Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Domen Soklič

Integracija kamere Raspberry Pi v sistem EPICS

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Robert Rozman

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Porazdeljeni kontrolni sistemi za upravljanje kompleksnejših procesov so dandanes vedno bolj iskani. Večinoma so vezani na uporabo specializiranih izdelkov in tovrstnih storitev višjih cenovnih razredov. Tehnologija računalniških sistemov je medtem doživela intenzivni razvoj, ki se kaže tudi v široki ponudbi cenovno dostopnejših in tehnološko naprednejših naprav oziroma sistemov. Izdelajte prototip takšnega cenovno dostopnega sistema za vključitev preproste kamere v obstoječi programski krmilni sistem EPICS. Sistem dopolnite z lastnimi rešitvami tam, kjer je to potrebno in smiselno. Sistem naj zagotovi uporabnikom udobno, prijazno in cenovno dostopno vključitev kamere kot naprave za nadzor poteka posameznih procesov v sistemu EPICS. Pri tem zagotovite vse potrebne funkcionalnosti za popolno integracijo v omenjeni sistem.

Izjava o avtorstvu zaključnega dela

Spodaj podpisani Domen Soklič, vpisna številka 63120098, avtor zaključnega dela z naslovom:

Integracija kamere Raspberry Pi v sistem EPICS (angl. *Integration of the Raspberry Pi Camera into EPICS*)

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom viš. pred. dr. Roberta Rozmana;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 30. januarja 2017 Podpis študenta/-ke:

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Uporabljene tehnologije in programska oprema **5**

2.1 Kontrolni sistem EPICS 5

2.1.1 Vtičnik areaDetector 9

2.1.2 Vtičnik AsynDriver 10

2.2 Spletno ogrodje AngularJS 12

2.3 Skupek predlog Bootstrap 13

2.4 Spletni strežnik Flask 14

2.4.1 Vtičnik PyEpics 15

3 Implementacija gonilnika za vtičnik areaDetector **17**

3.1 Zgradba gonilnika 19

3.2 Potek razvoja 23

4 Spletni uporabniški vmesnik **25**

4.1 Programski vmesnik 25

4.2 Spletna stran 27

5 Sklepne ugotovitve **31**

Seznam uporabljenih kratic

kratica	angleško	slovensko
EPICS	Experimental Physics and Industrial Control System	Sistem za postavitev in upravljanje s kontrolnimi sistemi
GUI	Graphical User Interface	Grafični uporabniški vmesnik
ICS	Integrated Control System	Integriran kontrolni sistem
PV	EPICS Process Variable	Procesna spremenljivka EPICS
CDN	Content Delivery Network	Omrežje za dostavo vsebine
API	Application programming interface	Programski vmesnik
REST	Representational state transfer	Predstavitveni prenos stanj
DOM	Document Object Model	Drevesna struktura elementov spletne strani

Povzetek

Naslov: Integracija kamere Raspberry Pi v sistem EPICS

Povzetek: Cilj diplomske naloge je izdelati sistem za zajem slike z Raspberry Pi kamero z uporabo sistema EPICS. EPICS je sistem, ki poenostavi postavitev in upravljanje kontrolnih sistemov. V tem diplomskem delu smo razvili programski gonilnik, ki služi kot posrednik med obstoječo nizko-nivojsko knjižnico (API) za upravljanje kamere in sistemom EPICS.

S tem se je skrajšal čas integracije, ki je potreben za vključitev računalnika Raspberry Pi v kontrolnih sistemih pospeševalnikov delcev, v industriji ali drugih okoljih, kjer se uporablja EPICS.

Poleg tega smo razvili spletni uporabniški vmesnik, na katerem se prikazuje slika in se lahko spreminja nastavitve kamere. Razvit spletni vmesnik omogoča uporabniku nadzor nad vsemi nastavitvami kamere, ki jih podpira API.

Ključne besede: raspberry pi, epics, kamera, areaDetector, flask, angularjs.

Abstract

Title: Integration of the Raspberry Pi Camera into EPICS

Abstract: The aim of this work was to create a system that can capture images with the use of EPICS. EPICS is a system that simplifies the deployment of control systems and provides an easy way to operate them. We have developed a driver that acts as an intermediary between the low level camera access library (API) and EPICS.

This work has shortened the integration time for systems that wish to use the Raspberry Pi in control systems for particle accelerators, industrial applications or other environments that use EPICS.

We have also developed a web graphical interface which can display the captured picture and allows the user to change the settings of the camera that are supported by the API.

Keywords: raspberry pi, epics, camera, areaDetector, flask, angularjs.

Poglavje 1

Uvod

Kontrolni sistem je naprava oziroma skupek naprav, ki se med seboj nadzirajo in upravljajo [1]. Kontrolni sistemi imajo tipično več senzorjev, s katerimi zaznavajo pogoje, v katerih se izvaja nadzorovan proces, na proces pa vplivajo z izhodnimi napravami kot so motorji in magneti. Povratno informacijo tipal kontrolni sistemi uporabijo za prilagajanje izhodnih naprav, da sistem doseže pogoje, ki so potrebni za izvajanje željenega procesa.

V večini primerov kontrolni sistemi uporabljajo drago, specializirano opremo za zajem slik. Uporaba takih naprav je dobra v produkcijskem okolju, saj so robustne in odporne na težke razmere. Pogosto pa bi želeli med razvojem uporabiti kakšno cenejšo napravo, ki jo kasneje lahko zamenjamo s profesionalno, če bi se izkazalo, da cenejša naprava ni dovolj zanesljiva ali robustna za produkcijsko okolje.

Problem smo rešili tako, da smo napisali EPICS gonilnik za kamero NoIR [3] računalnika Raspberry Pi [2]. S tem smo dosegli, da se lahko hitro doda računalnik Raspberry Pi in njegovo kamero NoIR v kontrolni sistem, ki uporablja EPICS.

EPICS je skupek programov in programske opreme, ki nudijo programsko infrastrukturo, ki se jo lahko uporablja v velikih distribuiranih kontrolnih sistemih. Z njim se lahko upravlja pospeševalnike delcev, velike eksperimente in teleskope. Taki porazdeljeni kontrolni sistemi tipično obsegajo več deset

ali celo sto računalnikov, ki so med seboj povezani v računalniško omrežje. Z uporabo EPICS-a je zelo preprosto posredovati sporočila različnim računalnikom, ter prejemati povratne informacije naprav. Pregled informacij in pošiljanje ukazov se lahko izvaja iz centralne nadzorne sobe ali preko spleta.

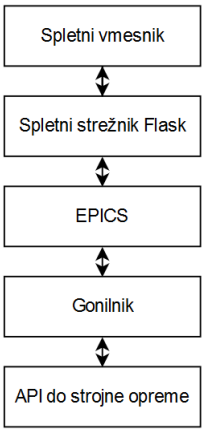
Med vsemi programskimi ogrodji za upravljanje kontrolnih sistemov je bil EPICS izbran zato, ker je že uveljavljen na področju velikih fizikalnih eksperimentov. Uporabljajo ga inštituti kot so ESS, ITER in SLAC [4]. Uporaba sistema EPICS je najbolj smiselna, saj se s tem pokrije največje število inštitutov in uporabnikov. V sistemu EPICS so podatki predstavljeni kot zapisi (*ang. record*), iz katerih lahko preberemo stanje naprav, ali v njih zapišemo vrednost, da napravam pošljemo ukaz za izvedbo nekega dejanja.

