



## Uvod

U okviru ove laboratorijske vježbe ćemo se upoznati sa još nekim alatima i sistemima koji će se koristiti.

Razvojni alati koje ćemo koristiti su:

- Online Mbed simulator,
- ARM Keil Studio (cloud verzija).

Kao razvojne sisteme (hardver) ćemo koristiti:

- LPC1114ETF, sistem baziran na mikrokontroleru NXP LPC1114FN28.
- Freescale FRDM-KL25Z, sistem baziran na mikrokontroleru Freescale KL25Z128VLK.

U nastavku će biti data pojašnjenja i svi koraci neophodni za kompajliranje i izvršavanje aplikacija korištenjem pomenutih okruženja, a za navedene sisteme.

## Razvojni alati

### Mbed simulator

Online Mbed simulator predstavlja alat koji se izvršava u prozoru browser-a, a koji do određene mjere oponaša izvršavanje aplikacija pisanih uz korištenje Mbed OS-a, za mikrokontrolere sa nekim od ARM Cortex-M jezgri. Raspoloživi resursi u Mbed simulatoru odgovaraju hardverskom sistemu sa mikrokontrolerom LPC1768, ali se razvijene aplikacije relativno jednostavno mogu prilagoditi i svakom drugom sistemu.

Osim simuliranog sistema, u okviru simulatora se mogu dodavati i različite vanjske komponente kao što su: LED diode, tasteri, potenciometri, displeji i slično. Detaljnije se o Mbed simulatoru može pročitati na linku <https://os.mbed.com/blog/entry/introducing-mbed-simulator/>.

Mbed simulatoru se može pristupiti putem linka <http://mbed.webredirect.org> (hostan na serveru na ET). Ovaj simulator omogućava testiranje kodova koji koriste Mbed OS, ali treba imati na umu da su njegove mogućnosti ograničene kako obimom implementacije (postoje elementi koji nisu u potpunosti implementirani), tako i činjenicom da se cjelokupna interakcija sa simuliranim sistemom odvija putem browsera uz korištenje Javascript-a. Radi toga ponašanje Mbed simulatora ne odgovara u potpunosti ponašanju stvarnog hardvera koji koristi Mbed OS. Tako npr. nisu moguće akcije kao što je pritisak i držanje tastera, nego jedino klik na taster.

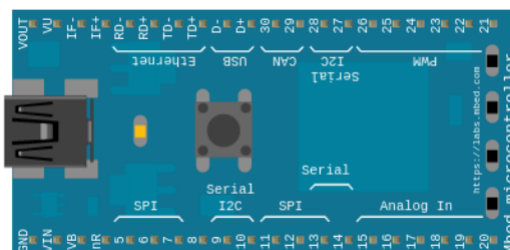
## Napomena

Treba voditi računa da Mbed simulator ne emulira stvarno izvršavanje kompajliranog koda na ARM sistemu, nego oponaša Mbed korištenjem Javascript-a. Zbog toga postoje odstupanja između efekata izvršavanja koda na Mbed simulatoru i na stvarnom sistemu. Ipak, Mbed simulator ima zadovoljavajuće ponašanje za potrebe laboratorijskih vježbi na Ugradbenim sistemima, tako da ga studenti mogu koristiti za pripremu svojih rješenja kod kuće.

## Hint

Ukoliko je potrebno, pritisak na taster se može simulirati tako što se klikne dugme miša na tasteru, a onda se pointer miša skloni sa tastera, dok se još dugme drži pritisnuto. Otpuštanje se može simulirati tako što se klikne dugme miša izvan tastera, onda se pointer dovede na taster dok se dugme drži pritisnuto i nakon toga se na tasteru otpusti.

U okviru Mbed simulatora se aplikacije izvršavaju na simuliranom sistemu predstavljenom na slici 1.



Slika 1: Izgled virtualnog sistema u Mbed simulatoru.

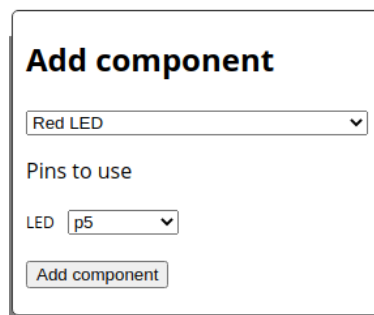
Ovaj virtualni sistem ima ugrađene komponente:

- četiri LED (LED1, LED2, LED3, LED4),
- jedan taster (BUTTON1).

