Programming **Project**

- **Objective:** is the implementation of an application in C, that is executed in a microcontroller system
    - to analyse a medium-size programming problem
    - plan and manage the execution of such project with representative industry standards

Programming Project

<u>At the end</u> of the course you should be able to:

- Identify and list the basic elements of the architecture of a microcontroller
- Describe and explain basic mathematical operations in fixed point format
- Apply and demonstrate the use of registers in a microprocessor
- Analyse a medium-size programming problem
- Apply C language and structured programming in C
- Implement and synthesize a program targeting a microprocessor
- Design and realize an application in real time
- Organize, plan and document the workflow of a software project
- Write a technical report including references and citations

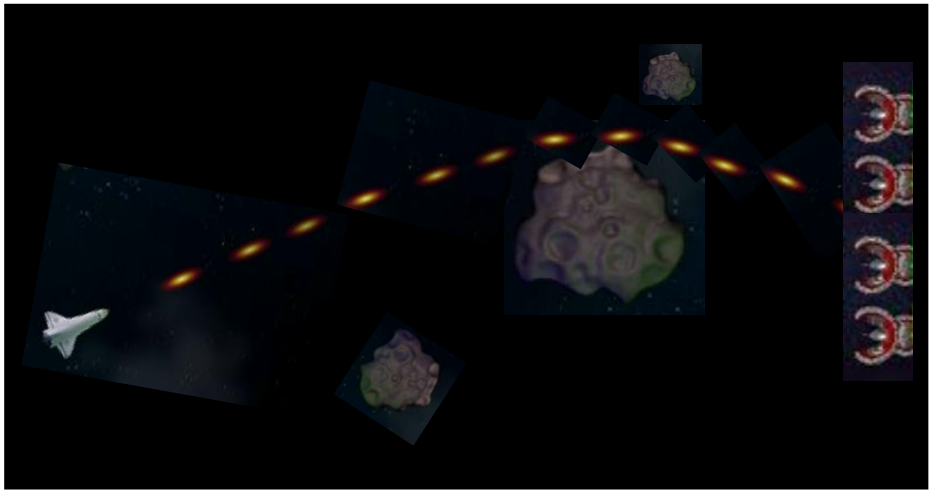
DTU Space
National Space Institute

30010 Course Plan

DTU

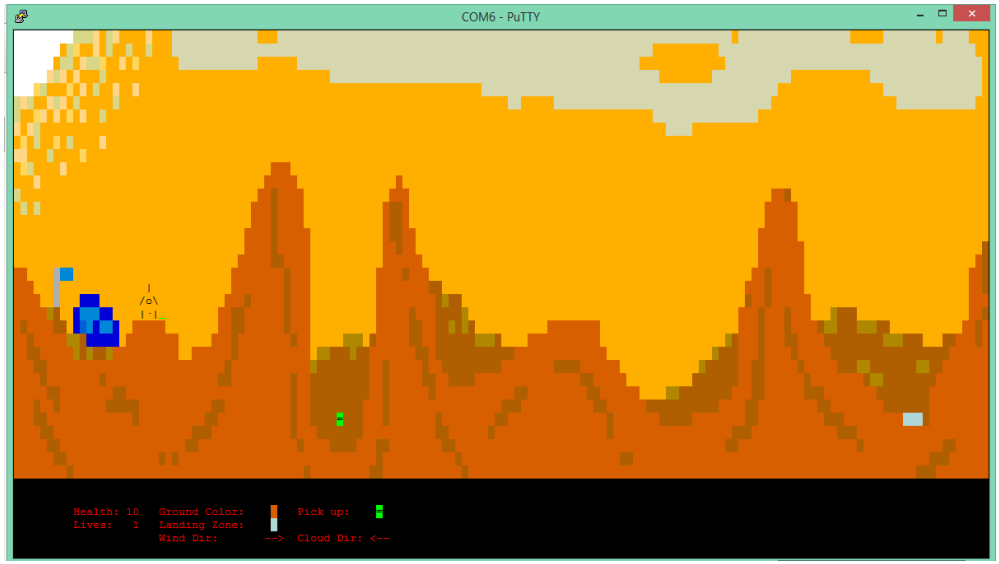
1st week – Introduction ver 0.1. Preliminary!					
	Thursday (2.6.)	Friday (3.6.)	Tuesday (7.6.)	Wednesday (8.6.)	Thursday (9.6.)
Preparation Read!!	EssentialC.pdf 1,2,5	EssentialC.pdf 3,4,6	Excerpts_STM32_Norris.pdf		
Theme background 09^{00}-10^{30} (341/22)	**Course Introduction C, Structural, HW & Compiler. Intro 1 & 2**	**C (cont.), Structural C, Array, Pointer, Structure**	**Microprocessor Architecture (1): Intro., I/O, Register**	**Exercises**	**Introduction to the Project Work**
Quiz	*~Quiz0 + Quiz1*	*Quiz2*	*Quiz3*	*Quiz4*	
10^{30}-12^{00} (341/003+015+019) (TA for Exercises)	**Exercises 1-8**	**Exercises 1-8**	**Exercises 1-8**	**Exercises 1-8**	**Exercises / Project Work** (PW, group level discussions)
Exercises 13^{00}-17^{00} (341/003+015+019) (TA for Exercises to ~16^{00})	**Exercise 1**: Learn/Use Compiler: Edit, Compile, Debug, Download & Execu **Exercise 2**: Draw window in putty with ANSI-Escape codes. Stucturing wi **Exercise 3**: Application of fixed point arithmetic. Using arrays & structure **Exercise 4**: Mini project: ball in movement within a window **Exercise 5**: Learn how to use the µP registers to read/write to the I/0. (Ref:Gpio,Irq,Timer). **Exercise 6**: Application of the onboard Timer with IRQ for implemen-ting a stop watch. **Exercise 7 (+8)**: Use the LCDs to show text and scroll, using Timer & IRQ (8 can be replaced by PW extra)			**Microprocessor Architecture (2): Display, UART**	
2nd week – Project Work					
	Friday (10.6.)	Monday (13.6.)	Tuesday (14.6.)	Wednesday (15.6.)	Thursday (16.6.)
Project Work (+ rest of Exercises) 9^{00}-12^{00} & 13^{00}-17^{00} (TA Mon/Tue)	**Milestone: Preliminary Review** +Midterm Evaluation1	**Milestone: Preliminary Review** +MidtermEvaluation1			**Written Exam 2 h., d. 16/6 (13:30 TBC)**
3rd week – Project Work					
	Friday (17.6.)	Monday (20.6.)	Tuesday (21.6.)	Wednesday (22.6.)	Thursday (23.6.)
Project Work 9^{00}-12^{00} & 13^{00}-17^{00} (TA Mon)	**Milestone: Status Review** +Midterm Evaluation2	**Milestone: Status Review** +Midterm Evaluation2			**Report Delivery & HW Delivery**

