Detekcija i prepoznavanje objekata korištenjem mikrokontrolera u nastavi informatike

Logarušić, Ana

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:739208

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-27**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO – MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

ANA LOGARUŠIĆ

Detekcija i prepoznavanje objekata korištenjem mikrokontrolera u nastavi informatike

Diplomski rad

Voditelj rada:

Dr. sc. Goran Igaly

Zagreb, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Zahvaljujem se mentoru dr. sc. Goranu Igalyju na podršci, strpljenu i brojnim savjetima tijekom izrade ovog diplomskog rada.

Zahvaljujem se svojim roditeljima, bratu i sestri na podršci tijekom studiranja, na njihovom veselju za svaki moj položen ispit te na tome što su uvijek vjerovali u mene.

Zahvaljujem se svom suprugu Josipu na beskonačnom razumijevanju, ljubavi i vjeri u mene, na svakodnevnom motiviranju i ohrabrivanju tijekom studiranja i tijekom pisanja ovog rada.

Sadržaj

Uvod 1
1 Kurikulum nastavnog predmeta informatika 2
1.1 Osnovna škola 2
1.2 Srednja škola 3
2 Nastava programiranja u školama 5
2.1 Usporedba programskih jezika Logo i Scratch 5
2.2 Ulazne jedinice 7
2.3 Senzori kao ulazne jedinice 7
3 O Pixyju 8
3.1 PixyMon 8
3.1.1 Raw video 9
3.1.2 Default program 9
3.1.3 Cooked video 10
3.2 Potpisi (*signatures*) 11
3.3 Kodiranje bojama (Color Code - CC) 14
4 Pixy u mBlocku 17
4.1 Razlika između I2C i SPI 18
4.2 Naredbe iz biblioteke PixyCAM I2C 20
4.2.1 Naredba Get objects 21
4.2.2 Naredba Serial Print data of object No. i 26
5 Programi u mBlocku 32
5.1 Aktivnost 1: RGB paleta boja 32
5.2 Aktivnost 2. Semafor 36
6 Arduino IDE 38
6.1 Naredbe za rad s Pixyjem 39
6.2 Funkcije Setup i Loop 40
6.3 Pixy u Arduinu IDE 42
6.3.1 i2c (Hello world) 42
6.3.2 Modifikacija programa Hello world 46
6.3.3 mBot prati objekt 49
7 Zaključak 53

8 Literatura 55
Sažetak 56
Summary 57
Životopis 58

Uvod

Mikrokontroleri se sve češće koriste u nastavi informatike. Zbog mogućnosti spajanja različitih ulaznih i izlaznih jedinica učenicima je programiranje mikrokontrolera zanimljivo i zabavno.

U ovom radu će se pokazati primjeri korištenja senzora objekata realiziranog pomoću kamere Pixy koja mikrokontroleru prosljeđuje informaciju o položaju pojedinih objekata. Na temelju te informacije mikrokontroler može poduzimati različite akcije. Pokazat će se razni primjeri, zadaci i projekti primjereni učenicima osnovnih i srednjih škola u kojima se može koristiti ova vrsta ulaznih podataka.

1 Kurikulum nastavnog predmeta informatika

U kurikulumu nastavnog predmeta Informatika za osnovne i srednje škole navedene su četiri domene kojima će se realizirati ciljevi predmeta Informatika. Te domene su: Informacije i digitalna tehnologija (A), Računalno razmišljanje i programiranje (B), Digitalna pismenost i komunikacija (C) te e-Društvo (D). U nastavku će biti izdvojeni odgojno-obrazovni ishodi za domenu Računalno razmišljanje i programiranje za osnovnu i srednju školu koji su povezani s korištenjem mikrokontrolera i senzora, odnosno robota [1]. Uz odgojno-obrazovne ishode napisane su i preporuke za ostvarenje ishoda iz kurikuluma.

1.1 Osnovna škola

B.1.2: "učenik prati i prikazuje slijed koraka potrebnih za rješavanje nekoga jednostavnog zadatka."

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.2.1. "učenik analizira niz uputa koje izvode jednostavan zadatak, ako je potrebno ispravlja pogrešan redoslijed."

"Na temelju poznatoga zadatka učenik analizira niz uputa predočenih slikom ili riječima, otkriva pogrešan redoslijed u uputama i ispravlja ga (niz uputa za zadatke koji su bliski učenicima, igre sakrivanja, davanja uputa za kretanje, vođenje robota labirintom – učenik može glumiti robota). "

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.2.2. "učenik stvara niz uputa u kojemu upotrebljava ponavljanje."

"Primijeniti pomicanje likova računalom ili bez računala (kornjača, roboti)"

Preporuka: "Prema mogućnostima škole učitelj pokazuje učenicima upravljanje robotom unošenjem različitih naredbi. "

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.3.1. "učenik stvara program korištenjem vizualnoga okruženja u kojemu se koristi slijedom koraka, ponavljanjem i odlukom te uz pomoć učitelja vrednuje svoje rješenje."

Preporuka: "Prema mogućnostima škole učitelj pokazuje učenicima upravljanje robotom unošenjem različitih naredbi. "

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.4.1. "učenik stvara program korištenjem vizualnim okruženjem u kojemu se koristi slijedom, ponavljanjem, odlukom i ulaznim vrijednostima."

Preporuka: "Prema mogućnostima škole učitelj pokazuje učenicima upravljanje robotom unošenjem različitih naredbi."

"Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.5.1. "učenik se koristi programskim alatom za stvaranje programa u kojemu se koristi ulaznim i izlaznim vrijednostima te ponavljanjem."

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.5.2. "učenik stvara algoritam za rješavanje jednostavnoga zadatka, provjerava ispravnost algoritma, otkriva i popravlja greške."

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.6.1. "učenik stvara, prati i preuređuje programe koji sadrže strukture grananja i uvjetnoga ponavljanja te predviđa ponašanje jednostavnih algoritama koji mogu biti prikazani dijagramom, riječima govornoga jezika ili programskim jezikom."

Preporuka: "Prema mogućnostima koristiti se i hardverskim rješenjima za vizualizaciju programiranja (roboti i sl.)."

B.6.2. "učenik razmatra i rješava složeniji problem rastavljajući ga na niz potproblema."

Preporuka: "U pokaznim (odabranim) primjerima programskoga koda uočiti/prepoznati/istaknuti dijelove koda koji predstavljaju rješenje nekoga poznatog (manjeg) problema (zadatka), mijenjati/prilagoditi dijelove koda kako bi se uklopili u rješenje nekoga većeg problema."

B.7.1. "učenik razvija algoritme za rješavanje različitih problema koristeći se nekim programskim jezikom pri čemu se koristi prikladnim strukturama i tipovima podataka."

"Analizirati i predvidjeti moguće izmjene algoritma koje bi mogle poslužiti za rješavanje sličnih problema."

1.2 Srednja škola

Iako se u preporukama za ostvarenje odgojno-obrazovnih ishoda ne spominju roboti, u nastavku će biti navedeni ishodi koji se mogu u cijelosti ili dijelom realizirati korištenjem senzora i mikrokontrolera. Ishodi su za opće, jezične, klasične i prirodoslovne gimnazije sa 70 sati informatike godišnje.

B.1.1. "učenik analizira problem, definira ulazne i izlazne vrijednosti te uočava korake za rješavanje problema."

Ulazne vrijednosti mogu biti upravo podaci koje senzori šalju mikrokontroleru, a izlazne vrijednosti mogu biti npr. ispis na displayu, pokretanje motora odnosno robota, itd..

B.1.3. „"učenik razvija algoritam i stvara program u odabranome programskom jeziku rješavajući problem uporabom strukture grananja i ponavljanja."

Tijekom programiranja mikrokontrolera se upravo koriste strukture grananja i ponavljanja. Kako mikrokontroler radi dok god se ne isključi, naredbe se moraju neprestano ponavljati. Također, kako bi se „pokrili“ svi mogući slučajevi koji se mogu dogoditi, koriste se naredbe grananja. Ovo se može i lako demonstrirati pa je učenicima lakše razumijeti zašto se javlja potreba za korištenje naredbi grananja.

Npr. ako je robot preblizu prepreke, treba se okrenuti. Inače, robot može nastaviti ravno.

B.2.2. "učenik u zadanome problemu uočava manje cjeline, rješava ih te ih potom integrira u jedinstveno rješenje problema."

Na primjer, ako želimo koristiti dva senzora, najprije napišemo program koji koristi samo jedan, a zatim program koji koristi samo drugi. Na kraju učenici integriraju dva programa u jedan.

B.2.3. "učenik u suradnji s drugima osmišljava algoritam, implementira ga u odabranome programskom jeziku, testira program, dokumentira i predstavlja drugima mogućnosti i ograničenja programa."

Ovakav projektni zadatak se može zadati i za programiranje mikrokontrolera sa senzorima, odnosno robota.

2 Nastava programiranja u školama

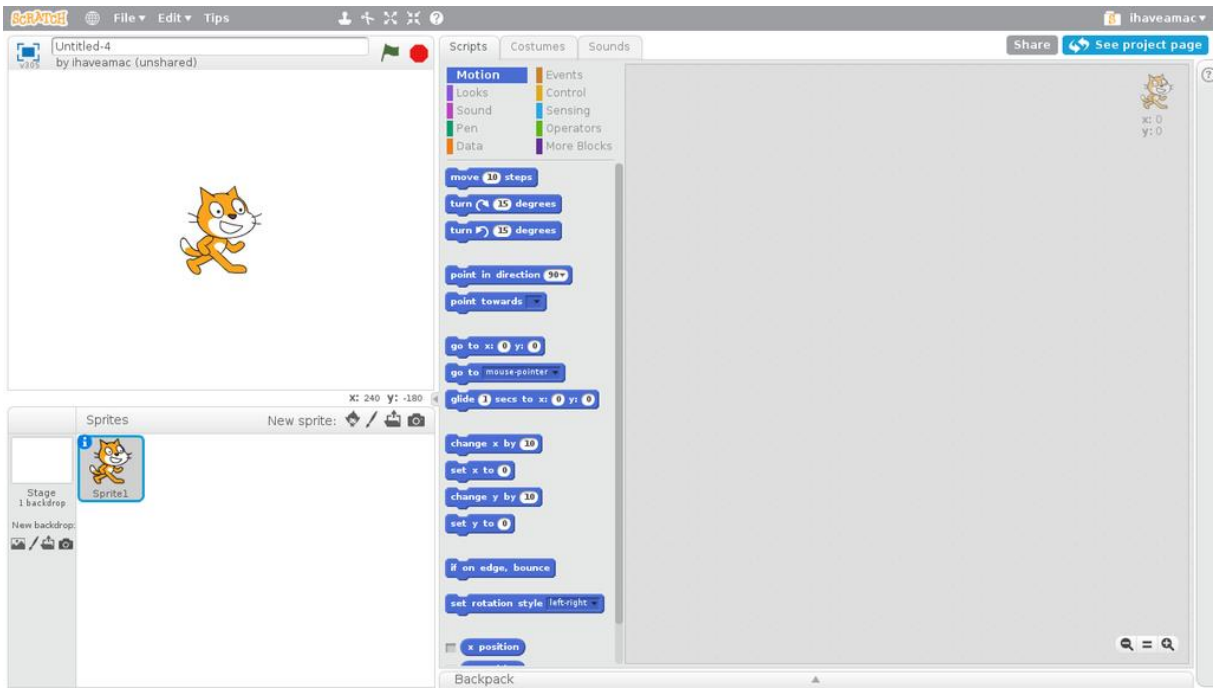
U Nacionalnom kurikulumu nastavnoga predmeta informatika za osnovne i srednje škole (2018.) su navedeni odgojno-obrazovni ishodi za domenu Računalno razmišljanje i programiranje, a na nastavniku je da izabere programski jezik pomoću kojega će ih učenici ostvariti.

U osnovnim školama učenici uče programirati u programskim jezicima Logo, Scratch, Basic i Python. U posljednje vrijeme se na informatici učenici sve više bave programiranjem mikrokontrolera, najčešće micro:bit i Arduino.

Za razliku od programskih jezika Logo i Basic, Scratch je programski jezik s grafičkim sučeljem u kojemu sastavljamo program pomičući već pripremljene blokove naredbi koji se zatim slažu u cjelinu. Takve programske jezike zovemo blokovskim programskim jezicima.

2.1 Usporedba programskih jezika Logo i Scratch

Scratch je besplatni blokovski programski jezik. Razvila ga je grupa Lifelong Kindergarten, dio MIT (Massachusetts Institute of Technology) Media Laba. Prvenstveno je namijenjen poučavanju učenika od 8 do 11 godina starosti. Koristi se kao programski jezik za poučavanje programiranja u školama jer je kreiranje programa jednostavno i zabavno, a time privlačno mlađim učenicima.



Program u Scratchu čine objekti koji se zovu likovi (*eng. sprites*). Likovi se mogu mijenjati dodavanjem različitih kostima. Glavni lik je mačak koji je i logotip programskog jezika Scratch.

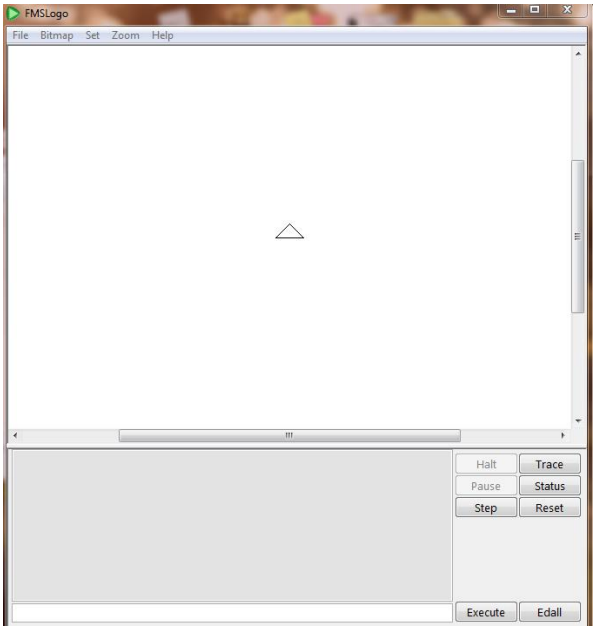
Scratch blokovi razvrstani su u deset skupina različitih boja: Kretanje, Izgled, Zvuk, Olovka, Podaci, Događaji, Upravljanje, Očitanja, Operacije i Varijable. Svaka skupina blokova ima drugačiji izgled koji sugerira na koji se način blokovi kombiniraju s blokovima iz drugih

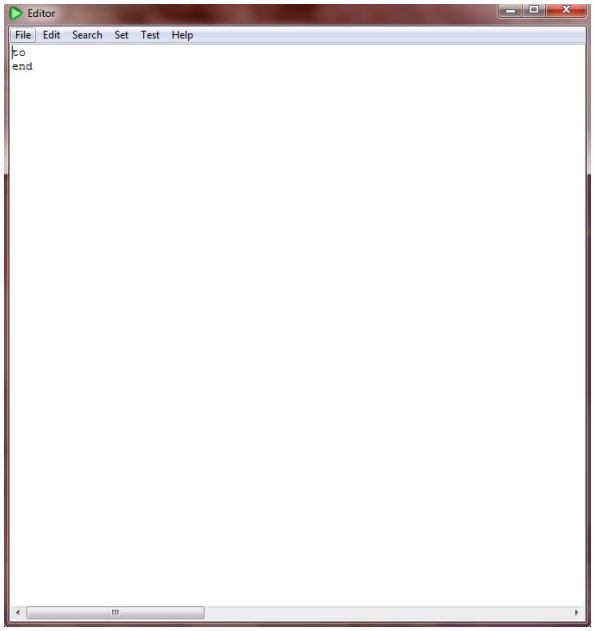
skupina (ili iz iste skupine). Program se sastavlja spajanjem zadanih blokova naredbi, a učenici mogu odmah vidjeti kako izvršavanje naredbi utječe na kretanje lika po ekranu.

Takav način programiranja omogućuje učenicima da bolje shvate logiku programiranja te da se ne opterećuju sintaksom pisanja programa.

Logo

Programski jezik Logo je stvoren za obrazovnu upotrebu. Stvorila ga je „MIT Artificial Intelligence Laboratory“ grupa predvođena Wallaceom Feurzeigom 1967. godine. Iako je programski jezik Logo opće namjene te može raditi sa riječima i listama, najpoznatiji je po korištenju kornjačine grafike. Pokazivač na ekranu pod imenom „kornjača“ prikazuje izvršavanje naredbi za pokretanje i crtanje. Kornjača ima olovku koju može podići ili spustiti, ovisno o tome želimo li ostaviti trag na ekranu ili ne. Može se pomicati ravno ili natrag te okretati u desnom ili lijevom smjeru za određeni broj stupnjeva.





Učenici u Logu koriste uobičajene naredbe poput `If` i `If-Else` te petlju `For`, `While` i `Repeat`. Sučelje programskog jezika Logo omogućava da je rezultat napisanog programskog koda odmah vidljiv. Naredbe se izvršavaju jedna po jedna. Mogu se pisati u traci za naredbe ili možemo napisati programski kod u Editoru. Napisani program treba pozvati u početnom prozoru, u kojemu će se vidjeti rezultat izvršavanja. Također, u početnom prozoru učenici mogu vidjeti poruke o greškama u kojima piše što je netočno napisano.

I Scratch i Logo namijenjeni su korištenju u poučavanju programiranja.

Scratch je osobito pogodan za poučavanje programiranja kod mlađih učenika zbog izgleda sučelja i unaprijed ponuđenih naredbi. Zahvaljujući tome, učenici ne moraju pamtiti naredbe napamet i mogu se usredotočiti na logiku programiranja umjesto na sintaksu. Također, pisanje programa je brže i jednostavnije jer ne moraju tipkati naredbe. Iako je to velika prednost za

učenike mlađih uzrasta, kasnije postaje nedostatak za naprednije učenike kojima Scratch više ne pruža dovoljno izražajnih mogućnosti.

Učenici koji koriste programski jezik Logo moraju naučiti naredbe napamet i tipkati sve naredbe. U početku će im trebati više vremena, ali ih to priprema za kasnije pisanje kompliciranijih programskih kodova. Također, Logo omogućava jednostavnu podjelu programa na manje dijelove i kasnije spajanje tih dijelova u smislenu cjelinu.

U oba programska jezika učenici mogu odmah vidjeti rezultat izvršavanja naredbi što pomaže učenicima da lakše shvate logiku programiranja te olakšava pronalazak grešaka [2].

2.2 Ulazne jedinice

Svaki program se može promatrati kao slijedni proces ULAZ-OBRADA-IZLAZ. **Ulaz** se realizira putem ulaznih jedinica, odnosno uređaja koji omogućuju unos podataka iz okoline u računalo. Jednostavniji programi imaju jednostavan ulaz – nekoliko brojeva ili znakovnih nizova unesenih korištenjem tipkovnice. Zatim se vrši **obrada** unesenih podataka provedbom određenih algoritama te nastaje **izlaz** koji se najčešće ispisuje na zaslonu računala.

Nešto složenija ulazna jedinica je npr. miš pomoću kojega se mogu raditi interaktivni programi koji na temelju akcije zadane putem miša (pozicija miša na zaslonu ili pritisak tipke) izvode određenu radnju.

Danas se sve više kao ulazne jedinice koriste senzori koji samostalno prikupljaju informacije iz okoline i na temelju tih informacija računalo izvodi određene radnje.

2.3 Senzori kao ulazne jedinice

Senzori su postali neizostavni dio života. Današnji pametni telefoni imaju u sebi desetak i više različitih senzora. No, senzori se nalaze i u mnogo jednostavnijim uređajima npr. automatskim vratima, svjetlu u hodniku zgrade, klima uređajima itd.

Senzor, kao sam pojam, izveden je iz latinske riječi „sensus“, koja znači „osjet“ ili „osjećanje“, a predstavlja uređaj koji detektira i reagira na neki unos iz fizičkog okruženja [3].

Senzori su zanimljivi za korištenje u nastavi programiranja u školama jer se pomoću njih postiže veća interaktivnost, pa je time učenicima programiranje zanimljivije. Umjesto zadatka: *Napišite program koji unosi dva broja te ispisuje veći od njih*, zadatak može glasiti: *Napišite program koji javlja treba li biljku zaliti*. Koristeći senzore u nastavi programiranja, učenici će postati svjesni kako funkcioniraju uređaji koje svakodnevno koriste.

Postoje razne vrste senzora. Neki od njih su senzor za zvuk, temperaturu, svjetlost, vlagu, udaljenost, dodir, gibanje, te, kao jedan od kompleksnijih, senzor koji analizira sliku dobivenu kamerom.