Virtualni sistem ima ukupno 40 pinova, od kojih se dio ne može koristiti, nego predstavljaju pinove za napajanje i sl. Svi pinovi koji se mogu koristiti u okviru Mbed aplikacije su numerisani i nose oznake p5, p6, p7, itd. Ove oznake se koriste za referenciranje na željene pinove u okviru Mbed aplikacije. Osim ugrađenih komponenti, Mbed aplikacija može koristiti i vanjske komponente, koje se mogu dodavati na dozvoljene pinove. Te vanjske komponente su:

- LED diode (Red LED, Blue LED, Yellow LED, White LED);
- Tasteri (Button);
- Termistori (Analog Thermistor);
- Senzor temperature i vlažnosti (SHT31 temperature/humidity sensor);
- LCD displej (C12832 LCD display);
- RGB displej osjetljiv na dodir (ST7789H2 LCD + FT6x06 Touch Screen).

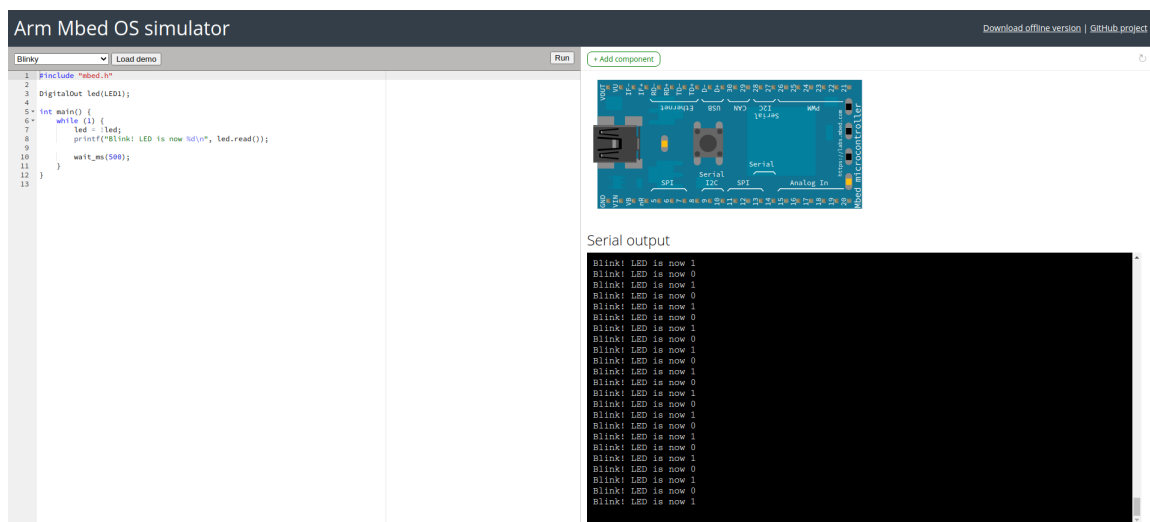
Ove komponente se dodaju klikom na dugme + Add component u desnom dijelu prozora simulatora. Svaka od pomenutih komponenti se može dodati na neke od raspoloživih pinova, a odabir pinova je ograničen u dijalogu za dodavanje komponente (slika 2):



Slika 2: Dijalog za dodavanje komponenti.

Mbed simulator se može koristiti za testiranje Mbed aplikacija, ali treba voditi računa da simulator ne može izvršavati cjelokupne aplikacije napisane za konkretnu platformu korištenjem razvojnih alata kao što je ARM Keil Studio ili sličan. Bez obzira na to, simulator je dovoljno funkcionalan da sasvim dobro ilustrira ponašanje klasa Mbed OS-a, te se uz malo vještine u korištenju, simulatorom mogu testirati prilično složene aplikacije. Prozor Mbed simulatora (slika 3) je podijeljen u dva osnovna dijela:

- dio za unos programskog koda;
- dio za prikaz efekata izvršavanja koda i interakciju sa virtualnim sistemom.



Slika 3: Izgled prozora Mbed simulatora.

U lijevom dijelu prozora se na uobičajen način može pisati kod, koji se kompajlira klikom na dugme **Run**. Ukoliko je aplikacija ispravno kompajlirana, započeti će njeno izvršavanje. Ukoliko kompajler otkrije grešku, poruke o razlogu prekida kompajliranja će biti prikazane u desnom dijelu prozora. Nakon toga je potrebno ispraviti kod, te ponoviti postupak kompajliranja.

### Napomena

Treba voditi računa da u okviru `while(1)` petlje mora postojati poziv funkcije `wait`, `wait_ms` ili `wait_us`, inače će aplikacija u potpunosti blokirati browser.