SpaceShip



SpaceShip interstellar travel mission

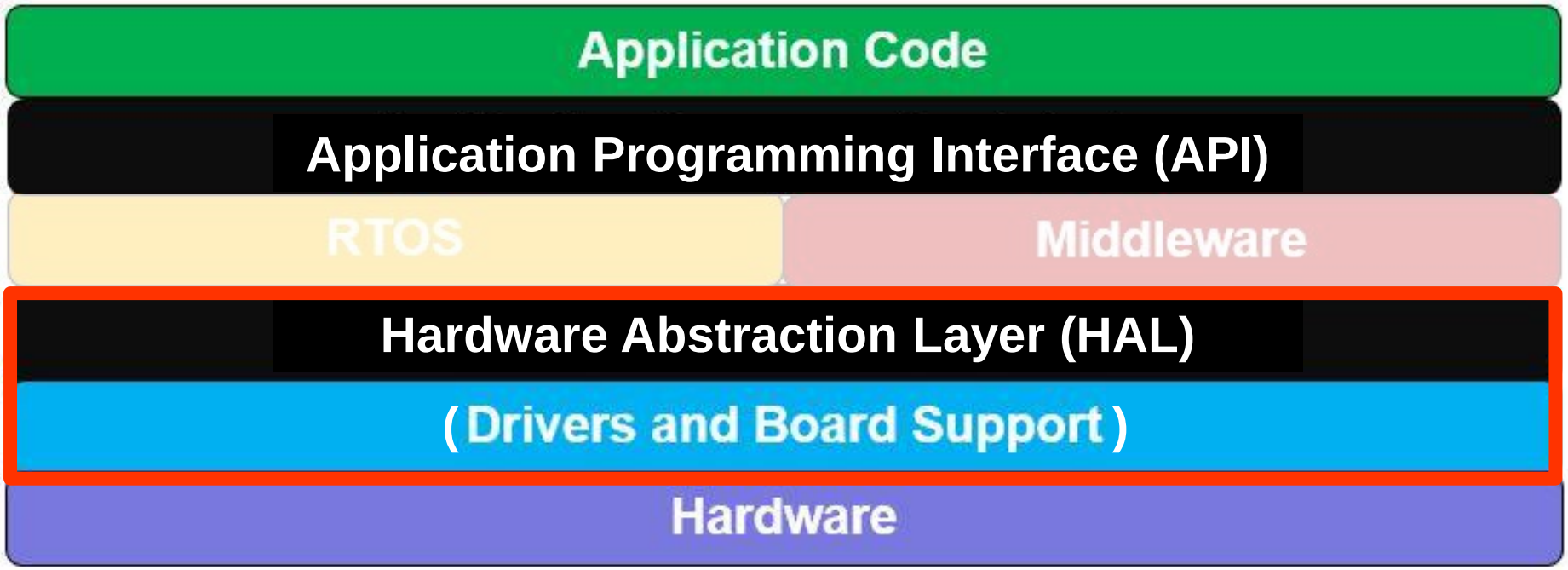
SpaceShip planet rescue mission



General Requirements for the project

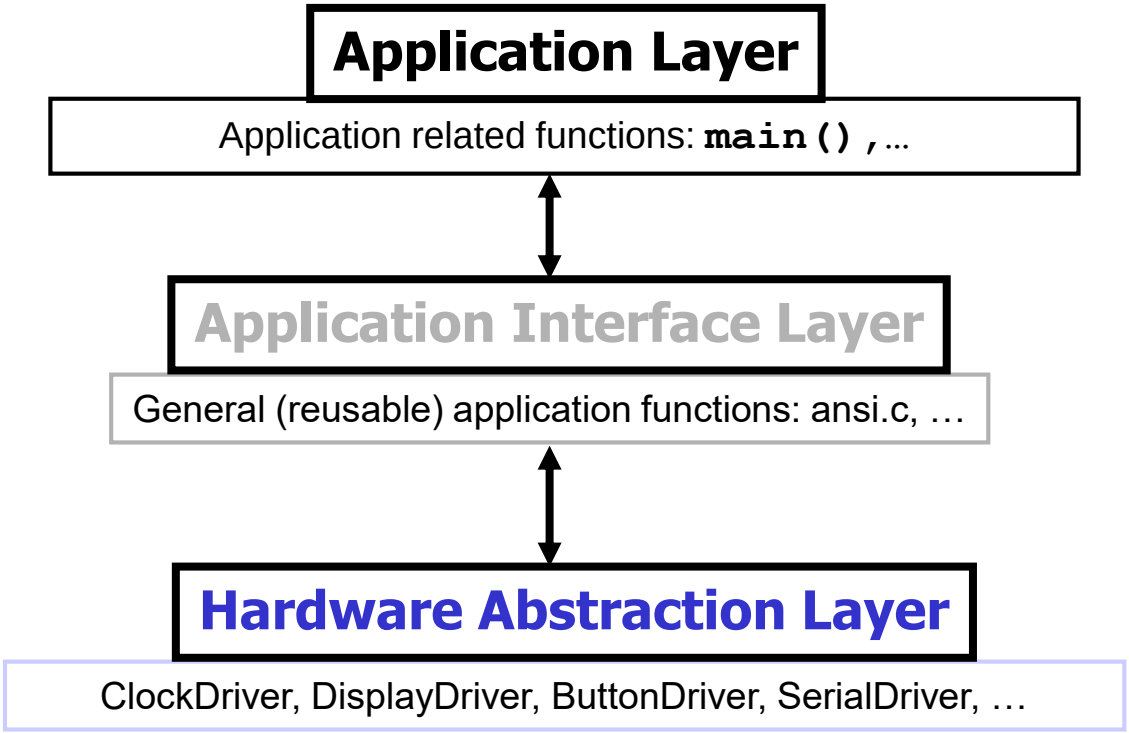
- Req. 1.- The implementation shall be carried out using the ARM Cortex M4 CPU & STM32CubeIDE v1.5.1.
- Req. 2.- The program C language code files shall be structured with clear and logical separation between functional libraries, and therefore with .h files. (Ref: Slide #13 of Course Lecture 1).
- Req. 3.- The program abstraction (or architecture) shall contain at least three layers: Application Layer (project specific functionality), API (Application Interface Layer; functions that can be reused without modification in other projects) and HAL (Hardware Interface Layer). (Ref: Slides #10-11 of Lecture 2).
- Req. 4.- The application shall use GPIOs, Timer, IRQ, a screen via Putty and the onboard LCD display.
- Req. 5.- The Application Layer shall be documented using flowcharts. (Ref: Slide #12 of Course Lecture 2, and fx. https://www.draw.io/).
