Modeling, Control and Automatic Code Generation for a Two-Wheeled Self-Balancing Vehicle Using Modelica

Joint Master’s Thesis project: Department of Automatic Control LTH, Dassault Systèmes AB Lund

By

Carlos Javier Pedreira Carabel, Andrés Alejandro Zambrano García

I. INTRODUCTION

A two-wheeled self-balancing vehicle consists of an unstable platform with two wheels and a handle bar. This vehicle keeps its balance and is able to safely transport a rider using his or her tilt angle as drive command. When the rider leans forward, the vehicle moves forwards and when the user leans backward, the vehicle moves backwards. A popular example of a two-wheel self-balancing vehicle is the Segway PT.

Modelica is a programming language which allows the modeling and simulation of complex systems. There are many computer tools based on Modelica, Dymola is one of them. During this project various features of Modelica and Dymola were tested on the design of a control program for the ElektorWheelie which is a programmable Segway-like vehicle (Figure 1).

The main goal of this project was to create a control program for the vehicle’s computer using Modelica features. Additionally, a communication program which allows the remote controlling of the vehicle using a personal computer was created in Dymola.

II. METHODOLOGY

a) Modeling
In order to design a control program for the vehicle’s main computer, it was useful to create a computer model which behaves like the real vehicle. Modelica’s tools in mechanical design were used for this purpose.

b) Control
A set of mathematical control equations was determined in order to maintain the balance of the vehicle. The control equations were tested using Modelica features in control systems and the 3D model of the vehicle.
The control equations had as inputs measurements coming from sensors in the platform and the wheels, and as output a velocity signal to drive the wheels. Different methods for obtaining measurements from the sensors were tested as part of the control design.

c) Programming
After the control equations were defined, these had to be programmed in the vehicle’s main computer. The computer in the vehicle is just a small microprocessor so the complex control equations had to be programmed considering its limited capabilities.

A new Modelica library (Embedded Systems Library) has tools to automatically generate computer code for simple microprocessors. These tools were compared to manual programming of the control equations.

d) Communication Program
A computer program in Dymola was created to establish communication between a personal computer and the vehicle. Figure 2 shows an overview of the communication program.

III. RESULTS

a) Modeling
The computer-generated mechanical model was made to behave as the vehicle. Figure 3 shows the generated 3D animated model.

b) Control
Using Dymola, the control equations were simulated on the computer model. The simulations allowed to select the most suitable sensor processing equations as well as to tune the calculated control equations.

c) Programming
The control equations were programmed in the vehicle’s computer.
Two programs were tested, first the manually generated program and second the automatically generated program. Some minor modifications of the automatically generated code were done in order to adapt it to the vehicle’s computing capabilities.

Test rides were carried out in order to compare the performance of both codes (Figure 4). There were no differences between both codes; during both tests the ride experience was smooth and the vehicle was easy to control after some minutes of practice.

![Figure 4. Outdoor test ride](image)

**d) Communication Program**

The communication link between the computer and the vehicle was set up using a Bluetooth device in order to avoid having to use wires. The incoming data from the vehicle was processed as plots in the Dymola simulation environment. Different turn and go forward commands were sent to the vehicle in order to test the remote control mode (Figure 5). The vehicle responded to the commands in the expected way showing that the communication was established correctly.

![Figure 5. Self-stabilizing and remote control mode test](image)

**IV. CONCLUSIONS**

During this project many Modelica features were tested in the modeling, control, and communication of a complex system. Modelica was successfully implemented in the design of a control strategy and the generation of code for a simple external computer.

The designed control code implemented on the vehicle let non-expert users to ride vehicle in a smooth way after some minutes of practice.