Na računalnik Raspberry Pi lahko priključimo posebno kamero, ki se priključi neposredno na tiskano vezje računalnika. Prednost takšnega priklopa je, da sliko neposredno obdeluje grafična enota, brez da bi s tem obremenili še procesor. Posebnost kamere NoIR je, da nima filtra za svetlobo v infrardečem območju. Prednost tega je, da infrardeča svetloba, ki jo predmeti lahko absorbirajo ali odbijejo, včasih vsebuje pomemben podatek o lastnostih opazovanega predmeta. Primer takega predmeta so rastline in postopek fotosinteze. Klorofil absorbira modro in rdečo svetlobo, ne pa zeleno in infrardečo. En način, kako lahko znanstveniki zaznajo potek fotosinteze je, da poskusijo zaznati prisotnost odbite zelene svetlobe. Še boljši način pa je, da zaznamo prisotnost infrardeče in odsotnost modre svetlobe. Za to bi uporabili infrardečo kamero, pred katero postavimo filter modre svetlobe [6].

Pri svojem delu smo si pomagali z EPICS-ovo knjižnico za kamere *areaDetector* [7], ki služi kot osnova za pisanje našega novega gonilnika. Ta knjižnica poenostavi pisanje gonilnikov za nove kamere. Določa tudi standardni vmesnik za dostop do slikovnih podatkov in nekaterih drugih parametrov, kar omogoča hitro zamenjavo ene kamere z drugo.

Za prikaz zajetih slik smo razvili spletni uporabniški vmesnik, ki omogoča uporabnikom, da spreminjajo nastavitve kamere kot so svetlost, kontrast in ločljivost slike. Strežnik za grafični vmesnik teče na istem računalniku

Raspberry Pi, na katerega je priklopljena kamera.



Slika 1.1: Diagram toka podatkov med komponentami tega diplomskega dela.

Potek podatkov v tem diplomskem delu je prikazan na sliki 1.1. Na najvišjem nivoju je spletni vmesnik, ki prikazuje sliko in pošilja spremembe nastavitev spletnemu strežniku. Spletni strežnik služi kot posrednik med sistemom EPICS in spletnim vmesnikom. EPICS skrbi za komunikacijo med napravami. V tem diplomskem delu se vse odvija na isti napravi, EPICS pa dovoljuje, da bi bila kamera na eni napravi, spletni strežnik pa na drugi. Naslednji korak je naš gonilnik, ki je posrednik med EPICS-om in knjižnico za dostop do strojne opreme kamere. Zadnji nivo je knjižnica za dostop do kamere (API), ki nudi dostop do strojne opreme.

Diplomsko delo sestoji iz naslednjih tematskih sklopov oziroma poglavij.

V drugem poglavju bomo govorili o uporabljenih tehnologijah in obstoječi programski opremi. Povzeli bomo značilnosti uporabljenih tehnologij in njihovo uporabo.

V tretjem poglavju se bomo poglobili v sistem EPICS, ki predstavlja največji delež tega diplomskega dela. Razkrili bomo delovanje razvitega gonilnika in izzive, s katerimi smo se soočili med razvojem.

V četrtem poglavju bomo govorili o spletnem uporabniškem vmesniku. Razložili bomo, kako smo ga izdelali in kako komunicira z ostalimi kompo-

nentami, ki smo jih razvili.

V petem poglavju bomo povzeli naše delo in predstavili svoj zaključek. Predstavili bomo tudi potencialne izboljšave in spremembe.

Poglavje 2

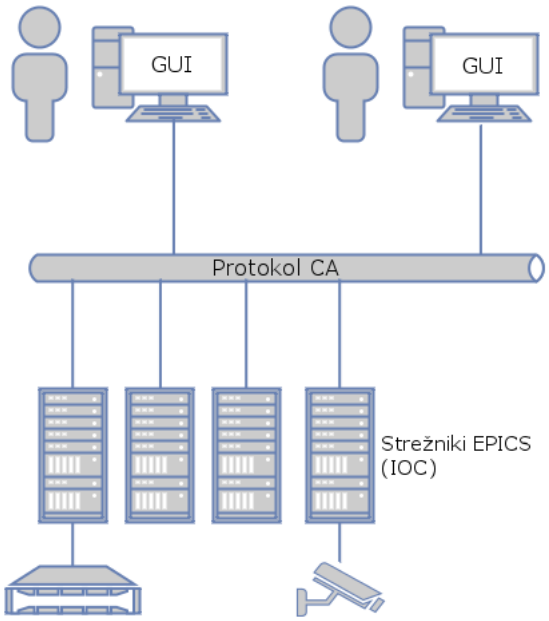
Uporabljene tehnologije in programska oprema

> "Če sem videl dlje, sem zaradi tega, ker sem stal na ramenih velikanov."
>
> Isaac Newton

Vsako delo vsebuje veliko tehnologij in programske opreme, ki so jo ustvarili drugi. To poglavje bo predstavilo in opisalo obstoječe knjižnice, ki so bile uporabljene v tem diplomskem delu.

2.1 Kontrolni sistem EPICS

EPICS uporablja arhitekturo Odjemalec/Strežnik in Objavi/Naroči za komunikacijo med posameznimi računalniki. Večina strežnikov (ki se imenujejo Input/Output Controller ali IOC) izvajajo dejanja v fizičnem svetu, svoje podatke pa objavijo drugim odjemalcem s protokolom Channel Access (CA). Ta protokol je bil izdelan specifično za aplikacije, ki močno obremenijo pasovno širino omrežja in pričakujejo informacije v realnem času. Zaradi tega je EPICS odličen za uporabo v kontrolnih sistemih, ki so zgrajeni iz več sto računalnikov.



Slika 2.1: Shema povezanih naprav v kontrolnem sistemu EPICS.

Slika 2.1 predstavlja delovanje sistema EPICS. Ponazarja porazdeljeno naravo kontrolnih sistemov, kjer več strežnikov komunicira med seboj in deli podatke. Strežniki lahko tako drugim napravam v omrežju omogočijo dostop do svoje strojne opreme.