3 O Pixyju

Pixy (CMUcam5) je senzor objekata.

CMU je skraćenica od Carnegie Mellon University, što je ime sveučilišta gdje je CMUcam projekt nastao. Sjedište sveučilišta se nalazi u Pittsburgu, Pennsylvania.

CMUcam je projekt čiji je cilj pružiti jednostavne mogućnosti vida u obliku inteligentnog senzora. CMUcam senzori otvorenog koda (*open source*) s ugrađenim senzorom boja su relativno jeftini niskoenergetski senzori za mobilne robote [4]. CMU kamera se sastoji od male video kamere i mikrokontrolera sa serijskim sučeljem. CMUcam-ovo jednostavno sučelje omogućuje jednostavno povezivanje s mikrokontrolerom. Također je važno to da ugrađeni mikroprocesor podržava jednostavnu obradu slike i praćenje nakupina[1] boja, što je osnova računalnog vida. Time je računalni vid omogućen u sustavima koji bi inače bili preslabi za takvu vrstu obrade [5].

Postoje starije verzije CMUcam1, 2, 3 i 4, a CMUcam5 je nazvan Pixy. Od nedavno je također dostupna manja i brža verzija Pixy2.

Vid kao senzor je koristan kada želimo da robot izvrši zadatak poput podizanja, praćenje ili pronalaženje objekta. No, takvi senzori često šalju previše podataka koje procesor mikroračunala mora obraditi zbog čega ne može obaviti druge zadatke.

Pixy rješava ovaj problem tako što ima vlastiti procesor za obradu slika. On obrađuje slike dobivene putem kamere te šalje mikrokontroleru samo nužne informacije (na primjer, *x* i *y* koordinate detektiranog objekta, veličinu objekta, boju itd.). Ovaj senzor obrađene informacije šalje mikrokontroleru čak 50 puta u sekundi što je dovoljno za programiranje edukacijskog robota odnosno sličnih sustava koji služe za učenje programiranje sustava upravljanih senzorima [6].

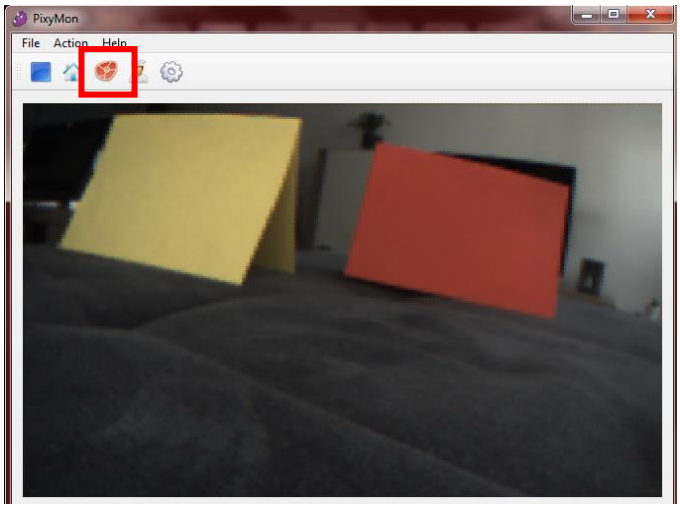
3.1 PixyMon

PixyMon je program koji nam omogućuje da na zaslonu računala vidimo što Pixy vidi odnosno koje objekte prepoznaje. U njemu možemo konfigurirati Pixyja i upravljati potpisima (*signature*) tj. svojstvima objekata koje ovaj senzor treba prepoznavati i slati informacije mikrokontroleru.

U ovom programu postoje tri načina prikaza [8]:

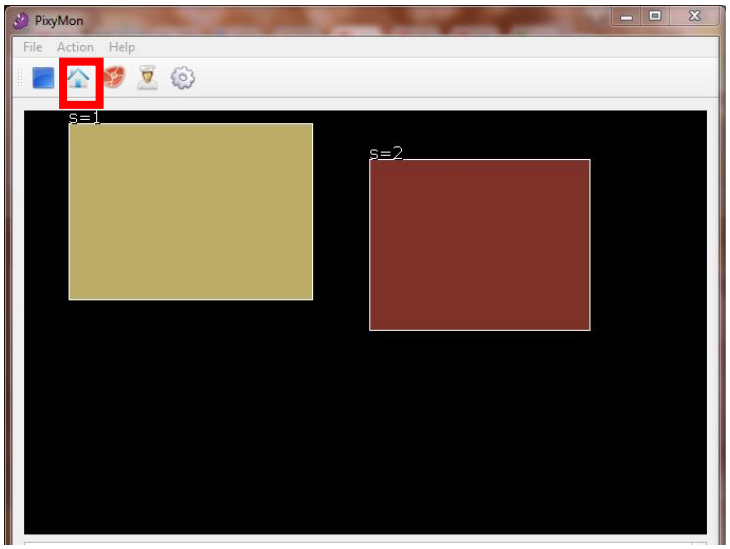
[1]Nakupina (eng. *blob)* je grupa povezanih piksela na slici koji imaju neka zajednička svojstva [7].

3.1.1 Raw video



Pritiskom na ovaj gumb vidimo „sirovi“, neobrađeni video tj sliku koja je snimljena kamerom koja se nalazi na senzoru Pixy.

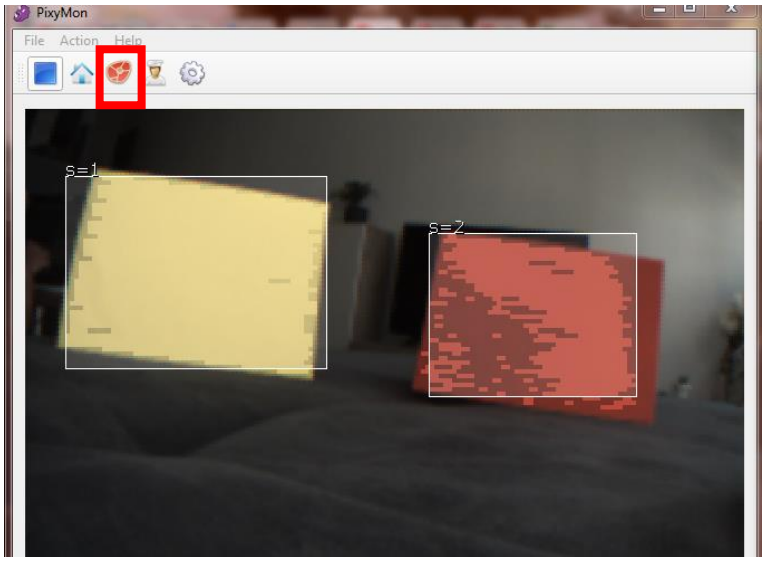
3.1.2 Default program



To je zadani program koji se izvršava kada se Pixy uključi. On obavlja svu obradu podataka i šalje rezultate. Pritiskom na ovaj gumb ne vidimo sliku koju Pixy dobiva preko kamere, već samo obrađene podatke (gdje se objekt nalazi, koje je boje odnosno koji je njegov potpis).

Kada priključimo Pixy na mikrokontroler (npr. Orion pločicu), možemo koristiti samo Default program ako želimo na zaslonu računala vidjeti što Pixy vidi, odnosno koje objekte prepoznaje.

3.1.3 Cooked video



Pritiskom na ovaj gumb vidimo kako Pixy obrađuje slike. Na njemu istovremeno vidimo video koji Pixy dobiva od kamere te obrađene podatke (gdje se objekt nalazi i koji je njegov potpis).

Napomena

Bilo bi korisno za učenike da možemo vidjeti Cooked video u isto vrijeme kada mikrokontroler koristi Pixy. Naime, može se dogoditi da zbog drugačijeg osvjetljenja u prostoriji Pixy ne prepozna označeni objekt. Učenicima to može biti zbunjujuće jer ga oni vide, ali ga Pixy svejedno ne detektira, odnosno ne znaju kakve ulazne informacije ima program u odnosu na stvarnu sliku. Korištenjem Cooked video u isto vrijeme kada mikrokontroler koristi Pixyja, učenici bi mogli vidjeti točnu sliku koju Pixy dobiva te prepoznaje li Pixy objekt ili ne.

Primjer - Problem osvjetljenja

Zamračeni prozori u prostoriji

Više svjetla

Kao što smo već spomenuli, ako je tijekom izvođenja programa osvjetljenje u prostoru drugačije od osvjetljenja u onom trenu kada smo Pixy naučili prepoznavati određeni objekt, može se dogoditi da ga Pixy ne prepozna zbog čega se program koji ovisi o prepoznatim objektima neće ponašati na očekivani način.

Ako radimo samo s jednim objektom, npr. želimo da edukacijski robot mBot prati lopticu, tada možemo postaviti više potpisa istog objekta u različitim osvjetljenjima kako ne bismo morali iznova označavati objekt u svakoj prostoriji.

Takav problem se pojavio Otvorenom danu matematike 2018. (Prirodoslovno-matematički fakultet Sveučilišta u Zagrebu). U sklopu Otvorenog dana smo napravili prezentaciju pratećeg robota koristeći dva mBota i Pixy. Jedan mBot je imao crveni balon, a drugi ga je pratio koristeći Pixy. Problem je nastao kada smo u PixyMonu označili balon u učionici s puno prirodnog svjetla (prva slika), a prezentacija se odvijala u hodniku gdje nema prirodnog svjetla (druga slika). Zbog velike razlike u osvjetljenju, Pixy u hodniku nije mogao detektirati balon.

Učionica

Hodnik

3.2 Potpisi (*signatures*)

Pixy može zapamtiti sedam potpisa, što znači da može detektirati sedam objekta različitih boja. U slučaju kada trebamo zapamtiti više od sedam objekata, možemo koristiti kodiranje bojama (*color codes*) [9].

Kako Pixy prepoznaje objekte na temelju njihove boje, bitno je da su objekti prepoznatljive nijanse. Iz tog razloga, nije poželjno da je objekt kojeg želimo detektirati crne, sive ili bijele boje. Možemo naučiti Pixy da prepoznaje neki objekt i koristeći dugme na samom Pixyju.

Nakon što odaberemo objekt, spojimo Pixy na izvor struje. Uključit će se LED lampice. Nakon što se lampice isključe, treba pritisnuti (i držati) bijelo dugme na vrhu Pixy kamere. Nakon jedne sekunde, uključit će se LED lampica bijele boje, a zatim crvene. Tada treba pustiti dugme. Nakon toga namjestimo objekt u centar kadra. Pixy u toj fazi uključuje LED lampice da svijetle onom bojom koje boje je i sam objekt. Kada smo zadovoljni s nijansom boje, pritisnemo i odmah pustimo dugme (kao klik na mišu). Ako je postavljanje potpisa uspjelo, LED lampica će zasvijetliti nekoliko puta. U protivnome, lampica neće zasvijetliti.

Na prethodno opisani način ćemo označiti objekt s potpisom 1. Ako želimo naučiti Pixy da prepozna još jedan objekt, odnosno želimo mu dati potpis 2, ponovno pritisnemo bijelo dugme na vrhu Pixy kamere i čekamo da se uključi narančasto svjetlo (nakon crvenog) te pustimo dugme. Ponovno moramo potvrditi boju klikom na dugme, kao i u prethodnoj situaciji.

Označavamo objekte različitim potpisima ovisno o tome kojom bojom svijetle LED lampice kada pustimo dugme.

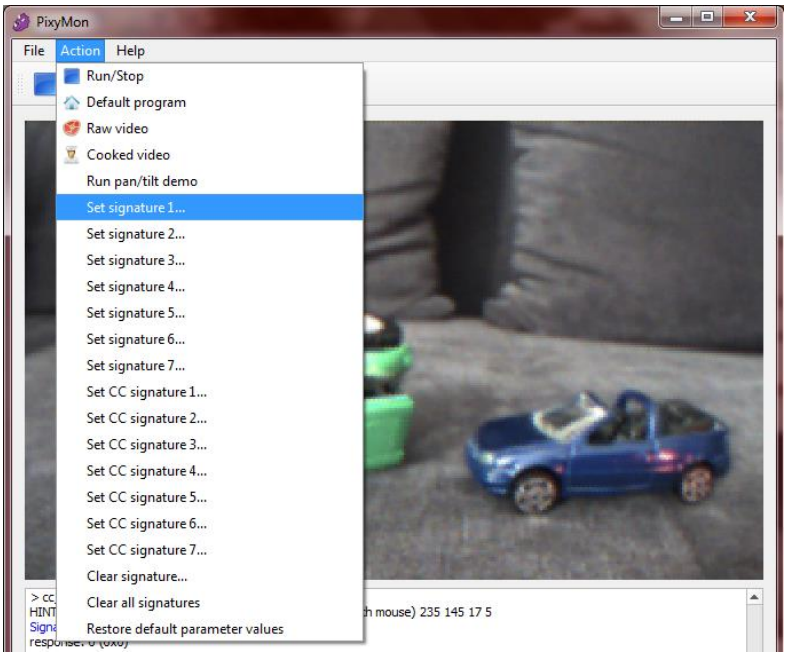
Ovo je redoslijed boja, odnosno potpisa:

1. crvena
2. narančasta
3. žuta
4. zelena
5. cijan
6. plava
7. ljubičasta

Ako pustimo dugme u trenutku kada lampica svijetli plavom bojom, objekt će dobiti potpis broj 6. Važno je napomenuti da boja lampice nije povezana s bojom objekta kojega želimo označiti. Boja lampice nam samo označava kojim potpisom ćemo označiti objekt.

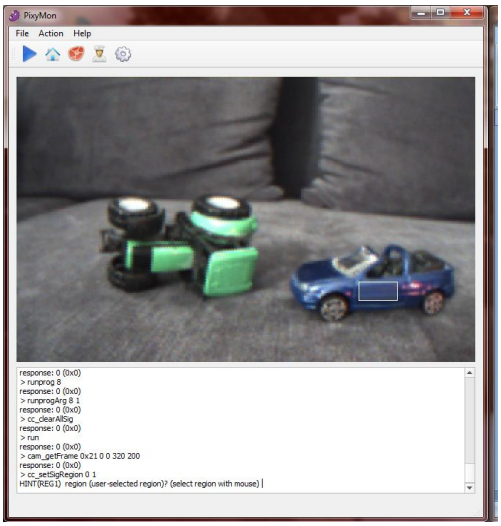
Također možemo naučiti Pixy da detektira objekt koristeći program PixyMon.

Nakon što povežemo Pixy s računalom, pokrenemo PixyMon. Na izborniku odaberemo *Action* i zatim *Set signature 1...*

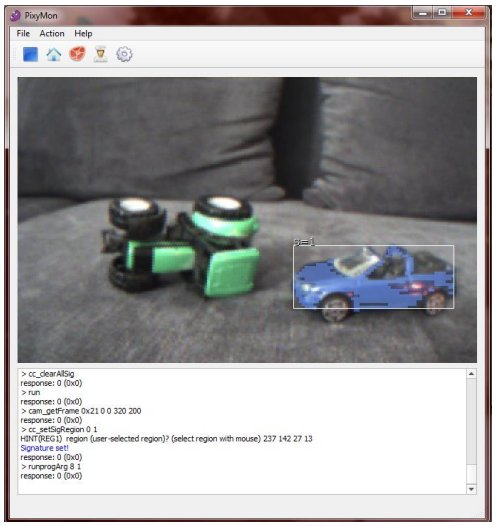


Zatim obilježimo područje za koje želimo da Pixy iskoristi kako bi naučio prepoznati objekt.

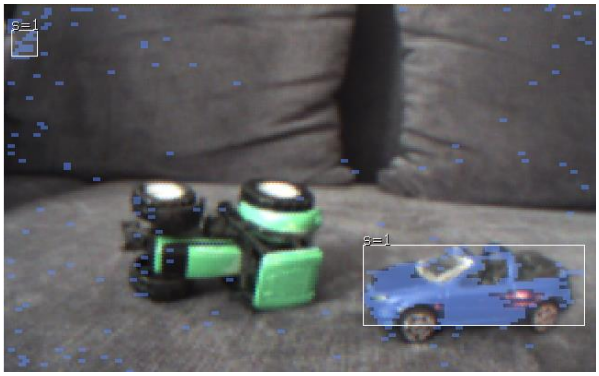
Označeno područje



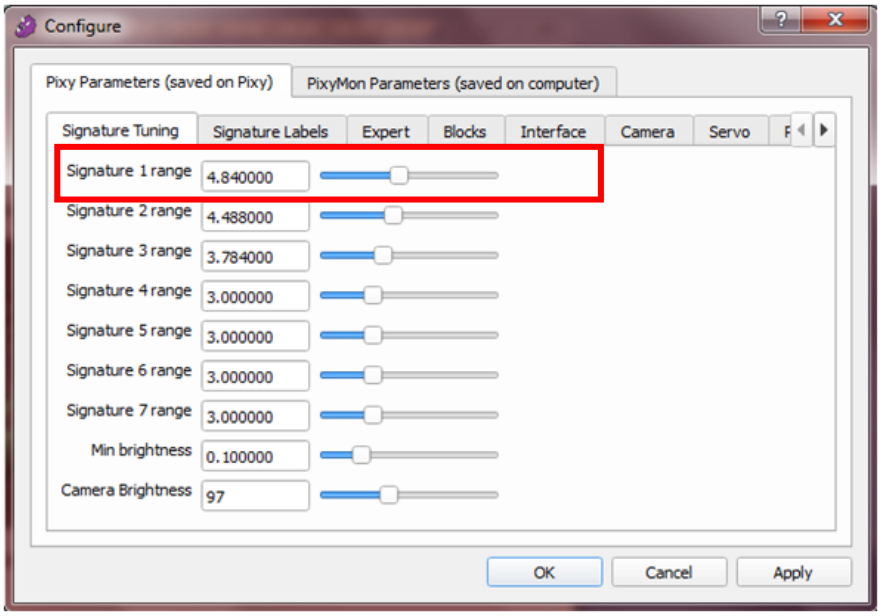
Detektirani objekt s potpisom 1



Može se dogoditi da Pixy detektira piksele iste boje u ostatku kadra, kao na sljedećoj slici.

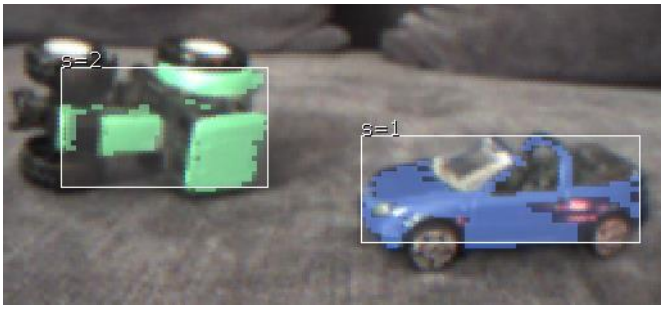


Tada možemo prilagoditi osjetljivost tako da na izborniku odaberemo *File* → *Configure.* Otvori se sljedeći dijaloški okvir:

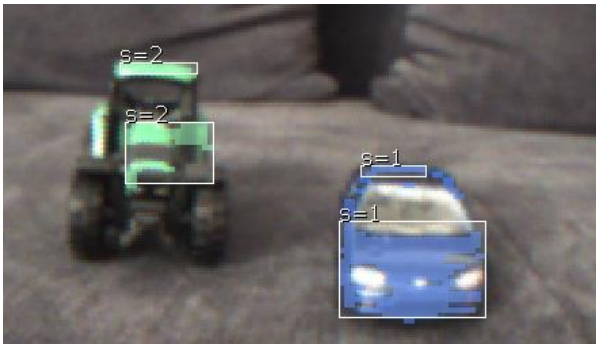


Smanjimo *Signature 1 range* na vrijednost dovoljno veliku da Pixy detektira objekt koji želimo, a dovoljno nisku da ne detektira piksele te boje u ostatku kadra.

Na isti način naučimo Pixy da prepozna i drugi objekt koji će dobiti potpis 2. Ovaj put odaberemo *Set signature 2...*.



Napomena: Ako objekt kojega želimo detektirati nije jednobojan, može se dogoditi da Pixy umjesto jednog objekta prepozna dva objekta iste boje, kao što je prikazano na slici ispod.



Ako želimo izbrisati sve potpise, na izborniku u programu PixyMon odabremo *Action*, a zatim *Clear all signatures.* Ako želimo obrisati samo pojedini potpis, na primjer potpis broj 2, odaberemo *Clear signature...* Dobijemo sljedeću poruku:

```
INT8 signature (numerical index of signature, can be 1-7)?
```

Zatim upišemo redni broj potpisa kojeg želimo obrisati i pritisnemo tipku *Enter*.