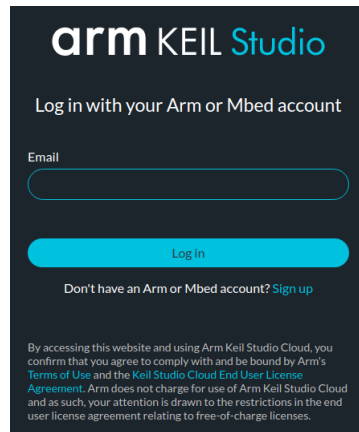
U desnom dijelu prozora se nalazi prikaz virtualnog sistema sa dodatnim komponentama, te prikaz terminala koji se može koristiti za ispis iz aplikacije. Ukoliko Mbed aplikacija radi sa ugrađenim komponentama virtualnog sistema, efekat izvršavanja aplikacije će biti vidljiv. Ukoliko aplikacija koristi i druge pinove virtualnog sistema, prikaz je moguć samo ukoliko su

odgovarajuće komponente dodane na prethodno opisani način. Bez obzira da li su vanjske komponente dodane, aplikacija će se izvršavati ukoliko je ispravno kompajlirana (s tim da efekti izvršavanja neće biti vidljivi).

U okviru Mbed simulatora postoji i određeni broj primjera koji mogu pomoći pri korištenju različitih klasa Mbed-a, a primjeri se učitavaju odabirom u lijevom gornjem dijelu prozora. Kada se primjer učitava, potrebno ga je kompajlirati na opisani način.

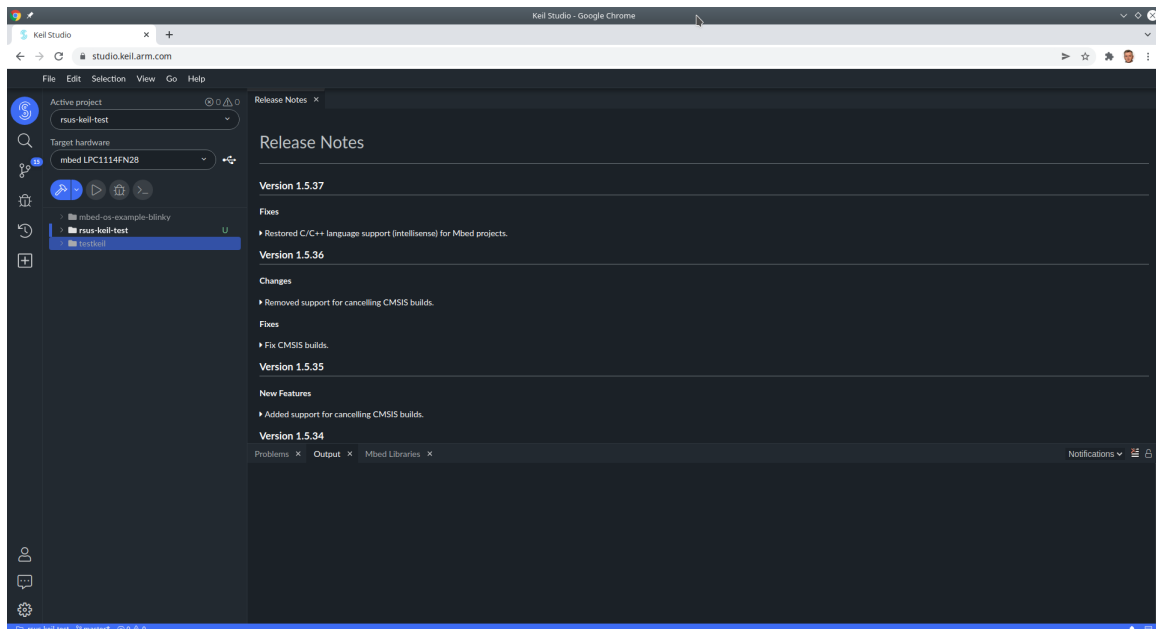
## ARM Keil Studio

Za razvoj i kompajliranje aplikacija za razvojne sisteme (stvarni hardver) ćemo koristiti cloud verziju razvojnog alata ARM Keil Studio. Forma za prijavu na ovaj razvojni sistem se nalazi na linku <https://studio.keil.arm.com/auth/login/> (slika 4), a svaki student će trebati kreirati korisnički račun na sistemu.



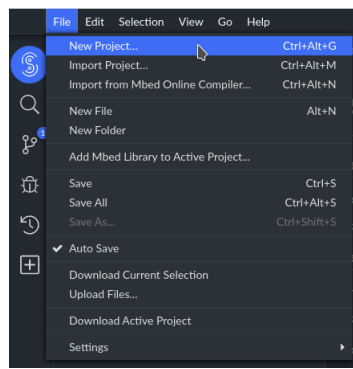
Slika 4: Forma za prijavu na ARM Keil Studio.