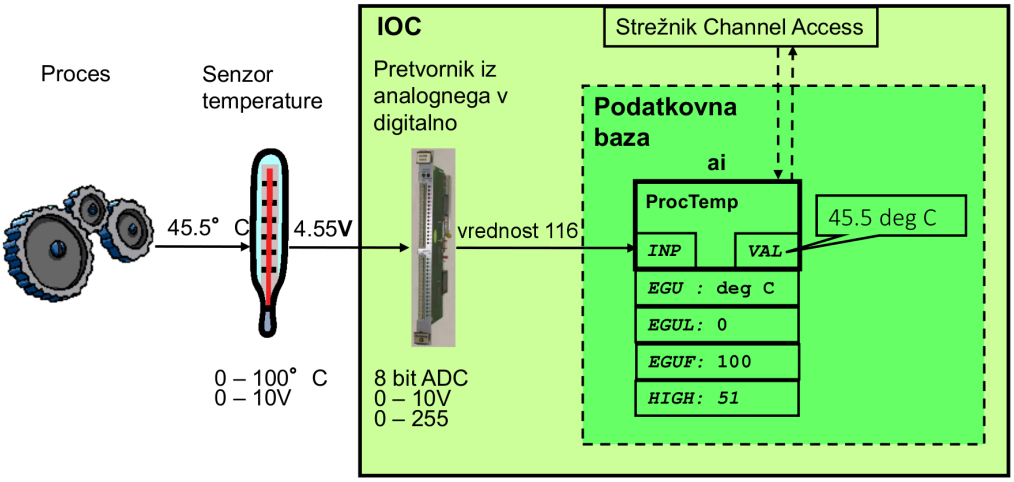
EPICS mora biti zanesljiv, saj lahko napaka v delovanju sistema poškoduje drago opremo, ki stane več tisoč dolarjev in lahko zahteva več dni ali celo tednov, da se jo popravi. Sistem EPICS se ponaša z več kot 95% zanesljivostjo delovanja na pospeševalniku APS (Advanced Proton Source) [5].

Podatki so zbrani v tako imenovanih podatkovnih bazah (database), vsak podatek oziroma zapis (record) pa je lahko več različnih vrst. Glede na vrsto zapisa ima vsak zapis nabor atributov.

Princip delovanja EPICS si lahko predstavljamo z naslednjim primerom: predstavljajte si, da imate električno pečico. Taka pečica bi imela več zapisov za hranjenje podatkov in sprejemanje ukazov, ter nastavitev. Pečica bi imela zapise, kot so:

- vklopljeno: Binarni vhod, ki določa, če je pečica vklopljena ali izklopljena.
- ventilator: Binarni vhod, ki določa, če naj pečica uporablja ventilator.
- nastavljena temperatura: Analogni vhod, na katero temperaturo se mora pečica segreti.
- dejanska temperatura: Analogni izhod, kakšna je trenutna temperatura pečice.

Take zapise bi lahko načeloma imel vsak krmilnik, prednost EPICS-a pa je, da ima že veliko vgrajenih funkcionalnosti, ki jih tipično podpirajo kontrolni sistemi. EPICS ima že vgrajeno upravljanje z alarmi, upoštevanje merskih enot in omogoča razširitev funkcionalnosti s programsko kodo, napisano v jeziku C ali C++. Zapise lahko preberejo in spreminjajo tudi druge naprave v omrežju, zaradi česar je EPICS odličen za porazdeljene sisteme z velikim številom naprav.



Slika 2.2: Prikaz zapisov sistema EPICS.

Za boljše razumevanje EPICS zapisov si oglejte sliko 2.2. Nek zunanji proces vpliva na temperaturo na termometru. Termometer zazna tempera-

tip	**angleško**	**slovensko**
aai	Array Analog Input	Polje analognih vhodov
aao	Array Analog Output	Polje analognih izhodov
ai	Analog Input	Analogni izhod
ao	Analog Output	Analogni vhod
aSub	Array Subroutine	Podprogram s tabelo
bi	Binary Input	Binarni vhod
bo	Binary Output	Binarni izhod
calc	Calculation	Izračun
calcout	Calculation Output	Izračun z izhodom
compress	Compression	Kompresija
dfanout	Data Fanout	Podvojitev podatkov
event	Event	Dogodek
fanout	Fanout	Podvojitev
histogram	Histogram	Histogram
longin	Long Input	Vhod tipa long
longout	Long Output	Izhod tipa long
mbbi	Multi-Bit Binary Input	Več-bitni binarni vhod
mbbiDirect	Multi-Bit Binary Input Direct	Neposredni več-bitni binarni vhod
mbbo	Multi-Bit Binary Output	Več-bitni binarni izhod
mbboDirect	Multi-Bit Binary Output Direct	Neposredni več-bitni binarni izhod
permissive	Permissive	Popustljiv
sel	Select	Izbira
seq	Sequence	Sekvenca
state	State	Stanje
stringin	String Input	Besedilni vhod
stringout	String Output	Besedilni izhod
subArray	Sub-Array	Pod-tabela
sub	Subroutine	Podprogram
waveform	Waveform	Polje z vzorci signala

Tabela 2.1: Tipi EPICS zapisov.

turo in jo predstavi kot napetost med 0 in 10 volti. Pretvornik iz analognih vrednosti v digitalne pretvori to napetost v diskretno vrednost temperature in jo zapiše v zapis *ProcTemp*.

Na zapisu *ProcTemp* uporabljamo več atributov. Prvi atribut je INP, ki določa, od kod zapis pridobi vrednost. Vrednost se zapiše v atribut VAL. To bi načeloma že zadostovalo za delovanje kontrolnega sistema, želimo pa uporabiti še dodatne funkcionalnosti EPICS-a oziroma tipe atributov:

- atribut EGU za enoto podatka,
- atribut EGUL predstavlja najnižjo vrednost, ki jo lahko vsebuje zapis,
- atribut EGUF predstavlja najvišjo vrednost, ki jo lahko vsebuje zapis,
- atribut HIGH določa mejo normalnih vrednosti.

Če vrednost zapisa preseže vrednost, ki je določena v atributu HIGH, se sproži alarm.

Vsi podprti tipi zapisov v EPICS različici 3.14 so predstavljeni v tabeli 2.1.

2.1.1 Vtičnik areaDetector

Vtičnik areaDetector v sistemu EPICS poenostavi upravljanje detektorjev površine. Detektor površine je kakršenkoli senzor, ki vrne dvo-dimenzionalne podatke. Najbolj pogosto se uporablja za zajem podatkov s kamerami [7].

Glavni cilji tega vtičnika so:

- zmanjšati količino programske kode, ki jo je potrebno napisati, da se podpre nov detektor;
- nuditi standardni vmesnik, ki določa vse funkcije in parametre, ki jih mora podpreti detektor;
- nuditi nabor EPICS zapisov, ki bodo prisotni pri vsakem detektorju. To omogoča splošnim prikazovalnikom slike, da uporabljajo detektorje;
- omogočati mora preprosto razširitev osnovnega nabora zapisov, da se lahko podpre tudi bolj specifične funkcije detektorjev;

- visoka zmogljivost;
- omogočati od naprave neodvisno realno-časovno obdelavo podatkov;
- podpreti nabor najbolj pogostih detektorjev.

Na sliki 2.3 imamo narisan diagram arhitekture vtičnika. Slika se prenaša iz nivoja 1 do nivoja 6, kjer se nahaja odjemalec oziroma prikazovalnik. Nastavitve se prenašajo v drugo smer, iz nivoja 6 do nivoja 1, kjer je strojna oprema.