3.3 Kodiranje bojama (Color Code - CC)

Kodiranje bojama (Color code - CC) se sastoji od dviju ili više oznaka u različitim bojama koje su postavljene jedna kraj druge. Kako s potpisima (*signatures*) možemo detektirati samo sedam objekata različitih boja, ovakvo kodiranje je korisno ako imamo puno objekata koje želimo detektirati. Na primjer, pomoću CC-ova s dvije oznake i četiri različite boje možemo generirati šest jedinstvenih objekata. Pixy može detektirati CC-ove s najviše pet oznaka raspoređenih u retku/stupcu. To nam omogućuje generiranje tisuće jedinstvenih CC-ova.

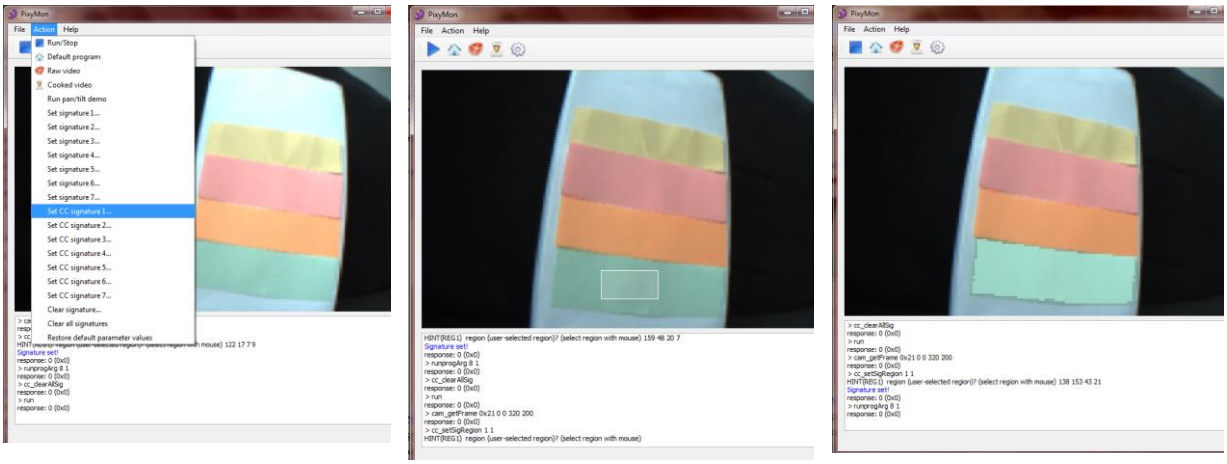
Korištenjem kodiranja bojama se također poboljšava točnost detekcije, to jest, postoji manja vjerojatnost da će se specifične boje pojaviti u određenom redoslijedu i jedna pored druge. Osim toga, možemo staviti veći opseg prepoznavanja pojedine boje budući da će se kompletan kod boja prepoznati tek ako su sve boje u točnom redoslijedu. Također, omogućuje nam detekciju objekata koji nemaju specifičnu žarku boju koja se lako detektira kao na primjer sivi i bijeli objekti.

CC-ovi također mogu biti korisni za navigiranje robota. Na primjer, možemo lako označiti vrata svake prostorije različitim kodiranjima bez većih troškova, a robot bi na taj način znao u kojoj se prostoriji nalazi. Nedostatak je da moramo postaviti ovakvo kodiranje na svaki objekt koji želimo detektirati [10].

Primjer: Postavljanje CC potpisa koristeći četiri boje

U načinu prikaza *Cooked video* odaberemo *Action→Set CC signature 1...* i označimo prvu boju (slike ispod).

Set *CC signature 1...* Označeno područje Detektirana boja

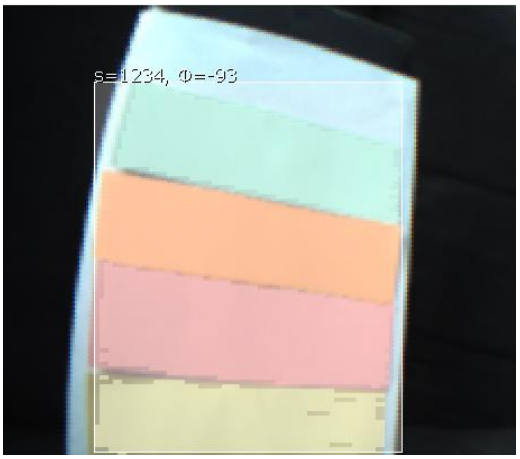


Za drugu boju odaberemo *Set CC signature 2...* Ponovimo postupak i za ostale boje. Vidimo rezultat u različitim načinima prikaza na slikama ispod.

Cooked video Default program

Ako odaberemo Default program, ne vidimo boje, kao u slučaju kada imamo običan potpis (samo jedna boja).

Zelena boja ima CC potpis 1, narančasta 2, ružičasta 3, a žuta 4. Vidimo na prethodnoj slici da je potpis detektiranog objekta s=1234. To je ustvari kodiranje bojama (*Color Code*) koje nam govori u kojem su poretku boje. Ako boje poredamo u suprotnom smjeru, dobit ćemo situaciju kao na sljedećoj slici.

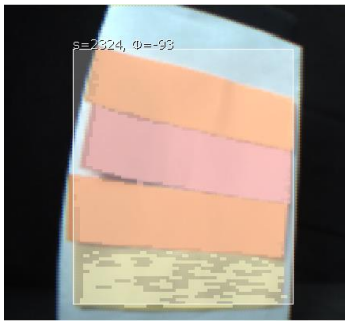


Vidimo da je potpis detektiranog objekta opet s=1234. To znači da dva CC- a imaju identične potpise ako im je redoslijed boja isti, bez obzira s koje strane krenuli. Ako promijenimo redoslijed boja, na primjer zamijenimo narančastu i ružičastu boju, dobivamo sljedeću situaciju.

Potpis objekta s ovim poretkom boja je 1324.

Boje ne moraju nužno biti različite, ali ako koristimo više istih boja, moraju biti odvojene nekom drugom bojom. Ako zamijenimo zelenu boju s narančastom, dobivamo potpis 2324. Ako ju pak zamijenimo sa ružičastom, dobit ćemo potpis 324. Drugim riječima, ako su dvije trake iste boje jedna kraj druge, Pixy će ih detektirati kao jednu boju. Možemo vidjeti oba primjera na slikama ispod.

s=2324

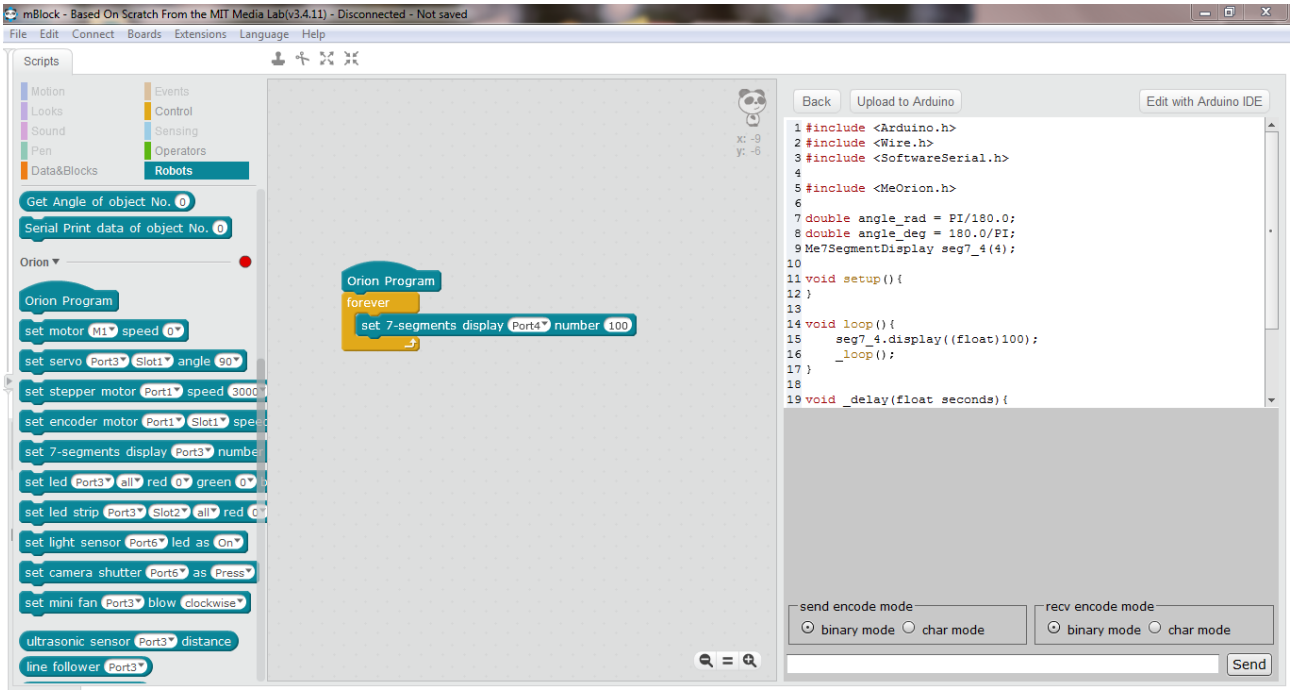


s=324

4 Pixy u mBlocku

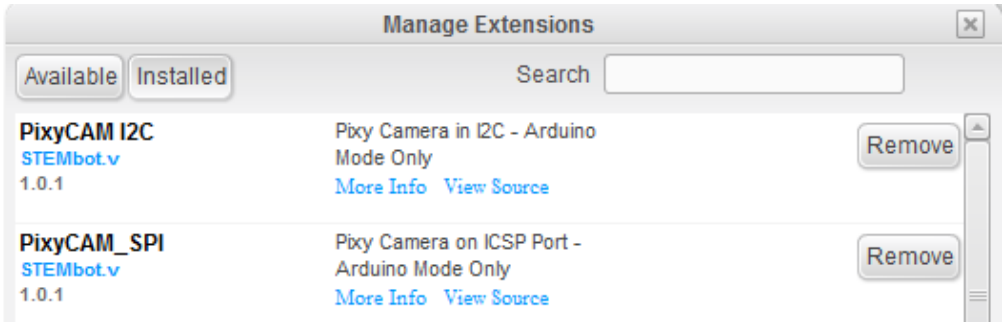
mBlock je programerski softver dizajniran za STEM obrazovanje. Inspiriran je programskim jezikom Scratch. Kao i Scratch, bazira se na blokovskom programiranju te je besplatan. Sadrži iste naredbe kao i Scratch uz koje su još dodane naredbe za Makeblock robote. Kako je mBlock baziran na Scratchu, učenicima koji su do tada programirali u Scratchu je jednostavno prijeći na mBlock. Umjesto pokretanja likova na ekranu, u mBlocku će pokretati robote.

Dostupne su dvije verzije mBlocka, mBlock 3 i mBlock 5 (beta verzija). Kada slažemo blokove u mBlocku, on ih u stvarnom vremenu prevodi u tekstualni programski jezik koji se može otvoriti u Arduino IDE. mBlock 3 prevodi blokovski program u programski jezik C (slika ispod), a mBlock 5 u programski jezik Python. To omogućuje učenicima da jednostavnije i intuitivnije prijeđu na tekstualni programski jezik u višim razredima osnovne škole ili u srednjoj školi. Također, ako učenici žele napisati kompliciraniji program, nisu ograničeni blokovima naredbi koji su ponuđeni u mBlocku [11]. Kako je mBlock 5 još uvijek beta verzija, u ovom radu će se koristiti mBlock 3.



Osim s robotom mBot te drugim robotima tvrtke Makeblock, Pixy je kompatibilan i s Arduino i micro:bit hardverom.

mBlock nudi dvije ekstenzije za rad s Pixyjem, PixyCAM I2C i PixyCAM_SPI (slika ispod).



Klikom na „More Info“ za obje ekstenzije dolazimo do web stranice https://stembot.vn/.

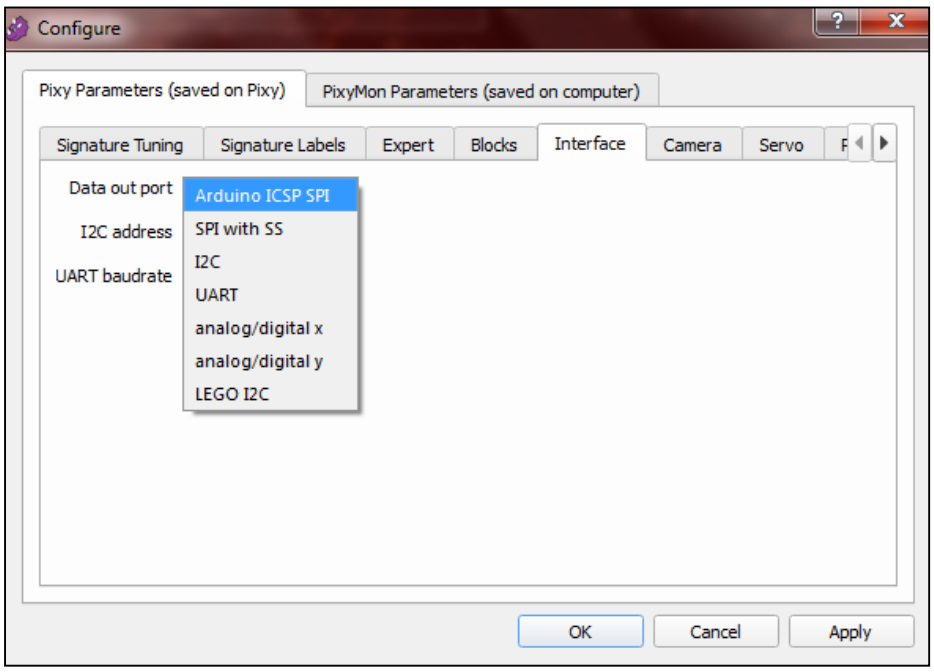
4.1 Razlika između I2C i SPI

Pixy može komunicirati s mikrokontrolerom na tri različita načina:

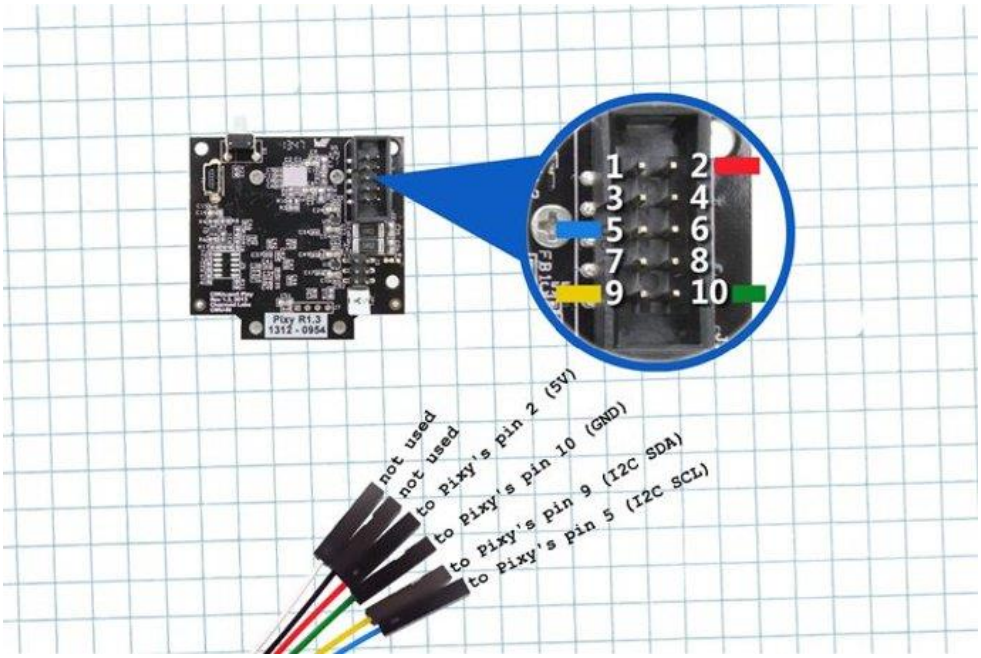
- Serijska komunikacija: uključuje SPI (Serial Peripheral Interface), I2C (Inter-Intergrated Circuit) i UART (Universal Asynchronous Receiver/Transmitter) sučelje. Ovaj način komunikacije Pixy koristi za komuniciranje s Arduinom.
- USB: USB sučelje je namijenjeno za mikrokontrolere koji imaju više memorije. Pixy na ovaj način komunicira s Raspberry Pi i BeagleBone Black mikrokontrolerima. Ovaj način komunikacije se također koristi za korištenje PixyMon programa.
- Analogno/digitalno: najjednostavniji način komunikacije

Nakon što se odlučimo za način komuniciranja, moramo konfigurirati Pixy koristeći program PixyMon. Na izborniku odabremo *File→Configure...*

Odaberemo karticu *Pixy Parameters (saved on Pixy)*, a zatim *Interface.* Odaberemo koji način komunikacije želimo koristiti tako da otvorimo padajući izbornik pokraj *Data out port*, kao što je prikazano na slici ispod [12].



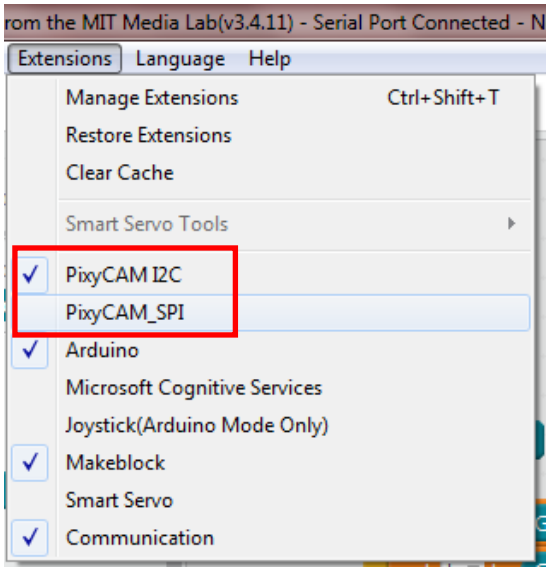
U nastavku rada ćemo koristiti I2C način komunikacije. Pixy moramo povezati pomoću Dupont kabela na sljedeći način [13]:



Kako smo odabrali I2C način komunikacije, u nastavku rada ćemo koristiti ekstenziju PixyCAM_I2C.

Da bismo mogli koristiti ekstenziju PixyCAM_SPI, moramo najprije spojiti Pixy s mikrokontrolerom na način da koristi SPI način komunikacije. Na stranici https://arduino.stackexchange.com/questions/16348/how-do-you-use-spi-on-an-arduino se nalaze upute za spajanje na nekoliko Arduino mikrokontrolera. U ovom radu se nećemo baviti SPI načinom komunikacije jer nam je dovoljno imati jedan način koji radi.

VAŽNA NAPOMENA: Kada smo odabrali PixyCAM_I2C, ne smijemo zaboraviti maknuti drugu ekstenziju. Ovo je prikazano na sljedećoj slici:



U suprotnom se dogodi da program napisan korištenjem naredbi iz ekstenzije PixyCAM_I2C ispravno radi, a nakon spremanja i ponovnog otvaranja programa, mBlock automatski prijeđe na ekstenziju PixyCAM_SPI, što se vidi u kodu generiranom za Arduino IDE.

