c = '.';  
  else if (t < 25) c = '-';  
  else if (t < 30) c = '*';  
  else if (t < 35) c = '+';  
  else if (t < 40) c = 'x';  
  else if (t < 45) c = '%';  
  else if (t < 50) c = '#';  
  else if (t < 55) c = 'X';  
  else c = '0';  
  Serial.print(c);  
  
  if(t > 90) count = count + 1;  
#endif
```

Programski kod 3.10. Predprocesorske naredbe i ispisivanje vrijednosti

Drugi način rada zahtjeva uvjetno grananje. Korištenjem **if/else** vrijednost temperature zamjenjuje ASCII simbol radi lakše vizualizacije toplinske slike koju senzor daje. Ukoliko više piksela prepozna vrlo visoku temperaturu pali se alarmantno stanje.

Posljednji dio koda je alarmantno stanje. Sastoji se od jednog uvjetnog grananja koje glasi: ako više od 5 piksela prepozna vrlo visoku temperaturu, naizmjenično pali/gasi LE diodu – u suprotnom ju drži ugašenom. Dodana je i pauza u slučaju paljenja/gašenja LE diode kako bi ljudsko oko moglo prepoznati promjenu – zbog tromosti oka potrebno je imati pauzu programa dužu od 40 ms kako bi oko prepoznalo da se dioda ugasila, odnosno upalila [17].

```
if(count>5){
digitalWrite(pinToControl, HIGH);
delay(50);
digitalWrite(pinToControl, LOW);
}
else
digitalWrite(pinToControl, LOW);

Serial.println();
}
```

Programski kod 3.11. Upravljanje LE diodom u alarmantnom stanju

4. EKSPERIMENT

Kako bi se dokazala ispravnost mikroupravljačkog sustava odrađen je eksperiment. U prvom dijelu eksperimenta iznad senzora je postavljen hladan predmet. Za temperature niže od 15°C senzor šalje brojčanu vrijednost, a program na zaslon ispisuje prazan prostor.

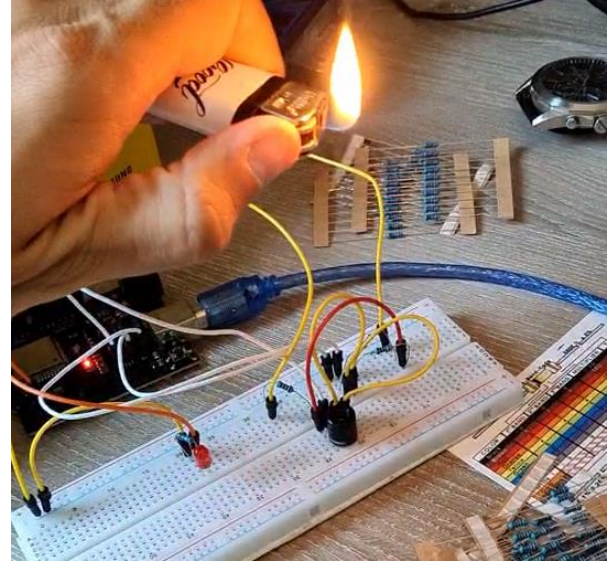
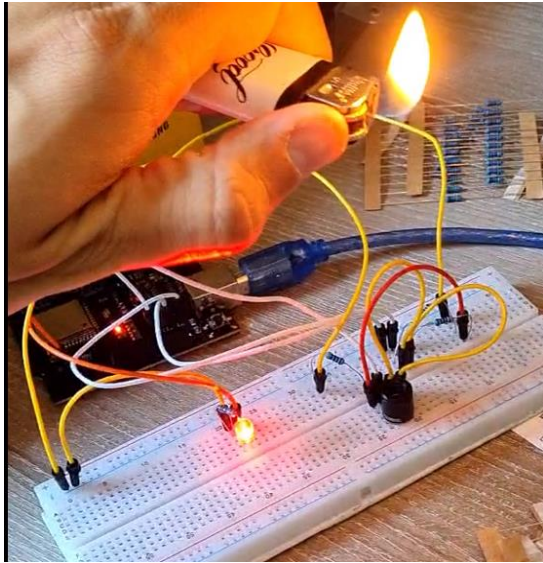


Sl. 4.1. Hladni predmet

XX+X+	++XXX++X++	XX+X++++	XXX+X+X++X+X+X
X++X	+XX+XXX+	X++XX+X+	XX+X+X++X+XXX+X+
X+	++X+X++	X+XX+X+X+	XX+X++X+X+X+X++X
X	+X+X+X	XXX+X+X++	XX+X+X++X+X+XXX+
X	XX+X++	XX+X+X++X+	+X+X+X+X+X +X+X
X	+X++X	XX+X++X+X+	+XX+X+X+ +X+
X	+XXX+	XX+X+X++X+X+	X+X+X++ +X
+	++X+X	+X+X+X+X+X+++	X+XX+X X
+X	+X++X+	+XX+X+X+X++X++	+X+XXX
X+	X++X+X	X+X+X++X+X++X+X	XX+X++
X+X	X+X++XX	X+XX+X+X+X+X++XX	X++XX+
+X+X	X+XX+X++	+X+XXXX+X+XX+X++	X+XX+XX

Sl. 4.2. Pomicanje hladnog predmeta po vidokrugu senzora i pomicanje prostora niske temperature na ispisu

U drugom dijelu eksperimenta korišten je upaljač kao izvor plamena te simulacija požara. Nakon što je plamen doveden u senzoru vidljivo područje, pali se „blicanje“ LE diode koja označava alarmantno stanje.



Sl. 4.3. Paljenje i gašenje LE diode tokom alarmantnog stanja