- Req. 6.- The API (Application Interface Layer) and HAL (Hardware Interface Layer) shall be documented using a box containing name, variables and internal functions if any. A description for the module and resources used (eg. Hardware) if any. (Ref: Slide #13 of Course Lecture 2).
- Req. 7.- Global variables may only be used if strictly necessary. If used, a strong argumentation shall be reported as well as a clarification with the reasons for not using local variables and passing arguments to functions. Also use of global variables requires additional commenting in the source code.
- Req. 8.- All operations and numbers must use fixed-point representation / arithmetic

Programming Architecture Abstraction Model



Source: https://www.beningo.com/embedded-basics-apis-vs-hals/

Software Architecture



Application Layer: This is the part of the program that contains the high level specifications. In other words, it is independent of platform and hardware.

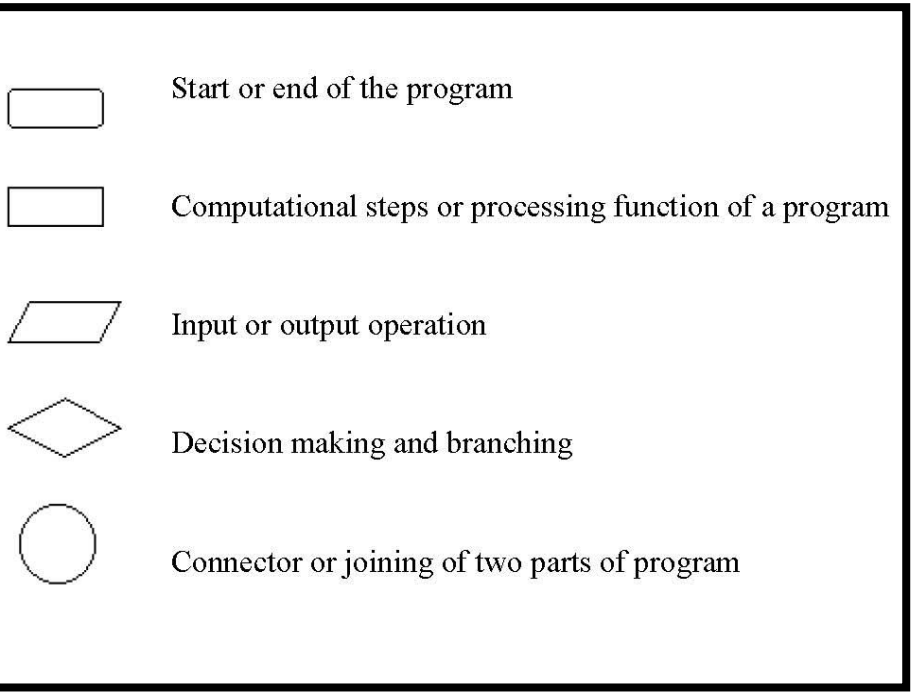
Application Interface Layer: This part contains a library with general functions. They are platform independent and core. These can be some basic/standard functions as well as some more specific.