PixyCAM_I2C

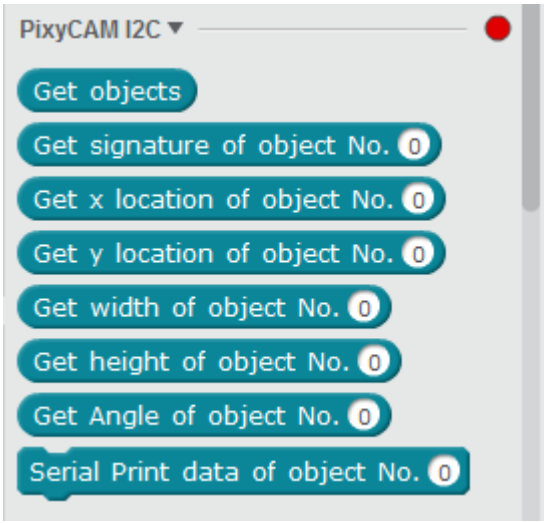
```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <Wire.h>
#include "PixyI2C.h"
#include <MeOrion.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
double a;
double b;
PixyI2C pixy;
Me7SegmentDisplay seg7_4(4)

void setup(){
    pixy.init();
    pixy.init();
    Serial.begin(115200);

}

void loop(){
...
}
```

PixyCAM_SPI

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <SPI.h>
#include "Pixy.h"
#include <MeOrion.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
double a;
double b;
Pixy pixy;
Me7SegmentDisplay seg7_4(4);

void setup(){
    pixy.init();
    Serial.begin(115200);

}

void loop(){
...
}
```

4.2 Naredbe iz biblioteke PixyCAM I2C

Ovo su ponuđene naredbe iz biblioteke PixyCAM I2C za rad s Pixyjem:



Naredba **Get objects** vraća broj objekata koje je Pixy detektirao te listu u kojoj su spremljeni podaci o tim objektima. Ako Pixy nije detektirao niti jedan objekt, ova naredba vraća 0.

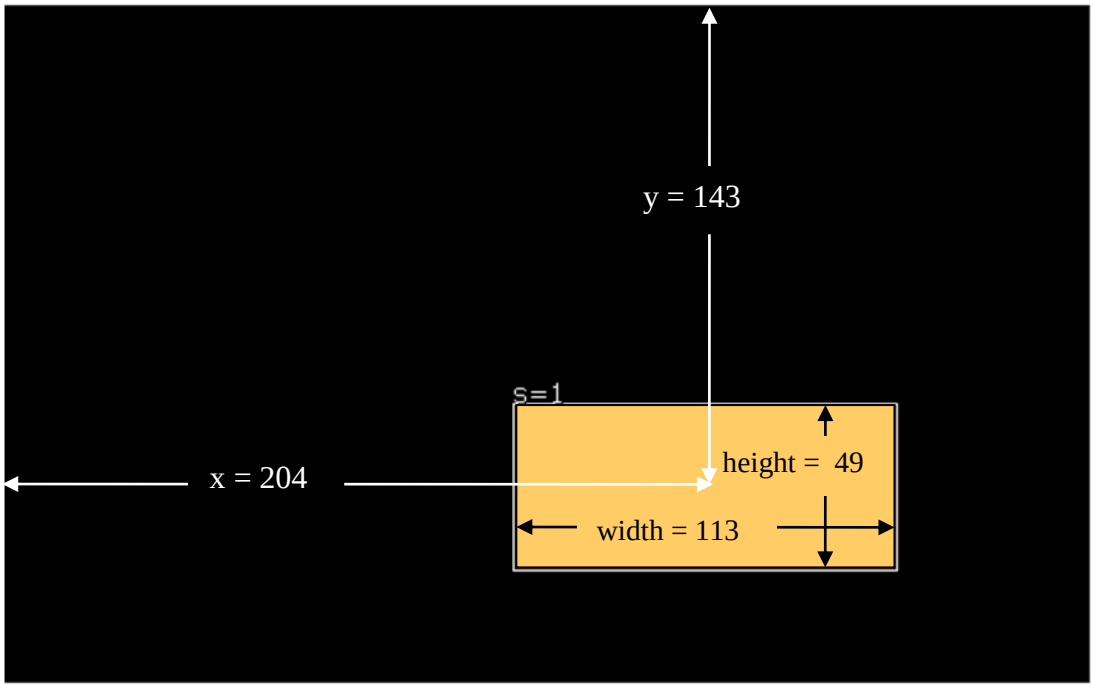
Do elemenata liste koju vraća naredba **Get blocks** možemo doći koristeći ostale naredbe.

Na primjer, naredba **Get signature of object No. i** pristupa listi koju vraća **Get objects** i vraća potpis objekta na *i*-tom mjestu liste. Analogno za naredbe **Get x location**, **Get y location**, **Get width**, **Get hight** i **Get Angle of object No. i**.

Preostaje naredba **Serial Print data of object No. i**. Ona ispisuje sve podatke o objektu na *i*-tom mjestu liste (potpis, *x* i *y* koordinate, širina, visina i kut).

Slika ispod prikazuje položaj detektiranog lika, njegove *x* i *y* koordinate, širinu i visinu. Pixy za *x* i *y* koordinate uzima središte pravokutnika.

Podaci o objektu su sljedeći: `sig: 1 x: 204 y: 144 width: 113 height: 48.`

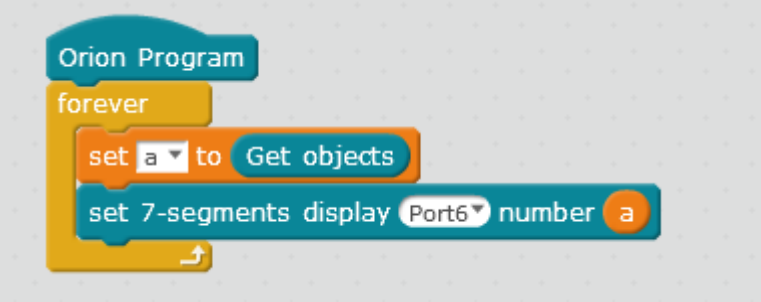


VAŽNA NAPOMENA! Prije korištenja bilo koje naredbe obavezno moramo prethodno zadati naredbu Get objects.

4.2.1 Naredba Get objects

Kao što je već rečeno, naredba **Get objects** vraća broj detektiranih objekata.

Napišimo program u mBlocku koji će na 7-segmentni display ispisati broj detektiranih objekata.



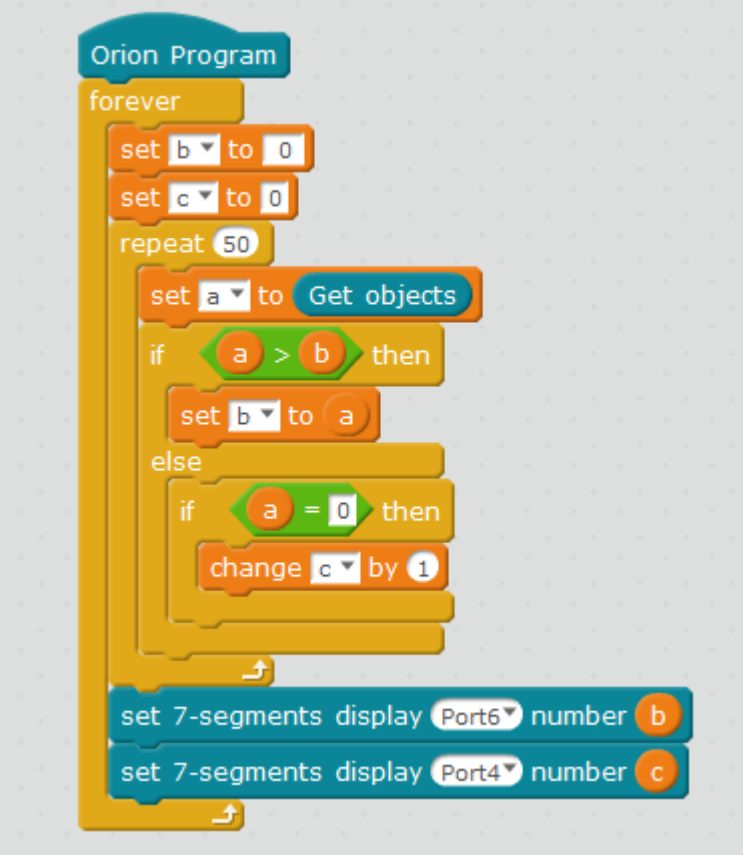
U varijablu **a** spremimo broj objekata koji vraća naredba **Get objects**. Zatim na 7-sgmentnom displayu koji je spojen na port 6 ispišemo taj broj.

Problem: Na 7-segmentnom displayu se nalazi broj 0. Povremeno display zatreperi na dijelić sekunde kada prikaže neki drugi broj. Iz ovog primjera bismo mogli doći do zaključka da Pixy nije detektirao niti jedan objekt ili da naredba **Get objects** u mBlocku ne radi dobro.

Rješenje problema

Pixy šalje podatke mikrokontroleru čak 50 puta u sekundi. No, kada Pixy nije spreman poslati podatke, on na upit mikrokontrolera vraća broj 0 kao da nije detektirao niti jedan objekt.

Ovaj zaključak možemo provjeriti pomoću sljedećeg primjera:



Spojili smo dva 7-segmentnog displaya na Orion pločicu.

Najprije smo postavili varijable **b** i **c** na nulu. U varijabli **b** ćemo pamtiti najveći broj koji je Pixy vratio, a u varijabli **c** ćemo brojati koliko puta je Pixy vratio broj 0. Ponavljamo petlju

`repeat` 50 puta. Svaki put ćemo „pitati“ Pixy koliko objekata je detektirao pomoću naredbe **Get objects**.

50 puta ponavljamo sljedeći postupak:

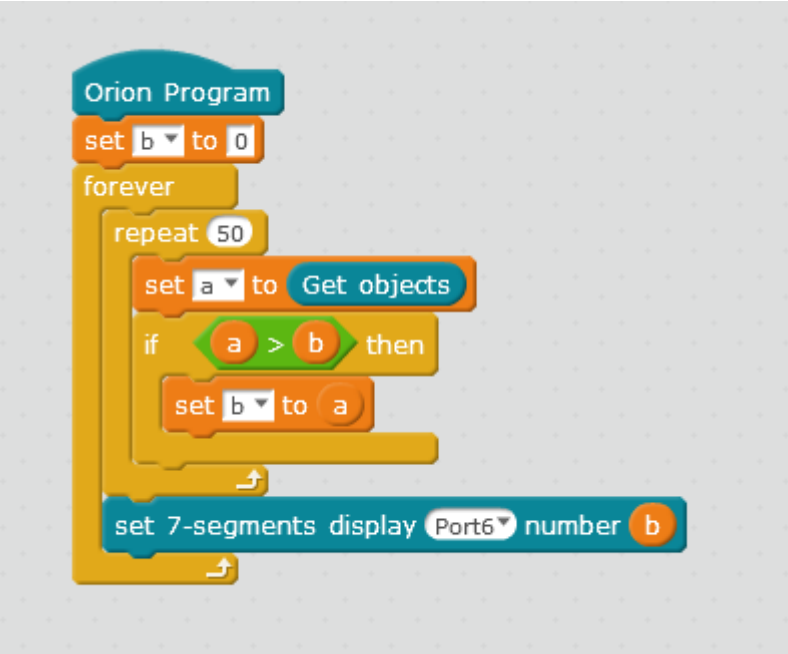
- pitamo Pixy koliko je objekata detektirao
- ako je Pixy vratio broj veći od 0, spremimo taj broj u varijablu `b`
- inače, ako je Pixy vratio broj 0, povećamo varijablu `c` za 1.

Na kraju ispišemo na jednom 7-segmentnom displayu koji je najveći broj kojega je Pixy vratio (varijabla `b`), a na drugom displayu koliko je puta u 50 upita Pixy vratio broj 0 (varijabla `c`). Ponavljamo gore navedeni postupak zauvijek (petlja `forever`).

Rezultat: Sada jedan 7-segmentni display pravilno prikazuje broj detektiranih objekata, a drugi display prikazuje da je Pixy na 47 (ili 48) od ukupno 50 upita vratio broj 0.

Zaključak: Da bismo u mBlocku dobili stvaran broj objekata koje je Pixy detektirao, trebamo „pitati“ Pixy barem 50 puta koliko objekata je detektirao. Kako Pixy od 50 puta čak 47 puta (ili čak više) šalje broj 0, iz čega bismo mi zaključili da nije detektirao niti jedan objekt, velika je vjerojatnost da za broj upita manji od 50 ne bismo dobili dobru informaciju.

Sljedeći program u mBlocku ispravno prikazuje na 7-segmentnom displayu broj detektiranih objekata.

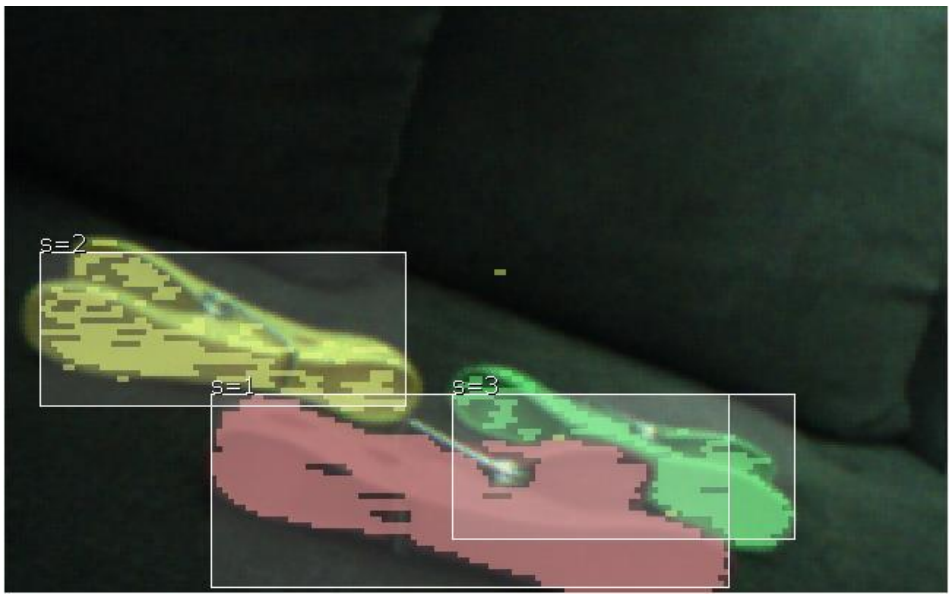


Ako u mBlocku želimo proći kroz listu koju vraća **Get objects** naići ćemo na problem. Naime, naredbe za Pixy se mogu koristiti samo u načinu rada ***Arduino Mode***, a kada smo u tom načinu rada, mBlock ne dopušta rad s listama. To znači da je teško doći do podataka koje vraća **Get objects**.

Do elemenata liste može se doći „pješke“, koristeći petlju **Repeat** i povećavajući brojač za 1. Tu se javlja novi problem – mBlock postavlja varijablu kao tip podataka `double`, a ne `integer` pa ovakva varijabla ne može poslužiti za dohvat elementa liste na određenoj poziciji. Možemo se snaći pomoću trika koristeći **pick random (i,i)**. Tako ćemo dobiti varijablu `i` koja je cjelobrojnog tipa (`integer`). No, to znatno komplicira naš program, a ovakve zaobilazne načine rješavanja problema posebno treba izbjegavati u nastavi kod poučavanja programiranja.

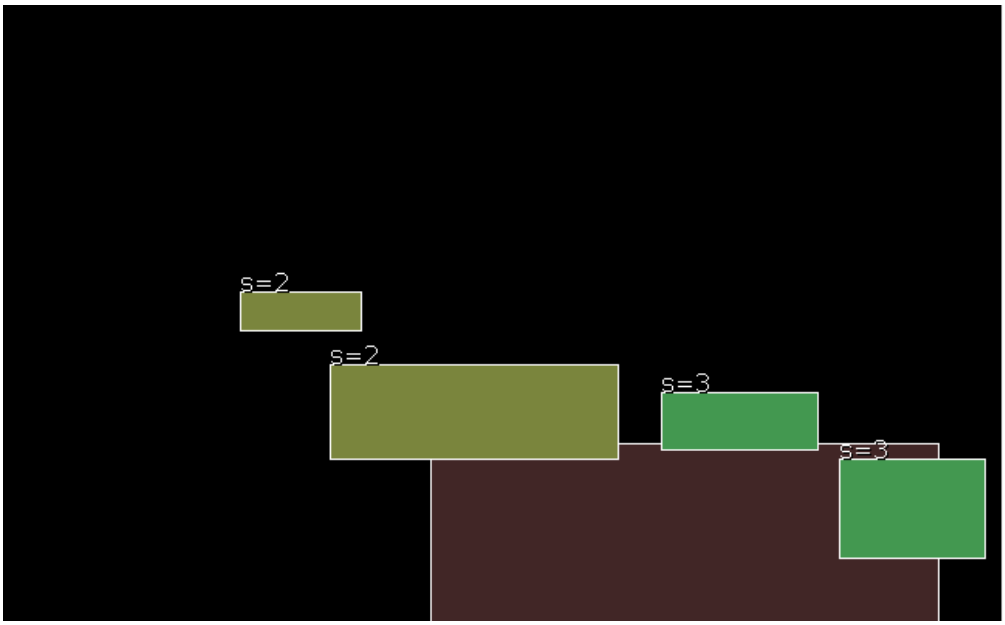
U nastavku će biti opisan sljedeći problem kod dohvaćanja broja detektiranih objekata.

U kadru imamo sljedeću situaciju (tri štipaljke za rublje):



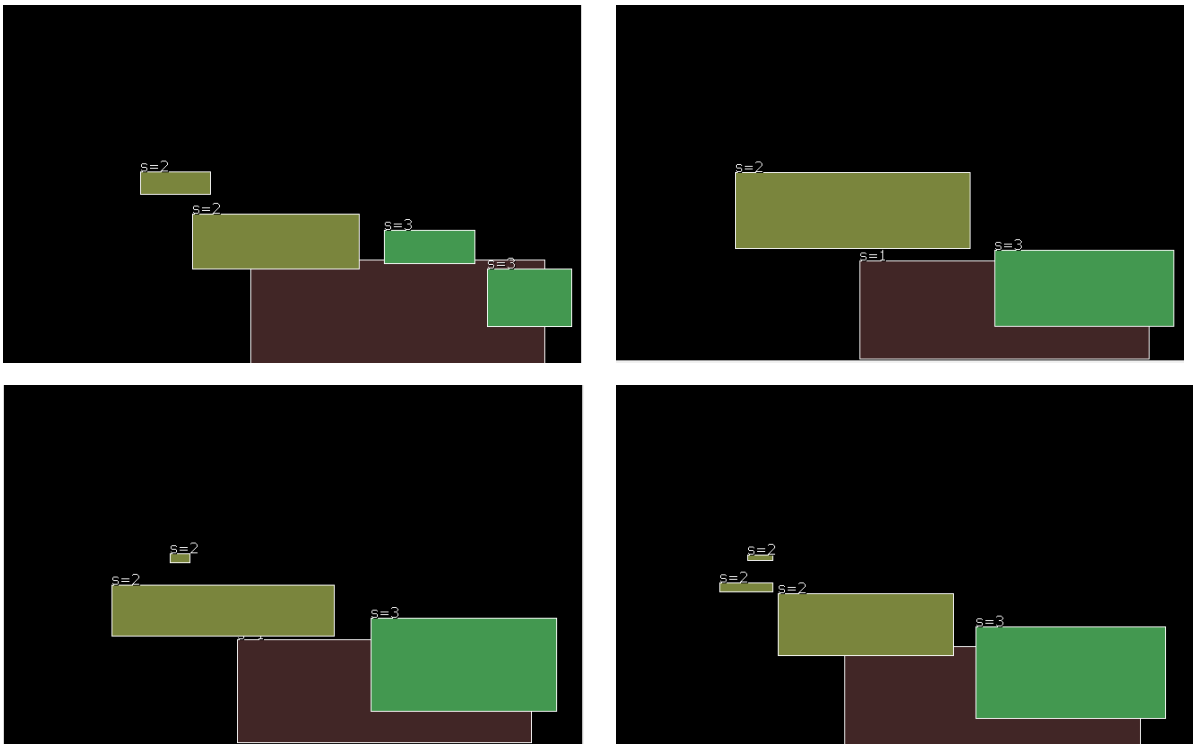
Vidimo da imamo tri objekta, ružičasti s potpisom 1, žuti s potpisom 2 i zeleni s potpisom 3.

Kada koristimo Default pogram u PixyMonu, dobijemo sljedeći prikaz:



Žuti i zeleni objekti su „rastavljeni“ na dva manja objekta.

No, broj detektiranih objekata se vrlo brzo mijenja. Na sljedećim slikama je prikazano stanje unutar nekoliko sekundi:



Kada bismo koristili program iz prošlog primjera u kojemu smo ispisali najveći mogući broj objekata koje je Pixy detektirao, ne bismo dobili realno stanje. Ispisani broj objekata bi bio pet umjesto tri.

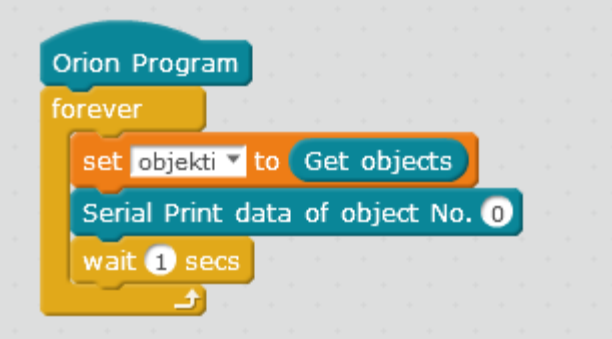
Bilo bi dobro niz očitanja spremiti u datoteku, a zatim filtrirati dobivene podatke. Na primjer, objekt prikazan kao puno manjih objekata prikazati jednostavnije, odnosno doći do zaključka da je to ustvari jedan objekt.

Napomena: Da bi biblioteka bila iskoristiva za niže uzraste učenika, ne smijmo se baviti detaljima i „trikovima“ kako bismo došli do željenih informacija

Za buduće korištenje Pixya i mBlocka u nastavi programiranja, potrebno je pojednostavniti korištenje naredbi, odnosno prilagoditi biblioteku tako da se lakše dođe do potrebnih informacija.

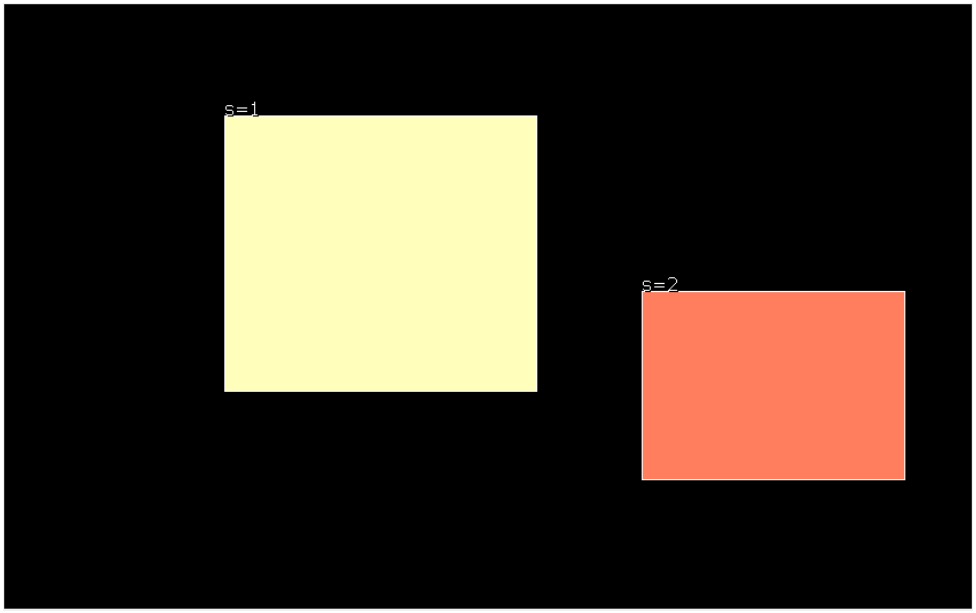
4.2.2 Naredba Serial Print data of object No. i

Sljedeći primjer pokazuje način korištenja naredbe **Serial Print data of object No. i**:

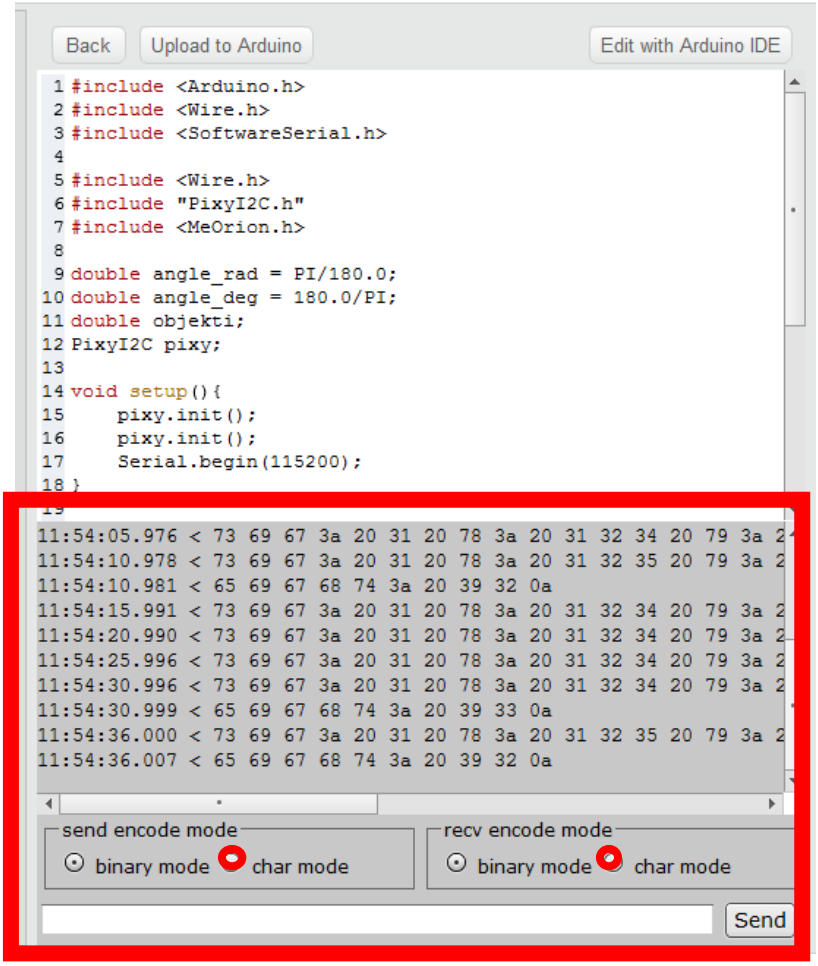


Najprije koristimo naredbu **Get objects** kako bi dobili listu detektiranih objekata s informacijama o svakom objektu. Zatim koristimo naredbu **Serial Print data of object No 0** koja će ispisati podatke koji se nalaze na nultom mjestu liste. Kako bismo mogli pročitati dobivene podatke, stavimo pauzu od jedne sekunde između svakog ispisa. Označena su dva objekta, žuti i crveni.

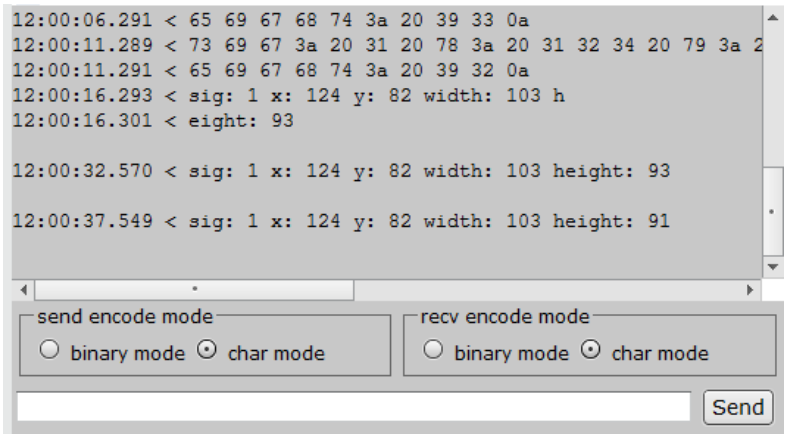
Koristeći PixyMon možemo vidjeti sljedeću sliku:



Vidimo da žuti objekt ima potpis 1, a crveni potpis 2. Nakon što prenesemo program na Orion pločicu (*upload*), moramo se opet spojiti na odgovarajući serijski port (*Serial Port*). Kako bismo mogli pročitati podatke koje Pixy šalje, moramo odabrati znakovni način rada (*char mode*) kod **send encode mode** i **recv encod mode** (slika ispod).

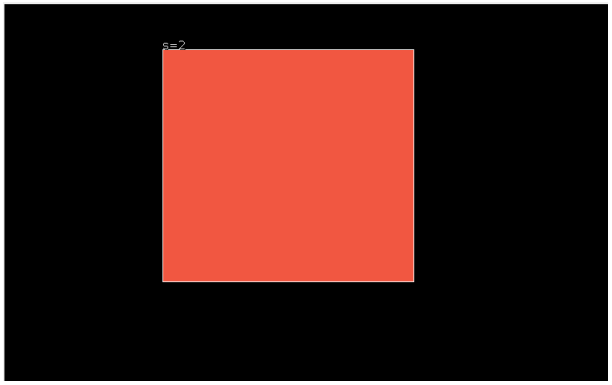


Time dobijemo prikaz na slici ispod. Sada možemo pročitati potpis detektiranog objekta, njegovu širinu i visinu.



Get objects sprema podatke o objektima u listu redoslijedom prema potpisima. U kadru smo imali dva objekta, žuti s potpisom 1 i crveni s potpisom 2. Pomoću prethodnog ispisa vidimo da se na nultom mjestu nalaze informacije o objektu s potpisom 1, odnosno o žutom objektu.

Ako maknemo žuti objekt iz kadra, ostat će samo crveni objekt s potpisom 2 pa će podaci o njemu biti na nultom mjestu liste (slike ispod).