```

0000++++XXX++X++
0000x+x+x+xx+xxx
00000x+x+xxx+x+x
0000000++x+x+x+x
0000000+x+xx+x++
00000000x+x+x++x
00000000x+x+xxx+
000000+x+x++xx+x
000000+x++xx+xx+
00000++xx+xx+++x
000x+x+x+x+x++xx
0x+xxxxx+x+xx+x++

```

Sl. 4.4. Izlaz prilikom prilaska plamena senzoru s lijeve strane – simboli O označavaju visoke temperature

Cijeli video eksperimenta dostupan je na linku [18].

5. ZAKLJUČAK

Mikroupravljački sustav za prepoznavanje požara uspješno je napravljen. Omogućena je prevencija potencijalno opasnih situacija za čovjeka i za imovinu. MLX90641 senzor nalazi se u kompaktnom pakiranju te je cijelim sustavom lako rukovati i prenijeti ga. Moguća je i njegova ugradnja na mjesta bez pristupa internetu budući da se sva komunikacija događa interno – unutar samog senzora. Spajanjem uređaja na računalo moguće je bilježiti svaku sliku koja nastaje te ih spremati radi dokumentacije unutar Serial Monitora ugrađenog u Arduino IDE. Ovakvim mikroupravljačkim sustavima lako je rukovati te ih nadograditi, a uz pomoć strojnog učenja i osamostaliti. ESPduino-32 platforma za testiranje pokazala se kao odličan alat kako za početničke elektroničke projekte, tako i za napredne mikroupravljačke sustave. Prednost nad ostalim razvojnim platformama je velika količina dostupne memorije bez koje ovaj rad ne bi bio moguć – kroz rad otkriveno je da Arduino UNO nema dostatnu memoriju za izvođenje zadatka.

Moguća unaprjeđenja ovog mikroupravljačkog sučelja su spajanje senzora na mrežu te, ako mreža postoji, automatski pozvati u pomoć u slučaju otkrivanja požara. Spajanje mlaznice protupožarnog aparata bilo bi privremeno rješenje problema budući da mala rezolucija samog senzora neće biti dovoljna za precizne akcije gašenja.

LITERATURA

- [1] Shedboy71, ESP32learning, A LOOK AT THE ESPDUINO-32 BORD, 12th November 2017, dostupno na [<http://www.esp32learning.com/hardware/a-look-at-the-espduino-32-board.php>] Pristupljeno dana 28.8.2023.
- [2] Mouser Electronics, MLX90641ESF-BCB-000-SP službena dokumentacija, dostupno na: <https://www.mouser.co.uk/ProductDetail/Melexis/MLX90641ESF-BCB-000-SP?qs=17cgNqFNU1j9Pjwrdu1MIg%3D%3D> Pristupljeno dana 1.6.2023.
- [3] electronics-lab, MLX90641 – FAR INFRARED THERMAL SENSOR ARRAY WITH 16X12PX RESOLUTION, dostupno na: <https://www.electronics-lab.com/mlx90641-far-infrared-thermal-sensor-array-16x12px-resolution/> Pristupljeno dana 1.6.2023.
- [4] Arduino Docs, Arduino službena web-stranica, dostupno na: <https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics> Pristupljeno dana 23.8.2023.
- [5] Scott Campbell , Circuit Basics, HOW TO INSTALL AND CONFIGURE THE ARDUINO IDE, dostupno na [<https://www.circuitbasics.com/wp-content/uploads/2020/05/Arduino-IDE-1-1024x878.png>] Pristupljeno dana 28.8.2023.
- [6] Visual Studio Code, 8.3.2023. dostupno na: URL [<https://code.visualstudio.com/docs/editor/whyvscode>] Pristupljeno dana 28.8.2023.
- [7] Barry, A Total Solution, Inc., FIRE PROTECTION: THE FOUR TYPES OF FIRE DETECTORS, 21.12.2017. dostupno na: URL [<https://www.atotalsolution.com/blog/fire-protection-the-four-types-of-fire-detectors/>] pristupljeno dana 31.8.2023.
- [8] Herschel, Frederick William. Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2021. dostupno na: <https://enciklopedija.hr/natuknica.aspx?ID=25231> Pristupljeno dana 25.8.2023.
- [9] infracrveno zračenje. Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2021. dostupno na: <https://enciklopedija.hr/natuknica.aspx?ID=27417> Pristupljeno dana 25.8.2023.
- [10] Wikipedia, Infrared, dostupno na: URL [https://upload.wikimedia.org/wikipedia/commons/c/cf/Ir_girl.png] Pristupljeno dana 31.8.2023.

[11] EESemi, dostupno na: URL [<https://www.eesemi.com/to39.htm>] Pristupljeno dana 25.8.2023.

[12]] Scott Campbell , Circuit Basics, BASICS OF THE I2C COMMUNICATION PROTOCOL, dostupno na: URL [<https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>] pristupljeno dana 31.8.2023.