Nakon uspješne prijave na ARM Keil Studio, bit će prikazan prozor razvojnog okruženja, kao na slici 5.



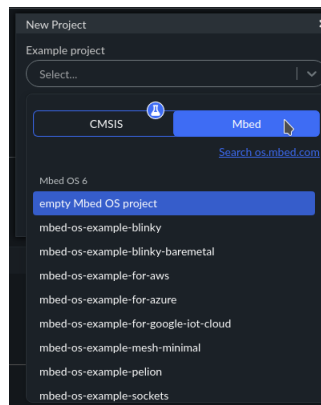
Slika 5: Glavni prozor okruženja ARM Keil Studio.

Novi projekat se kreira odabirom opcije menija File->New Project (slika 6).

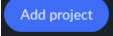


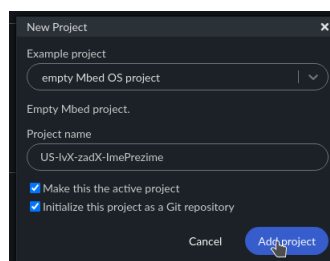
Slika 6: Kreiranje novog projekta.

Pri kreiranju novog projekta je potrebno odabrati predložak za novi projekat (slika 7), pri čemu se obavezno treba odabrati neki od predložaka za Mbed OS. Obzirom da ćemo uglavnom koristiti kod za Mbed OS, koji je razvijen u okviru Mbed simulatora, preporučuje se odabir predloška "mbed-os-example-blinky-baremetal". Ukoliko neki drugi predložak bolje odgovara, moguće ga je koristiti.



Slika 7: Odabir predloška za novi projekat.

Projektu je potrebno dodijeliti ime i kliknuti na dugme  (slika 8).

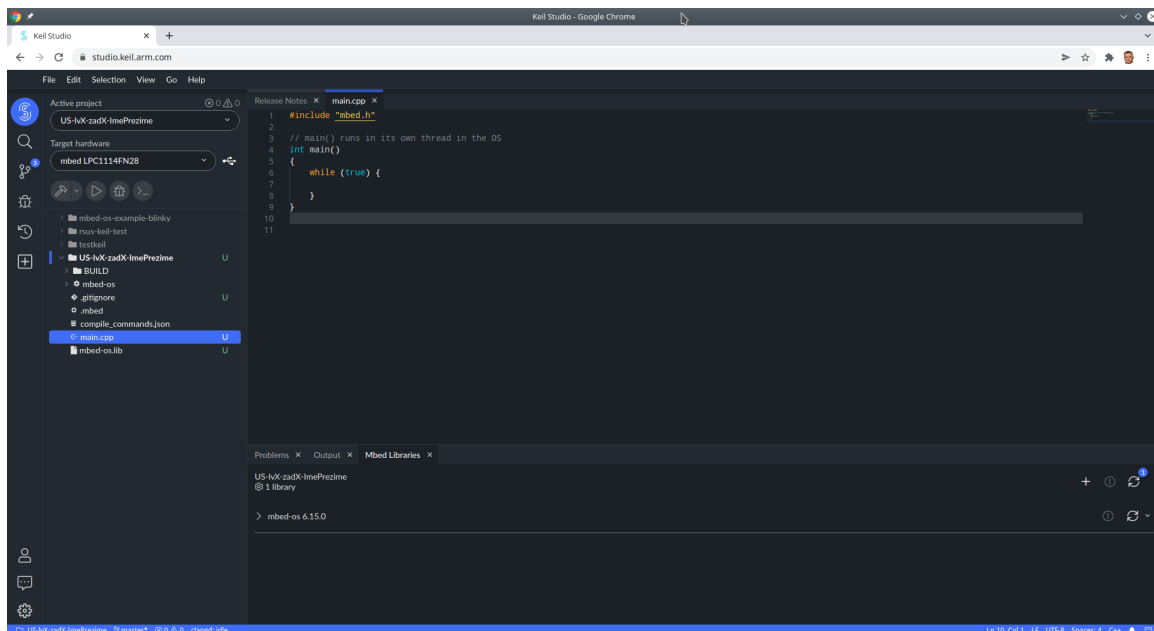


Slika 8: Dodjela imena i kreiranje projekta.

### Napomena

Radi jednostavnijeg snalaženja prilikom slanja i pregledanja rješenja zadataka, projekte obavezno imenovati kao `US-lvX-zadX-ImePrezime`.

Nakon što je projekat kreiran, potrebno je otvoriti fajl `main.cpp` (slika 9), nakon čega se u prozoru editora može uređivati programski kôd.