```
12:25:23.848 < height: 142

12:25:28.829 < sig: 1 x: 264 y: 116 width: 111
12:25:28.832 < height: 147

12:25:33.846 < sig: 1 x: 264 y: 117 width: 109 height: 147

12:25:38.849 < sig: 1 x: 262 y: 118 width: 104 height: 150

12:25:43.859 < sig: 2 x: 155 y: 97 width: 121 height: 87
```

send encode mode: ○ binary mode ⊙ char mode

recv encode mode: ○ binary mode ⊙ char mode

Prethodni program možemo koristiti i ako smo koristili CC potpise.

U kadru se nalazi objekt označen CC-ovima. Koristeći naredbu **Serial Print data of object No. 0** dobivamo potpis tog objekta (s=1234), koordinate, njegovu širinu, visinu te kut (slike ispod).

Objekt s potpisom 1234

Ispis u mBlocku

```
15:30:12.410 < CC block! sig: 1234 (668 decimal
15:30:12.413 < ) x: 210 y: 100 width: 138 heigh
15:30:12.420 < t: 160 angle -85
```

Na ovim jednostavnim primjerima možemo vidjeti nekoliko nedostataka kod korištenja Pixyja u mBlocku.

Nedostatak 1. Treba prilagoditi biblioteku kako bismo pomoću naredbe **Get objects** lakše došli do informacije koliko je objekata Pixy detektirao. Također bi bilo dobro da možemo dohvaćati elemente liste, odnosno da možemo inicijalizirati cjelobrojnu varijablu (`integer`).

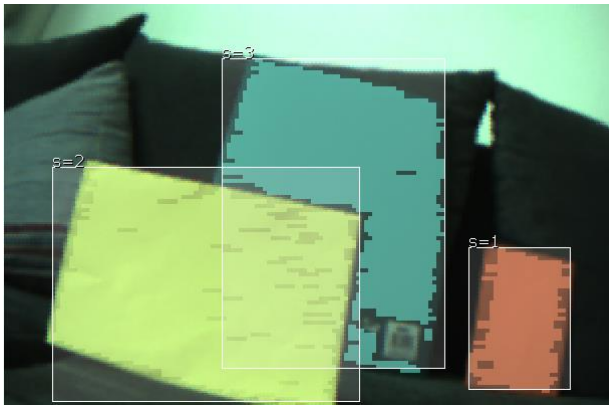
Nedostatak 2. Ako imamo više objekata s različitim potpisima u kadru, nakon što maknemo jedan od njih, na zadnjem mjestu liste ostaje stari podatak, odnosno imamo situaciju u kojoj se čini kao da postoji detektirani objekt iako takvog objekta nema u kadru.

Oba ova nedostatka bi se mogla barem djelomično izbjeći ako bi npr. vrijednosti 0 za **width** i **height** značile da se na tom mjestu u listi (niti nakon njega) više ne nalazi pronađeni objekt

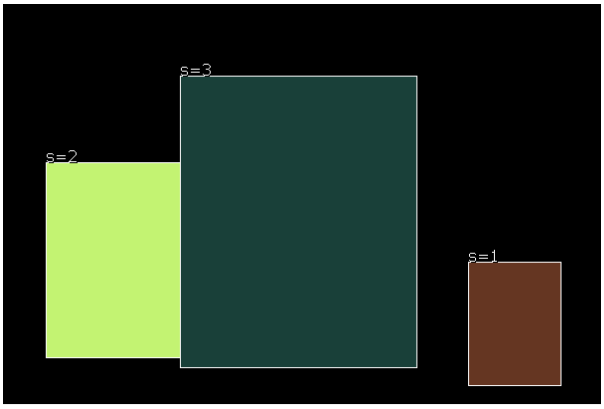
Primjer
U kadru se nalaze tri objekta različitih boja. Potpisi objekata su:

Crveni objekt – potpis 1
Žuti objekt – potpis 2
Zeleni objekt – potpis 3

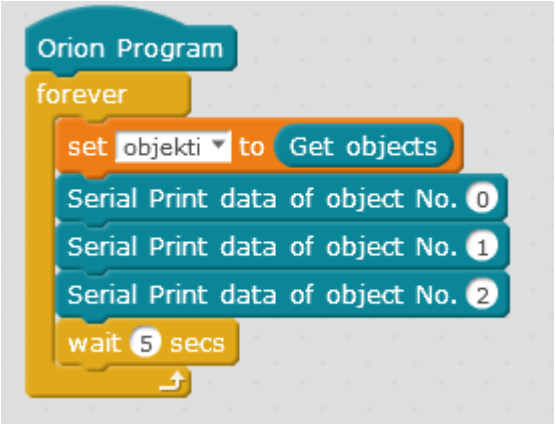
Cooked video



Default program



U mBlocku imamo sljedeći program:



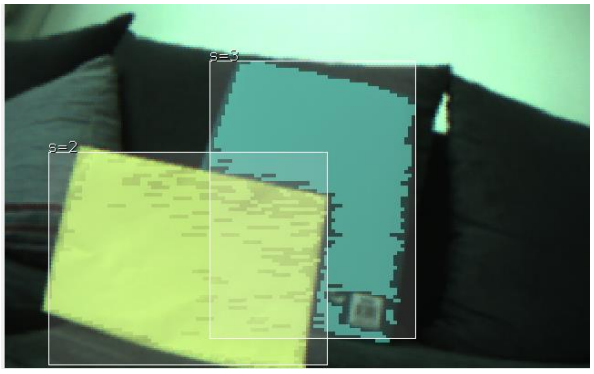
Koristimo naredbu **Serial print data** kako bismo vidjeli koje podatke Pixy šalje.

Dobiveni ispis:

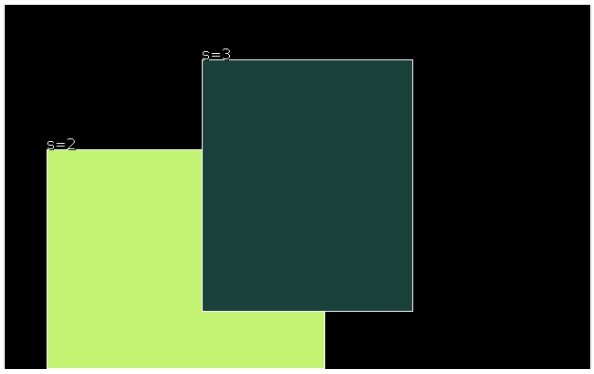
```
sig: 1 x: 254 y: 156 width: 51 height: 73
sig: 2 x: 98 y: 139 width: 151 height: 120
sig: 3 x: 166 y: 102 width: 113 height: 143
```

Maknemo objekt s potpisom 1 iz kadra. Sada u kadru imamo sljedeću situaciju:

Cooked video



Default program



Dobiveni ispis:

```
sig: 2 x: 99 y: 139 width: 152 height: 120
sig: 3 x: 166 y: 107 width: 113 height: 153
sig: 3 x: 166 y: 102 width: 113 height: 143
```

Usporedbom ispisa možemo primijetiti da je sada na prvom mjestu objekt s potpisom 2, a na drugom mjestu objekt s potpisom 3. Trećeg objekta nema, ali je ostao podatak iz prethodne situacije kada su na trećem mjestu u listi bili podaci o objektu s potpisom 3.

Prva situacija

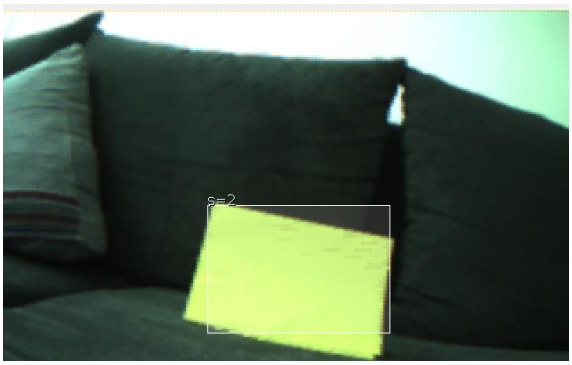
```
sig: 1 x: 254 y: 156 width: 51 height: 73
sig: 2 x: 98 y: 139 width: 151 height: 120
sig: 3 x: 166 y: 102 width: 113 height: 143
```

Druga situacija

```
sig: 2 x: 99 y: 139 width: 152 height: 120
sig: 3 x: 166 y: 107 width: 113 height: 153
sig: 3 x: 166 y: 102 width: 113 height: 143
```

Isto se dogodi kada iz kadra maknemo objekt s potpisom 3.

Cooked video



Default program

Dobiveni ispis je identičan prethodnom:

```
sig: 2 x: 99 y: 139 width: 152 height: 120
sig: 3 x: 166 y: 107 width: 113 height: 153
sig: 3 x: 166 y: 102 width: 113 height: 143
```

Kada bismo gledali samo ispis, došli bi do zaključka da se u kadru nalaze objekti s potpisima 2 i 3, iako je stvarna situacija drugačija. Kada bismo znali koliko je objekata Pixy detektirao, mogli bismo riješiti problem tako da ne gledamo ostatak liste.

Također bi bilo dobro da se u slučaju kada Pixy detektira manje objekata nego u prethodnoj situaciji, ispisuju samo informacije o stvarnom stanju.

Nedostatak 3. Bilo bi korisno u PixyMonu imati istovremeni prikaz slike i detektirane objekte (Cooked video) i kada mikrokontroler koristi Pixy tj. za vrijeme izvođenja programa. Na taj bi način učenici bolje mogli analizirati ponaša li se program u skladu s očekivanjem tj. ako program ne radi onako kako bismo željeli, je li problem u detektiranju objekata ili u dijelu programskog koda koji ovisi o detektiranim objektima.

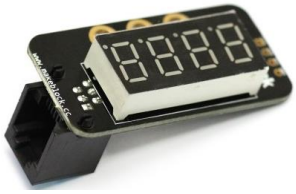
5 Programi u mBlocku

Za sljedeće aktivnosti ćemo koristiti Me Orion pločicu (slika ispod).

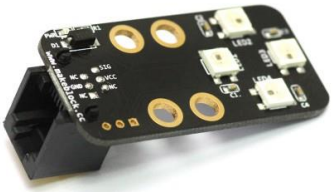
Me Orion pločica je mikrokontroler baziran na Arduino Uno s nekoliko poboljšanja u svrhu korištenja u obrazovanju. Ima osam priključaka za jednostavno spajanje mikrokontrolera s ulaznim i izlaznim jedinicama te je podržana odgovarajućim naredbama u mBlocku.

Jedna od izlaznih jedinica je ranije spomenuti 7-segmentni display na kojemu možemo prikazati četveroznamenkaste brojeve i nekoliko posebnih znakova. Osim 7-segmentnog displaya također ćemo koristiti RGB LED lampice. Sa svakom od 4 RGB lampice može se upravljati zasebno, kontrolirati jačinu boje te dobiti različite boje miješajući crvenu, plavu i zelenu boju [14].

7-segmentni display



RGB LED lampice



Kao ulaznu jedinicu ćemo koristiti Pixy.

Pixy povežemo s Orion pločicom.

5.1 Aktivnost 1: RGB paleta boja

Svaki piksel u boji sastoji se od tri bajta informacija koju čine osnovne boje: crvena, zelena i plava. To je tzv. RGB paleta boja (eng. *Red Green Blue*) [15].

Prema kurikulumu za informatiku za osnovnu školu, ova tema je prikladna za 7. razred osnovne škole. U nastavku se nalaze neki od ishoda iz kurikuluma koji se mogu realizirati ovom aktivnošću [1].

A. 7. 4 "učenik opisuje, uspoređuje i koristi se različitim formatima zapisivanja grafičkih, zvučnih podataka i videopodataka na računalu."

"Istražiti osnovna obilježja nekih grafičkih/zvučnih/video zapisa (veličinu, broj boja, razlučivost…)."

B. 6. 2 "učenik razmatra i rješava složeniji problem rastavljajući ga na niz potproblema."

"U pokaznim (odabranim) primjerima programskoga koda uočiti/prepoznati/istaknuti dijelove koda koji predstavljaju rješenje nekoga poznatog (manjeg) problema (zadatka), mijenjati/prilagoditi dijelove koda kako bi se uklopili u rješenje nekoga većeg problema."

B. 7. 1 "učenik razvija algoritme za rješavanje različitih problema koristeći se nekim programskim jezikom pri čemu se koristi prikladnim strukturama i tipovima podataka."

"Analizirati neki problem te prepoznati ulazne vrijednosti potrebne za rješavanje toga problema te moguće izlazne vrijednosti programa."

"Analizirati i predvidjeti moguće izmjene algoritma koje bi mogle poslužiti za rješavanje sličnih problema."

Cilj aktivnosti: Učenici će napisati program koji koristi RGB paletu boja

Nastavni oblik: diferencirana nastava u obliku grupnog rada

Nastavne metode: eksperimentalna metoda, problemska nastava

Nastavni materijal: Me Orion pločica, Pixy, RGB LED lampice, računalo, program mBlock, papiri crvene, zelene i plave boje

Tijek aktivnosti:

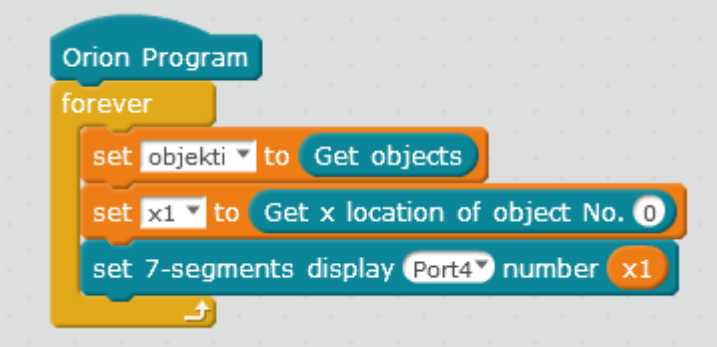
Učenici se podijele u grupe po troje.

Korak 1.

Koristeći PixyMon označimo crveni papirić potpisom 1.

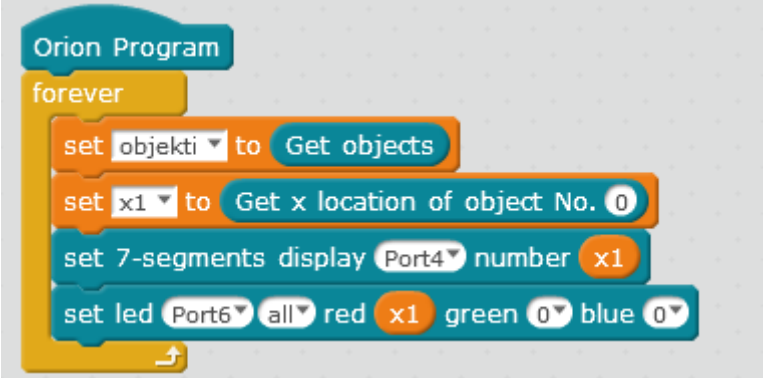
Na Orion pločicu spojimo Pixy, 7-segmentni display i LED lampice. Dohvatimo podatke o objektima koje Pixy detektira. Zatim dohvatimo *x*-koordinatu prvog objekta u listi, spremimo je u varijablu **x1** i prikažemo na 7-segmentnom displayu.

Pomicanjem objekta lijevo-desno učenici zaključuju koje su minimalne i maksimalne vrijednosti *x*-koordinate (min=1, max=319).



Korak 2.

Sada želimo uključiti LED lampice crvenom bojom tako da jačina svjetla ovisi o *x*-koordinati prvog objekta.



Problem: maksimalna koordinata je 319, a maksimalna vrijednost za svaku boju je 255.

Možemo riješiti problem na nekoliko načina.

Prvo rješenje je da promatramo što će se dogoditi kad *x*-koordinata prijeđe 255. Učenici će lako zaključiti da LED lampice tada svijetle maksimalnom jačinom te da se jačina nakon 255 ne mijenja. Iz tog razloga možemo jednostavno ignorirati problem.

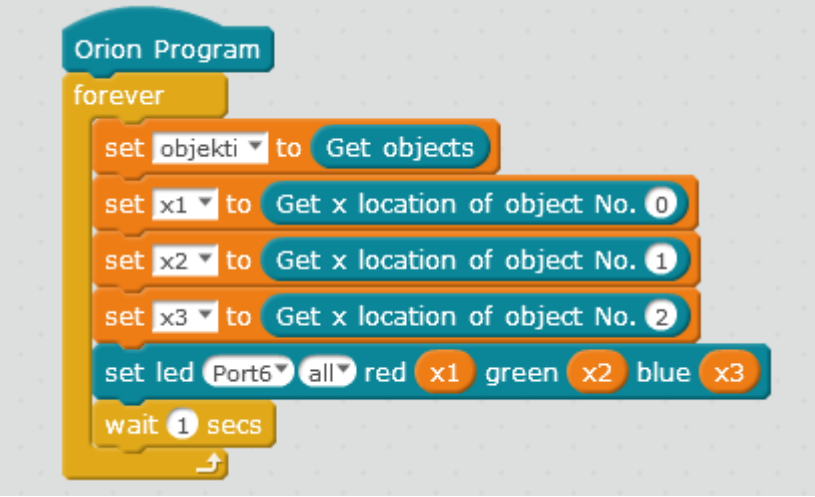
Drugo rješenje je da koristeći naredbu `if` promijenimo jačinu ako je koordinata veća od 255 (npr. da se nakon 255 jačina postepeno smanjuje ili da nakon 255 opet kreće od 0).

Treće rješenje je da skaliranjem svedemo maksimalnu koordinatu na 255 tako trenutačnu koordinatu pomnožimo faktorom $\frac{255}{318}$.

U nastavku će se koristiti prvo rješenje.

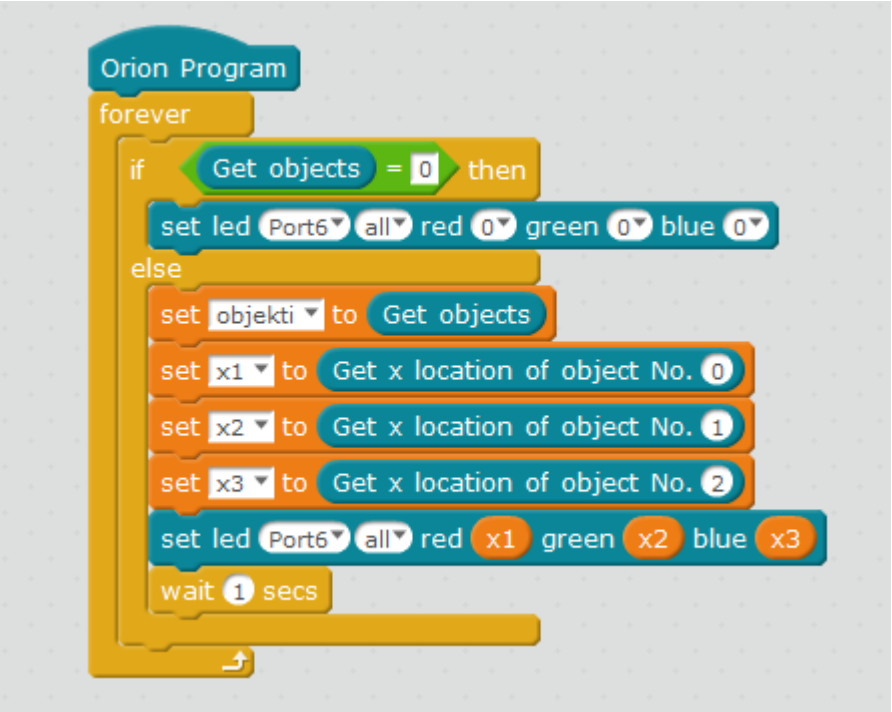
Korak 3.

Dodajmo još dva objekta, zeleni s potpisom 2 i plavi s potpisom 3. Spremimo njihove *x*-koordinate u varijable `x2` i `x3` te uključimo zelenu boju jačinom `x2` i plavu jačinom `x3`.



Svaki učenik pomiče po jedan objekt lijevo, odnosno desno te zajedno pokušavaju dobiti zadane boje (ljubičasta, žuta, narančasta, tirkizna itd.).

Program možemo još malo poboljšati tako da isključimo lampice u slučaju kada Pixy nije detektirao niti jedan objekt.



Problem: Niti jedan objekt ne smije izaći iz kadra da bi ovaj program radio. Ako objekt izađe iz kadra, zapamćena je njegova posljednja x-koordinata.

U ovom primjeru dolazi do izražaja potreba da znamo koliko je objekata Pixy detektirao. Kada bismo znali tu informaciju, mogli bismo ići po listi detektiranih objekata, pogledati koji je potpis od svakog od njih i spremiti koordinate. Tada bi se smjela dogoditi situacija da neki od objekata nije u kadru.

Rješenje ovog problema je isključivo u poboljšanju biblioteke, inače je veliki problem napraviti ovu aktivnost s učenicima jer u nastavi ne možemo objasniti učenicima da biblioteka ne radi ispravno.

5.2 Aktivnost 2. Semafor

Ova aktivnost se nadovezuje na prethodnu. Učenici su u prethodnoj aktivnosti otkrili kako dobiti narančastu boju korištenjem RGB palete boja.

Cilj aktivnosti: Učenici će napisati program koji uključuje crveno svjetlo ako je detektiran crveni objekt, zeleno svjetlo ako je detektiran zeleni objekt i narančasto svjetlo ako su detektirana oba objekta.

Nastavni oblik: diferencirana nastava u obliku individualnog rada

Nastavne metode: problemska nastava

Nastavni materijal: Me Orion pločica, Pixy, RGB LED lampice, računalo, program mBlock, papiri crvene i zelene boje

Tijek aktivnosti:

Objekt s potpisom 1 je zelene boje, a s potpisom 2 je crvene boje.

Ako je Pixy detektirao samo objekt crvene boje, LED lampice svijetle crvenom bojom.

Ako je detektirao samo objekt zelene boje, lampice svijetle zelenom bojom.

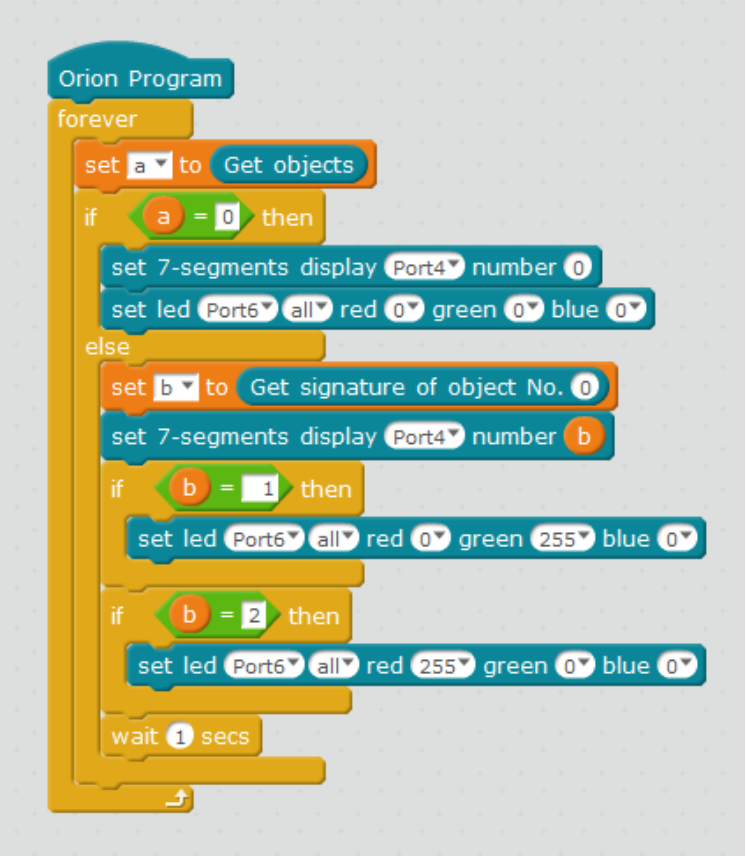
Ako je detektirao i jedan i drugi objekt, lampice svijetle narančastom bojom

Ako Pixy nije detektirao niti jedan objekt, lampice se isključuju.

Korak 1.

U ovom rješenju će LED lampice svijetliti zelenom bojom ako je Pixy detektirao i jedan i drugi objekt. Kako je potpis zelenog objekta 1, on će biti na nultom mjestu u listi *a*.

Lampice će svijetliti crveno ako je Pixy detektirao samo objekt crvene boje. Njegov potpis je 2, pa će biti na nultom mjestu u listi samo ako nema objekta zelene boje s potpisom 1.

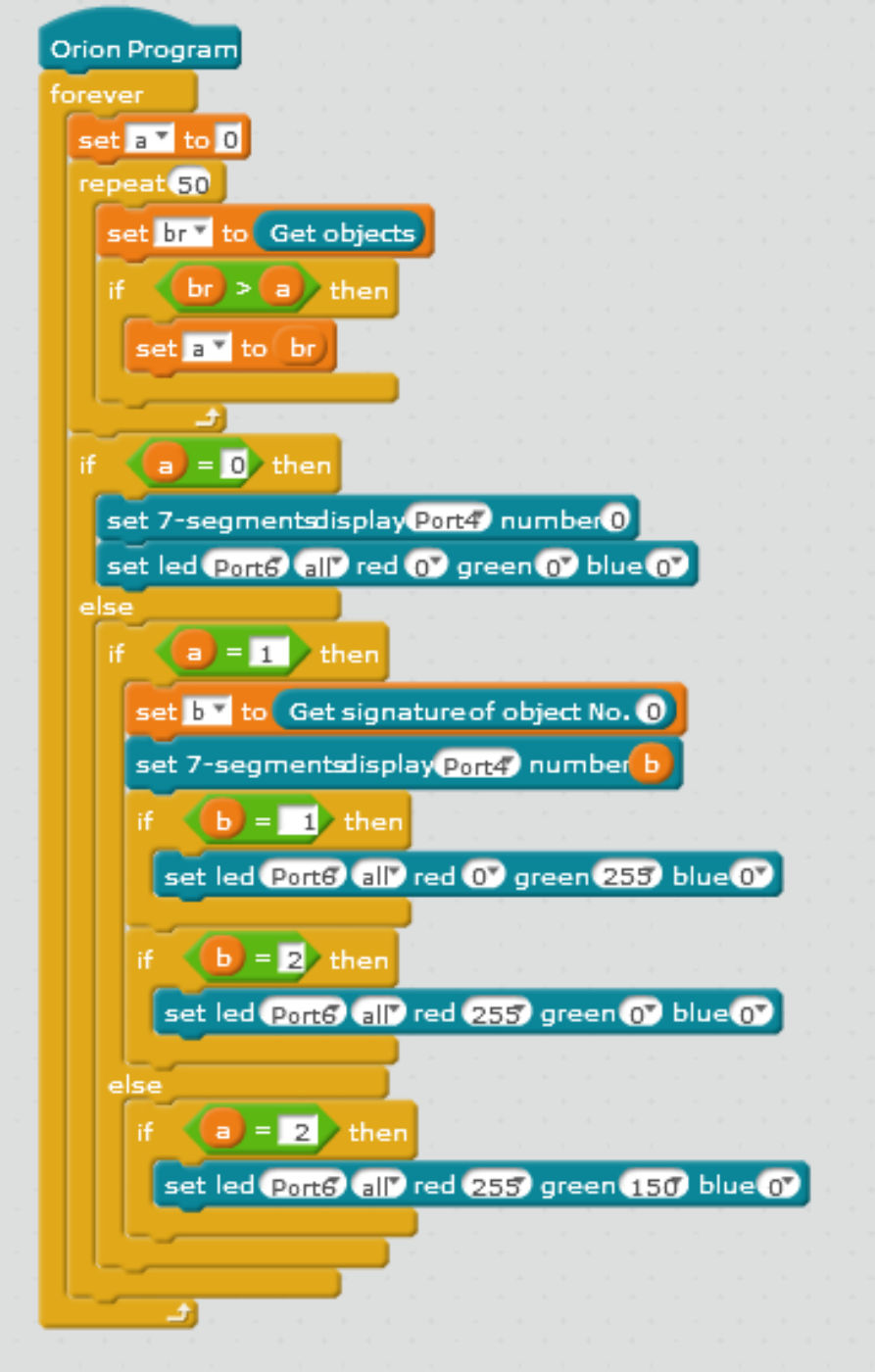


Najprije provjerimo je li Pixy detektirao ijedan objekt. Ako nema niti jednog objekta u kadru, isključimo lampice. U suprotnom, pogledamo koji je potpis objekta na nultom mjestu u listi. Ako je potpis objekta 1, tada uključujemo lampice zelenom bojom, a ako je potpis objekta 2, uključujemo lampice crvenom bojom.

Problem: Problem nam stvaraju stari podaci koji su ostali u listi nakon što se objekt maknuo iz kadra. Prema ispisu ne možemo zaključiti imamo li jedan ili dva objekta u kadru.

Korak 2.

Kako bismo riješili problem, trebamo najprije utvrditi koliko objekata imamo u kadru. Poslužit ćemo se s rješenjem opisanim u poglavlju 4.2.1.



Ako je detektiran samo jedan objekt, provjerimo koji je njegov potpis i u ovisnosti o potpisu uključimo lampice zelenom ili crvenom bojom. Ako su detektirana dva objekta, uključimo lampice narančastom bojom.

6 Arduino IDE

Arduino softver (IDE) je programsko okruženje otvorenog koda (*open source*) koje olakšava pisanje i prenošenje koda na mikrokontroler. Okruženje je napisano u programskom jeziku Java, a temelji se na programu Processing i ostalim programima otvorenog koda. Može se koristiti za bilo koju Arduino pločicu.

Svi programi koji se pišu u programu mBlock automatski se prevode u tekstualni programski jezik C koji se može otvoriti u Arduino IDE. To omogućuje učenicima lakši prijelaz s blokovskog programiranja u tekstualne programske jezike. Također omogućuje onim učenicima kojima naredbe u mBlocku nisu dovoljne za ono što žele napraviti, da se prebace u Arduino IDE gdje imaju više mogućnosti i gdje mogu napisati složenije programe [16].

6.1 Naredbe za rad s Pixyjem

Najvažnija naredba za rad s Pixyijem je **`getBlocks()`** koja vraća broj objekata koje je Pixy detektirao. Zatim možemo doći do informacija o detektiranim objektima u listi **`pixy.blocks[]`**. Svaki element liste sadrži sljedeće informacije:

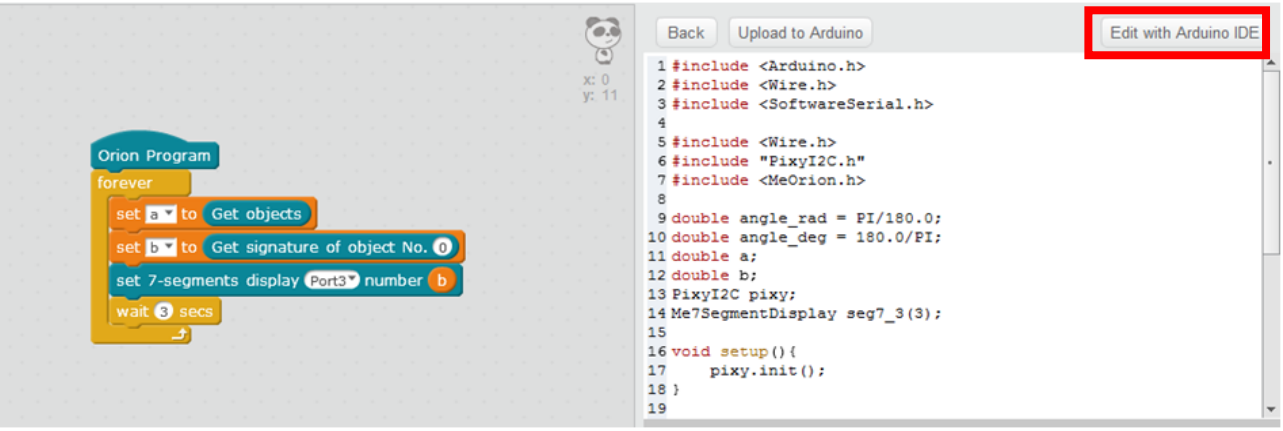
- **`pixy.blocks[i].signature`** vraća potpis objekta na i-tom mjestu
  `pixy.blocks[i].x` vraća x-koordinatu i-tog objekta (od 0 do 319)
- **`pixy.blocks[i].y`** vraća y-koordinatu i-tog objekta (od 0 do 199)
- **`pixy.blocks[i].width`** vraća širinu i-tog objekta (od 1 do 320)
- **`pixy.blocks[i].height`** vraća visinu i-tog objekta (od 1 do 200)
- **`pixy.blocks[i].angle`** vraća kut i-tog objekta ako je objekt označen kodiranjem boja (CC)
- **`pixy.blocks[i].print()`** ispisuje informacije o *i*-tom detektiranom objektu.

Naredba **`getBlocks()`** prihvaća broj tipa **`integer`** koji govori koji je maksimalni broj blokova koje želimo da naredba **`getBlocks()`** vrati.

Također, u listi `pixy.blocks[]` se nalaze informacije o objektima redom po potpisima, od najmanjeg prema najvećemu. Ako imamo više objekata s istim potpisom, poredani su po veličini, od onog s najvećom površinom do onoga s najmanjom površinom [17].

6.2 Funkcije Setup i Loop

U mBlocku je napisan jednostavan program koji na 7-segmentnom displayu prikazuje potpis objekta koji se nalazi na nultom mjestu u listi `a`. U mBlocku možemo odmah vidjeti kod generiran za Arduino IDE, a klikom na *Edit with Arduino IDE* ga možemo i otvoriti u programu Arduino IDE.



Pogledajmo kako izgleda kod generiran u Arduino IDE za program napisan u mBlocku iz prethodnog primjera.

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <Wire.h>
#include "PixyI2C.h"
#include <MeOrion.h>

double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
double a;
double b;
PixyI2C pixy;
Me7SegmentDisplay seg7_3(3);

void setup(){
    pixy.init();
}

void loop(){
    a = pixy.getBlocks();
    b = pixy.blocks[0].signature;
    seg7_3.display((float)b);
    _delay(3);
    _loop();
```

```
}

void _delay(float seconds){
    long endTime = millis() + seconds * 1000;
    while(millis() < endTime)_loop();
}

void _loop(){
}
```

Najprije se uključuju sve potrebne biblioteke te deklariraju sve varijable i po potrebi postavljaju na početne vrijednosti. Svaki program napisan u Arduino IDE se sastoji od dvije glavne funkcije – `setup()` i `loop()`.