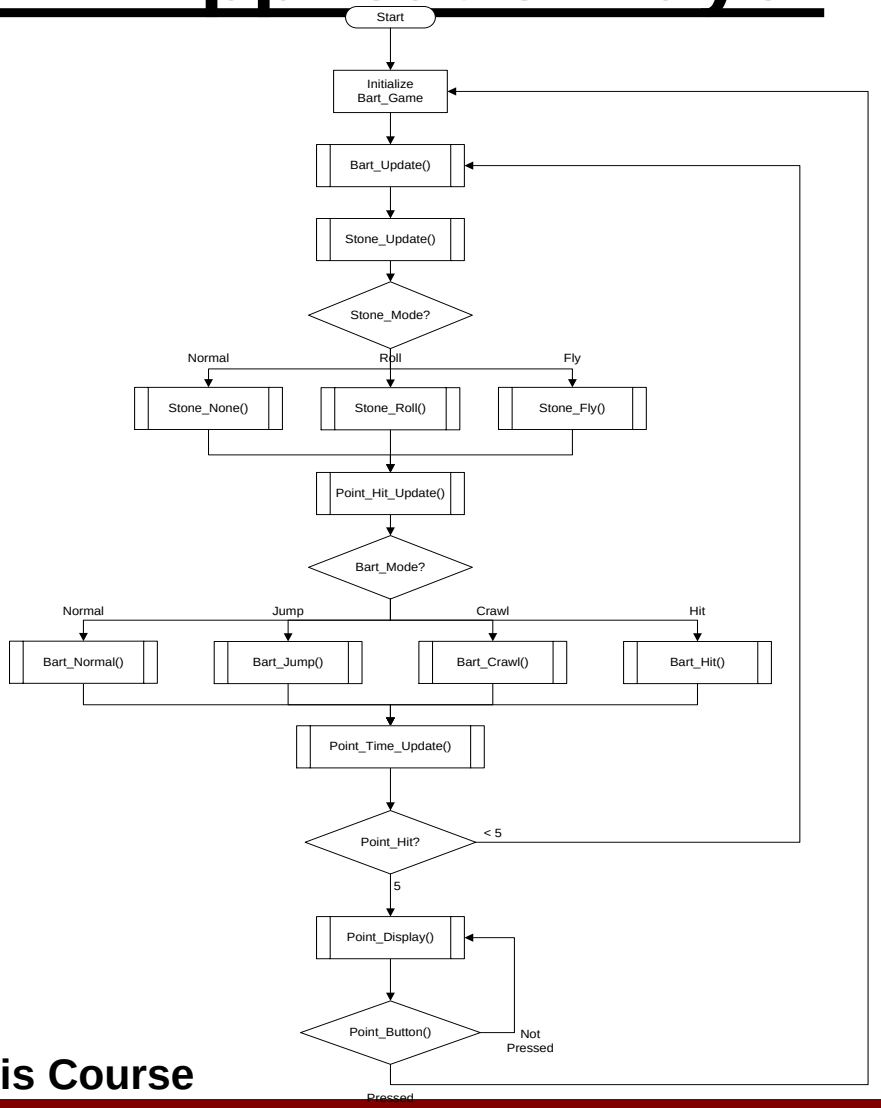
Hardware Abstraction Layer: It is the collection of functions that are strongly hardware dependent. *This is the solely layer that is allowed to "touch" the hardware*. The functions here give the possibility of interfacing between the hardware and the Application Interface Layer.

This is one of the reasons why:
It is FORBIDDEN to use *Global variables* in this Course

Software Documentation – Application layer

- Flowchart for `main()`
- Symbols for Flowcharting:





This is another reason why:
It is FORBIDDEN to use *Global variables* in this Course

Software Documentation – API & HW

Application Interface Layer

Bart.c

Public (Bart.H)

Global

char Bart_Mode

Functions

void Bart_update()[1,2,3]

void Bart_Normal()[1,3]

void Bart_Jump()[1,3]

void Bart_Crawl()[1,3]

void Bart_Hit[1,3]

1 Use Display_Bart in HAL Display.H
2 Use the buttons PD7 and PD6 of Button_Read() in HAL Button.H
3 Use virtual timer 0 in HAL Clock.H

void Bart_Jump()

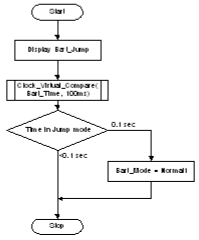
Description:

Displays Bart in lifted position in 1 second. After completion the Bart_Mode is returned to normal mode (0).

Resources:

Uses a virtual timer (0) of Clock.H to keep track of time.

