# Laboration 1c

Syfte med labben:

Studenten ska kunna programmera ett system som kan hantera olika saker samtidigt.

Studenten ska kunna använda funktioner i olika c-filer och bygga ett nytt projekt med hjälp av dessa.

Krav för godkänt:

Ha genomfört följande labbmoment enligt denna handledning:

1. Ladda ner ett projekt som används i första delen av labben från github
2. Köra TDD för funktionerna som styr AD-omvandlare och initialisering av Delay-timern.
3. Sätta samman de testade filerna i ett nytt projekt.
4. Bygg, ladda ner och kör programmet.
5. Checka in båda projekten till det centralt GIT-repo

Du visar upp för läraren först när du är helt klar med steg 4 och 5.

Väl godkänt:

För väl godkänt skall du skriva minst två tester för A/D-omvandlaren och skriva ut på LCD-displayen vilken knapp som tryckts in, inte bara det uppmätta värdet på A/D-ingågnen.

## Krav innan laborationen

Om annan dator än labsalens dator används (t.ex. egen dator) måste följande program finnas installerat på den datorn:

* Atmel Studio 6.1 (finns endast för Windows)
* Bossac flashprogrammering + tillhörande bat-fil
* Git
* Unity testramverk (unity.c, unity.h och unity\_internals.h)

## Dra senaste versionen av kursrepots master-branch från github

Ladda ner labbens kod till ert eget lokala repo och er egen branch. OBS: Om ni sparade er kod undre förra labben i mappen CourseMaterial kan ni att få kollisioner med vilken versions som är giltig.

1. Ställ er i master-branchen i ert lokala git-repo  
   >git checkout master
2. Syncha mot master-branchen på kursens repo på github  
   >git pull origin master
3. Ställ er i er egen branch på ert lokala repo (behöver ingen –b flagga för ni skapade ju branschen redan i förra labben)  
   >git checkout StudentNN
4. Merga in master-branchen i er egen branch för att få tillgång till labbens projekt i katalogen Coursematerial/Lab1c  
   >git merge master

## Bygga labbsetup

Utrustning: Utvecklingskort Arduino Due

LCD display

För att skydda AD-ingången från överspänning har ett extra motstånd lötts till som gör att R2 är större än ursprungliga 2kΩ[[1]](#footnote-1).

Hur stort bör R2 vara om spänningen på AD0 skall högsta vara 3,3V när ingen knapp är intryckt? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Programmering

Labben bygger på att sätta samman tre testade c/h-filer till ett fungerande program som läser värden på AD-ingången och skriver ut det på LCD-displayen.

1. Skriv tester och därefter kod för A/D-omvandlaren enligt TDD. Använd Due pinne A0 som läser värdet för knappsatsen på LCD-shielden.  
   Dokumentation och kodexempel för AD-omvandlaren finns på Atmels sida för ASF[[2]](#footnote-2). Lämplig tester kan vara:
   1. A/D-omvandlaren är aktiv efter initialisering
   2. För *väl godkänt* bör man testa att A/D-omvandlaren inte ger ifrån sig värdet 0 (eftersom ingången inte kan ha värdet 0 enligt kopplingsschemat för knappsatsen på LCD-shielden).
2. Skapa ett nytt Atmel studio-projekt i en lämplig katalog i ert repo. Projektet ska använda sig av kod från tre olika källor:
   1. Kod (c & h-filer) för LCD-displayen du testade och skrev i förra labben (det finns en backup-lösning i src-mappen om ni inte har klarat av den labben än).
   2. Kod (c & h-filer) för Delay-timern kopierar du från lab1b till en egen c/h-fil. (Eller tar de som finns i src-katalogen, de fungerar förhoppningsvis…)
   3. De funktioner ni precis skrivit för AD-omvandlaren (c & h-filer).
3. Använd ASF Wizard för att inkludera biblioteken för timer och A/D-omvandlare.
4. Skriv en main.c som i en oändlig loop läser värdet (12 bitar) på A/D-ingången från spänningsdelningen för knapparna och skriver ut värdet på LCD-displayen varannan sekund. Delay-timern håller reda på hur länge man behöver vänta mellan varje gång. Quick start-guiden visar hur enkelt det är att läsa senaste värdet på AD-omvandlaren.
5. För väl godkänt ska man ändra koden så att den skriver ut på LCD-Displayen vilken knapp som har tryckts in (Select/Left/Right/Up/Down). Lämpligtvis använder man den tidigare utvecklade funktionen LCDWrite().

# Appendix: A/D channels versus Due pin mapping

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Instance** | **Signal** | **I/O Line** | **Peripheral** | **Due Pin** |
| ADC | ADTRG | PA11 | B |  |
| ADC | AD0 | PA2 | X1 | Analog In 7 |
| ADC | AD1/WKUP1 | PA3 | X1 | Analog In 6 |
| ADC | AD2 | PA4 | X1 | Analog In 5 |
| ADC | AD3 | PA6 | X1 | Analog In 4 |
| ADC | AD4 | PA22 | X1 | Analog In 3 |
| ADC | AD5 | PA23 | X1 | Analog In 2 |
| ADC | AD6 | PA24 | X1 | Analog In 1 |
| ADC | AD7 | PA16 | X1 | Analog In 0 |
| ADC | AD8 | PB12 | X1 |  |
| ADC | AD9 | PB13 | X1 |  |
| ADC | AD10 | PB17 | X1 | Analog In 8 |
| ADC | AD11 | PB18 | X1 | Analog In 9 |
| ADC | AD12 | PB19 | X1 | Analog In 10 |
| ADC | AD13 | PB20 | X1 | Analog In 11 |
| ADC | AD14/WKUP13 | PB21 | X1 |  |

1. Se kretsschema på <http://www.dfrobot.com/image/data/DFR0009/LCDKeypad%20Shield%20V1.0%20SCH.pdf> [↑](#footnote-ref-1)
2. <http://asf.atmel.com/docs/latest/sam3x/html/group__sam__drivers__adc__group.html>

   <http://asf.atmel.com/docs/latest/sam3x/html/sam_adc_quickstart.html> (man behöver inte implementera några interrupts!) [↑](#footnote-ref-2)