Nivo 6 Grafični vmesnik oziroma odjemalec. Manjši del mojega dela se nahaja tu.

Nivo 5 Tu se nahajajo definicije zapisov, kjer se hranijo podatki.

Nivo 4 Splošno asinhrono delovanje "asyn"[8]. Princip asinhronega delovanja je razložen podrobneje v poglavju 2.1.2.

Nivo 3 Gonilnik, ki veže višje nivoje s programskim vmesnikom. Večina mojega dela se nahaja tu.

Nivo 2 Programski vmesnik, za interakcijo s strojno opremo.

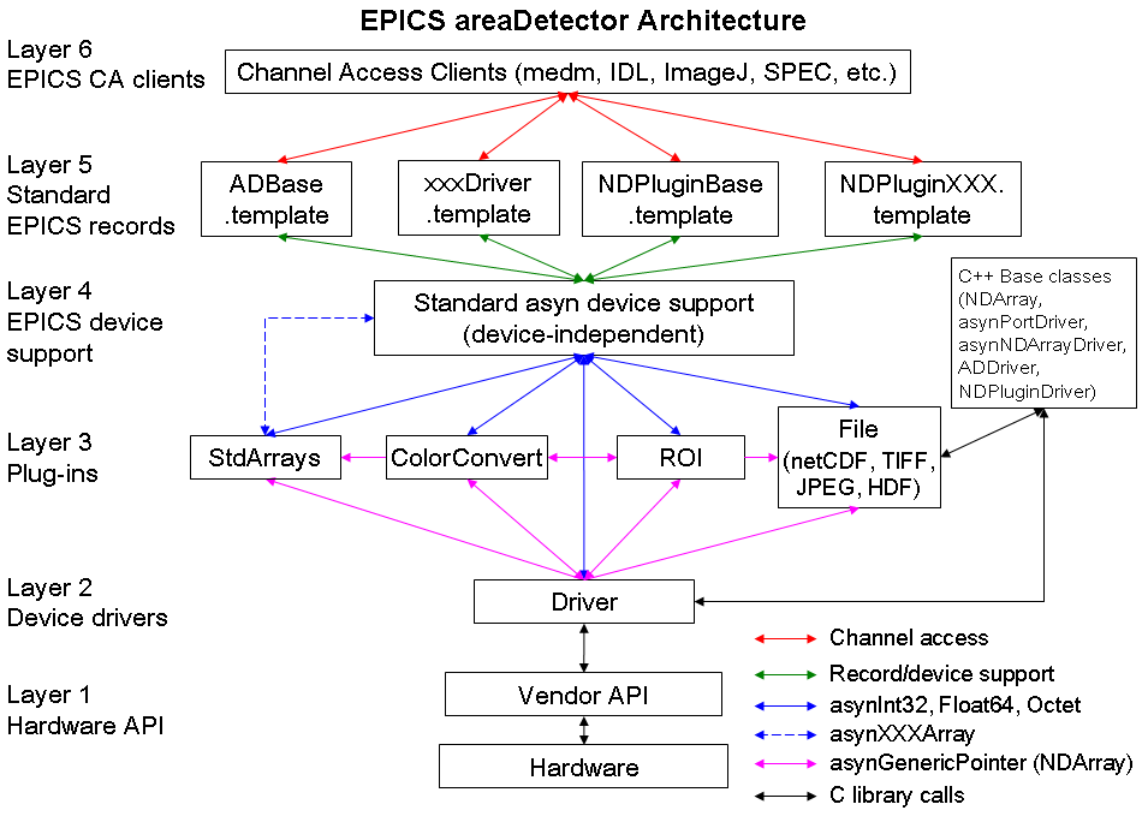
Nivo 1 Strojna oprema.

2.1.2 Vtičnik AsynDriver

AsynDriver [8] je vtičnik za EPICS, ki omogoča asinhrono pridobivanje podatkov. Tipično se ga uporablja za komunikacijo z napravami, ki potrebujejo daljši čas, da se odzovejo. Priročen pa je tudi, če se mora izvesti neka programska koda, ko nek odjemalec spremeni vrednost v EPICS zapisu.

Prednost asinhronega delovanja vtičnika je, da našemu gonilniku ni treba neprestano preverjati, če se neka vrednost spremeni. Ob spremembi se sproži funkcija, ki izvede programsko kodo.

Vtičnik našemu gonilniku omogoča asinhrono zaznavanje, kdaj odjemalec naredi spremembo nastavitev ali zahteva zajem slike. S tem smo se izognili



Slika 2.3: Arhitektura delovanja vtičnika areaDetectorja.

hranjenju starih vrednosti in neprestanemu preverjanju, če se je zgodila sprememba zapisa.

2.2 Spletno ogrodje AngularJS

AngularJS je ogrodje za izdelavo interaktivnih spletnih aplikacij. Omogoča uporabo HTML-ja za izdelavo predlog in razširitev sintakse HTML. Vezava podatkov odstrani veliko programske kode, ki bi jo sicer morali napisati. Vse to pa se izvede v spletnem brskalniku, zaradi česar je idealni partner za sodelovanje s strežnikom.

HTML je dober za definicijo dokumentov, ni pa dovolj za razvoj spletnih aplikacij. Tipično se težave reši tako, da se uporabi knjižnico, kot je *jQuery* [9], ali pa ogrodje, kot je *durandal* [19] ali *ember* [20].

Angular ima pri tem drugačen pristop. Stremi k temu, da bi bil čim manjši razkorak med običajnimi HTML elementi in dodatnimi podatki, ki jih potrebuje aplikacija. Angular nauči brskalnik novo sintakso s pomočjo konstruktov, ki se imenujejo direktive (ang. directive). Primeri teh konstruktov so:

- vezava podatkov z {{ }},
- kontrolne strukture DOM za ponavljanje, prikazovanje in skrivanje elementov,
- podpora za obrazce in validacijo obrazcev,
- dodajanje novih lastnosti elementov, kot je obnašanje dogodkov DOM,
- združevanje HTML-ja v ponavljajoče se skupine [10].

Preprost primer, ki prikaže prednosti ogrodja je viden na sliki 2.4. Programska koda spletne strani je na levi strani slike, kako bi pa taka spletna stran izgledala v brskalniku, pa na desni. Tu je še najbolje vidna uporaba vezave podatkov. Besedilno polje je povezano z besedilom tako, da ko uporabnik tipka v besedilno polje, se njegov vnos med tem že avtomatično izpisuje v

besedilu spodaj. Ta povezava je ustvarjena z uporabo *ng-model=”yourName”* in {{*yourName*}}. Če bi želeli implementirati podobno obnašanje spletne strani brez uporabe AngularJS, bi porabili neznatno več časa in napisali veliko programske kode.