Flowchart:



Hardware Abstraction Layer

GPIO_ReadInputData

Function Name	**uint16_t GPIO_ReadInputData (*GPIO_TypeDef* * GPIOx)**
Function Description	Reads the specified input port pin.
Parameters	• **GPIOx :** where x can be (A, B, C, D, E or F) to select the GPIO peripheral.
Return values	• **The input port pin value.**
Notes	• None.

TIM_PrescalerConfig

Function Name	**void TIM_PrescalerConfig (*TIM_TypeDef* * TIMx, uint16_t Prescaler, uint16_t TIM_PSCReloadMode)**
Function Description	Configures the TIMx Prescaler.
Parameters	• **TIMx :** where x can be 1, 2, 3, 4, 8, 15, 16 or 17 to select the TIM peripheral. • **Prescaler :** specifies the Prescaler Register value • **TIM_PSCReloadMode :** specifies the TIM Prescaler Reload mode This parameter can be one of the following values: – ***TIM_PSCReloadMode_Update :*** The Prescaler is loaded at the update event. – ***TIM_PSCReloadMode_Immediate :*** The Prescaler is loaded immediatly.
Return values	• None.
Notes	• None.

STM32F30xx31xxSPL, page 239/459

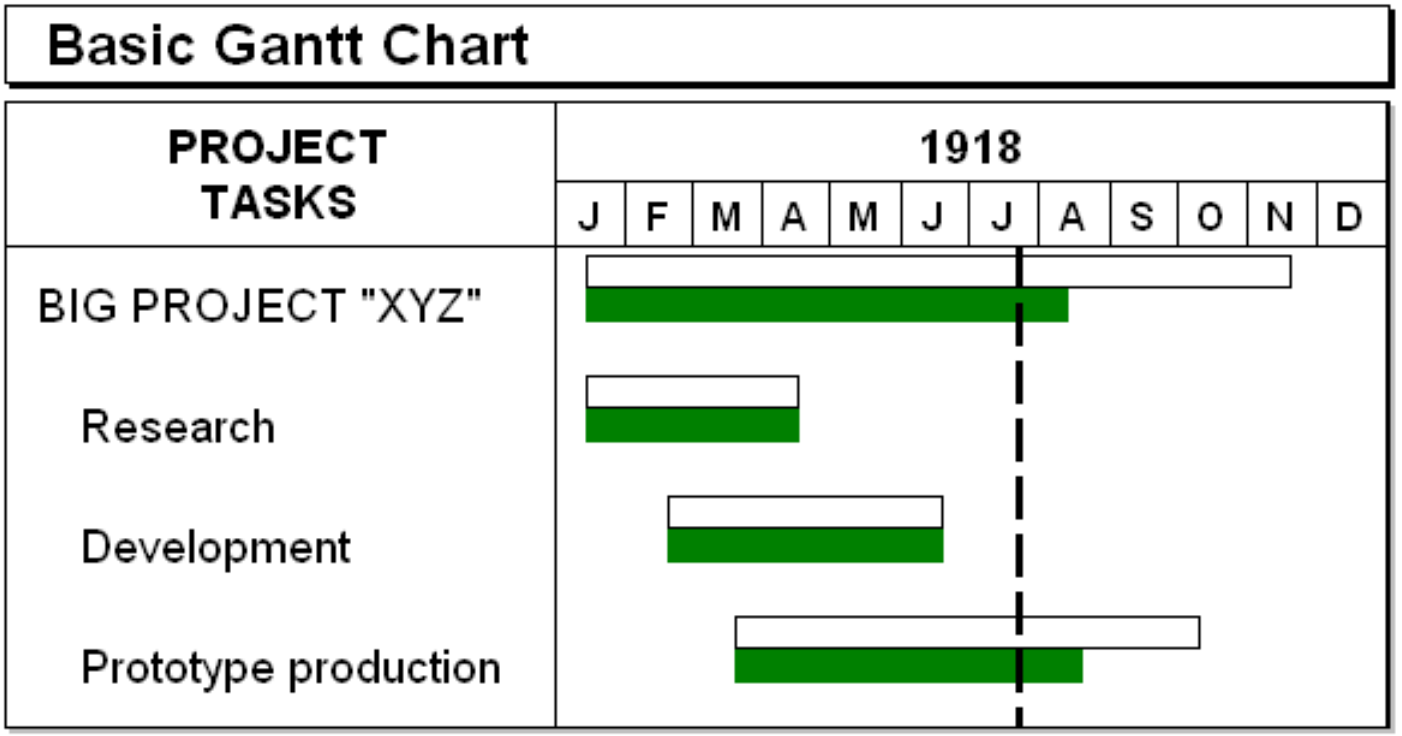
Project Schedule

- A **Gantt chart** is a graphical representation of the duration of tasks against the progression of time. It is an useful tools for planning and scheduling projects:

  - They allow you to assess how long a project should take.

  - Gantt charts lay out the order in which tasks need to be carried out.

(Source: http://www.ganttchart.com/) (FreeSoftware:http://www.smartdraw.com)



Block Diagram

- It is a pictorial representation of some process or model of a complex system. geometric shapes are connected by lines to indicate association and direction/order of traversal.

