Uppgift 1401a

Programmering med handledningstid i labbsalen
Ulrik Eklund, 2014-11-06

Syfte med uppgiften:

Studenterna ska använda moderna utvecklingshjälpmedel för programutveckling för inbyggda system, som IDE (Integrated Development Environment), utvecklingskort och versionshanteringssystem.
Uppgiften görs normalt två och två, men det går bra att jobba ensam också. Det finns tid med handledare i labbsalen för frågor.

Det rekommenderas att ha ett eget USB-minne för katalogerna som med dina personliga kod repositories även om det inte är nödvändigt, all kod du gjort i labben kommer att finnas ”i molnet” när du är klar.

Krav för godkänt:

Ha genomfört följande tre moment enligt denna handledning:

1. Skapa ett eget lokalt repository baserat på det gemensamma repot på github som finns för uppgiften.
2. Starta Atmel Studio, modifiera källkoden i projektet, kompilera och ladda ner den till utvecklingskortet och köra programmet.
3. Committa filerna till ditt personliga repo, synka med github, och göra en ”pull request” till det centrala kursrepot.

Läraren kommer att kolla alla pull requests vid veckans slut, om du lämnar in senare bör du meddela via e-post.

Lämpliga förberedelser

1. Labben kommer att innehålla grundläggande C-programmering, så repetera vad som gåtts igenom i kursen datateknik från första året. Det finns ett diagnostiskt test på It’s learning. Gör det innan du går till labbsalen som en koll på att du kan grunderna!
2. Läs igenom grunderna i versionshantering med git, t.ex. denna självstudiekurs: http://try.github.io/levels/1/challenges/1
3. Du skall också ha läst igenom *hela* denna handledning innan du går till labbsalen!

Förutsättningar för att kunna börja på uppgiften

1. Om annan dator än labbsalens dator används (t.ex. egen dator) måste följande program finnas installerat på den datorn:
    a. Atmel Studio 6.1 eller senare (finns endast för Windows)
    b. Bossac flashprogrammeringsdriver, inklusive DOS-macro (finns om man laddar ner uppgiftens repo)

   c. En Git-klient, t.ex. Github client[1] eller TortoiseGit[2] (för andra operativsystem än Windows finns andra Git-klienter)
2. Alla studenter i labbgruppen måste också ha registrerat en användare på github[3]! Om du redan har en, använd den!
   Meddela Ulrik på ulrik.eklund@mah.se vilka era githubanvändare är.

Versionshantering med Git

Grunderna hur man arbetar med git-klienten för Windows finns på
https://help.github.com/articles/getting-started-with-github-for-windows
https://help.github.com/articles/synchronizing-repositories

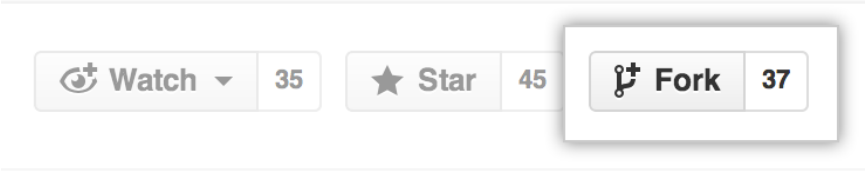
Själva uppgiften

Skapa ett personligt repo för denna programmeringsuppgift med hjälp av Git

För att fortsätta jobba och ändå ha kontroll på vem som gör vad så låter vi Git hålla reda på det för alla studenter och lärare i kursen, vilket är en av Gits styrkor (och varför det används i projekt med hundratals utvecklare, t.ex. open-source-projektet Linux).

Det finns ett centralt repo för kursen på github med webbadressen:
https://github.com/MalmoUniversity-DA139A/Task1401a

1. Skapa en lokal ”fork” av detta repository[4] på kursuppgiftens webbsida. Då kan du jobba vidare på ditt eget projekt i ett eget repository utan att oroa dig för att förstöra kursens gemensamma repot. Enklast är att skapa en fork från githubs webbgränssnitt till kursrepot



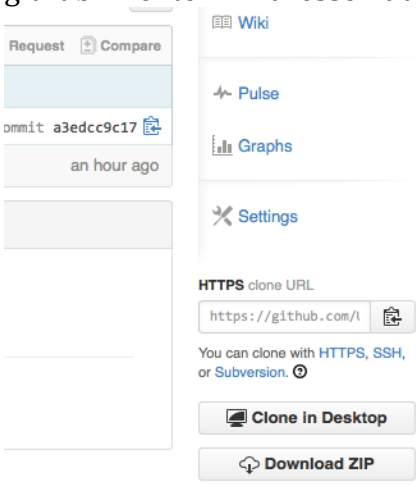
Figur 1: Fork-knappen på githubs webbsida

2. Ange att du skall forka repot till din egen användare på github (den har du redan skapat). I resten av handledningen kallas denna användare *StudentNN.*

3. Än så länge finns bara ditt nya repo på github-servern, men du vill ju kunna jobba med projektet lokalt på din dator. Därför måste du *klona* det nyligen skapade repot till din lokala dator, vilket betyder att du skapar en identisk kopia som hela tiden kan synkas med github med kommandona *pull* och *push*.
   1. Välj en katalog på datorn där du vill ha din lokala kopia av repot.