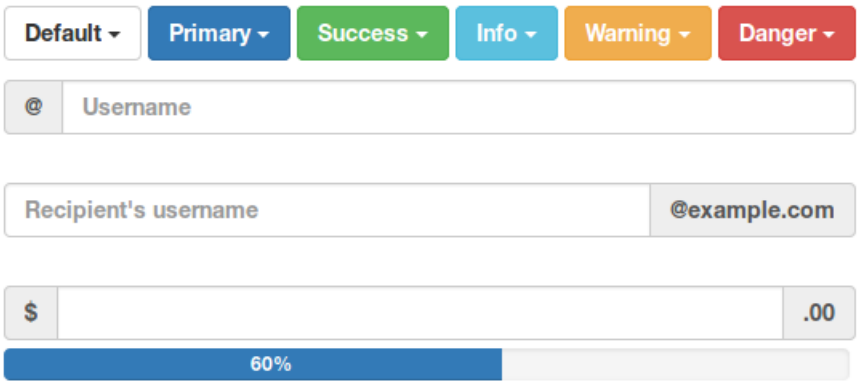
Slika 2.4: Primer vezave podatkov z ogrodjem AngularJS.

2.3 Skupek predlog Bootstrap

Bootstrap [11] je skupek predlog CSS, ki poenostavijo oblikovanje spletnih strani. Bootstrap je izdelan tako, da je izgled za mobilne naprave na prvem mestu.

To pomeni, da mora oblikovalec razvrstiti elemente HTML po takem vrstnem redu, kot naj bi bili na mobilni spletni strani. Elementi se nato prilagodijo večjemu zaslonu.

Bootstrap vsebuje številne vnaprej pripravljene predloge za izgled obrazcev, gumbov, okvirjev in podobno. Na sliki 2.5 je prikazanih nekaj izbranih gradnikov, ki uporabljajo te predloge. Ima tudi zelo priročen sistem razdelitve zaslona v mrežo.



Slika 2.5: Prikaz izbranih gradnikov okolja Bootstrap.

2.4 Spletni strežnik Flask

Flask [13] je mikro-ogrodje za preprosto izdelavo spletnih REST programskih vmesnikov (REST API). Temelji na orodju Werkzeug [14] in pogonu za predloge Jinja2 [15].

Flask se imenuje mikro-ogrodje, ker od razvijalca ne zahteva uporabo točno določenega orodja ali knjižnice. Podpira razširitve, ki lahko dodajo funkcionalnost na isti način, kot da bi bile del samega Flaska. Obstajajo razširitve za preverjanje obrazcev, prenos datotek, razne avtentikacijske tehnologije in mnoga orodja, povezana z ostalimi ogrodji [16].

Tak spletni REST programski vmesnik smo želeli ustvariti, ker smo potrebovali vmesno programsko opremo, ki pošlje zahtevek spletnega brskalnika za zajema slike sistemu EPICS, ter posreduje nastale slike nazaj.

To ogrodje smo izbrali, ker lahko z njim zelo hitro razvijemo delujoči strežnik. Preprost strežnik je viden v izseku programske kode 2.1, ki se na zahtevke odzove z "Hello World". Poleg tega pa obstaja za programski jezik Python [12], za katerega je bilo to ogrodje razvito, vtičnik PyEpics, ki poveže strežnik z okoljem EPICS.

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route("/")
5 def hello():
6     return "Hello World!"
7
8 if __name__ == "__main__":
9     app.run()
```

Izsek 2.1: Preprost primer uporabe strežnika Flask.

2.4.1 Vtičnik PyEpics

PyEpics [17] nudi Python vmesnik za dostop do EPICS zapisov preko kanalnega dostopa (ang. *Channel Access* ali CA). Vsebuje metodi za branje in pisanje procesnih spremenljivk. Poleg tega omogoča, da je obveščen vsakič, ko se spremeni vrednost spremenljivke. Kljub preprosti uporabi osnovnih operacij nudi knjižnica dostop tudi do bolj naprednih zmožnosti kanalnega dostopa.

Poglavje 3

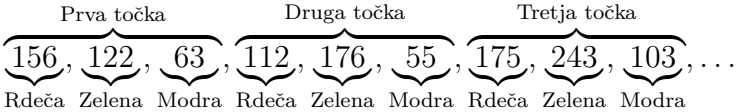
Implementacija gonilnika za vtičnik areaDetector

Za izdelavo podpore kamere smo najprej morali ugotoviti, kako so sestavljeni gonilniki in kako lahko napišemo svojega. Za učenje smo pregledali programsko kodo gonilnika *ADExample*. ADExample vsebuje gonilnik simulirane kamere in IOC (strežnik sistema EPICS), na katerem ta gonilnik teče [18].

Naš gonilnik je narejen iz dveh programskih niti. Prva nit čaka na spremembe nastavitev, ki so zapisane v EPICS zapisih. Druga nit čaka na ukaz za zajem slike in ga tudi izvede.

Slika je sestavljena iz zaporedja barvnih komponent vsake točke slike. Prvo število predstavlja vrednost rdeče komponente prve točke, drugo vrednost zelene komponente prve točke, tretje modro komponento. Četrto število vsebuje rdeče komponento druge točke, in tako naprej. Za boljše razumevanje lahko uporabite prikaz v sliki 3.1.

Vrednosti komponent so omejene na nepredznačena osem-bitna števila.



Slika 3.1: Vizualizacija podatkov slike.

V sledeči tabeli 3.1 imamo izbor nekaterih EPICS zapisov, ki jih lahko uporabimo za komunikacijo z gonilnikom. Vsako ime zapisa je predznačeno z (P)(R). To je zato, ker so vsi zapisi v sistemu EPICS globalni. Če želimo več instanc istega gonilnika, mora imeti vsak zapis edinstveno ime. Vsaka instanca gonilnika ima drugačne vrednosti za makro $(P) in $(R). Če za makro uporabimo preslikavo

- P = "Oddelek1-,"
- R = "Kamera1-,"

se $($P$)$(R)*Acquire* spremeni v ime zapisa *Oddelek*1-*Kamera*1-*Acquire*.

Ime zapisa	Opis
(P)(R)Acquire	Zapišite 1 za začetek zajema slike. Zapišite 0 za nasilno prekinitev.
(P)(R)ColorMode	Nastavitev barvnega načina kamere. Možnosti so: • 'RASPICAM_FORMAT_YUV420', • 'RASPICAM_FORMAT_GRAY', • 'RASPICAM_FORMAT_BGR', • 'RASPICAM_FORMAT_RGB'.
(P)(R)ArrayCounter	Število zajetih slik.
(P)(R)SizeX_RBV	Širina zajete slike.
(P)(R)SizeY_RBV	Višina zajete slike.
(P)(R)EnableCallbacks	Omogoči zapis.
(P)(R)RESOLUTION	Izbira ločljivosti. Možne vrednosti so: • '1280x960', • '640x480', • '320x240'.

(P)(R)BRIGHTNESS	Svetlost slike. Vrednosti morajo biti v intervalu $[0, 100]$.
(P)(R)SHARPNESS	Ostrina slike. Vrednosti morajo biti v intervalu $[-100, 100]$.

Tabela 3.1: Nekaj EPICS napisov za nadzor kamere.

Prva stvar, ki jo lahko v tabeli opazite je, da so nekatera imena zapisana z velikimi začetnicami, druga imena pa uporabljajo izključno velike črke. Razlika je v tem, da sam vtičnik areaDetector uporablja stil poimenovanja *CamelCase*, medtem ko implementiran gonilnik uporablja stil poimenovanja z izključno velikimi črkami.

Kot drugo ste morda opazili, da nekatera polja sprejmejo numerično vrednost, medtem ko druga pričakujejo vrednost s seznama. To razliko lahko v EPICS-u uvedemo z uporabo zapisa *mbbo* za izbirna polja in uporabo zapisa *ao* za numerična polja.

Kot tretje pa lahko opazite, da se nekatera polja končajo z '_RBV'. Ta kratica pomeni *Read Back Value*. To pomeni, da so ta polja namenjena izključno branju. EPICS sicer dovoli, da nekdo v to polje zapiše drugo vrednost, ne bo pa ta vrednost vplivala na delovanje gonilnika [23].

3.1 Zgradba gonilnika

Zgradbo gonilnika v veliki meri določa vtičnik areaDetector. Kot prvo moramo v konstruktorju gonilnika definirati vse parametre kamere, do katerih lahko uporabnik dostopa preko EPICS zapisov. Nastaviti moramo tudi njihove prevzete vrednosti. Ta postopek je viden v izseku kode 3.1.

Ustvarimo novo programsko nit, ki bo zadolžena za zajem slike, saj traja zajem slike dalj časa od drugih operacij, med tem pa želimo, da je gonilnik še vedno odziven na druge zahtevke in spremembe. Da pa bo nova programska nit pametno čakala na ukaz za zajem slike, ustvarimo še objekt za sproženje

in čakanje na dogodke. V niti lahko tako preprosto čakamo na dogodek, namesto da neprestano preverjamo stanja EPICS zapisov. Koda za čakanje je prikazana v izseku kode 3.2.