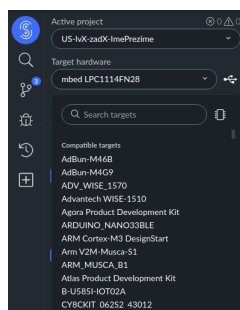


Slika 9: Programski kod se piše u fajlu `main.cpp`, korištenjem prozora editora.

### Hint

Ukoliko je programski kôd rješenja već razvijen u okviru Mbed simulatora, može se prekopirati u prozor editora. Pri tome je **obavezno** potrebno adekvatno preimenovati imena objekata (ulazi, izlazi i ostalo), u skladu sa imenima resursa na konkretnom razvojnom sistemu za koji će se kôd kompajlirati. Isto tako, u novijim verzijama Mbed OS-a ne postoji funkcija `wait`, nego treba koristiti `wait_us` ili `wait_ns`.


Prije kompajliranja programskog kôda je potrebno odabrati razvojni sistem (hardversku platformu) za koju se kôd kompajlira (slika 10).



Slika 10: Odabir razvojnog sistema za koji se kôd kompajlira.

Na laboratorijskim vježbama ćemo koristiti sisteme:

- mbed LPC1114FN28 - za razvojni sistem LPC1114ETF,
- FRDM-KL25Z.

Nakon odabira željenog razvojnog sistema, kompajliranje se pokreće klikom na dugme . Ukoliko se proces kompajliranja završi uspješno, fajl sa izvršnim kodom će biti preuzet na lokalni disk. Ovaj fajl je potrebno prebaciti (uploadirati) na razvojni sistem za koji je kompajliran, pri čemu postupak prebacivanja fajla ovisi o kojem sistemu je riječ.

# Razvojni sistemi

Izvršni kôd aplikacije će se izvršavati na konkretnom mikrokontroleru, pri čemu treba voditi računa da je kôd kompajliran za mikrokontroler na koji se fajl sa izvršnim kodom prebacuje. Ovo je neophodno pošto različiti mikrokontroleri raspolažu različitim resursima, te se aplikacija neće ispravno izvršavati ukoliko neki od resursa nije raspoloživ na mikrokontroleru (bez obzira što jezgro mikrokontrolera može biti identično).

Kako je već napomenuto, na laboratorijskim vježbama ćemo koristiti razvojne sisteme:

- LPC1114ETF, koji ima jezgro ARM Cortex-M0,
- FRDM-KL25Z, koji ima jezgro ARM Cortex-M0+.

Iako su jezgra ova dva mikrokontrolera gotovo identična, raspoloživi resursi (hardverski moduli, broj pinova, vrste ulaza i izlaza, dodatne komponente) se do određene mjere razlikuju. Isto tako, različit je i način prebacivanja (uploadiranja) fajla sa izvršnim kodom na svaki od ovih razvojnih sistema.

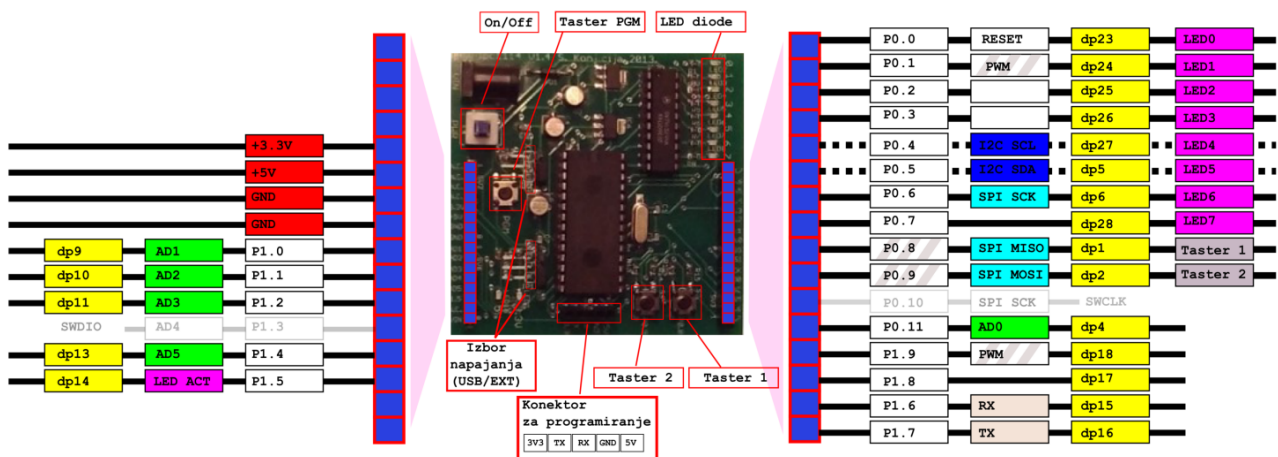
## LPC1114ETF

### Osnovne karakteristike

Razvojni sistem LPC1114ETF je namjenski dizajniran za potrebe laboratorijskih vježbi na predmetu Ugradbeni sistemi, a na njemu se nalazi:

- mikrokontroler LPC1114FN28,
- 8 LED dioda,
- 2 tastera,
- headeri za povezivanje eksternih komponenti.