- It provides a high-level view and is used to :

  1) Establish the boundaries of a system under consideration,

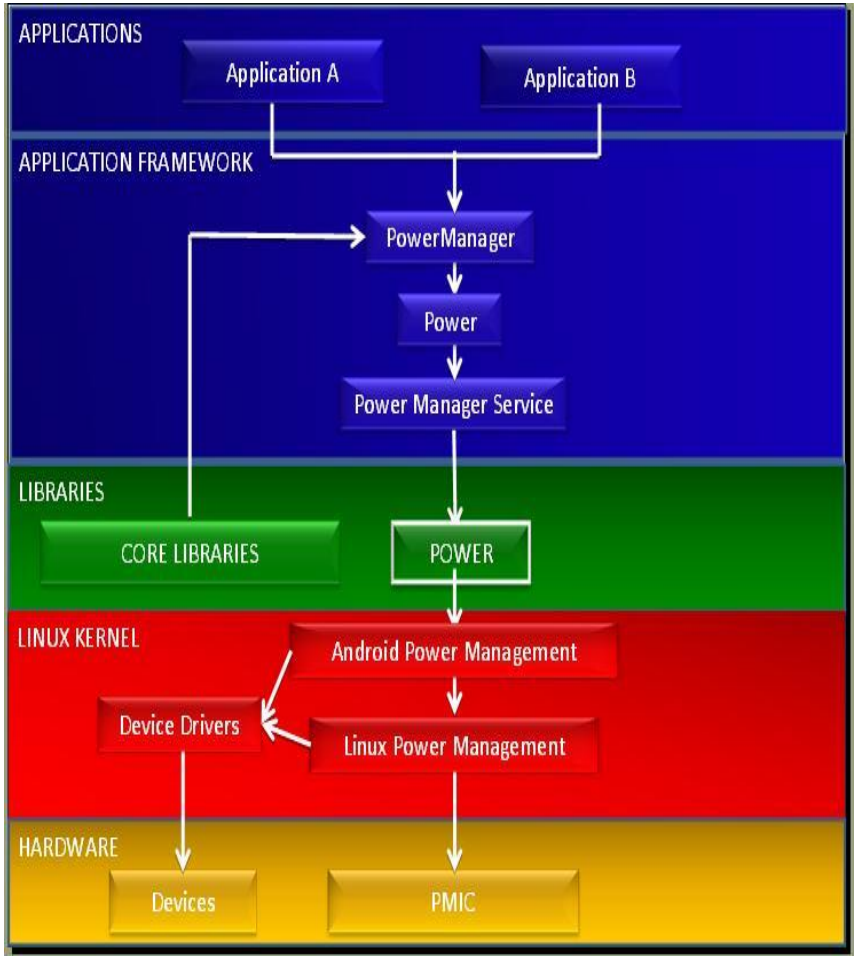
  2) Outline the elements contained within the scope of a task - helps Flow Chart

  3) Identify inputs and outputs for components within a system,

  4) Identify relationships between systems/components,

  5) Identify redundancies in systems,

  6)Establish critical paths through systems

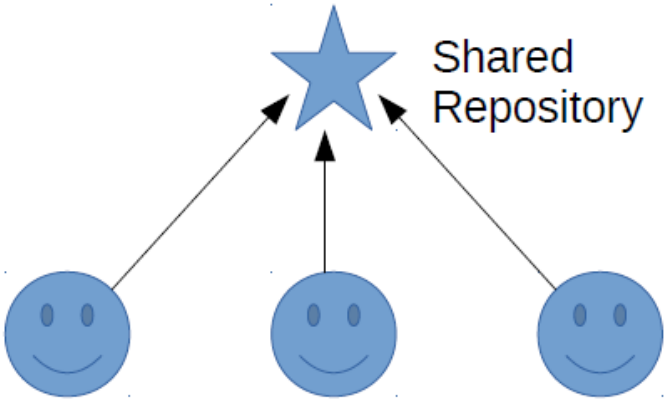


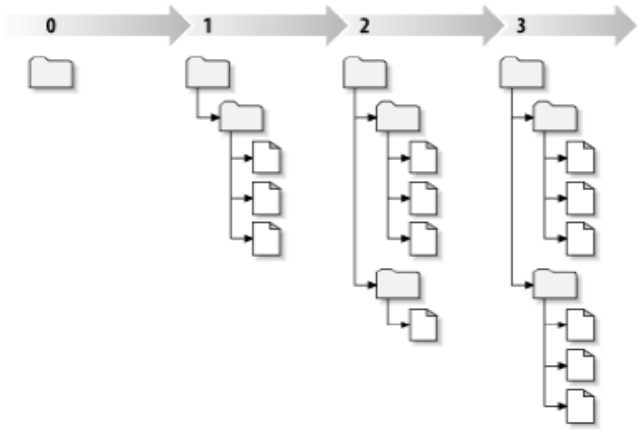
(Source: http://en.wikipedia.org/wiki/Block_diagram
http://thequalityportal.com/q_block.htm)

Example: Android GingerBread

Version Control

- What is Version Control?
  - Distributed software development tool
  - Revision control backup
- Version control software
  - Git, SVN, CSV
- Software and documentation
  - http://git-scm.com/
- Free repositories
  - www.github.com
  - www.assembla.com





Version Control - Git

Your First Git Project

- Register a user on www.github.com
  - Use your student mail for free private repositories.
- Make a new repository on github
  - Set a name and chose private.
- Download a Git tool
  - http://www.sourcetreeapp.com
- Setup the repository
  - Log in to your github account.
  - Press “clone/new”, then the globe icon.
  - Select the github repository you made.
  - Use the “destination path” to save the files at a proper location.
- Interactive Git tutorials:
  - https://try.github.io
  - http://pcottle.github.io/learnGitBranching/

Basic Git Functions

git clone git@github.com:user/project.git
- Initial command to download a repository.

git add main.c
- Adding a file to revision control.

git commit -m “Fixed bug #101”
- Saving the current state of the added files.
- Revision number is incremented.

git push
- Send all commits to the repository (github.com etc.)

git pull
- Download the newest revisions from the repository.