Funkcija `setup()` se poziva na početku programa. Koristi se za inicijalizaciju varijabli, za provedbu radnji potrebnih za kasnije korištenje funkcija iz određenih biblioteka i slično. Funkcija `setup()` se poziva samo jednom, nakon svakog uključivanja ili resetiranja Arduino pločice [18].

U prethodnom primjeru se unutar funkcije `setup` inicijalizirala varijabla `pixy`:

```
void setup(){
    pixy.init();
}
```

Nakon funkcije `setup()`, koja incijalizira i postavlja početne vrijednosti, funkcija `loop()` uzastopno pokreće petlju koja omogućava programu da odgovori na promjenu. U toj funkciji se nalaze naredbe koje će se neprekidno izvršavati. Koristi se za aktivno upravljanje Arduino pločicom. U prethodnom primjeru funkcija `loop()` izgleda ovako:

```
void loop(){
    a = pixy.getBlocks();
    b = pixy.blocks[0].signature;
    seg7_3.display((float)b);
    _delay(3);
    _loop();
}
```

Blokovska naredba **Get objects** se u Arduinu IDE prevodi u naredbu `getBlocks()` koja dohvaća podatke koje šalje Pixy te u varijablu `a` sprema podatak koliko je objekata Pixy detektirao. U varijabli `b` je spremljen potpis objekta koji se nalazi na nultom mjestu u listi `pixy.blocks`. Kako se te naredbe nalaze unutar petlje `loop()`, one će se neprekidno iznova izvršavati.

Blokovska naredba **forever** u mBlocku služi da se naredbe unutar tog bloka ponavljaju zauvijek. Pogledajmo kako se ona prevela u Arduinu IDE. Nakon prethodnog opisa funkcije `loop()` u Arduinu IDE, već je jasno da će se upravo pomoću te funkcije prevesti blokovska naredba **forever**. Osim funkcije `loop()`, za prevođenje naredbe **forever** trebamo i funkciju `setup()` kako bismo počeli koristiti biblioteku „PixyI2C.h".

6.3 Pixy u Arduinu IDE

U Arduino IDE postoje primjeri programa za korištenje različitih senzora, među njima i senzora Pixy CMUcam5. Primjeri za Pixy nisu uključeni u Arduinu IDE, već moramo najprije instalirati biblioteku za Pixy s kojom dobijemo i primjere.

Biblioteku možemo naći na adresi http://www.cmucam.org/projects/cmucam5/wiki/Latest_release. Najprije preuzmemo biblioteku u *.zip* formatu, zatim na izborniku odaberemo *Sketch→ Include Library →* *Add .ZIP Library* te odaberemo datoteku koju smo preuzeli. Nakon toga će se u primjerima nalaziti i oni programi koji koriste Pixy.

Do primjera se dolazi pomoću izbornika gdje odaberemo *File → Examples → pixy*. Ponuđeni su sljedeći primjeri:

code_mbot_pixy
hello_world
i2c
led_cycle
pantilt
servo_mode
uart

Primjeri **hello_world** i **i2c** su isti, osim što se u i2c koristi I2C komunikacija, a u hello_world SPI komunikacija. Jedina razlika u kodu je pri pozivu biblioteka što je prikazano ispod.

hello_world

```
#include <Pixy.h>

Pixy pixy;
```

i2c

```
#include <PixyI2C.h>

PixyI2C pixy;
```

Kako smo spojili Pixy s Orion pločicom na način da koristimo I2C komunikaciju, koristit ćemo taj primjer.

6.3.1 i2c (Hello world)

Primjeri „hello_world“ i „i2c“ služe za upoznavanje s naredbama iz biblioteke za rad s Pixyjem. Ovi programi ispisuju detektirane objekte kroz serijski monitor (*Serial Monitor).*

U nastavku se nalazi kod programa „i2c“.

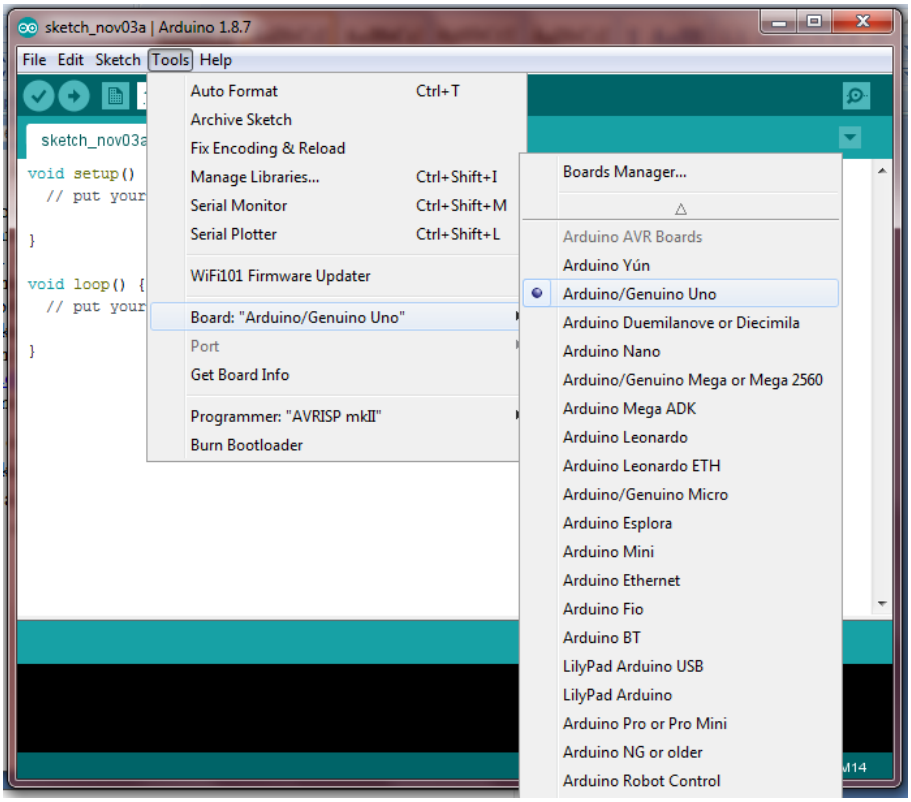
```
#include <Wire.h>
#include <PixyI2C.h>


PixyI2C pixy;

void setup()
{
  Serial.begin(9600);
  Serial.print("Starting...\n");
```

```
  pixy.init();
}


void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];

  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(buf);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, "  block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

Najprije pozivamo biblioteke potrebne za komunikaciju između Orion pločice i Pixy kamere. Stvaramo objekt tipa `PixyI2C` koji zatim inicijaliziramo u funkciji `setup()`. U istoj funkciji je ostvarena i serijska komunikacija s računalom pomoću naredbe `Serial.begin(9600)`. Naredba otvara serijski port i postavlja brzinu prijenosa podataka na 9600 bps (bita po sekundi) [18]. Zatim na serijski monitor ispisujemo „Starting...“ kako bismo znali da je ispis započeo.

Unutar funkcije `loop()` nalazi se naredba `getBlocks()` koja vraća varijablu tipa `integer`. Ona nam govori koliko se objekata nalazi u kadru Pixy kamere. Ako u kadru nema prepoznatih objekata, naredba će vratiti nulu. Za svaki od detektiranih objekata mogu se dobiti informacije o koordinatama, dimenzijama ili potpisu pojedinog objekta.

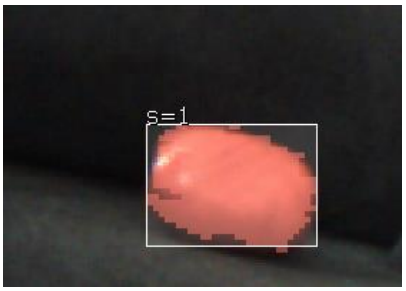
Pomoću petlje `if (blocks)` provjeravamo je li Pixy detektirao ijedan objekt. U slučaju da nema detektiranih objekata, program neće uči u petlju. Ako je detektiran barem jedan objekt, povećavamo varijablu `i` za jedan. Kako Pixy šalje podatke čak 50 puta u sekundi, ispisat ćemo samo svaki 50. podatak, odnosno jednom u sekundi. Da smo ispisali sve podatke koje Pixy šalje, ne bismo ih stigli pročitati. To se ostvarilo pomoću uvjeta `if (i%50==0)`. Najprije ispisujemo koliko je objekata Pixy detektirao, a zatim koristimo petlju `for()` kako bismo ispisali svaki od detektiranih objekata [19].
Kako bismo mogli prenijeti program na Orion pločicu, najprije moramo odabrati koju ćemo Arduino pločicu koristiti. Na izborniku odaberemo *Tools* → *Board* → *Arduino/Genuino Uno* kao što je prikazano na slici ispod.



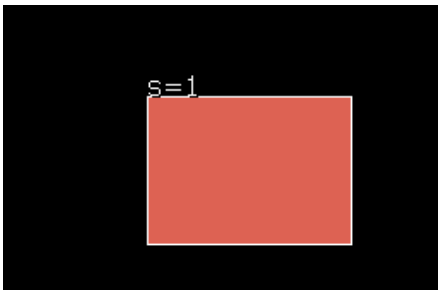
Zatim moramo označiti port koji koristimo tako da na izborniku odaberemo *Tools→Port → COM14*. Port može biti i neki drugi, ovisno o tome u koji port smo priključili Orion pločicu. Kliknemo na gumb za prenošenje programa (*Upload*), a zatim kliknemo na gumb za otvaranje serijskog monitora (*Serial Monitor*). U kadru se nalazi objekt crvene boje s potpisom 1.

Napomena: Ako želimo istovremeno vidjeti što se nalazi u kadru i koristiti Pixy kameru, moramo koristiti način prikaza Default program u PixyMonu.

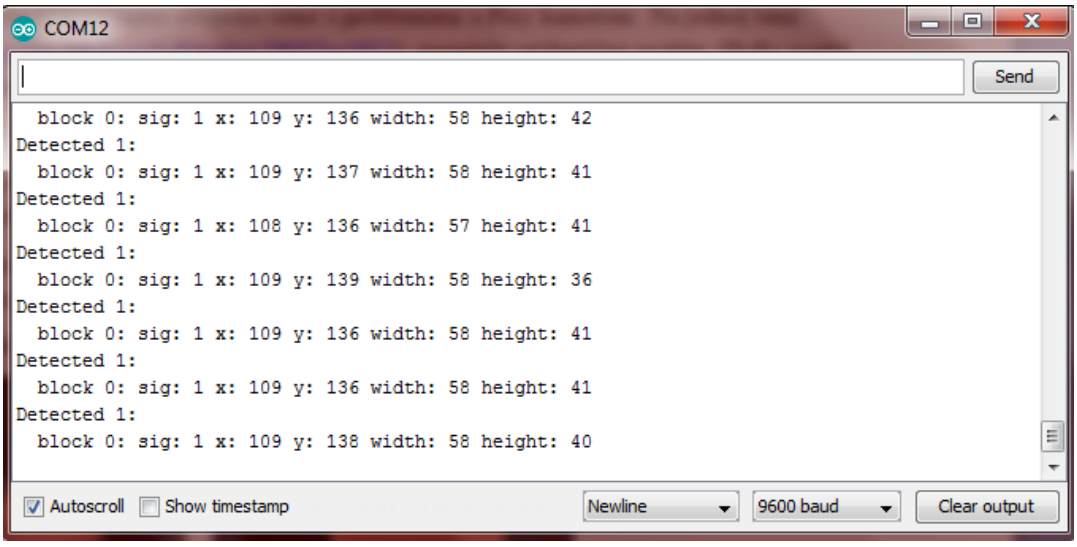
Cooked video



Default program



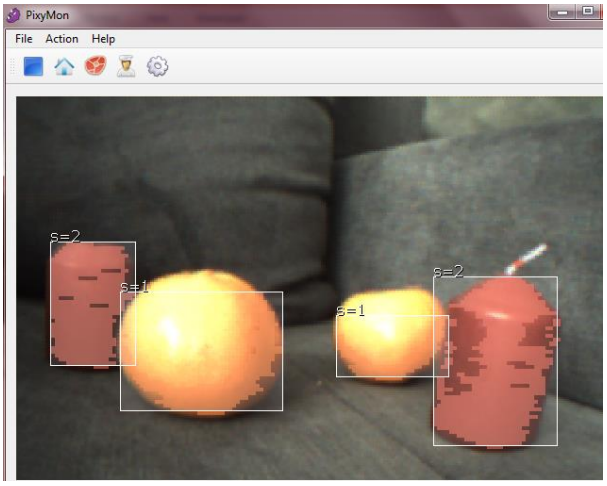
Ispis je sljedeći:



Vidimo da je Pixy detektirao jedan objekt čiji je potpis 1. Također možemo pročitati koordinate te širinu i visinu tog objekta.

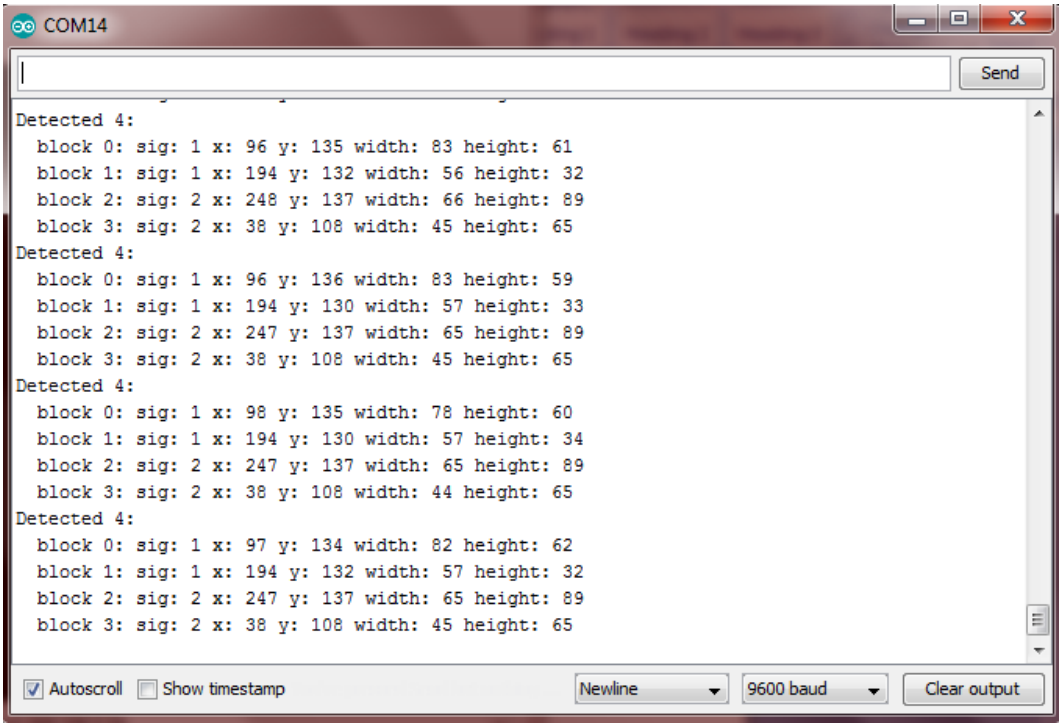
Primjer s više objekata u kadru:

Cooked video



Default program

Ispis:



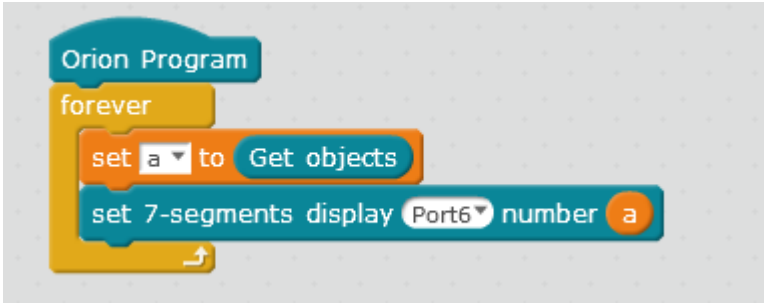
Sada je Pixy detektirao četiri objekta. Dva objekta imaju potpis 1, a dva potpis 2. Možemo pročitati koordinate, širinu i visinu za svakoga od njih.

6.3.2 Modifikacija programa Hello world

Pomoću primjera Hello world smo naučili kako spremiti podatke koje Pixy šalje mikrokontroleru te kako koristiti serijski monitor u Arduino IDE. Možemo modificirati program Hello world tako da iskoristimo i 7-segmentni display na kojemu ćemo ispisati koliko objekata je Pixy detektirao.

Program „i2c“ (odnosno „hello_world“) je jednostavno modificirati. Ako su se učenici do sada upoznali s programiranjem u mBlocku, za početak mogu složiti program u mBlocku koji koristi 7-segmentni display, a zatim pogledati kako se taj dio koda preveo u Arduino IDE. Mogućnost da se u mBlocku vidi i programski kod za Arduino IDE uvelike olakšava prijelaz učenicima s blokovskog na tekstualni programski jezik.

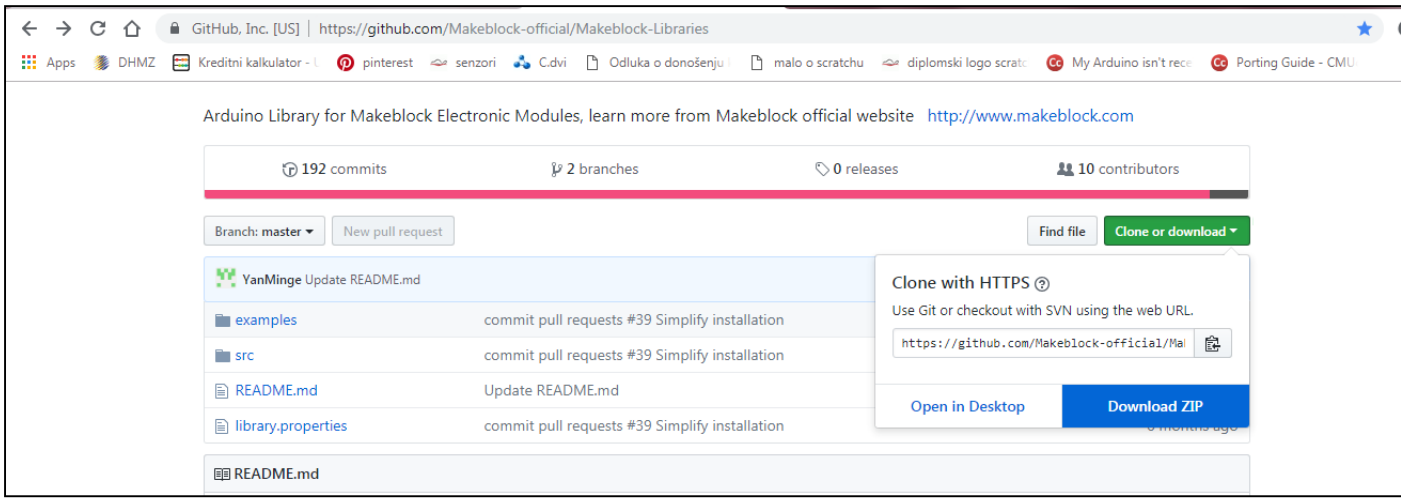
Napišimo program u mBlocku koji koristi 7-segmentni display kojega smo spojili na port 6.



Pogledajmo kako su se prevele naredbe vezane za 7-segmentni display u Ardunino IDE.