```
1  /* Create parameters */
2  createParam(RasPiCamBrightnessString, asynParamInt32,
3              &RasPiCamBrightness);
4  createParam(RasPiCamSharpnessString, asynParamInt32,
5              &RasPiCamSharpness);
6  createParam(RasPiCamContrastString, asynParamInt32,
7              &RasPiCamContrast);
8  createParam(RasPiCamISOString, asynParamInt32,
9              &RasPiCamISO);
10 // ...
11
12 /* Set some default values for parameters */
13 status = setStringParam(ADModel, "Rasp Pi Camera");
14 status |= setIntegerParam(ADMaxSizeX, 1280);
15 status |= setIntegerParam(ADMaxSizeY, 960);
16 // ...
17
18 // set custom param defaults
19 status |= setIntegerParam(RasPiCamBrightness, 50);
20 status |= setIntegerParam(RasPiCamSharpness, 0);
21 status |= setIntegerParam(RasPiCamContrast, 0);
22 status |= setIntegerParam(RasPiCamISO, 400);
23 // ...
```

Izsek 3.1: Definicija parametrov in prevzetih vrednosti.

```
1  setStringParam(ADStatusMessage, "Waiting for signal");
2  status = epicsEventWait(this->startEventId);
```

Izsek 3.2: Čakanje na dogodek, ki naznani zajem slike.

Gonilnik ima dve metodi, ki se sprožita ob spremembi EPICS zapisov. Metodi se imenujeta "writeInt32", ko se spremeni zapis za cela števila, ter "writeFloat64", ko se spremeni zapis za realna števila. Metodi prejmeta podatek, kateri zapis se je spremenil, ter kakšna je nova vrednost. V metodah spremembe vrednosti zapisov uveljavimo na strojni opremi, kot je vidno v izseku 3.3. Pri večini zapisov se po spremembi prebere nova vrednost ravno nastavljenega parametra, da lahko uporabnik opazi, če se kakšen parameter ni pravilno nastavil, zaradi omejitev na strojni opremi.

```
1  if(function == ADAcquire){
2      if(value){
3          // send signal to capture image
4          epicsEventSignal(this->startEventId);
5      } else {
6          this->stopContinuous = 1;
7      }
8  } else if(function == NDColorMode){
9      Camera.setFormat(
10         static_cast<raspicam::RASPICAM_FORMAT>(value));
11     status = setIntegerParam(NDColorMode,
12         Camera.getFormat());
13
14 } else if(function == ADSizeX || function == ADSizeY){
15     // We don't allow the user to change
16     // SizeX and SizeY directly.
17     // Set values back to what they were.
18     setIntegerParam(ADSizeX, Camera.getWidth());
19     setIntegerParam(ADSizeY, Camera.getHeight());
20 } else if(function == RasPiCamResolution){
21     // set both width and height
22     if(value < RHigh || value > RLow){
23         // value out of range
```

```
24            status = asynError;
25        } else {
26            Camera.setWidth(resolutions[value][RWidth]);
27            Camera.setHeight(resolutions[value][RHeight]);
28            setIntegerParam(ADSizeX, Camera.getWidth());
29            setIntegerParam(ADSizeY, Camera.getHeight());
30        }
31    } else if(function == RasPiCamBrightness){
32        Camera.setBrightness(value);
33        status = setIntegerParam(RasPiCamBrightness,
34                    Camera.getBrightness());
35    } else if(function == RasPiCamSharpness){
36        Camera.setSharpness(value);
37        status = setIntegerParam(RasPiCamSharpness,
38                    Camera.getSharpness());
39    } else if(function == RasPiCamContrast){
40        Camera.setContrast(value);
41        status = setIntegerParam(RasPiCamContrast,
42                    Camera.getContrast());
43    } else if(function == RasPiCamISO){
44        Camera.setISO(value);
45        status = setIntegerParam(RasPiCamISO,
46                    Camera.getISO());
47    } else if(function == RasPiCamSaturation){
48        // ...
```

Izsek 3.3: Uveljavitev novih nastavitev na strojni opremi.

3.2 Potek razvoja

Razvoj se je začel s tem, da smo izbrali nekaj parametrov in jih poskusili spremeniti, da bi ugotovili, da se asinhrone metode v gonilniku uspešno zaženejo. Ko smo ugotovili, da ta del deluje, smo dodali še ostale parametre.

Razvoj niti za zajem slike je bil najbolj zahteven. Prva težava je bila, da dokumentacija vtičnika areaDetector ni lahko berljiva. Nismo našli nekaterih podrobnosti, kot na primer v kakšni obliki mora biti slika zapisana v EPICS zapis. Vtičnik vsebuje metodo za pretvorbo slikovnih podatkov, ni pa jasno, kako se to metodo uporabi.

Ker ob razvoju gonilnika še nismo imeli spletnega grafičnega vmesnika, smo morali uporabiti drug program za prikaz slike. Najprej smo poskusili program ImageJ [21]. Program je ves čas prikazoval črno sliko, zaradi česar smo predvidevali, da se slika ne zapiše pravilno. Z uporabo orodja za izpis surovih podatkov smo videli, da se slikovni podatki pravilno prenesejo. Za hitri preizkus pravilnosti podatkov smo kamero prekrili z roko, nakar so se vrednosti slikovnih podatkov spustile na nič. To pomeni, da je slika postala črna. Napisali smo Python skripto za prenos podatkov in pretvorbo podatkov v sliko, ter tako ugotovili, da lahko uspešno zajamemo sliko. Prišli smo do sklepa, da je program ImageJ izvor težave. Ker smo že od začetka imeli namen zgraditi svoj prikazovalnik slik, smo se odločili, da ne bomo iskali natančnega razloga za to napako, saj programa ne bomo uporabljali, porabili bi pa lahko zelo veliko časa.

Poglavje 4

Spletni uporabniški vmesnik

Spletni uporabniški vmesnik omogoča uporabniku, da si ogleda zajeto sliko in spreminja nastavitve kamere. Zajeta slika se najprej shrani v zapis sistema EPICS. Ta zapis ob zahtevi uporabnika programski vmesnik prebere, pretvori v format JPEG in pošlje spletnemu brskalniku. Poglavje o grafičnem vmesniku je razdeljeno na dva dela: programski vmesnik in spletno stran.

4.1 Programski vmesnik

Zahvaljujoč uporabi ogrodja Flask je programski vmesnik izjemno preprost. Dve metodi, razvidni v izseku 4.1, omogočata enostaven dostop do EPICS zapisov.