Raspored funkcija na pojedinačne pinove (izvode na headerima) ovog razvojnog sistema je prikazan na slici 11.



Slika 11: Raspored funkcija po izvodima na razvojnem sistemu LPC1114ETF.

Nazivi izvoda i funkcija su deklarirani u fajlu `lpc1114etf.h`. Radi lakšeg korištenja izvoda i funkcija (u skladu sa nazivima kao na slici) je potrebno ovaj header fajl uploadirati na projekat i uključiti u okviru fajla `main.cpp`. Ukoliko se ovaj fajl ne uključi, onda je moguće koristiti

jedino imena pinova (dp9, dp10 itd.). Međutim, i tada je neophodno definirati ime pina dp23, dodavanjem u `main.cpp` linije koda:

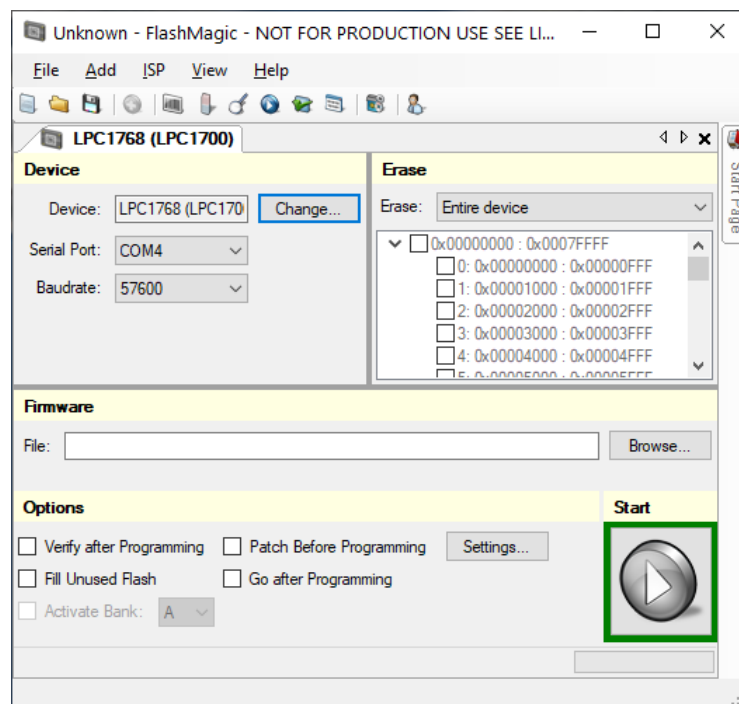
```
// Dodati liniju ispod u fajl main.cpp!  
  
# define dp23 P0_0
```

### Hint

Ukoliko se u projekat uključi header fajl `lpc1114etf.h`, u programskom kôdu je moguće koristiti nazive funkcija u skladu sa nazivima navedenim na slici 11.

## Način programiranja sistema LPC1114ETF

Da bi se aplikacija mogla izvršavati na razvojnom sistemu LPC1114ETF, neophodno je u programsku memoriju mikrokontrolera ovog sistema upisati izvršni kôd aplikacije. Ovo se postiže prebacivanjem sadržaja fajla sa izvršnim kôdom na razvojni sistem pomoću odgovarajuće aplikacije za upload. Postoji više takvih aplikacija, pri čemu neke imaju grafički interface, dok se druge koriste preko komandnog prompta. Primjer aplikacije sa GUI je Flash Magic (<https://www.flashmagictool.com/>), a izgled glavnog prozora ove aplikacije je prikazan na slici 12.



Slika 12: Glavni prozor aplikacije Flash Magic.

U laboratoriji ćemo koristiti konzolnu aplikaciju za Linux `lpc21isp`, koja nema GUI i koristi se putem komandnog prompta. Ova aplikacija se jednostavno instalira sa zvaničnih repozitorija svih uobičajenih Linux distribucija, a može se preuzeti i sa linka <https://sourceforge.net/p/lpc21isp/wiki/Home/>.



Bez obzir o kojoj aplikaciji za prebacivanje fajla sa izvornim kodom se radi, sadržaj fajla se putem fizičkog ili virtualnog serijskog porta prenosi na razvojni sistem i upisuje u programsku memoriju mikrokontrolera.

