1 INLÄMNINGSUPPGIFT/LABB 01

1.1 ÖVERGRIPANDE

Uppgiften utförs och examineras individuellt. Det är självklart tillåtet att be andra om hjälp och att använda resurser från internet — så länge du inte bryter mot några licensavtal och korrekt anger källor på tredjepartskod du använt. Att inte göra det utgör akademiskt fusk, och resulterar i underkänd uppgift och/eller disciplinära åtgärder. Du ska kunna stå för och förklara varje rad kod du lämnar in.

Deadline för labben är **21:00 Tisdag 28/10**

Är något otydligt, eller om du inte förstår uppgiften, är det ditt ansvar att påtala det och begära ett förtydligande.

1.2 INLÄMNING

Lösningen lämnas i samma form som du får ut baselinen, dvs ett `.tar.gz`-arkiv innehållande ett gitrepo med lösningarna. Repot skall innehålla en (1) uppsnyggad och väl beskriven commit innehållande respektive deluppgift, i nedanstående ordning. Det ska *tydligt* framgå ur respektive commit-meddelande vilken deluppgift commiten avses lösa. Koden ska vara prydlig och tydlig — *Linux Kernel Coding Style* (https://www.kernel.org/ doc/html/v4.10/process/coding-style.html) är en lämplig grundplåt att utgå från.

1.3 BETYGSKRAV

För *Godkänt* krävs inlämnade lösningar på deluppgifter 1–3 med den efterfrågade funktionaliteten. Mindre misstag kan tolereras, men funktionen överlag ska finnas och indikera förståelse för problemet. Koden skall kompilera. Du skall kunna förklara din kod.

För *Väl Godkänt* krävs, förutom kraven för godkänt, att laborationen lämnats in i tid, är fullständigt korrekt, samt att en korrekt lösning på deluppgift 4 tillhandahålls.

1.4 UPPGIFT

Labben går i stora drag ut på att använda GPIO på Arduinon för att blinka en LED samt få igång seriekommunikation mot den. Syftet med det är i första hand att bekanta dig med miljön, och att få möjlighet till debug-utskrifter i senare labbar.

Du tillhandahålls en ’baseline’ med nödvändiga byggscript och källkodsfiler — utgå från den när du löser labben. Du får även ut en uppsättning nödvändiga elektronikkomponenter. Arduino UNO R3 samt kopplingsdäck och kablar tillhandahåller du själv. Du tillhandahålls också en uppsättning elscheman och relevanta datablad, dessa finns under ’labbrelaterade resurser’ på studentportalen.

1.4.1 DELUPPGIFT 1: BLINKA LED

1. Läs databladen för tillhandahållen LED och resistorer och räkna ut en lämplig resistorstorlek
2. Koppla in LED och lämplig resistor till pinne 8 på Arduinons expansionsport med hjälp av kopplingsdäck och kablar
3. Använd elschemat för Arduinon för att lista ut vilken port och pinne på ATMegan det motsvarar
4. Konfigurera upp pinnen som utgång
5. Skapa en oändlig loop i `main()`, och omväxlande tänd/släck LEDen med 500 ms av- respektive på-tid.
6. Städa upp din kod och skapa en git-commit med ovanstående och redogör för resistorberäkningen i commit- meddelandet

1.4.2 DELUPPGIFT 2: KONFIGURERA UPP UART FÖR SÄNDNING (TX)

Samtliga nämnda funktioner deklareras i `serial.h` och implementeras i `serial.c`.

1. Implementera funktionen `uart_init` så att den konfigurerar upp USART0 för sändning (TX) enligt 8N1 i 38400 baud.

2. Implementera funktionen `uart_putchar` så att den skickar ett enskilt tecken över USART0, och lägger till tecknet ’`\r`’ (radretur) om tecknet vi vill skicka är ’`\n`’ (radbrytning).

3. Testa funktionen med hjälp av en serieterminal på din dator — anropa periodiskt `uart_putchar` från din loop i `main`.

4. Implementera funktionen `uart_putstr` så att den med hjälp av `uart_putchar` skriver ut en hel sträng på serieterminalen.

5. Visa att funktionen fungerar genom att anropa `uart_putstr` från `main` med ditt namn + radbrytning som argument.
6. Städa upp din kod och skapa en git-commit med ovanstående förändringar.

1.4.3 DELUPPGIFT 3: KONFIGURERA UPP UART FÖR MOTTAGNING (RX)

1. Ändra implementationen av `uart_init` så att den även medger mottagning (RX).
2. Implementera funktionen `uart_getchar` så att den tar emot ett enskilt tecken.
3. Visa att funktionen fungerar genom att implementera `uart_echo` så att den väntar på ett inkommande tecken med hjälp av `uart_getchar`, och direkt skickar tillbaka samma tecken med `uart_putchar`. Anropa funktionen från `main`.
4. Städa upp din kod och skapa en git-commit med ovanstående förändringar.

1.4.4 DELUPPGIFT 4 (VG-KRAV): STYR LED VIA UART

1. Deklarera, implementera, och kommentera en funktion som via UART tar emot strängarna `'ON\r\n'` och `'OFF\r\n'` och tänder respektive släcker en LED (samma som ovan, en ny på godtycklig pinne, eller den inbyggda på pinne 13).
2. Städa upp och skapa en git-commit med ovanstående förändring. Dokumentera beteendet och vilken pinne LEDen förväntas sitta på.