```
1 @app.route('/caget/<record>/')
2 def api_caget(record):
3     return str(caget(record))
4
5 @app.route('/caput/<record>/<value>/')
6 def api_caput(record, value):
7     return str(caput(record, value))
```

Izsek 4.1: Preprosti metodi za dostop do EPICS zapisov.

Metodi *caget* in *caput*, ki sta vsebovani v vtičniku PyEpics, se uporabljata za branje in pisanje v zapise EPICS.

Programski vmesnik mora najprej prebrati slikovne podatke in jih shraniti v slikovni objekt. Sliko se nato pretvori v format JPEG in vrne odjemalcu. Celotni postopek je razviden v izseku 4.2.

```
    """ Load image from camera """

    res = caget("RASPICAM1:cam1:RESOLUTION_RBV")
    if res == 1:
        size_x = 640
        size_y = 480
    elif res == 2:
        size_x = 320
        size_y = 240
    else:
        size_x = 1280
        size_y = 960

    # read array data into image
    imageRecord = "RASPICAM1:image1:ArrayData"
    img = Image.frombytes('RGB',
                          (size_x, size_y),
                          caget(imageRecord))

    # convert PIL image into JPEG
    byte_io = BytesIO()
    img.save(byte_io, 'JPEG')
    byte_io.seek(0)

    # send png to client
    return send_file(byte_io, mimetype='image/jpeg')

@app.route('/info/', methods=['POST'])
```

Izsek 4.2: Prenos slikovnih podatkov in pretvorba v sliko.

Preden so na voljo ažurni slikovni podatki, jih je potrebno najprej zajeti. Zato smo napisali metodo, predstavljeno v izseku 4.3, ki pošlje ukaz za zajem novih slikovnih podatkov [24].

```
1  @app.route('/acquire/')
2  def acquire_photo():
3      """ Call acquire and wait for new image """
4      numRecord = 'RASPICAM1:cam1:ArrayCounter_RBV'
5      prev_num = caget(numRecord)
6      caput('RASPICAM1:cam1:Acquire', 1)
7      while prev_num == caget(numRecord):
8          # wait for the number to change
9          sleep(0.2)
10     return str(caget(numRecord))
```

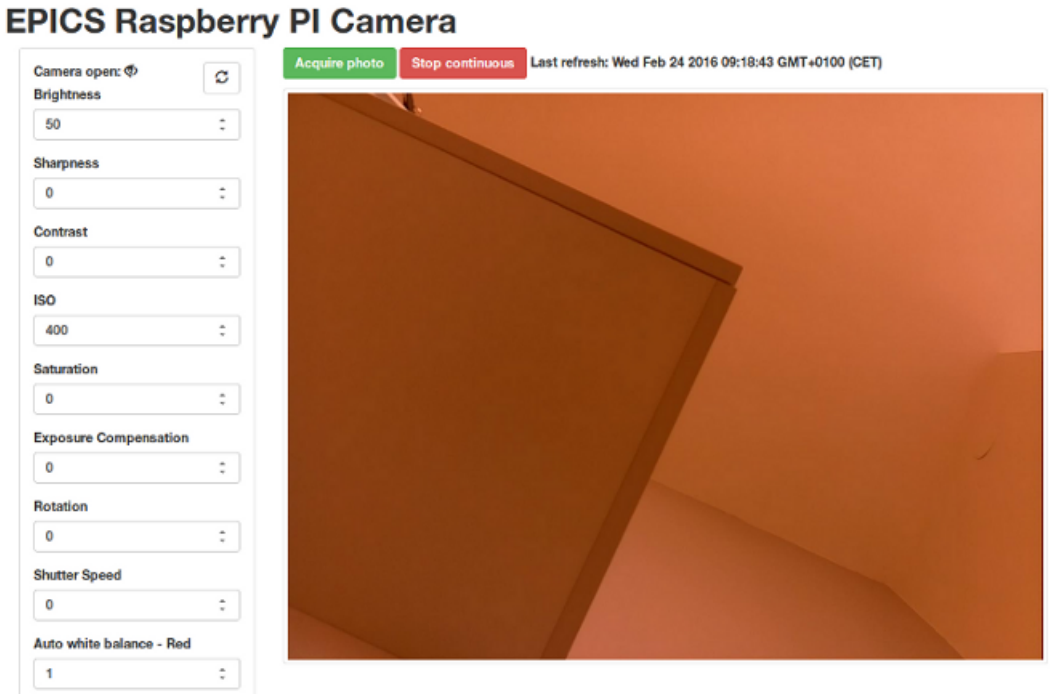
Izsek 4.3: Metoda za zajem slike.

4.2 Spletna stran

Spletna stran, vidna na sliki 4.1, je glavni način uporabe kamere, zato je morala biti stran vizualno privlačna in enostavna za uporabo. Za vizualni izgled je bila uporabljena knjižnica Bootstrap [11]. V modernih aplikacijah je pomembna zanesljivost in hitrost, zato sem uporabil arhitekturo MVC, ki jo nudi spletno ogrodje AngularJS.

