Laboration 2: ”Gissa talet”

1 Inledning

Laborationen ska bidra till en grundläggande förståelse för:

- Hur man programmerar inbyggda system i C.
- Hur man använder pekare och adressreferenser till variabler.
- Hur textsträngar är uppbyggda samt hur de kan hanteras i C.
- Hur programkod kan delas upp i flera abstraktionsnivåer.

1.1 Utrustning

Det behövs inga nya komponenter.

2 Beskrivning av laboration

Den här laborationen utgör en fortsättning på Laboration 1. Laboration 1 bidrar med: fördröjningsrutiner, drivrutiner för LCD och tangentbord. Utöver detta tillkommer nu en del nya saker: strängar och användning av pekare.

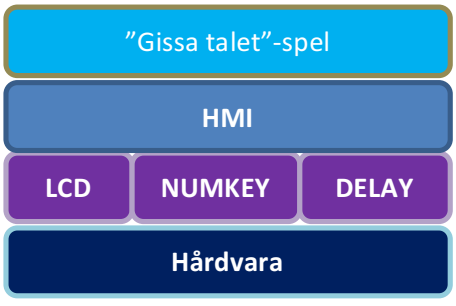
Laborationen består av följande moment:

1. Interaktion med användaren
2. Färdigställning av ”Gissa talet”

I laborationens mapp på It’s Learning finns en zip-fil med lite komponenter (**lab2_files.zip**). Dessa komponenter består av flera filer som tillsammans med det som kommer från Laboration 1 utgör grunden för enkelt spel, ”Gissa talet”, som går ut på att spelaren ska försöka gissa på ett slumpmässigt tal mellan 1-100. Laborationens olika moment går ut på att ni ska komplettera programkoden på olika abstraktionsnivåer. När ni sedan är klara kan ni ”slappna av” och testa spelet.

2.1 Syfte och mål

I föregående laboration var det en del fokus på att skapa en applikation baserat på kod med tydliga gränssnitt (API:er). I denna laboration blir detta ännu tydligare med fördelarna av att skapa mindre kodmoduler med tydligt definierade och avgränsade ”ansvarsuppgifter”. Med denna metod skapar man flera abstraktionsnivåer av programkod (se illustration i Figur 2-1).



Figur 2-1: Illustration av en applikation, baserat på flera abstraktionsnivåer av kod.

Genom att använda flera nivåer (”lager”) så blir det lättare att utveckla, underhålla och vidareutveckla applikationer. Man behöver exempelvis inte programmera om spelet bara för att man ska ”porta” det till en annan mikrokontroller. Dessutom blir det lättare att fördela uppgifterna mellan olika programmerare i ett projekt, då varje person kan fokusera på en avgränsad del av den slutgiltiga applikationen.

”Gissa talet”-spelet har avsikten att exemplifiera hur en applikation kan konstrueras på ett strukturerat vis. I takt med att ni utför laborationens uppgifter, genom att modifiera kod och göra vissa tillägg på olika abstraktionsnivåer, kommer ni att se fördelarna med att skapa avgränsade kodmoduler.

2.2 Examination

2.2.1 Inlämning av rapport och programkod

Instruktioner för inlämning finns beskrivet på inlämningssidan för denna laboration (på It’s Learning).

2.2.2 Praktisk och muntlig redovisning

Den praktiska delen av laborationen examineras efter att laborationens sista moment har avslutats. Följande ska redovisas:

- Programkod (med god struktur och kommentarer).
- Demonstration av systemet, fullt fungerande enligt specifikation.

Ni ska även kunna redogöra för funktionalitet avseende programkod.

3 Moment 1: Interaktion med användaren

3.1 Beskrivning

Med hjälp av grundläggande drivrutiner för att skriva text på displayen, hantera tangenttryckningar och även skapa fördröjningar, kan man skapa ytterligare en abstraktionsnivå i programkoden. På denna nivå finns funktioner för att erbjuda interaktion med användaren. Dessa funktioner finns i filen **hmi.c**. HMI står för *Human-Machine-Interaction*, dvs. interaktion mellan människa (användaren) och maskin (ert system).
I detta moment ska ni komplettera funktionen **input_int()**, som är en funktion för att låta användaren mata in ett tresiffrigt tal. Funktionen fungerar som en ”dialogruta”, vilket kan jämföras med något ni har sett i en GUI-miljö. Detta innebär att samtidigt som ni trycker på en tangent, så ska den visas på displayen. När man trycker på någon av tangenterna 0-9 så ska dessa visas på displayen. Tangenten med fyrkant (#) används för att avsluta och bekräfta inmatat tal. Tangenten med stjärna (*) används för att radera inmatade tecken, ett i taget. Tangenterna ’A’, ’B’, ’C’, ’D’ skall inte vara aktiva (inget skall hända då man trycker på någon av dem).

3.2 Skapa projektet och kopiera in filer från Lab1

Uppgift 7.2.1 (redovisas i programkod)
Skapa ett nytt ”Executable”-projekt - processorn är ATmega2560!(kom ihåg att lägga det där det inte försvinner, tex på ert M:-konto) och kopiera in filerna (inte lab1.c!) från Laboration 1. Om man först (I högerfönstret – Solution Explorer – högerklickar på projektet och väljer ”Add/Existing project” för att lägga till Lab1 i det nya projektet, drar filer från detta till lab2-projektet och slutligen tar bort lab1-projektet (högerklicka på projektet och välj ”Remove”) så slipper man ta in filerna en och en.