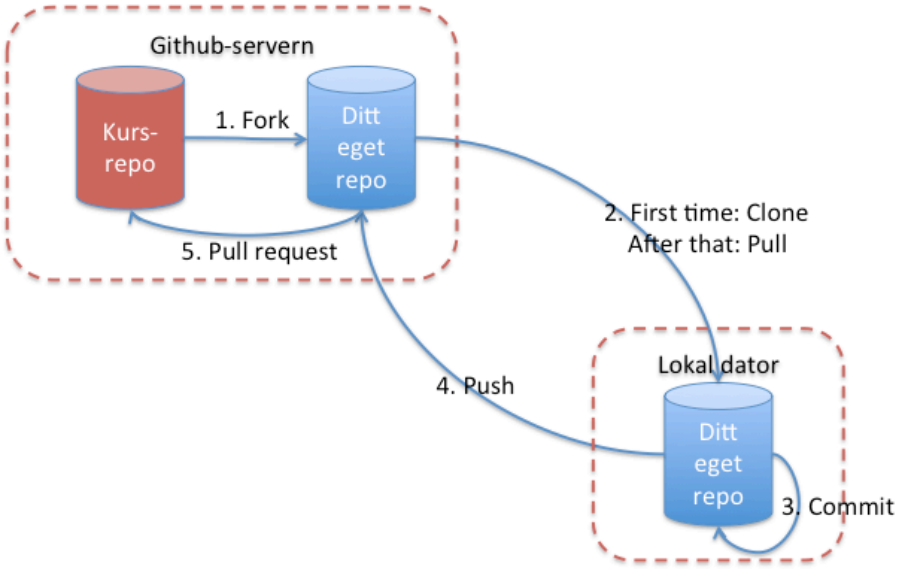
[1] http://windows.github.com/
[2] http://code.google.com/p/tortoisegit/
[3] http://www.github.com/
[4] https://guides.github.com/activities/forking/

2. Klona repot på github repo till din valda katalog på datorn med hjälp av github-klienten[5]. Adressen du ska använda finns under ***HTTPS*** *clone URL*.



Figur 2: Adressen till ett kodrepo på githubs webbsida.

Under labben är det enklast om man jobbar mot ett och samma forkade repo i hela labben, även om det tillhör bara en specifik student. Om ni är två som vill jobba med varsitt lokalt repo kan ni ge båda studenterna rättigheter att jobba mot samma repo och sen klona repot på github till två olika datorer.



Figur 3: Översikt över hur man forkar, klonar och synkar kodrepot i kursen. Kursens gemensamma repo är rött, ditt privata repo är blått.

Arbetskatalogen för denna uppgiften

I katalogen `Task1401a` finns denna handledning och en projektmapp för Atmel studio vid namn `Project1401a`

---

[5] Kommandoraden blir i Unix eller gitbash om man står i önskad katalog:
`>git clone https://github.com/StudentNN/Task1401a.git`
där *StudentNN* är ditt användarnamn på github.

Använda Git

Med jämna mellanrum vill man committa sina ändringar till sitt lokala repo (Steg 3 i Figur 3)[6]. Varje commit bör vara en ”unit of functionality” som alltid kompilerar, t.ex. refaktorera namnet på en funktion, en ny feature, eller förbättrad dokumentation. Varje commit måste också ha en beskrivande kommentar så man vet vad det faktiskt var man ändrade[7].
Git sparar alla tidigare commits, så man kan närsomhelst backa till ett tidigare commit-tillfälle om det skulle behövas.

Om en fil aldrig har blivit committad tidigare måste man ange att den ska adderas till Git-repot[8]. Efter man gjort det första gången håller Git reda på att filen ”ingår” i repot (men den måste committas första gången ändå!). Git-klienterna burkar hålla reda på nyskapade filer i katalogerna som tillhör repot och frågar om de ska adderas.

Bygga labbsetup

Utrustning: Utvecklingskort Arduino Due
USB-kabel (USB <-> USB micro)

Vi ska driva lysdioden som finns på Due-kortet. Den drivs från en av utgågnarna på Due-kortet, vilken? _____
Vilken I/O-port på processorn driver denna utgång?[9] _____

Due-kortet kan drivas med strömmen genom en USB-kabel från datorn om man inte behöver för mycket effekt till utgångarna och det räcker gott för denna labben. USB-kabeln kopplas till USB-micro-kontakten märkt `PROGRAMMING`. När man programmerar eller ”flashar” kortet gör man det också genom samma USB-kabel.

