Slutuppgift: Inbyggda System
Arkitektur och Design

Bakgrund:
Arduino-bräden och likvärdiga nybörjarkort med färdigskrivna drivrutiner och bibliotek har format en grund för studerande att anta sig grundläggande utvecklingsprinciper. Men att kalla på funktioner och vidarefordran variabler är inte helheten bakom att vara en utvecklare för Inbyggda System eller IoT.
Det är därför viktigt för studerande att arbeta ännu närmre hårdvaran med ”öppnare” utvecklingskort där de behöver nyttja Embedded C/C++ samt Assembly.

Detta dokument kommer därför utforma grunden och strukturen till kursens slutuppgift där den studerande kommer tvingas att genomföra och uppvisa kunskap inom tre huvudsakliga kategorier:

- Hantering, Förståelse och Användande av Dokumentation
- Uppsättning och Konstruktion av en Effektiv Utvecklingsmiljö
- Programmering i Maskinkod samt C eller C++ för Periferienheter

Slutuppgiften i sig kommer vara strukturerad i, och bedömas på, två mindre projekt. Ett projekt där fokuset hamnar på maskinkod, och ett andra projekt där fokus landar på lokal utveckling av OS för ett inbyggt system.

Vid frågor om;

- Arbetsuppgifterna
- Kodrevy eller Stöd
- Tidsomfattningar
- Resurser/Dokumentation
- Eller Likvärdigt

Skall detta medlas till utbildaren över email: Ludwig@sihc.education

Förutsättningar för ett Lyckat Genomförande:
Många av de delar som krävs kunskapsmässigt för att genomföra slutuppgiften kommer som meddelat att presenteras löpande utmed kursen. Tanken med detta format är att ny kunskap skall presenteras vardera lektionstillfälle så att studerande på egen tid utanför lektionstid eller under handledningar kan arbeta med att tillämpa sina nyvunna kunskaper och kännedomar. På så sätt implementeras även egen research och nyfikenhet.
Men med detta noterat finns ett par tekniska förutsättningar som görs för samtliga studerande och som den studerande själv ansvarar för att inneha.
Dessa kategoriseras nedan:

Utvecklingsmiljö:
- Visual Studio Code
- C/C++ Compiler och Dependencies
- STM32CubeIDE
- Proteus Simulation Tool
- Github Desktop / Alt. Github

Dokumentation:
- Anges senare

Virtualisering:
- VirtualBox

Operativsystem:
- Ubuntu 22.04

Studerande förväntas sätta upp ovan nämnda på egen hand i enlighet med länkade resurser. Vid problem, ta kontakt med utbildaren.

Struktur: Hur Slutuppgiften Förväntas Hanteras;
I en verklighetstrogen arbetsmiljö så förväntas utvecklaren att ta sitt ansvar i att hålla sitt arbete strukturerat, organiserat och väldokumenterat. Detta kommer även vara fallet för detta första arbetsvärlds-trogna programmeringsprojekt. Nedan kommer nu beskrivas hur studerande förväntas att vidhålla struktur under hela kursens gång;

Filstruktur:
Utmed hela arbetet så skall vi följa en god dokumentstruktur. Detta innebär att från start tänka på hur man för sig emellan filer och information som är lokalt lagrad. Nedan följer ett arbetsträd så som jag rekommenderar det:

<Huvudmapp med Projekt-Titel> (En huvudmapp för P1 och en för P2)
- <ReadMe.md>
  (Innehåller en övergripande text som förklarar projektets innehåll, mål och syfte samt dess struktur)

- <Hårdvarumapp>
  (Innehållande samtliga dokument avseende den hårdvara som projektet konstruerats för / baserats på)

- <Mapp med Källkod>
  (Innehållande samtliga filer som är en del av de program som genererats utmed kursen och den slutuppgift som genomförs)

- <Mapp med Rapport>
  (Innehållande text-rapport om utförandet av arbetet i form av akademisk rapport)

Denna filstruktur förväntas att lagras lokalt och kontinuerligt synkas emot Github i ett repo i samma namn och med utbildaren som inbjuden att se projektet.