Lägg därefter till filerna från Lab2_files.zip (finns på It’s learning) på detta sätt:

- **hmi.h** och **hmi.c** skall ligga i en katalog som heter **hmi** (skapa en sådan katalog genom att högerklicka på projektnamnet i högerfönstret (Solution Explorer) i Atmel studio. Välj sedan ”Add..”. Filerna läggs in på samma sätt – välj ”Existing file…”).
- **guess_nr.h** och **guess_nr.c** skall inte ligga i någon underkatalog
- **Lab2.c** skall inte heller ligga i någon underkatalog (filen skall ersätta den fil som skapades då ni skapade det nya projektet). Om ni har kallat ert projekt något annat än Lab2, så skall ni **INTE ändra namnet** på huvudprogrammet!

3.3 Komplettering av inmatningsfunktion

Uppgift 3.3.1 (redovisas i programkod)
Efter att en tangent har tryckts ner och dess funktion har utförts, så ska programmet ”loopa” tills att tangenten har släppts upp. Komplettera funktionen **input_int()** så att detta sker. Ändra i huvudprogrammet, så att ni kan lägga in testkod för att testa denna funktion. (hela programmet fungerar inte förrän alla delar i moment 1 och 2 har slutförts!).

Uppgift 3.3.2 (redovisas i programkod)
Komplettera funktionen så att ett tecken kan raderas från displayen när man trycker på knappen med stjärna (*).
OBS! Ni kommer att behöva använda en LCD-instruktion för att stega åt vänster! Finns en sådan funktion? Om inte, hur gör man annars?

Uppgift 3.3.3 (redovisas i rapport)
I början av funktionen deklareras en tecken-array (**numbers**), som används för att skapa en sträng av siffertecken. Funktionen tillåter endast att maximalt tre siffror kan matas in. Som ni ser så dimensioneras arrayen till att rymma fyra tecken. Vad används den sista positionen till?

Uppgift 3.3.4 (redovisas i programkod)
Komplettera inmatningen, så att tangenterna 'A', 'B', 'C', 'D' inte är aktiva. Detta skall göras på lämpligt ställe i **hmi.c**. Ni skall alltså **INTE** ändra o tangenbordsdrivern!

Uppgift 3.3.5 (redovisas i programkod)
Redovisa ert bidrag i koden genom att komplettera kommentarsblocket i början av filen.

4 Moment 2: Färdigställning av "Gissa talet"-spelet

4.1 Beskrivning

Spelet går ut på att ett tal slumpas fram, som spelaren ska försöka lista ut. Om man inte gissar rätt, så får man en ledtråd om talet var för lågt eller för högt. Spelet är nästan färdigt för att demonstreras! Det enda som kvarstår är några mindre modifieringar av två funktioner i filen **guess_nr.c**.

4.2 Färdigställning av funktioner

Uppgift 4.2.1 (redovisas i rapport)
Ändra siffran i "srand-funktionsanropet" i huvudprogrammet till något annat värde (2-255). Kompilera/bygg programmet. Det bör inte bli uppstå några fel eller varningar vid kompileringen. Beskriv i rapporten hur effekterna av att använda en pseudo-slumptalsfunktion uppträder när man kör programmet.

Uppgift 4.2.2 (redovisas i programkod)
I funktionen **get_nr()**, gör så att **input_int()** anropas. Den första parametern är en textsträng, som ska vara "ENTER NUMBER:". Den andra parametern ska vara en adressreferens till variabeln **guessed_nr**, vilket möjliggör att **input_int()** kan modifiera denna variabel. Returvärdet från **input_int()** ska lagras i variabeln **input_length**. Testkör programmet för att se att det fungerar som det ska!

Uppgift 4.2.3 (redovisas i programkod)
När spelaren har gissat rätt tal ska antalet gissningar visas på displayen. För varje gissning ska en variabel ökas med 1. I nuvarande versionen fungerar detta inte alls. För att detta ska fungera måste ni göra en mindre utökning i funktionen **playing_game()**. Testkör programmet för att se att det verkligen fungerar!

Uppgift 4.2.4 (redovisas i programkod)
När spelet fungerar som det ska så är det dags att avsluta genom att redovisa ert bidrag i koden, dvs. genom komplettering av kommentarsblocket i början av filen **guess_nr.c**. Därefter kan ni redovisa för labhandledare.

Uppgift 4.2.5 (redovisas i rapport)
Redogör för era erfarenheter och kunskaper från denna laboration (minst en halv A4-sida):

- Vad har ni lärt er?
- Om ni får välja en sak, upplevde ni något som var intressant/givande?
- Fanns det något som upplevdes som svårt?
- Gick allting bra eller stötte ni på problem? Om allting gick bra, vad var i så fall anledningen detta? Om ni stötte på problem, hur löste ni i så fall dem?

Fördjupande uppgifter (görs om tid finns)

5 Moment 7: Inmatning av ”seed”

5.1 Beskrivning

I detta, extra steg, skall ni lägga till rutiner för att läsa in ett startvärde (seed) för slumptalsgeneratorn.

- **Uppgift 5.1.1 (redovisas i programkod)**
  Använd inmatningsrutinen (**input_int**) för att låta användaren mata in ett tal, som sedan används för att initiera slumptalsgenereringen (funktionen ”srand” i början av huvudprogrammet). Gör även **utskrift av lämpliga texter** för att be användaren mata in ett tal. Testa att det fungerar!