Labbens programmeringsuppgift

Om du lyckats med att klona det centrala git-repot finns nu i katalogen `Task1401a\Project1401a` en projektfil för Atmel Studio: `Project1401a.atsln`
Öppna den genom antingen genom att dubbelklicka eller genom ”open project” inifrån Atmel Studio.

Labben går ut på att ni ska göra om existerande kod till att göra en snarliknande uppgift. Något som är betydligt vanligare än att man skriver all kod från början på ett vitt papper. All kod som är relevant för labben finns i `main.c`
Testa att kompilera projektet under menyn ”Build”. Det här kollar främst att syntaxen är rätt, variabler är definierade och lite annat. C har annars få begräsningar vad som är rätt eller fel i språket.

---

[6] Git-shell-kommando: `>git commit -am ”mywork”`
[7] Ett exempel: https://github.com/akka/akka/commit/adfeb2c1f07153b7eec11705fda956f62b1bbb04
[8] `>git add filnamn.xxx`
[9] Se http://arduino.cc/en/Hacking/PinMappingSAM3X

Du kommer att få två felmeddelanden och få eller inga varningar[10]. Felmeddelandena måste åtgärdas, och de beror på att datatyperna som behövs för 32-bitarsregisterna inte är definierade. Läser du koden i `main.c` noga ser du att den raden som inkluderar definitionerna i `inttypes.h` är bortkommenterad. Avkommentera och till att den läses av kompilatorn och kompilera på nytt.

Bygg programmet under ”Build”-menyn.

Ladda ner den färdiga binärfilen till Due-kortet via USB-kabeln med kommandot `BossacArduinoDue(Debug)` under Tools-menyn[11].

När programmering är klar ska lysdioden blinka regelbundet. Om den inte gör det är det dags att börja felsöka och förstå vad som gått fel.

Om allt funkar som det ska kan det vara dags att göra en commit i ditt personliga repo.

Skapa separata filer för definitioner och funktioner

För att begränsa programmeringen i labben till bara just de ställen det behövs ska du klippa bort de definitioner och funktioner som inte kommer att behöva ändras och skapa en egen fil med dem.

Börja med att skapa en header-fil (som får ändelsen .h) genom att ställa dig i src-katalogen i ”Solution Explorer”-fönstret inuti Atmel Studio. Högerklicka och välj Add/New Item…
Atmel kallar header-filer för ”Include File”. Glöm inte att döpa filen till något meningsfullt, på formen `abc.h` längst ner, innan du trycker på Add-knappen.

Nu ska du klippa ur alla deklarationer ur `main.c` och in i filen du nyss skapat. Det står kommenterat var du ska klippa om du är osäker.
Eftersom deklarationerna fortfarande behövs i main-programmet så måste du tala om för kompilatorn var de finns nu genom att inkludera din nyskapade header-fil på samma ställe i `main.c` som du klippte ur dem:
`#include ”abc.h”`

Testa att kompilatorn fortfarande tycker att alla definitioner fortfarande stämmer innan du fortsätter.

Nu ska du göra samma sak för funktionerna efter main(). Skapa en ”C File” i src-katalogen inifrån Atmel Studio genom att högerklicka och ”Add/New Item…”. Döp den till samma sak som din h-fil ovan (om header-filen heter `abc.h` skall kod-filen heter `abc.c`, det är praxis i all C-programmering).

Klipp ur funktionerna som definieras efterhuvudprogrammet `main()`

---

[10] Varningar som kan finnas är ”bara” för att en del variabler är definierade som aldrig används (i den här labben), så vi bryr oss inte om dem.
[11] Finns inte det kommandot i menyn följ instruktionerna här:
https://github.com/ctbenergy/BossacArduinoDue
Både bossac-programmet och bat-filen finns i katalogen i uppgiftens git-repo.

`void PIOB_init(int PinNumber)` och
`register_data pin(int n)`
ur `main.c` och klistra in dem in din nya c-fil.

Eftersom dessa två funktionera också använder sig av deklarationerna i headerfilen måste den också inkluderas med `#include`

Bygg hela projektet. Om det funkar utan felmeddelande kan du flasha ner programmet till Due-kortet och se att lysdioden fortfarande blinkar likadant.

Om allt funkar som det ska kan det vara dags att göra en commit igen i din branch mot ditt personliga repo.

OBS! Eftersom filerna har modifierats så skall givetvis den/de som gjort förändringarna stå som författare i källkoden, inte Ulrik!

