**Self-driving/app-controlled car**

**Configuration management**

Sicoe Oana-Maria

Universitatea Politehnica din Timișoara,

Facultatea de Automatica și Calculatoare

**Table of contents:**

1. Project overview................................................................................................3
2. Project requirements..........................................................................................3
3. Project specifications.........................................................................................3
4. Tasks.................................................................................................................3
5. Gantt chart for the completion of tasks..............................................................4
6. Configuration management – an overview........................................................4
7. Configuration management in the context of the “Self-driving/app-controlled car” project.........................................................................................................5
8. Conclusions.......................................................................................................7
9. **Project overview:**

The project’s aim is to design and build a car of small dimensions that has two operating modes:

* self-driving mode;
* app-controlled mode;

1. **Project requirements:**

**Functional requirements**:

* self-driving mode must ensure that the car will successfully avoid collisions;
* in situations where the car is surrounded by obstacles on a 50 cm radius, the car will come to a full stop;
* in self-driving mode, when the car will reach an obstacle, it will assess its surroundings, and take the route where there are either no obstacles or the obstacle is the furthest;
* in app-controlled mode the application controlling the car must be designed for iOS devices;
* in app-controlled mode, the communication between the car and the application must be successful, if the distance between the two is at most 10 meters;

**Technical requirements**:

* iOS implementation of required software;
* application must be accompanied by documentation;
* when in app-controlled mode, only one application must be able to interact with the car at a time;

1. **Project specifications:**

The project will use the following components:

* detection of the obstacles will be achieved by a rotating ultrasonic sensor (HC-SR04);
* rotation of the sensor will be ensured by a servomotor (TowerPro MG 90S Microservo);
* all processing will be done by an Arduino nano board;
* connection between the device and car will be ensured by a bluetooth sensor (ZS040);
* two power supplies will be used:

1. 2x1.5V : will power the motors connected to the wheels;
2. 2x3.7V: will power the Arduino board;

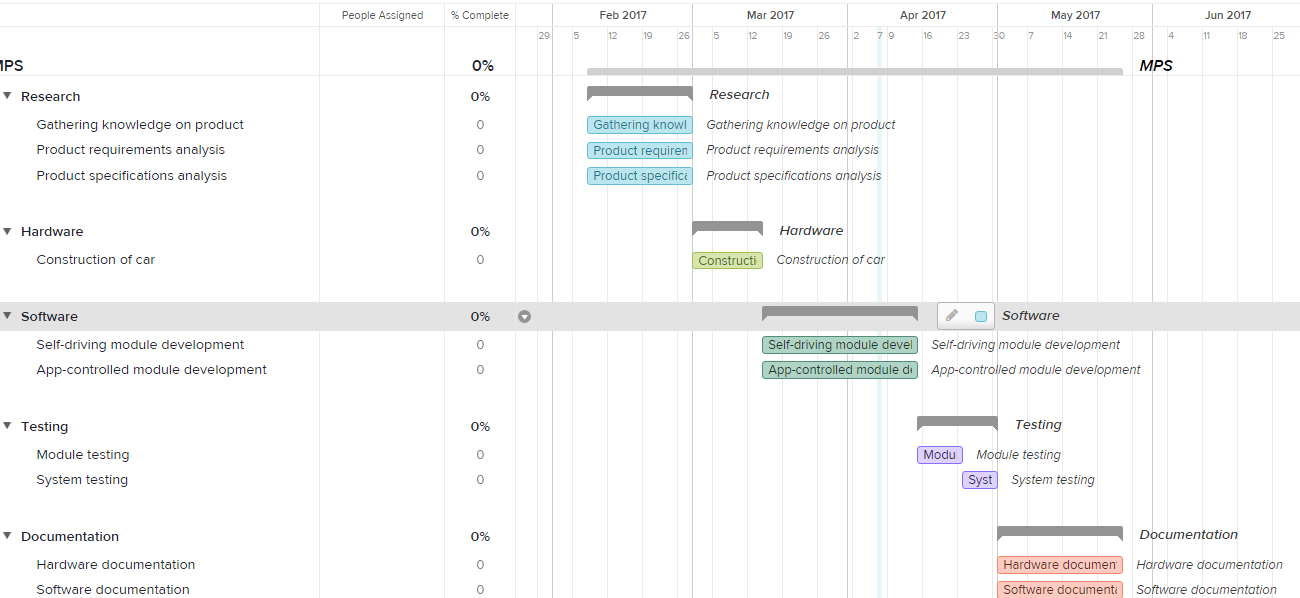
**4. Tasks:**

1. Project research and product specifications and requirements analysis;
2. Hardware construction of the car;
3. Software development: -self-driving mode module;

-app-controlled mode module;

-iOS application to control the car;

1. Software module testing and debugging;
2. Software system testing and debugging;
3. Documenting the product;
4. **Gantt chart for the completion of tasks**



1. **Configuration management – an overview**

Configuration management (CM) applied over the life cycle of a system provides visibility and control of its performance, functional, and physical attributes. CM verifies that a system performs as intended, and is identified and documented in sufficient detail to support its projected life cycle. The CM process facilitates orderly management of system information and system changes for such beneficial purposes as to revise capability; improve performance, reliability, or maintainability; extend life; reduce cost; reduce risk and liability; or correct defects.