Opcije pri pozivu aplikacije `lpc21isp` su pojašnjene u help tekstu:

```
Portable command line ISP
for NXP LPC family and Analog Devices ADUC 70xx
Version 1.97 compiled for Linux: Feb 10 2020, 13:04:50
Copyright (c) by Martin Maurer, 2003-2013, Email: Martin.Maurer(at)libb.de
Portions Copyright (c) by Aeolus Development 2004, www.aeolusdevelopment.com

Syntax: lpc21isp [Options] file[ file[ ...]] comport baudrate Oscillator_in_kHz

Example: lpc21isp test.hex com1 115200 14746

Options: -bin          for uploading binary file
        -hex          for uploading file in intel hex format (default)
        -term         for starting terminal after upload
        -termonly     for starting terminal without an upload
        -localecho    for local echo in terminal
        -detectonly   detect only used LPC chiptype (NXPARM only)
        -debug0       for no debug
        -debug3       for progress info only
        -debug5       for full debug
        -donotstart   do not start MCU after download
        -try<n>       try n times to synchronise
        -wipe         Erase entire device before upload
        -control      for controlling RS232 lines for easier booting
                      (Reset = DTR, EnableBootLoader = RTS)
        -boothold     hold EnableBootLoader asserted throughout sequence
        -controlswap  swap RS232 control lines
                      (Reset = RTS, EnableBootLoader = DTR)
        -controlinv   Invert state of RTS & DTR
                      (0=true/assert/set, 1=false/deassert/clear).
        -verify       Verify the data in Flash after every writes to
                      sector. To detect errors in writing to Flash ROM
        -logfile      for enabling logging of terminal output to lpc21isp.log
        -halfduplex   use halfduplex serial communication (i.e. with K-Line)
        -writelay     Add delay after serial port writes (for compatibility)
        -ADARM        for downloading to an Analog Devices
                      ARM microcontroller ADUC70xx
        -NXPARM       for downloading to a chip of NXP LPC family (default)
```

Sadržaj fajla sa izvornim kôdom može biti zapisan u različitim formatima, a dva osnovna su:

- binarni format,
- Intel HEX format.

Obzirom da će sadržaj fajla preuzet nakon kompajliranja korištenjem razvojnog okruženja ARM Keil Studio biti zapisan u binarnom formatu, prilikom poziva aplikacije `lpc21isp` će

biti potrebno specificirati da se radi o binarnom formatu. Isto tako, bit će potrebno navesti ime fajla sa izvršnim kodom, te brzinu komunikacije i frekvenciju (u kHz) oscilatora razvojnog sistema. Radi toga će poziv aplikacije `lpc21isp` biti:

```
lpc21isp -bin ime_fajla.bin /dev/ttyUSB0 115200 12000
```

### Napomena

Na Linux operativnom sistemu virtualni serijski portovi preko USB imaju nazive `tttyUSB0`, `tttyUSB1` itd. Svaki serijski port ima svoj fajl u folderu `/dev`. Radi toga pri pozivu `lpc21isp` navodimo punu stazu do fajla virtualnog serijskog porta preko koga se ostvaruje komunikacija sa razvojnim sistemom.

Međutim, da bi razvojni sistem započeo komunikaciju sa aplikacijom `lpc21isp`, potrebno ga je prebaciti u *boot* način rada (odnosno pokrenuti *bootloader* na mikrokontroleru). Ovo se postiže tako da se drži pritisnut taster PGM (slika 11) prilikom uključanja napajanja razvojnog sistema LPC1114ETF. Statusne poruke i rezultat izvršavanja aplikacije će biti ispisani na konzoli. Ukoliko je prebacivanje izvršnog koda bilo uspješno, razvojni sistem će preći iz *boot* načina radu u normalni način rada i započeti će se izvršavanje aplikacije.

## FRDM-KL25Z

### Osnovne karakteristike

Razvojni sistem FRDM-KL25Z je jedan od demonstracijskih razvojnih sistema kompanije Freescale za njihove mikrokontrolere, a na njemu se nalazi:

- mikrokontroler KL25Z128VLK,
- jedna RGB LED dioda,
- kapacitivni slider,
- akcelerometar,
- headeri za povezivanje eksternih komponenti.

Raspored funkcija na pojedinačne pinove (izvode na headerima ovog razvojnog sistema je prikazan na slici 13.