```
1  <div class='form-group' ng-repeat='row in integerFields'>
2      <label for='{{row.name}}'>{{row.title}}</label>
3      <input type='number' ng-change='updateSetting(row)'
4             class='form-control' id='{{row.name}}'
5             min='{{row.min}}' max='{{row.max}}'
6             step='{{row.step}}'
7             ng-model='row.value'
8             ng-disabled="{{row.disabled}}">
9  </div>
```

Izsek 4.4: Uporaba AngularJS in Bootstrap.



Slika 4.1: Spletni uporabniški vmesnik.

Oglejmo si primer uporabe AngularJS in Bootstrap v izseku 4.4. Razčlenimo posamezne dele tega primera (omejimo se na specifike ogrodja AngularJS in izpustimo osnovne HTML oznake):

class HTML element *div* se začne z atributom *class*, ki vsebuje vrednost ‘form-group‘. Ta del uporabi izgled skupine obrazcev.

ng-repeat Ta atribut določa, da se bo element *div* ponovil za vsak element v naboru ‘integerFields‘.

ng-change Ob vsaki spremembi tega vnosnega polja, se bo sprožila metoda ‘updateSettings‘.

ng-model Ta atribut določa, katera spremenljivka se bo vezala na to vnosno polje. Ko se spremeni spremenljivka, se posodobi pogled, ko se spremeni pogled, se posodobi spremenljivka.

ng-disabled Ko je ta atribut 'true', je vnosno polje onemogočeno.

Uporaba ogrodja nam je omogočila, da razčlenimo posamezne dele spletne aplikacije. Tako je nastala jasna meja med tem, kateri del kode skrbi za izgled in kateri za upravljanje s podatki.

Upravljanje s podatki (ang. *controller*) je razvidno v izseku 4.5.

```
1  $scope.updateSetting = function(row){
2      // send data with ajax
3      $http.get("/caput/" + row.record + "/"
4                              + row.value + "/")
5              .success(function(data){
6          // done
7      });
8  }
```

Izsek 4.5: Upravljanje s podatki.

Izsek kode se uporablja za sporočanje sprememb EPICS-u.

Poglavje 5

Sklepne ugotovitve

Z uporabo sistema EPICS in vtičnika areaDetector nam je uspelo implementirati ustrezen gonilnik in s tem ustvariti zanesljiv in modularen sistem za zajem slike. Ker je trenutno Raspberry Pi zelo popularna rešitev za mnoge uporabe in ponuja zelo veliko za nizko ceno, sklepamo, da bodo številne organizacije želele uporabiti to napravo za vizualno spremljavo eksperimentov ali procesov v industriji.

Tipično EPICS teče na klasičnih strežnikih, ki lahko stanejo par tisoč evrov. Robustne industrijske kamere lahko stanejo podobno količino denarja. Naša rešitev je na voljo že za približno 50 evrov.

Prihranili smo tudi čas drugim razvijalcem, saj jim ni potrebno pisati gonilnikov na novo. Razvit gonilnik lahko hitro in enostavno vključijo v svoj projekt.

Z uporabo Flask in AngularJS smo pokazali, da obstajajo alternative obstoječim grafičnim vmesnikom za EPICS. Grafični vmesnik je tako lahko bolj vizualno privlačen in se ga da uporabljati na več različnih napravah, tudi mobilnih.

V prihodnosti bi lahko izboljšali zajem videa. Trenutno deluje prenos z največ okoli petimi slikami na sekundo, kar včasih ni dovolj. Uporabili bi lahko postopek kompresije video posnetkov H.264.

Literatura

[1] JJ Di Steffano, AR Stubberud, IJ Williams, "Feedback and control systems", Schaums outline series, McGraw-Hill 1967, pogl. 1.1.

[2] Raspberry Pi. [Online]. Dosegljivo: https://www.raspberrypi.org/. [Dostopano 15. 1. 2017].

[3] Raspberry Pi NoIR Camera. [Online]. Dosegljivo: https://www.raspberrypi.org/products/pi-noir-camera-v2/. [Dostopano 15. 1. 2017].

[4] EPICS Users. [Online]. Dosegljivo: http://www.aps.anl.gov/epics/sites.php. [Dostopano 7. 1. 2017].

[5] EPICS - Experimental Physics and Industrial Control System. [Online]. Dosegljivo: http://www.aps.anl.gov/epics/. [Dostopano 15. 1. 2017].

[6] What's that blue thing doing here? - Raspberry Pi. [Online]. Dosegljivo: https://www.raspberrypi.org/blog/whats-that-blue-thing-doing-here/. [Dostopano 7. 1. 2017].

[7] areaDetector: EPICS software for area detectors. [Online]. Dosegljivo: http://cars.uchicago.edu/software/epics/areaDetector.html. [Dostopano 15. 1. 2017].

[8] asynDriver. [Online]. Dosegljivo: http://www.aps.anl.gov/epics/modules/soft/asyn/. [Dostopano 15. 1. 2017].

[9] jQuery. [Online]. Dosegljivo: https://jquery.com/. [Dostopano 15. 1. 2017].

[10] AngularJS: Developer Guide: Introduction. [Online]. Dosegljivo: https://docs.angularjs.org/guide/introduction. [Dostopano 15. 1. 2017].

[11] CSS · Bootstrap. [Online]. Dosegljivo: http://getbootstrap.com/css/#overview. [Dostopano 15. 1. 2017].

[12] Welcome to Python.org. [Online]. Dosegljivo: https://www.python.org/. [Dostopano 15. 1. 2017].

[13] Flask (A Python Microframework). [Online]. Dosegljivo: http://flask.pocoo.org/. [Dostopano 15. 1. 2017].

[14] Welcome — Werkzeug (The Python WSGI Utility Library). [Online]. Dosegljivo: http://werkzeug.pocoo.org/. [Dostopano 15. 1. 2017].

[15] Welcome — Jinja2 (The Python Template Engine). [Online]. Dosegljivo: http://jinja.pocoo.org/. [Dostopano 15. 1. 2017].

[16] Flask - Extensions. [Online]. Dosegljivo: http://flask.pocoo.org/extensions/. [Dostopano 15. 1. 2017].

[17] Epics Channel Access for Python. [Online]. Dosegljivo: http://cars9.uchicago.edu/software/python/pyepics3/. [Dostopano 15. 1. 2017].

[18] ADExample. [Online]. Dosegljivo: https://github.com/areaDetector/ADExample. [Dostopano 15. 1. 2017].

[19] Home — Durandal. [Online]. Dosegljivo:
http://durandaljs.com/. [Dostopano 15. 1. 2017].

[20] Ember.js - A framework for creating ambitious web applications. Dosegljivo:
http://emberjs.com/. [Dostopano 15. 1. 2017].

[21] ImageJ. Dosegljivo:
https://imagej.nih.gov/ij/. [Dostopano 15. 1. 2017].

[22] EPICS R3.14 Channel Access Reference Manual. Dosegljivo:
http://www.aps.anl.gov/epics/base/R3-14/8-docs/CAref.htmlcaget.
[Dostopano 15. 1. 2017].

[23] Domen Soklič, "Raspberry Pi Camera - Design Document", CSL-DES-16-128506, Cosylab, 2016.

[24] Domen Soklič, "Control Sheet 27 (June 2016)", Cosylab, 2016, str. 6-8.

[25] Apache License, Version 2.0. [Online]. Dosegljivo:
http://www.apache.org/licenses/LICENSE-2.0. [Dostopano 15. 1. 2017].