1 INLÄMNINGSUPPGIFT/LABB 02

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 **Torsdag 4/11, KLOCKAN 23:59**.

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 (alternativs .zip) 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 1–2 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 3 tillhandahålls.

1.4 UPPGIFT

Labben går i stora drag ut på att använda timer/compare-enheten på Arduinon för att periodiskt blinka en LED, och sedan använda pulsbreddsmodulering för att variera ljusstyrkan på LEDen. Syftet med det är i första hand att bekanta dig med timers och dess olika användningsområden, samt konfigurering av något mer komplexa periferienheter.

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 scheman och relevanta datablad.

1.4.1 DELUPPGIFT 1: BLINKA LED

Använd samma LEDar och resistorer som i labb 1

1. Koppla in LED och lämplig resistor till pinne 8 på Arduinons expansionsport med hjälp av kopplingsdäck och kablar. Minns från labb 1 att det motvarar PORTB0, och konfigurera upp den som utgång (alternativt valfri led på din egentillverkade, handgjorda, lokalproducerade sköld!).

2. Konfigurera upp *timer0* mha kap 19 i databladet:

    - CTC-mod (Clear Timer on Compare Match mode)
    - Prescaler = 1024
    - Använd den informationen och databladet för att hitta ett output compare-värde som motsvarar en periodtid på 10 millisekunder. Minns att CPU-klockan är ställd till 16 MHz. Ta med uträkningen och värdet i git-committen.

3. Skapa en oändlig loop i `main()`. I den, vänta på att räknaren når/matchar output comparevärdet (övervaka matchflaggan), och när så sker:

    - Rensa flaggan
    - Inkrementera en räknarvariabel, och när denna når 10; nollställ den och toggla LEDen. Detta för att förenkla, ögat ser inte blinkningarna i 100 Hz, så vi delar ner frekvensen till 10 Hz. Skriv och testa gärna den här delen innan du kopplar in timer-koden.

4. Städa upp din kod och skapa en git-commit med ovanstående.

1.4.2 DELUPPGIFT 2: VERIERA LJUSSTYRKAN MED PWM

1. Koppla upp en LED med resistor på den Arduino-expansionsport som motsvarar `OC0A` på ATMegan. Sätt upp den som utgång.

2. Konfigurera om *timer0*:
    - *Fast PWM*-mod, `0xFF` som `TOP`, Non-inverting mode.
    - Prescaler = 64. Ange i git-commiten vilken PWM-frekvens det resulterar i samt beräkningen du använde.
3. Prova att variera ljusstyrka/duty cycle genom att skriva ett par olika värden till `OCR0A`, och visa att det fungerar genom att periodiskt växla mellan dem med en enkel loop och fördröjning mellan varje ändring. Välj värdena så att man tydligt ser skillnader.
4. Städa upp din kod och skapa en git-commit med ovanstående förändringar.

1.4.3 DELUPPGIFT 3 (VG-KRAV): LED-RAMPNING

1. Konfigurera *timer0* som Fast PWM enligt ovan, och *timer2* som enkel timer i CTC-mod, periodtid **16** ms.
2. Städa upp ovanstående och bryt ut till en `timer_init()`-funktion
3. Deklarera och implementera en funktion `uint8_t simple_ramp()` som vid varje anrop returnerar ett värde mellan 0–255. Värdet ska börja på 0, och för varje anrop inkrementeras, tills det når 255. Därefter ska det dekrementeras ner till 0, varpå cykeln börjar om.
4. Anropa funktionen periodiskt med hjälp av *timer2* och använd returvärdet som duty cycle för LEDens PWM-styrning. Förväntat beteende är att LEDen synligt pulserar av och på
5. Städa upp och skapa en git-commit med ovanstående förändring.