[13] Robot Electronics, dostupno na: URL [<https://www.robot-electronics.co.uk/images/i2ca.GIF>] pristupljeno dana 31.8.2023.

[14] Goodness Woke, Redgate Hub, The difference between libraries and frameworks, 24.3.2023. dostupno na: URL [<https://www.red-gate.com/simple-talk/development/other-development/the-difference-between-libraries-and-frameworks/#:~:text=A%20library%20is%20a%20collection,be%20called%20upon%20as%20needed.>] Pristupljeno dana 26.8.2023

[15] Mauser Electronics, MLX90640 službena dokumentacija, dostupno na: <https://hr.mouser.com/c/ds/?marcom=145139989> Pristupljeno dana 28.8.2023.

[16] Nikko Phongchit, Setra, What is Baud Rate & Why is it important?, 4.1.2016. dostupno na: URL [<https://www.setra.com/blog/what-is-baud-rate-and-what-cable-length-is-required-1>] Pristupljeno dana 26.8.2023

[17] John V. Forrester, Andrew D. Dick, Paul G. McMenamin, Fiona Roberts, Eric Pearlman, The Eye, Chapter 5 Physiology of vision and the visual system, Elsevier, 2016

[18] https://drive.google.com/file/d/1F7ZPYqR_jst2oBAkMNUwqWG_Uy85IsBh/view?usp=drive_link

SAŽETAK

Tema ovog rada bila je upoznati se s tehnikama te skupiti alate, dizajnirati i programirati mikroupravljački sustav za otkrivanje požara. Potrebno je pomoću infracrvenog zračenja zapisati temperaturu okoline te u slučaju pojavljivanja visokih vrijednosti temperature obavijestiti korisnika. Za bilježenje temperatura korišten je MLX90641, infracrveni senzor topline a 16x12 matričnim prikazom. Sa 192 infracrvena senzora (piksela) u sebi može zabilježiti temperaturu od -40°C do 300°C čineći ga pogodnim za prikaz toplinskih snimaka. Pomoću Arduino IDE sučelja korisnik komunicira sa senzorom prikupljajući s njega podatke, a može svoj kod prepustiti zajednici programera u obliku gotovih biblioteka. Komunikacija se odvija preko I2C protokola, vrlo jednostavnog i materijalno nezahtjevnog načina. Koriste se dvije žice za napajanje te dvije za samu komunikaciju. Dostupne su velike brzine prijenosa što u prijevodu znači rad senzora u stvarnom vremenu.

Ključne riječi: ESPduino-32, MLX90641, Melexis, otkrivanje plamena, I2C, mikroupravljački sustav

ABSTRACT

The topic of this work was to learn about the techniques and collect tools, design and program a microcontroller system for fire detection. It is necessary to record the ambient temperature using infrared radiation and inform the user if high temperature values appear. MLX90641, an infrared heat sensor with a 16x12 matrix display, was used to record said temperatures. With 192 infrared sensors (pixels), it can record temperatures from -40°C to 300°C, making it suitable for displaying thermal infrared images. Using the Arduino IDE interface, the user communicates with the sensor, collecting data from it, and can leave his code to the community in the form of ready-made libraries. Communication takes place via the I2C protocol, a very simple and materially undemanding method. Two wires are used for power and two for communication. High transfer rates are available, which translates into real-time sensor operation.

Keywords: ESPduino-32, MLX90641, Melexis, flame detection, I2C, microcontroller

ŽIVOTOPIS

Matko Grgić rođen je 17.10.2001. u Osijeku. Pohađao je Osnovnu školu Jagode Truhelke u Osijeku i završio ju 2016. godine. Te godine upisao je III. gimnaziju Osijek po prirodoslovno-matematičkom programu. Za vrijeme pohađanja srednje škole nastupio je na brojnim školskim i županijskim natjecanjima iz područja Informatike. Nakon ispita državne mature, 2020. godine, upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku (FERIT) gdje odabire preddiplomski smjer računarstva i blok računalnog inženjerstva. Od 2022. godine član je IEEE inženjerske organizacije, a od 2023. je i član studentskog zbora istog fakulteta. Uz fakultetske obaveze, član je i odbojkaškog kluba MOK Marsonia iz Slavenskog Broda za koji profesionalno igra hrvatsku Superligu.

PRILOZI

Kompletan programski kod mikroupravljačkog sustava za otkrivanje požara u Arduino IDE sučelju