Git status
- Show the status of the current state.

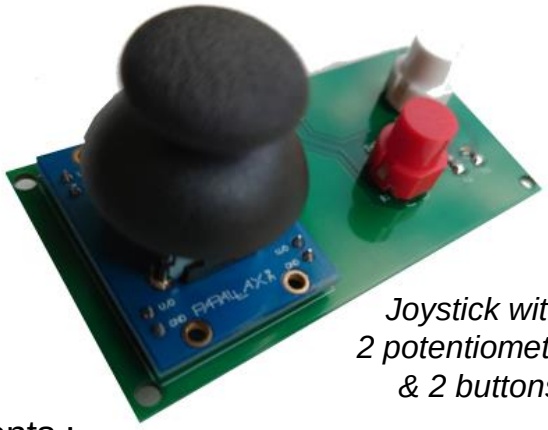
Git log
- Show all history of commits.

Git checkout
- Pulls a specific version of the code.

Implementation & Extensions for the project

There are many solutions to the problem and additional features that can be implemented. You are only limited by your imagination and the fact that you have two weeks to finalize the project.

Get inspiration at fx www.agame.com/game. The basic game shall look like one these arcade games.



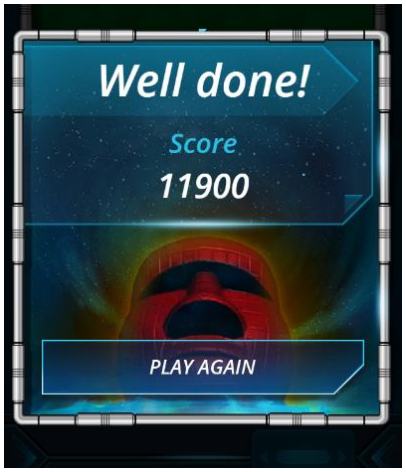
Joystick with 2 potentiometers & 2 buttons

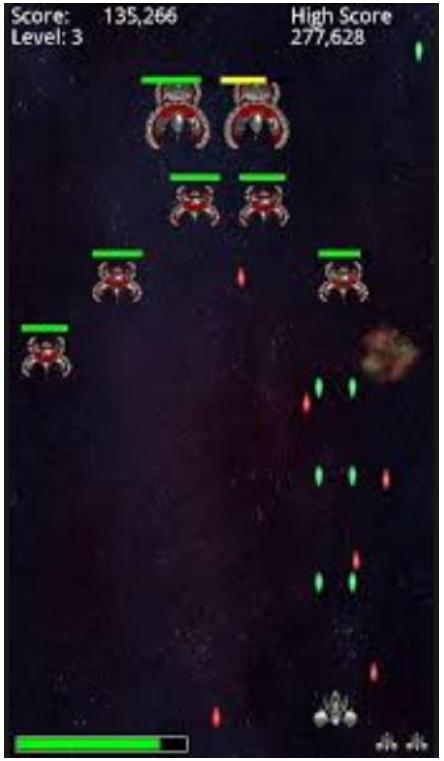
•The resulting game shall (**obligatorily**) include the following requirements :

- All of these: asteroids/planets with gravity effect, game levels with increasing speed controlled by the timer/irq, power-ups, number of lives on LCD/putty, number of points on LCD/putty, menu, help screen, and boss key.
- More than two of: multiple bullets, docking spaceships, animated figure instead of a bullet, score / high score, random delta-angle, two players, game controlled through PuTTY/Keyboard.
- More than two of: show info on RBG LED, game controlled by digital joystick, game controlled by 30010 joystick, full game on the LCD, sound from buzzer, use of onboard accelerometer.
- Other extras that may be included after completion of the above: using the RTC, using the PWM, using the DAC, using the flash memory, using an external joystick, using an ext. potentiometer, …..

At PDR we will agree on what you will implement for your project.

Inspiration





Phasing of the project

- It is important to control the implementation of a project, and therefore industrial standards recommend a balanced split of the tasks of the project into 1) resources (members of your group) and 2) time (last two weeks of the 3 week period). In order to manage the project, we foresee two **milestone** reviews:
  - Preliminary Design Review
  - Status Design Review

- **Preliminary Design Review (PDR)** / (*Fre-Mon, Jun 10th-13th*):
  Here you shall present how you plan to implement the project. The objective of this review is that you have analyzed the project scope and you present and converge into the concrete specifications requirements for the project you are going to implement. At this review it is required to present a **project block diagram**, **flow charts** for different levels and modules. A **project schedule** and plan shall also be presented, taking into account the tasks to be accomplished. (Ref: Slide #12-13 of Course Lecture 2). These will be part of your final report..

- **Status Design Review (SDR)** / (*Fre-Mon, Jun 17th-20th*):
  Here you shall present the project status and plans for its finalization. Final versions of block diagram, flow charts, schedule and plan are expected. The objective is to ensure that the project execution can be rounded-up including the documentation and report within the given time frame. You are required to present what has been completed & tested, what is undergoing and/or under testing, what is missing and the strategy for completion and testing of the ongoing tasks.