CM emphasizes the functional relation between parts, subsystems, and systems for effectively controlling system change. It helps to verify that proposed changes are systematically considered to minimize adverse effects. Changes to the system are proposed, evaluated, and implemented using a standardized, systematic approach that ensures consistency, and proposed changes are evaluated in terms of their anticipated impact on the entire system. CM verifies that changes are carried out as prescribed and that documentation of items and systems reflects their true configuration.

A structured CM program ensures that documentation (e.g., requirements, design, test, and acceptance documentation) for items is accurate and consistent with the actual physical design of the item. In many cases, without CM, the documentation exists but is not consistent with the item itself. As changes inevitably occur in the requirements and design, they must be approved and documented, creating an accurate record of the system status.

The CM process for both hardware- and software-configuration items comprises five distinct disciplines. These disciplines are carried out as policies and procedures for establishing baselines and for performing a standard change-management process.

The five disciplines are:

1. **CM Planning and Management**: a formal document and plan to guide the CM program that includes items such as:
   * personnel
   * responsibilities and resources
   * training requirements
   * administrative meeting guidelines, including a definition of procedures and tools
   * baselining processes
   * configuration control and configuration-status accounting
   * naming conventions
   * audits and reviews
   * subcontractor/vendor CM requirements
2. **Configuration Identification (CI):** consists of setting and maintaining baselines, which define the system or subsystem architecture, components, and any developments at any point in time. It is the basis by which changes to any part of a system are identified, documented, and later tracked through design, development, testing, and final delivery.
3. **Configuration Control**: includes the evaluation of all change-requests and change-proposals, and their subsequent approval or disapproval. It covers the process of controlling modifications to the system's design, hardware, firmware, software, and documentation.
4. **Configuration Status Accounting**: includes the process of recording and reporting configuration item descriptions (e.g., hardware, software, firmware, etc.) and all departures from the baseline during design and production. In the event of suspected problems, the verification of baseline configuration and approved modifications can be quickly determined.
5. **Configuration Verification and Audit**: an independent review of hardware and software for the purpose of assessing compliance with established performance requirements, commercial and appropriate military standards, and functional, allocated, and product baselines. Configuration audits verify that the system and subsystem configuration documentation complies with the functional and physical performance characteristics before acceptance into an architectural baseline.

The traditional software configuration management (SCM) process is the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

The SCM process further defines the need to trace changes, and the ability to verify that the final delivered software has all of the planned enhancements that are supposed to be included in the release.

1. **Configuration management in the context of the “Self-driving/app-controlled car” project:**

As work on the project will begin, certain specifications or requirements may need to be modified, and some may even need to be added; Changes in the code will appear, to reflect the new requirements and specifications;

Configuration management in this particular case will aim to update requirements and specifications, to ensure consistency of the product’s performance, functional and physical attributes with its requirements, design and operational information at any given time throughout the project’s life, and to track changes brought to the code. This will be done with the help of documents and a versioning system;

The project will be separated into tasks, and each participant will have tasks assigned to them; Participants will have access to the project Gantt chart, where the new tasks will appear, along with the name of the assigned responsible of the change needed; once the change has been made, tested and integrated the task can be set to DONE;

As a whole, the project will be separated into two components: the code component, and the documents component.

The code component will comprise of:

* self-driving module;
* app-controlled module;
* iOS application code;

The documents component will comprise of:

* Project requirements document;
* Project specifications document;
* General project documentation of the final product;
* Project test specification- one for each code component;

Changes will be tracked with the help of a versioning system (specifically PTC Integrity); each user will create an account, through which their own modifications can be tracked throughout the whole project development;

When a change of code is done, a series of steps will be followed:

1. the last revision of the project specifications and requirements will be checked;
2. the newly added functionality will be tested;
3. the code change will be checked-into the versioning tool, with a relevant description of the modification brought;
4. the general documentation of the product will be updated if needed;
5. a new test will be added in the project test specification, if possible and needed; this test will describe a way to test the newly added functionality;
6. checked-in code will be integrated in the base project;

Once the final version of the product is done, system testing will begin.

The first test will verify that all the requirements and specifications are consistent with the final product; to achieve this, the respective documents will be checked, and the product will be tested in parallel;

The second test will aim to verify that all the tests added in the projects test specification document pass in the final form of the product;

1. **Conclusions**

In the context of the “Self-driving/app-controlled car” project, configuration management ensures consistency of the product’s functionalities with the requirements and specifications throughout the whole product development, and an efficient and intuitive process.

By dividing the project into tasks and assigning them to project participants, an efficient workflow is achieved. Establishing a set of steps to be followed with each code change reduces the bug risks, and using a versioning tool for the code development and document changes makes it easy to find bug sources, or detect with which change one may have started to appear;

Using documents to track modifications and corresponding tests ensures that when the final product exists, it comes with a set of tests to verify that each functionality works. This makes it almost impossible to forget to test some parts of the code in the final product.