## Zadatak 1

### Mbed simulator

Napisati i izvršiti sljedeći program na Mbed simulatoru:

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(1);
        myled = 0;
        wait(1);
    }
}
```

Nakon toga modificirati kôd tako da se pale i gase redom LED1, LED2, LED3 i LED4, na način da se prvo upali LED1 u trajanju od 1s, zatim ugasi LED1 u trajanju od 1s. Nakon toga se upali LED2 u trajanju od 1s i ugasi LED2 u trajanju od 1s, itd.

### LPC1114ETF

Izvršiti obje verzije programa i na razvojnom sistemu LPC1114ETF.

#### Napomena

Voditi računa da je za prikaz na LED diodama na razvojnom sistemu LPC1114ETF potrebno pin LED\_ACT postaviti na 0V, inače su pinovi mikrokontrolera povezani na header.

## Zadatak 2

### Mbed simulator

Dodati 8 LED dioda i korištenjem tih dioda realizovati binarni brojač, koji se uvećava za jedan svake sekunde. Svakim klikom na BUTTON1 smjer brojanja se treba mijenjati. Modificirati kod takoda brojač broji unazad kada je BUTTON1 pritisnut (obratiti pažnju na napomenu o načinu simuliranja pritisnutog tastera).

#### Hint

Umjesto pojedinačnih digitalnih izlaza, za rješavanje ovog zadatka je pogodnije koristiti podršku za višebitne digitalne izlaze u Mbed-u.

### LPC1114ETF

Isti program izvršiti na razvojnom sistemu LPC1114ETF.

## Napomena

Voditi računa da je potrebno preimenovati nazive LED dioda, u skladu sa nazivima navedenim na slici 11.

### Zadatak 3

#### Mbed simulator

Korištenjem 8 LED dioda iz prethodnog zadatka realizovati “trčeće svjetlo” na LED diodama. Prvo se uključi prva dioda, zatim druga dioda (a prva se isključi) i tako redom. Dolaskom na kraj (do 8. diode), potrebno je uključiti sve LED, te onda realizirati trčeće svjetlo u obrnutom smjeru gašenjem LED. Brzina izmjene je 0.1s.

#### LPC1114ETF

Isti program izvršiti na razvojnom sistemu LPC1114ETF.

### Zadatak 4

#### Mbed simulator

Modificirati prethodni zadatak tako da se nakon pokretanja sistema svakih 0.5s pali i gasi LED0, što označava da se čeka pritisak na taster. Ako se pritisne **Taster\_1**, aktivira se trčeće svjetlo kao u prethodnom zadatku, sa vremenom izmjene od 0.1s. Ukoliko se pritisne **Taster\_2**, tada je vrijeme izmjene 0.5s. Nakon što se završi jedan ciklus, ponovo se počinje paliti i gasiti LED0, sve do novog pritiska na taster.

#### LPC1114ETF

Isti program izvršiti na razvojnom sistemu LPC1114ETF.

### Zadatak 5

#### Mbed simulator

Napisati i izvršiti program koji omogućava promjenu trajanja osvjetljenosti LED diode, i to na sljedeći način. U početnom trenutku LED dioda je uključena T sekundi, a T sekundi je isključena. Zatim se dužina uključenosti LED diode postepeno povećava, i to tako da u roku od 30T sekundi LED dioda dostigne vrijeme uključenosti 1.9T sekundi, a vrijeme isključenosti 0.1T sekundu. Zatim se, istom brzinom dužina uključenosti LED diode smanjuje, sve dok ne dostigne uključenost od 0.1T sekundu, a isključenost od 1.9T sekundi (za ukupno 60T sekundi). Nakon toga se vrijeme uključenosti opet povećava, i proces se nastavlja.

Analizirati razliku između izvršavanja programa za  $T=1$  i  $T=0.005$ .

#### Razvojni sistem po izboru studenta

Isti program izvršiti na razvojnom sistemu po izboru studenta.

### Dodatni zadatak (0.5b)

#### FRDM-KL25Z

Program iz prethodnog zadatka dopuniti tako da se mijenja osvjetljenost svake komponente u okviru RGB LED diode na razvojnom sistemu FRDM-KL25Z. Pri tome koristiti različite

varijable T za svaku od komponenti. Kakav se efekat dobije?

### Napomena

Voditi računa da je potrebno preimenovati nazive LED dioda, u skladu sa nazivima navedenim na slici 13 i linku <https://os.mbed.com/teams/Freescale/wiki/frdm-kl25z-pinnames>.

## Literatura

- [1] S. Konjicija, E. Sokić (2019) *Ugradbeni sistemi: Hardverrski aspekti, Elektrotehnički fakultet Univerziteta u Sarajevu, ISBN 978-9958-629-77-8*
- [2] S. Konjicija (2022) *Predavanje Ugradbeni sistemi: Digitalni ulazi i izlazi*
- [3] ARM Holdings (2022) *Mbed OS API Documentation*