```
#include <Adafruit_MLX90641.h>

Adafruit_MLX90641 mlx;
float frame[12*16]; // buffer for full frame of temperatures

// uncomment *one* of the below
//#define PRINT_TEMPERATURES
#define PRINT_ASCIIART
const int pinToControl = 26;
int count = 0;

void setup() {
  pinMode(pinToControl, OUTPUT);
  Serial.begin(115200);
  delay(100);

  Serial.println("MLX90641 Simple Test");
  if (! mlx.begin(MLX90641_I2CADDR_DEFAULT, &Wire)) {
    Serial.println("MLX90641 not found!");
    while (1) delay(10);
  }
  Serial.println("Found Adafruit MLX90641");

  Serial.print("Serial number: ");
  Serial.print(mlx.serialNumber[0], HEX);
  Serial.print(mlx.serialNumber[1], HEX);
  Serial.println(mlx.serialNumber[2], HEX);

  mlx.setMode(MLX90641_INTERLEAVED);
  //mlx.setMode(MLX90641_CHESS);
  Serial.print("Current mode: Interleave ");
  if (mlx.getMode() == MLX90641_CHESS) {
    Serial.println("Chess");
  } else {
    Serial.println("Interleave");
  }

}

mlx.setResolution(MLX90641_ADC_18BIT);
Serial.print("Current resolution: ");
mlx90641_resolution_t res = mlx.getResolution();
switch (res) {
  case MLX90641_ADC_16BIT: Serial.println("16 bit"); break;
  case MLX90641_ADC_17BIT: Serial.println("17 bit"); break;
}
```

```

        case MLX90641_ADC_18BIT: Serial.println("18 bit"); break;
        case MLX90641_ADC_19BIT: Serial.println("19 bit"); break;
    }

    mlx.setRefreshRate(MLX90641_64_HZ);
    Serial.print("Current frame rate: ");
    mlx90641_refreshrate_t rate = mlx.getRefreshRate();
    switch (rate) {
        case MLX90641_0_5_HZ: Serial.println("0.5 Hz"); break;
        case MLX90641_1_HZ: Serial.println("1 Hz"); break;
        case MLX90641_2_HZ: Serial.println("2 Hz"); break;
        case MLX90641_4_HZ: Serial.println("4 Hz"); break;
        case MLX90641_8_HZ: Serial.println("8 Hz"); break;
        case MLX90641_16_HZ: Serial.println("16 Hz"); break;
        case MLX90641_32_HZ: Serial.println("32 Hz"); break;
        case MLX90641_64_HZ: Serial.println("64 Hz"); break;
    }
}

void loop() {
    count = 0;
    delay(32);
    if (mlx.getFrame(frame) != 0) {
        Serial.println("Failed");
        return;
    }
    Serial.println();
    Serial.println();

    for (uint8_t h=0; h<12; h++) {
        for (uint8_t w=0; w<16; w++) {
            float t = frame[h*16 + w];
#ifdef PRINT_TEMPERATURES
            Serial.print(t, 1);
            Serial.print(", ");
#endif
#ifdef PRINT_ASCIIART
            char c = '&';
            if (t < 15) c = ' ';
            else if (t < 20) c = '.';
            else if (t < 25) c = '-';
            else if (t < 30) c = '*';
            else if (t < 35) c = '+';
            else if (t < 40) c = 'x';
            else if (t < 45) c = '%';
            else if (t < 50) c = '#';
            else if (t < 55) c = 'X';
            else c = '0';
            Serial.print(c);

```

```
        if(t > 90) count = count + 1;
    #endif
    }

    if(count>5){
        digitalWrite(pinToControl, HIGH);
        delay(50);
        digitalWrite(pinToControl, LOW);
    }
    else
        digitalWrite(pinToControl, LOW);

    Serial.println();
}
}
```