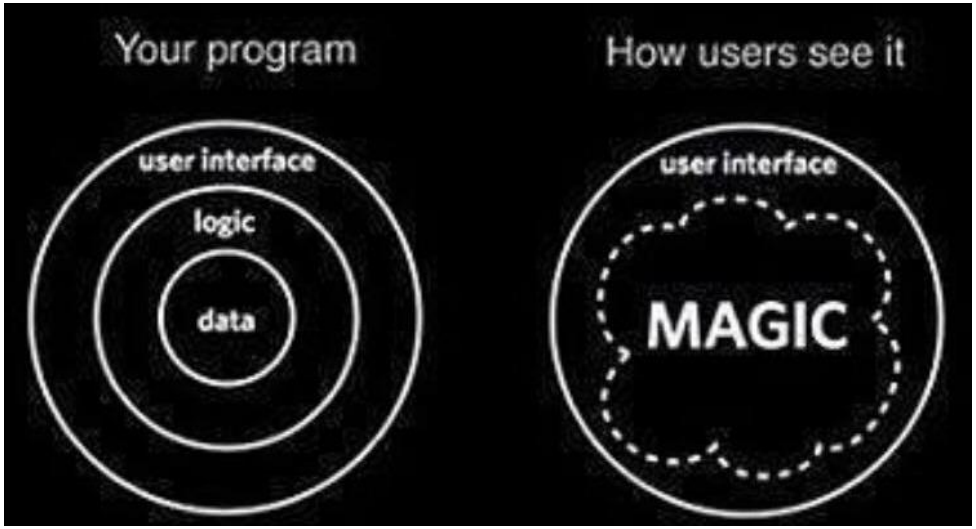
Documentation requirements

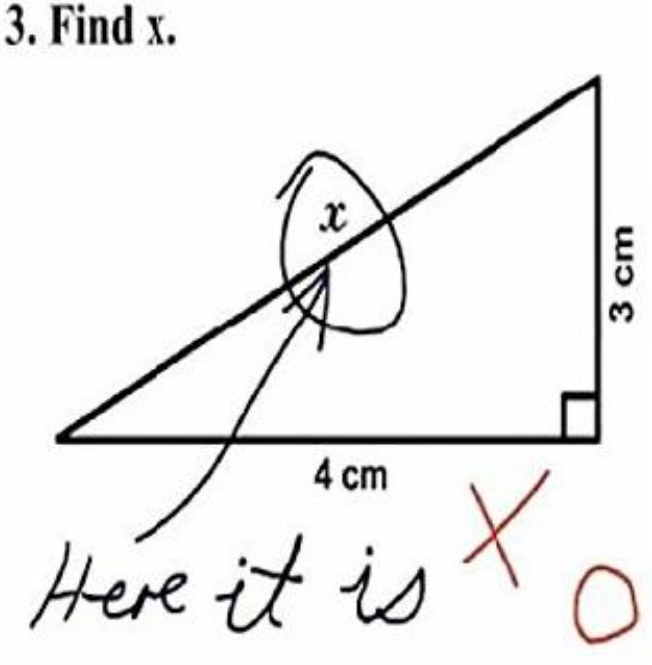
- Documentation is the most important outcome of any project. If the hardware and/or software implementation has been poorly documented it is very difficult to fix problems and/or add improvements later on. That is why you need to focus not only on the software but on the project as a whole including project management, HW, SW & documentation.

- The report (you may write it in Danish or English) documents the project work, and it should include the relevant technical aspects of the project. Here follows a suggestion for the report structure, but you may assess whether you use this or not. Formally, it shall contain at least the following information (chapters):

  - A front page containing: DTU Space, Technical University of Denmark, 30010 Programming Project, group number (you can check on cn or in the delivered hardware), group members, each: first name, last name, stud. nr., cn picture and signature.
  - Abstract (in English) **and** Resume (in Danish)
  - Introduction, which put the problem in perspective
  - Specifications requirements, for the project and with the reasons for choosing the given solution
  - Project Structure: What modules (one subsection for each) have been developed & **who created it**
  - Project verification: Strategy for project testing
  - Project plan: What elements (code, exec., test, doc., etc.) are done when (timeline), as who did it
  - Users manual for the application … that's because we want to play with your game!
  - Conclusion: what went well, what could be done better, what could be included in the future
  - References
  - Appendixes: **Developed source code**, **including .c and .h files**, journal of the exercises with the relevant code

Rapporten MATTERS!!!!!!





A journalist asked a programmer:-
What makes code bad?
No comment.

Delivery

The report shall be delivered latest on **Xxxday 2x/06 (draft on 23.6.)**:

- Two files shall be uploaded in DTU Learn:
  (under “Opgave” as group, one group member uploads &
  invites the others on the group, who have to confirm the delivery)

  - **30010_GroupXX.doc** or **pdf**: An electronic version of the report (ms word or pdf)
  - **30010_GroupXX.zip**: compressed file containing the project folder incl. all source code & a <50Mb avi/mpeg

- The **Hardware** shall also be delivered on Thu. Jun 23th latest 12:00 (my office 106 in building 327 or course databar in building 341).

Hjælpelærer

- Also next week Monday/Tuesday & gradually decreasing