Ändra hur lysdioden blinkar

Nu är uppgiften att ändra det regelbundna blinkandet av lysdioden så att den istället blinkar morsekoden för SOS, tre korta, tre långa, tre korta blink.
Blinkningarna ska uppfylla följande krav:

- Ett långt blink ska vara tre gånger så långt som ett kort blink.
- Pausen mellan varje blink inom en bokstav ska vara lika lång som ett kort blink
- Pausen mellan varje bokstav ska vara lika lång som ett långt blink.
- Pausen mellan varje SOS ska vara lika lång som fem korta blink utan paus.

Du behöver endast ändra koden som finns i kvar i `main.c` för att lyckas med ovanstående!

Det kan vara bra att regelbundet göra en commit i din branch mot ditt personliga repo när du har något som funkar, t.ex. den första bokstaven. På så sätt kan du alltid backa tillbaka till en fungerande version om något skulle gå fel.

Om du av någon anledning skulle behöva synka ditt lokala repo på datorn med kursen gemensamma repo finns det instruktion för hur man löser det[12].

Nu har du klarat av all programmering som behövs (men det är lite kvar med själva inlämningen för att få godkänt). Visa gärna upp det blinkande Due-kortet för handledaren om du befinner dig i labbsalen. Glöm inte att lägga till nya filer och committa det färdiga programmet till git på din dator.

Lagra arbetet på github och lämna in för bedömning

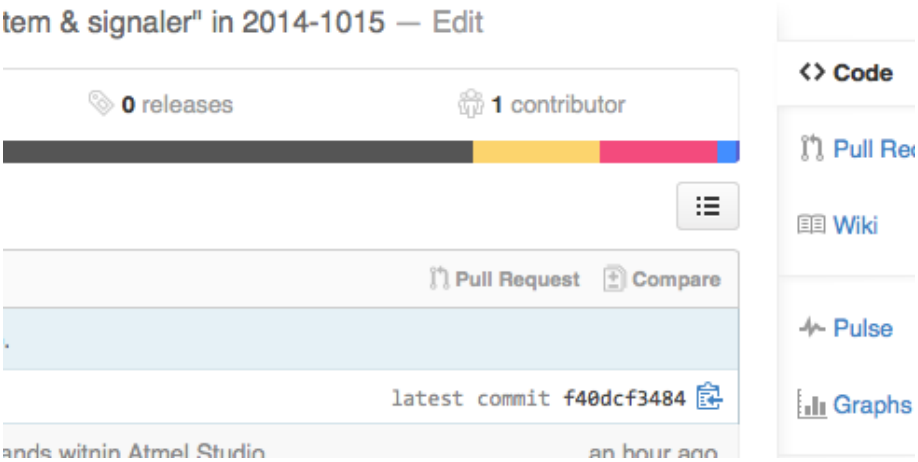
Om du är klar med all programmering är det vara dags att synka ditt lokala repo med det som du har på github.
En viktig sak som måste göras är att lägga till de nya filer till ditt repo, man måste alltså tala om explicit vilka filer som git ska hålla reda på[13]. Det går att fråga git-klienten om

[12] https://help.github.com/articles/configuring-a-remote-for-a-fork/
https://help.github.com/articles/syncing-a-fork/

några nya filer har tillkommit i repots katalog[14]. Innan man har lagt till filerna så uppdaterar en commit bara de filer som redan finns i repot.

Om du har gjort en slutlig commit, inklusive nya filer, kan det vara dags att görs med en push tillbaka till github[15] (steg 3 i **Fel! Hittar inte referenskälla.**). Din gitklient är oftast så smart att den kommer ihåg varifrån man klonade repot, därför kan man oftast använda *origin* istället för den långa webbaddressen[16].

Har du kommit så långt har du ett uppdaterat repo som du kan se på githubs webbsida. Nu ska du skicka in en begäran att ägaren till det gemensamma kursrepot ska titta på dina ändringar genom en *pull request* (steg 5 i Figur 3). Där anger du en rubrik (till exempel att du är klar med en viss uppgift), och fyller i en beskrivning av vad dina ändringar består av.
Glöm inte att ange namnen alla som bidragit med att göra klart uppgiften! Det går inte att komma i efterhand och påstå att du varit med om ditt namn inte finns med på originalinlämningen! I så fall får du göra en egen pull request efteråt.



Ulrik kommer att titta på alla som har lämnat in till fredag 2014-11-14 och de som är godkända får bonuspoäng till tentan. Om du lämnar in senare kommer de att bedömas vid terminens slut.

---

[13] `>git add –all *` säger till git att hålla reda på alla nya filer som skapats om man står i repots huvudkatalog.
[14] `>git status`
[15] `>git push origin`
[16] https://github.com/StudentNN/Task1401a.git