Arbetsstruktur:
Utmed hela arbetet så förväntas man inte endast att strukturera resultatet av dagens arbete, utan arbetet i sig. Detta kommer ned till två huvudsakliga byggstenar som skall nyttjas:
1. Veckoplanering
2. Dagbok

Veckoplanering skall ske individuellt och vardera vecka. Detta betyder att ni för er egen skull strukturerar er arbetsvecka precis som på en äkta arbetsplats igenom att spika punkterna:
- Vad är mina mål?
- Hur skall jag försöka uppnå satta mål?

Sedan för ni detta utmed veckan i er dagbok, vad ni har gjort för att arbeta emot dessa mål, hur det har gått (både fram och motgångar) samt vad projektet har kommit till vid den tidpunkten.
Detta är för att ge er bättre insikt i hur man kan bli effektivare på att strukturera sitt utvecklingsarbete.

Rapportstruktur:
Tillhörande projektet skall en rapport ta form för att kunna representera det arbete som ni utfört. Denna rapport skall följa formatet:

- Framsida med skolans logotyp samt för och efternamn samt mailaddress
- Innehållsförteckning som tydligt påpekar sidor där information kan återfinnas
- Inledning som förtydligar arbetets innehåll, mål och syfte
- Dagbok som förtydligar genomförandet av hela arbetet
- Resultatdel som uppvisar slutresultatet samt förtydligar användning, hantering och struktur av Github-repo
- Referensbilaga som refererar till alla de verktyg, källor och dokumentationer som nyttjats utmed arbetet

Kodstruktur:
Även den kod ni skriver skall följa kod programmeringsed och därmed hålla en professionell struktur. Följande krav kommer därför ställas på er källkod;

- Tydliga Namnsättningar
- Tydliga och Förklarande Kommentarer Löpande i Koden med Motivation
- Uppdelning av koden i flertalet filer

Detta för att garantera att en kollega eller utomstående part hade kunnat kliva in i er programmeringsmiljö och snabbt anta sig att förstå det arbete som utförts.

Projektdel 1: Drivrutins-konstruktion och Programmering i C/C++
I ert första projekt så skall ni genomföra ett arbete där fokuset ligger på att bemöta en plattform för en IoT/Embedded-lösning och sedan utveckla en lösning tillhörande denna plattform.
I vårt fall kommer vi att bygga på STM32-plattformen med anledning till dess tillgänglighet av dokumentation, information och starka communityredaktion vilket gör det till en bra inkörsport ifrån ert Arduinobräde.

Projektdel 1 kommer kräva ett par förutsättningar innan ni kan aktivt inleda ert utvecklingsarbete:

- Ladda ned dokumentationen
- Ladda ned Data-Sheet
- Ladda ned Användarguiden
- Bekanta er med dokumentationen och plattformen
- Introducera er själva till CubeIDE och STM32

Igenom detta så kommer ni vara mer beredda på ert åtagande för Projektdel 1.

När ni genomfört detta någorlunda komplexa förarbete så bör ni vara redo att etablera grunden till er första projektdel; Att välja en målsättning!
Ert första uppdrag är nämligen att själva välja en lösning baserad kring STM32-kortet och sedan konstruera den kod som krävs för att uppnå er satta målsättning.

De krav som er lösning dock måste innefatta är:

- Utvecklingen/Inkludering av ett eget library/egna drivrutiner
- Källkod baserad på Objekt-orienterad C/C++

Ni är helt fria att utefter dessa krav själva välja lösningar baserade på egen research eller egna intressen, se bara till att få er idé godkänd av utbildaren innan uppstart.
Om ni däremot inte vill välja ett projekt själva, så kan ni välja en av utbildarens två rekommendationer:

1. Hantera en LED med UART-protokollet
2. Hantera en LCD med UART-protokollet

Sedan är det ”bara” att starta!

