Prof. Bernd Finkbeiner, Ph.D.
Dipl.-Inf. Rüdiger Ehlers
Markus Rabe, M.Sc.
Sebastian Hahn, B.Sc.

Summer term 2012
Project

Embedded Systems

Scenario - Robots just wanna have fun!

The e-puck wants to have fun on a playground. Teach him how to use a seesaw!

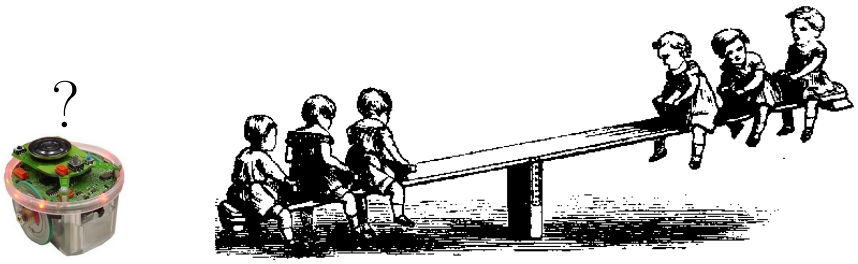

Figure 1: Seesaw in action.

Unfortunately, nobody wants to play with the e-puck, so he has to drive from one side of the seesaw to the other to make it move. Take a look at a our prototype: `http://www.youtube.com/watch?v=srOu3_l4G9M`

Task 1: Model

Download the Simulink model for this project from our homepage. It models already the dynamics of a robot on a seesaw. Extend the Simulink model, such that the discrete controller gets suitable input from the accelerometer and such that the actuators have an appropriate effect on the environment. That is, you are supposed to find out and model what data about the environment the accelerometer provides. Then, develop a discrete controller with Stateflow that lets the robot move back and forth on the seesaw.

Task 2: Implementation

(a) Write a feedback controller in C that implements the discrete controller you modelled in Taks 1.

(b) E-puck finally found a friend! A second robots wants to join the fun on the seesaw. However, driving on the seesaw at the same time might cause collisions and collisions hurt. Extend your implementation, such that your robot avoids collisions. Use the 8 IR sensors to detect close by objects and react appropriately. You may assume that the second robot will not actively crash into yours. You may include this in your Simulink model, but we do not require it.

How to start

In Task 2 you will implement C code for an embedded processor. You will need to work with a cross-compiler that is provided by the manufacturer of the processor, Microchip. Further, you probably want to work with the integrated development environment (IDE) MPLAB X, which is also provided by Microchip, and a project setup for this IDE that we provide. The following sections include pointers and important technical information that you will need for the project.

Hardware

GCtronic is an e-puck manufacturer and provides a great wiki (see links below) with technical information and HOW-TOs. You do not have to read through all of the documents, but if you feel you need to know more about the hardware platform, start searching there.

The e-puck works with the dsPIC30F6014A, a 16-bit RISC CPU based on the Harvard architecture that runs with 60 MHz (30 MIPS). Although the compiler hides the memory management, you should keep in mind that the platform has a total of 8 KB ram, of which only about 2 KB can be used for dynamic memory (stack+heap).

In the code you can find hints on how to work with the sensors and actuators. Libraries for sensors, actuators, and bluetooth communication are included. We recommend reading the important code sections to get an overview.

Tools

- Download and install the compiler MPLAB C30 or the newer version XC16. Depending on the precise version of the compiler, the installer will ask you for an activation key. Enter nothing, press "Next", and select the free compiler version. The following steps describe the configuration steps needed for the older version (C30). In the new version (XC16), the place of several options changed slightly.
- Download and install the IDE MPLAB X.
- Download and unzip the project file.
- Open the IDE and load the project via the "Open Project" dialog.
- In the "Open Project" dialog, check the boxes for "Open Required Projects" and "Open as Main Project."
- In the project view (box on the left side of the IDE) check the following things in the project setup: "Linker files" contains the linker script p30F6014A.gld.
- In the project properties (Right-click on main project → properties) you find on the left side different categories of properties. Please check the following items:
  - In the category "Conf:[default]" make sure your compiler is found and selected. Select not just the compiler type but a concrete version of the compiler.
  - In the category "Libraries" the projects "a_d_adv", "a_d", "motor_LED_adv", "motor_LED", and "e_uart_char" should occur in precisely this order.
  - In the category "C30 (Global Options)" make sure the output file format COFF is selected.
  - In the category "pic30-gcc" check whether the options "Additional warnings" are selected and whether the "Include directories" list the following strings: "lib/motor_led/advance_one_timer", "lib/uart", "lib/a_d/advance_ad_scan", and "lib/motor_led" in this order. Change the selector "Option categories" on the right side in the top area to "Memory Model" and make sure that "Large code model" as well as "Large data model" is selected.
  - Select the category "pic30-ld" and check that "Heap size" is set to 500 bytes, "Stack size" may be left empty. For the Option category (the selector on the top of the right side) "Libraries" make sure "Library directories" includes "`C:\Program Files (x86)\Microchip\xc16\v1.00\lib`" (if necessary, adapt to version and location of your compiler!). This directory provides processor specific header files. In the Option category "Diagnostics" the options "Display memory usage" and "Warn on section realignment" should be checked.

- If any of these options were not as expected, make sure also the sub-projects, which we use as libraries, have the correct project setup.
- Compile via the hammer symbol in the top bar.
- Voi là! You have created a '.hex' file that you can use to flash the e-puck.

Linux

You may use Linux for the project, but be aware that you will might encounter problems and less support from our side. You find an installation script for the tool chain in our wiki.

Testing code on the robots

Whenever you want to test your code on the robots, come to the office 533 in building E1.3. To prevent that too many groups want to try out their code at the same time, please "check in" via the doodle page (link below) we set up.

Flashing the e-puck

Flashing the e-pucks works via a bluetooth connection, and we offer you to flash it for you with our computers. In case you want to be able to flash the e-puck yourself, we refer to the section "Bootloader" in the GCtronic Wiki.

Resources

1. E-puck homepage:
   `http://www.e-puck.org/`
2. The wiki we created for our e-puck projects:
   `https://ludwig.cs.uni-saarland.de/robots/`
   Username: embeddedsystems2012, password: 123geheim
3. The wiki of GCtronic (one of the manufacturers of the e-pucks):
   `http://www.gctronic.com/doc/index.php/E-Puck`
4. More information about the processor:
   `ww1.microchip.com/downloads/en/DeviceDoc/70143D.pdf`
5. The bootloader required to flash the e-puck:
   `http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm`
6. The project files:
   `http://react.cs.uni-sb.de/teaching/embedded-systems-12/downloads/SeesawRobot.zip`
7. The doodle poll for requesting slots to work with the robots:
   `http://www.doodle.com/em4hhz3463ah7hd4`
8. MPLAB XC16 (the compiler) download page (left side, downloads → XC16 → Windows/Linux/OS X):
   `http://www.microchip.com/pagehandler/en_us/promo/mplabxc/`
9. MPLAB X (the IDE) download page:
   `http://www.microchip.com/pagehandler/en-us/family/mplabx/`
10. You can download the 'old' C30 compiler via these links:
    `http://ww1.microchip.com/downloads/mplab/X_Beta/mplabc30-v3.30c-linux-installer.run`
    `http://ww1.microchip.com/downloads/mplab/X_Beta/mplabc30-v3.30c-windows-installer.exe`.