```
#include <Wire.h>
#include "PixyI2C.h"
#include <MeOrion.h>

Me7SegmentDisplay seg7_6(6);

void loop(){
    objekti = pixy.getBlocks();
    seg7_6.display((float)objekti);
    _delay(2);
    _loop();
```

Biblioteku „MeOrion.h“ najprije moramo instalirati. Možemo preuzeti biblioteku na stranici https://github.com/Makeblock-official/Makeblock-Libraries. Odaberemo *Clone or download* i zatim kliknemo na *Download ZIP*.



Kako bismo mogli koristiti preuzetu biblioteku, još moramo na izborniku u Arduino IDE odabrati *Sketch* → *Include Library* → *Add .ZIP Library...* nakon čega odaberemo preuzetu biblioteku.
Koristeći naredbu `Me7SegmentDisplay seg7_6(6)` smo definirali varijablu `seg7_6` tipa `Me7SegmentDisplay`, a u zagradi navedemo broj porta u kojega smo priključili 7-segmentni display. Zatim smo unutar funkcije `loop()` ispisali broj detektiranih objekata na 7-segmentnom displayu.

Iskoristimo prethodni dio koda kako bismo modificirali program „Hello world“.
U nastavku se nalazi modificirani kod programa „Hello world“. Dodane linije koda su obojane crvenom bojom.

```
#include <MeOrion.h>
#include <Wire.h>
#include <PixyI2C.h>

PixyI2C pixy;
Me7SegmentDisplay seg7_6(6);

void setup()
{
  Serial.begin(9600);
  Serial.print("Starting...\n");

  pixy.init();
}

void loop()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];

  blocks = pixy.getBlocks();

  if (blocks)
  {
    i++;

    if (i%50==0)
    {
      seg7_6.display((float)blocks);
      sprintf(buf, "Detected %d:\n", blocks);
      Serial.print(buf);
      for (j=0; j<blocks; j++)
      {
        sprintf(buf, "  block %d: ", j);
        Serial.print(buf);
        pixy.blocks[j].print();
      }
    }
  }
}
```

Broj prepoznatih objekata na 7-segmentnom displayu ispisuje se samo ako je istinit uvjet `if (i%50==0)` kako bi se ispis ažurirao jednom u sekundi. Inače se na displayu mijenjaju brojke toliko brzo da se ne može pročitati broj.

6.3.3 mBot prati objekt

mBot je obrazovni robot tvrtke Makeblock čiji je „mozak“ mCore mikrokontroler baziran na Arduino Uno, posebno dizajniran za mBota [20].

mBot je namijenjen za početnike u programiranju fizičkih objekata. Učenici u našim školama uče programirati pomoću mBota. Croatian Makers liga je dio projekta Croatian Makers čiji je cilj omogućiti široko uključivanje robotike, automatike i programiranja u edukaciju u osnovnoškolskom uzrastu. Svi sudionici Croatian Makers lige dobivaju edukacijske robote (mBot) na posudbu [21].

mBot se može programirati u mBlocku ili u Arduino IDE.

Na web-stranici navedenoj u literaturi pod brojem [13] nalaze se detaljne upute (tutorial) kako omogućiti mBotu da prati neki objekt koristeći Pixy.

Najprije spojimo Dupont kabel s Pixyjem na način kao što je pokazano na slici u poglavlju Pixy u mBlocku. Opet ćemo koristiti I2C način komunikacije pa moramo u PixyMon programu namjestiti I2C komunikaciju (ako već nije). Pixy možemo spojiti na bilo koji od portova na mBotu.

Nakon što preuzmemo mapu s programom s navedene stranice, otvorimo datoteku „code_mbot_pixy.ino“ u Arduino IDE.

Kod programa:

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <PixyI2C.h>

#include "mBot.h"
#include "MePort.h"
MeBoard myBoard(mBot);
#include "MeDCMotor.h"

PixyI2C pixy;

int signature = 0;
```

```
int x = 0;
int y = 0;
unsigned int width = 0;
unsigned int height = 0;
unsigned int area = 0;
unsigned int newarea = 0;
int Xmin = 70;
int Xmax = 200;
int maxArea = 0;
int minArea = 0;
static int i = 0;

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);

void setup()
{
  Stop();
  pixy.init();
}

void loop()
{
  while(millis()<5000)
  {
    scan();
    area = width * height;
    maxArea = area + 1000;
    minArea = area - 1000;
  }

    scan();

  if(signature == 1)
  {
    newarea = width * height;

      if (x < Xmin)
      {
       left();
      }
      else if (x > Xmax)
      {
       right();
      }
      else if(newarea < minArea)
      {
       forward();
      }
      else if(newarea > maxArea)
      {
       backward();
      }
```

```
      else
      {
        Stop();
      }
   }
   else
   {
    Stop();
     }
}

void backward()
  {
  motor_9.run((9)==M1?-(-255):(-255));
  motor_10.run((10)==M1?-(-255):(-255));
  }

void forward()
  {
  motor_9.run((9)==M1?-(255):(255));
  motor_10.run((10)==M1?-(255):(255));
  }

void right()
  {
  motor_9.run((9)==M1?-(255):(255));
  motor_10.run((10)==M1?-(0):(0));
  }

void left()
  {
  motor_9.run((9)==M1?-(0):(0));
  motor_10.run((10)==M1?-(255):(255));
  }

void Stop()
  {
  motor_9.run((9)==M1?-(0):(0));
  motor_10.run((10)==M1?-(0):(0));
  }

void scan()
  {
  uint16_t blocks;
  blocks = pixy.getBlocks();
  signature = pixy.blocks[i].signature;
  x = pixy.blocks[i].x;
  y = pixy.blocks[i].y;
  width = pixy.blocks[i].width;
  height = pixy.blocks[i].height;
  }
```

Unutar funkcije `loop()` najprije pozivamo funkciju `scan()`. Funkcija `scan()` dohvaća podatke koje šalje Pixy, u varijablu `x` sprema x-koordinatu, a u varijablu `y` y-koordinatu `i`-tog objekta. Također u varijablu `width` sprema širinu, a u varijablu `height` visinu `i`-tog detektiranog objekta. Na početku programa smo inicijalizirali varijablu `i` na 0:

```
static int i = 0;
```

Stoga će se unutar funkcije `scan()` u prethodno navedene varijable spremati podaci o objektu koji se nalazi na nultom mjestu liste.

Nakon prvog poziva funkcije `scan()`, računamo površinu detektiranog objekta. Zatim računamo minimalnu i maksimalnu površinu tako da od početne površine oduzmemo, odnosno dodamo broj 100.

```
maxArea = area + 1000;
minArea = area - 1000;
```

Na početku pograma smo inicijalizirali varijable:

```
int Xmin = 70;
int Xmax = 200;
```

Varijable `xmin` i `xmax` su minimalne i maksimalne x-koordinate objekta kojega pratimo. U prijevodu, ako je x-koordinata praćenog objekta manja od 70, robot mora skrenuti lijevo kako bi praćeni objekt ostao u kadru, odnosno kako bi njegova x-koordinata postala veća od 70. Ako je x-koordinata praćenog objekta veća od 200, robot mora skrenuti desno kako bi x-koordinata praćenog objekta postala manja od 200.

To se ostvarilo unutar funkcije `loop()` ovim dijelom koda:

```
if (x < Xmin)
      {
       left();
      }
      else if (x > Xmax)
      {
       right();
      }
```

Nadalje, moramo provjeriti je li površina detektiranog objekta unutar dozvoljenih granica, odnosno je li veća od minimalne i manja od maksimalne površine.

```
     else if(newarea < minArea)
           {
            forward();
           }
           else if(newarea > maxArea)
           {
            backward();
           }
```

Ako je površina objekta manja od minimalne, robot se mora približiti objektu. Ako je površina objekta veća od maksimalne, robot ide unazad kako bi se udaljio od objekta.

Ako niti jedan od prethodno navedenih uvjeta nije zadovoljen, robot staje. Također, ako potpis detektiranog objekta nije 0, robot staje.

Prijedlog za poboljšanje:

U slučaju kada nema objekta u kadru, mBot se nastavlja kretati u smjeru u kojem je išao dok je objekt bio u kadru. Bilo bi dobro da se u tom slučaju mBot okreće oko sebe kako bi pokušao pronaći objekt.

Također, bilo bi dobro da mBot koristi ultrazvučni senzor kako bi izbjegao prepreke.

7 Zaključak

Mikrokontroleri i senzori se već neko vrijeme koriste u nastavi poučavanja programiranja u našim školama. To možemo vidjeti i po preporukama za ostvarenje obrazovno-odgojnih ishoda koje su navedene u kurikulumu za nastavni predmet Informatika [1]. U preporukama za osnovnu školu su navedeni upravo roboti. Kako se u školama već koristi programski jezik Scratch, učenicima ne bi trebao biti problem prijeći na programiranje u mBlocku. Umjesto likova na ekranu, u mBlocku pokreću motore, umjesto ispisa na ekran, učenici ispisuju na display, a umjesto ručnog unošenja ulaznih podataka, koriste senzore i stvarne podatke iz svoje okoline.

Pixy kao senzor ima mnogo mogućnosti jer pomoću njega dobivamo puno različitih podataka. Možemo dobiti podatak o boji objekta, gdje se on nalazi u kadru i koliko je objekt velik. Također možemo pomoću kodova boja jednostavno označiti prostorije kako bi robot znao u kojoj se prostoriji nalazi. Zbog svega toga Pixy je zanimljiv za korištenje. Ako na mikrokontroler spojimo i lampice (npr. RGB lampice), možemo napraviti vrlo interaktivne, šarene i zanimljive projekte.

Velika prednost Pixy kamere je ta što se može koristiti u mBlocku. Nažalost, kao što je opisano u ovom radu, ekstenzija za rad s Pixy kamerom u mBlocku ima nedostataka. U ovom radu je korištena ekstenzija PixyCAM I2C. Korištenjem naredbi iz ove ekstenzije smo došli do nekoliko problema. Veliki je problem naredba **Get Objects**. Vrlo je bitno da možemo jednostavno doći do podatka koliko objekata je Pixy detektirao, bez „trikova“. Nadalje, naredba **Serial Print of object no. i** ispisuje podatke o objektima koji više nisu u kadru. Možemo doskočiti problemu tako da najprije ustanovimo koliko objekata imamo u listi, a ostale podatke ignoriramo. Bilo bi dobro da se biblioteka prilagodi tako da ova naredba ispisuje samo stvarne podatke. U suprotnom, program se neće ponašati onako kako očekujemo. Osim toga, može se dogoditi da se neki objekt prikazuje kao više manjih objekata te da se stanje mijenja čak više puta u sekundi. Te podatke bi trebalo filtrirati, pojednostavniti i zaključiti da je to jedan objekt.

Dok se biblioteka za korištenje Pixyja u mBlocku ne pojednostavni, Pixy bi bilo bolje izbjegavati koristiti u osnovnoj školi.

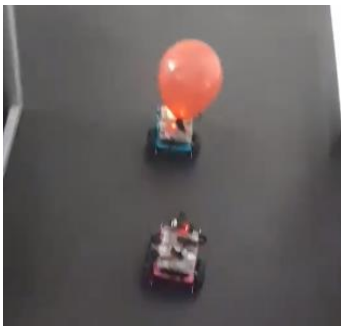
Još jedan manji nedostatak je taj što ne možemo vidjeti stvarnu sliku s podacima o detektiranim objektima (*Cooked video*) u isto vrijeme kada mikrokontroler koristi Pixy. Ta

mogućnost bi pomogla učenicima, pogotovo onima mlađeg uzrasta, da lakše shvate razliku između onoga što mi vidimo i podataka koje Pixy šalje.

Velika prednost mBlocka je ta što automatski prevodi program iz blokovskog u tekstualni programski jezik kojega zatim možemo otvoriti u Arduino IDE. Kako slažemo blokove s naredbama u mBlocku, u isto vrijeme vidimo kako se mijenja kod u Arduinu IDE. To omogućuje učenicima lakši prijelaz s blokovskog na tekstualni programski jezik. Osim toga, oni učenici koji žele napisati složenije programe nisu ograničeni ponuđenim naredbama u mBlocku.

Prilikom instaliranja biblioteke za korištenje Pixy kamere u Arduino IDE dobijemo i primjere programa u kojima se koristi Pixy. Programi su komentirani tako da je jednostavno shvatiti zašto služi pojedini dio programa. Na temelju tih primjera možemo napisati složenije programe u kojima koristimo i druge senzore ili izlazne jedinice (npr. lampice ili display).

U posljednjem primjeru u ovom radu je opisan kod za praćenje objekta korištenjem mBota i Pixy kamere. To je odličan primjer prikaza mogućnosti Pixy kamere kao senzora. Upravo ovaj kod se koristio na Otvorenom danu matematike 2018 (Prirodoslovno-matematički fakultet Sveučilišta u Zagrebu) [22].



Našim mladim posjetiteljima smo prezentirali kako jedan mBot prati drugoga. Učenici su odmah zaključili da mBot prati drugoga pomoću kamere koju ima na sebi i zato što drugi mBot ima na sebi crveni balon. Na pitanje kako možemo poboljšati praćenje, učenici su odmah primijetili što bi se moglo unaprijediti (npr. omogućiti izbjegavanje prepreka, okretanje kada više ne vidi balon itd.).

Iz svega ovoga možemo zaključiti da uključivanje senzora u nastavu programiranja otvara brojne nove mogućnosti i potiče kreativno razmišljanje učenika. Međutim, za učinkovito uključivanje složenijih senzora u nastavu, posebno u nastavu programiranja u nižim razredima, potrebno je pripremiti pouzdane biblioteke koje vraćaju podatke s kojima je jednostavno rukovati kako bi se učenici mogli posvetiti programskoj logici bez potrebe razumijevanja implementacijskih detalja rada samog senzora. U kasnijoj fazi učenja programiranja učenici mogu i samostalno dorađivati programsku potporu za rad senzora. Uz uspješno ispunjenje ovih pretpostavki, složeni senzori poput kamere Pixy sigurno će naći svoje mjesto u obrazovnom procesu, posebno u domeni Računalno razmišljanje i programiranje.

8 Literatura

[1] *Kurikulum nastavnog predmeta Informatika za osnovne i srednje škole,* Narodne novine, izdanje: NN 22/2018, https://narodne-novine.nn.hr/clanci/sluzbeni/2018_03_22_436.html

[2] M. Radošević, *Poželjne karakteristike za poučavanje programiranja u osnovnoj školi,* (2017)., https://urn.nsk.hr/urn:nbn:hr:217:096758, Diplomski rad, Zagreb: Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet

[3] M. Plantosat, *Primjena senzora u električnim strojevima,* (2017.), https://urn.nsk.hr/urn:nbn:hr:200:559555, Završni rad, Osijek: Sveučilište u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija

[4] CMUcam: Open Source Programmable Embedded Color Vision Sensors http://www.cmucam.org/

[5] *CMUcam,* https://en.wikipedia.org/wiki/CMUcam

[6] *PIXY Documentation,* https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:overview

[7] S. Mallick, *Blob Detection Using OpenCV (Python, C++),* (2015), https://www.learnopencv.com/blob-detection-using-opencv-python-c/

[8] *PixyMon Overview,* http://www.cmucam.org/projects/cmucam5/wiki/PixyMon_Overview

[9] *Teach Pixy an Object,* http://cmucam.org/projects/cmucam5/wiki/Teach_Pixy_an_object

[10] Using Color Codes, *http://www.cmucam.org/projects/cmucam5/wiki/Using_Color_Codes*

[11] *makeblock,* https://www.makeblock.com

[12] *How to talk to Pixy,* http://www.cmucam.org/projects/cmucam5/wiki/Porting_Guide

[13] *OBJECT FOLLOWER ROBOT (WITH MBOT + PIXY),* https://www.instructables.com/id/Object-Follower-Robot-with-MBot-Pixy/

[14] *Makeblock products* https://store.makeblock.com/

[15] I. Kniewald, V. Galešev, G. Sokol, V. Vlahović, D. Kager, H. Kovač, N. Kunštek, *Informatika+ 7, Udžbenik iz informatike za 7. razred osnovne škole,* 2. izdanje, Zagreb, 2018., SysPrint

[16] *ARDUINO Software,* https://www.arduino.cc/en/Main/Software

[17] *Arduino Library and API,* http://cmucam.org/projects/cmucam5/wiki/Arduino_API

[18] *Language Reference,* https://www.arduino.cc/reference/en/

[19] B. Landoni, (2014), *Pixy camera: detect the colour of the objects and track their position,* https://www.open-electronics.org/pixy-camera-detect-the-colour-of-the-objects-and-track-their-position/

[20] *mBot,* https://www.makeblock.com/steam-kits/mbot

[21] *Croatian Makers liga,* http://croatianmakers.hr/hr/croatian-makers-liga/

[22] *Otvoreni dan matematike 2018.* http://www.pmf.unizg.hr/primateh/informatika?@=1kta8#news_83043

Sažetak

Na početku ovog rada su navedeni odgojno-obrazovni ishodi iz kurikuluma nastavnog predmeta Informatike za domenu Računalno razmišljanje i programiranje za osnovnu i srednju školu koji su povezani s korištenjem mikrokontrolera i senzora, odnosno robota. Zatim su uspoređeni programski jezici Logo i Scratch te je opisan senzor objekata pod imenom Pixy. Također je opisan pomoćni program PixyMon koji omogućuje prikaz objekata koje ovaj senzor prepoznaje. Objašnjen je način definiranja jednostavnih potpisa (*signatures*) i kodiranja bojom (*color codes*). Opisane su naredbe iz ekstenzije PixyCAM I2C za rad s Pixyjem u mBlocku. Obrađene su dvije aktivnosti prikladne za učenike sedmih razreda osnovne škole i opisani određeni problemi prilikom korištenja naredbi iz ekstenzije PixyCAM I2C te dane sugestije za njihovo rješavanje.
U radu je opisano i programsko okruženje Arduino IDE s naglaskom na naredbe za rad s Pixyjem. Opisan je primjer osnovnog programa napisanog u Arduino IDE koji na serijski monitor ispisuje podatke o detektiranim objektima. Na kraju rada je opisan složeniji program za praćenje objekata za edukacijski robot opremljen kamerom Pixy.

Summary

At the beginning of this work, the learning outcomes from curriculum of the Informatics for domain "Computer Thinking and Programming" for Elementary and Secondary Schools that are related to the use of microcontrollers, sensors and robots are analised.
After that, the programming languages Logo and Scratch are compared, and a sensor of objects named Pixy is described, together with PixyMon, utility for presenting what objects this sensor detects. The way of defining signatures and color codes is explained in detail. Commands from the PixyCAM I2C Extension for Pixy Operation in mBlock are described. Two activities suitable for seventh grade students are presented. Some problems that are encountered while using the PixyCAM I2C extension are pointed out and suggestions for solving them are proposed.
In the last part of this work, the Arduino IDE programming environment is described, emphasising commands for working with Pixy. An example of the basic program written in Arduino IDE which sends data about detected objects to the serial monitor is presented. At the very end of this work, a more complex program for monitoring objects by educational robot equipped with the camera Pixy was described.

Životopis

Rođena sam 24. srpnja 1992. godine u Ljubljani, Slovenija. Kada sam imala šest godina, preselili smo se u mali slavonski grad, Našice. U Našicama sam završila Osnovnu školu kralja Tomislava i prirodoslovno-matematičku gimnaziju u Srednjoj školi Isidora Kršnjavoga. Maturirala sam 2011. godine s odličnim uspjehom. Završila sam preddiplomski sveučilišni studij Matematika; smjer: nastavnički, na Prirodoslovno-matematičkom fakultetu Sveučilišta u Zagrebu 2016. godine te sam iste godine upisala diplomski sveučilišni studij Matematika i informatika, smjer: nastavnički. Metodičku praksu iz matematike i informatike odradila sam u Osnovnoj školi Vrbani te u XV. gimnaziji u Zagrebu.