Rekommenderat Genomförande:
Det är rekommenderat att ni inleder med att bekanta er med dokumentationen. Försök avläsa exempelvis stycken om koncept som under lektionen diskuterats så som Registers, Pins, osv..
Efter detta kan det rekommenderas att man testar på att följa en guide eller två om drivrutinsutveckling bara för att få en praktisk förståelse om hur man refererar till materialet för det man är ute efter. Sedan rekommenderar jag att ni arbetar med att försöka programmera själva och se om ni kan kompilera era drivrutiner utan fel, och sedan vidaregå till er källkod för funktionen.
Kom även ihåg att vi kommer ta upp allt detta under lektionerna och att ni därmed bör försöka själva, men inte känna er förlorade om ni inte kan, då det är meningen att utforska och lära!

Projektdel 2: Utveckla OS-baserade lösningar för IoT/Embedded;
Som vi diskuterat och omnämnt under våra tidigare kurser så handlar inte alla inbyggda lösningar om att driftsätta den mest lättviktiga och hårdvarunära drivrutinen, utan ibland om att faktiskt nyttja den kapacitet systemet erbjuder och driftsätta ett komplett OS som applikationen sedan körs uppepå.
Detta är känt som ”Embedded Linux” och används främst när det man konstruerar kräver att linux-kärnan är närvarande; ex för webbkommunikation eller direktkontakt med molnet på ett smidigt sätt.
I vårt fall så kommer vi simulera detta igenom att vända oss till Yocto som är ett open source initiativ till att tillgängliggöra ”Embedded Linux” och bygga detta operativsystem för Raspberry Pi. (Uppmärksamma att detta kommer simuleras, inge inköp krävs för att genomföra detta projekt)

Vi kommer genomföra detta igenom att;
- Virtualisera Ubuntu på vår lokala hårdvara
- Sätta upp Ubuntu för Embedded-utveckling och synk emot Github
- Förbereda för att kompilera och Bygga vårt OS lokalt

Projektdel 2 kommer även här kräva att ni bekantar er med en del aspekter som möjliggör detta genomförandet;
- Linux
- Ubuntu
- Github
- Yocto

Men efter detta kommer Projektdel 2 förmodligen tyckas vara lättare än del 1 tack vare sitt mer administrativa programmerande än kodmässiga sådana. Men ni bör nu vara redo för genomförandet;

Det krav er lösning skall omfatta är:
- Arbete i Ubuntu
- Utveckling med Yocto
- Tillämpning av valfri funktion uppepå Yocto

För denna projektdel krävs heller inte samma självgående aspekt som kodutmaningarna i del 1. Detta då Yocto behöver hanteras på ett och samma sätt för att faktiskt fungera som planerat, utan här förekommer friheten i ert val av drift av funktionen uppepå vårt Embedded Operativsystem.

För genomförandet skall ni helt enkelt följa vad som uppvisats på lektionen och de resurser som angetts ovan. Även denna läsning rekommenderas:
https://medium.com/nerd-for-tech/build-your-own-linux-image-for-the-raspberry-pi-f61adb799652

Ert projekt skall sedan paketeras och synkas emot Github precis som del 1.

Betygsättning:
Betygsättningen kommer delas upp i två moment;

1. För att uppnå betyg G skall båda projekten vara genomförda enligt de kriterier som angetts i detta dokument. Dock kan Projektdel 2 inte generera ett högre betyg än G på grund av sitt guidade format.
2. Projektdel 1 kommer dock möjliggöra för den studerande att tilldelas betyg på den fullständiga skalan IG/G/VG. Kriterierna som här kommer bedömas av utbildaren är som följer;

- Arbetsstruktur (Hur man löpande arbetar med sina uppgifter och strukturerar sin tid och sina efterforskningar) (0-2p)
- Kodstruktur (Hur koden är skriven, motiverad och dokumenterad i kodfilerna) (0-4p)
- Dokumentationsstruktur (Hur vidare man dokumenterat både sin arbetsresa och sitt resultat på ett fullständigt och tydligt akademiskt sätt) (0-2p)
- Teoretiskt påvisande av kunskap (huruvida man har motiverat och förklarat samtliga delar av sitt arbete på ett sådant sätt att man tydligt ser att individen till fullo förstår vad denne utfört) (0-5p)
  IG = 0-7p
  G = 8-10p
  VG = 11-13p