Projekt AutoCar
Rapport

Diplomingeniør Elektronik
4. Semesterprojekt forår 2016

Ingeniørhøjskolen Aarhus Universitet
Vejleder: Lars G. Johansen

27. maj 2016

Jonas Baatrup
Studienr. 201405146

Troels Ringbøl Brahe
Studienr. 20095221

Nicolai H. Fransen
Studienr. 201404672

Jesper Kloster
Studienr. 201404571

Rasmus Harboe Platz
Studienr. 201408608

Nicolai Bonde
Studienr. 201404519

Emil Jepsen
Studienr. 20092013

Ansvarsområder Tabellen nedenfor viser de primære ansvarsområder for hver af gruppens medlemmer.

Navn	**Ansvarsområder**
Jonas Baatrup	Motorstyring, Motor-design, Regulering
Nicolai Bonde	RPi kode, Bil-design, Forsyning
Jesper Kloster	Servo, Vejbanesensorer, Tachometer
Nicolai Fransen	Servo, Vejbanesensorer, Tachometer
Troels Brahe	Sonar, Regulering, RPi kode
Emil Jepsen	Motorstyring. Regulering, SPI
Rasmus Platz	TCP, GUI

Tabel over ansvarsfordeling i projektet

Resumé

Denne rapport beskriver udviklingen af et 4. semester-projekt på IHA. Problemstillingen omhandler design og implementering af en selvkørende bil, AutoCar. AutoCar har en DC-motor til fremdrift, en servo-motor til retningsstyring, en sonarsensor til afstandsbedømmelse, en vejbanesensor i hver side til detektion af vejbanestriber, to batterisensorer til at vise batteriniveauerne, et tachometer til at finde AutoCar's fart, og intern logik, regulering, og kommunikation for at få det hele til at fungere sammen. Brugeren kan via interfacet starte og stoppe AutoCar, få udskrevet status fra sensorne, sætte farten og anmode om overhaling. AutoCar skal selv sørge for at køre inden for vejbanestriberne.

AutoCar er udviklet med en PSoC 4 og en Raspberry Pi 2b, der tilsammen fungerer som kontrolenhed for AutoCar. GUI'et er designet med QT Creator i en Linuxterminal.

Udviklingsprocessen har båret præg af iterative værktøjer som SCRUM og V-modellen, og med fokus på design fra bunden og op. ASE-modellen for projektudvikling er også benyttet i den tidsmæssige planlægning. Brugen af disse modeller har hjulpet i udviklingen af projekt. Projektprocessen er mundet ud i en prototype hvor størstedelen af funktionaliteten er implementeret.

Abstract

This report describes the devleopment of a 4th semester project at IHA concerning the design and development of a self-driving car, AutoCar. AutoCar has a DC-motor for propulsion, a servo for directional control, a sonar sensor used for ranging, a road sensor in each side to detect road markings, two battery sensors to detect and display battery levels, a tachometer to measure the speed of AutoCar, and a lot of internal logic, regulation, and communication to make it all work. The user can use the interface to start and stop AutoCar, print the status from the sensors, set the speed, and a request overtaking. AutoCar is itself responsible for driving within the road markings.

AutoCar is developed with a PSoC 4 and a Raspberry Pi 2b, which collectively function as the control unit. The GUI is designed with QT Creator in a Linux terminal.

The devleopment process has been influence by iterative tools such as SCRUM and the V model, and a strong focus on a bottom-up approach to the design process. The ASE Development Model has been used for chronological planning, and redmine along with SmartGit has been used to keep track of project files and resources. The use of these models and tools has ensured a competently devleloped system.

Indhold

1 Krav

Kravene til produktet er analyseret vha. MoSCoW-metoden[**MoSCoW**]. MoSCoW-metoden inddeler kravene til et system i fire kategorier. De fire kategorier er: Must, Should, Could og Won't.

Must
- Køre fremad og dreje
- Holde en fastsat brugerdefineret fart
- Holde sig inden for vejstriber
- Rapportere hastighed til bruger
- Trådløs kommunikation med brugeren

Should
- Tilpasse fart efter forankørende
- Monitorere og rapportere batteri-niveau
- Overhale ved brugerkommando
- Stoppe ved forhindring på kørebanen

Could
- Blinklys og kørelys
- Undvige forhindring
- Grafisk brugergrænseflade
- Køre baglæns

Won't
- Have navigation
- Læse vejskilte
- SmartPhone-applikation til styring

Ud fra ovenstående MoSCoW-model er der fremstillet en række krav. Der er funktionelle krav, som defineres ved use cases, og ikke-funktionelle krav.

1.1 Kravspecifikation

Kravspecifikationen indeholder kravene til projektet. Use casene er funktionelle krav til systemet set fra brugerens perspektiv. I diagrammet nedenfor beskrives brugerens og systemets sammenhæng til de funktionelle krav for produktet. Brugeren kan via kommandoer på grænsefladen styre bilen, ved at initiere de forskellige use cases. De andre aktører ses til højre på figuren, og linjerne viser hvilke use cases de er involveret i.

1.1.1 Use cases

De seks use cases forklares i dette afsnit. For yderligere beskrivelse af de forskellige use cases, se dokumentationens kravspecifikation.

Use case 1: Start bil
Use casen starter bilen når brugeren vælger "Start" via grænsefladen. Herefter vil bilen monitorere sensorinputs, udføre motortjek og rapportere status til brugeren via grænseflade.

Use case 2: Rapportér driftstatus
Use casen rapporterer om driftstatus når brugeren vælger "Status" via grænsefladen. Dette indebærer batteriniveau, bilens fart, status på vejbanesensorer og afstand til objekt foran.

Use case 3: Ændr fart
Brugeren indstiller den ønskede fart via grænsefladen og use casen justerer derefter farten til den ønskede værdi.

Use case 4: Følg vejbane
Use casen sørger for at bilen korrigerer sin retning i forhold til vejbanen. Dette er den eneste use case, som ikke initieres af brugeren.

Use case 5: Overhal
Use casen overhaler et forankørende objekt når brugeren anmoder om en overhaling via grænsefladen.

Use case 6: Sluk bil
Use casen slukker bilen når brugeren vælger "Sluk" via grænsefladen. Her vil bilens fart sættes til nul samt slukke for sensorer, lys og motor. Dette udskrives på grænsefladen.

1.1.2 Ikke-funktionelle krav

Kravene for kvaliteten af bilen beskrives ved ikke-funktionelle krav. Disse krav er beskrevet under kravspecifikationen i dokumentationen.
Kravene indebærer bilens dimensioner i form af størrelse og vægt. Der stilles krav til bilens topfart og sonarens rækkevidde. Yderligere er der krav til, at brugeren skal kunne kommunikere med bilen via en grafisk brugergrænseflade.