Haptic Human-Robot interfaces: Lab 0

Assistants : *Ali Reza Manzoori (ali.manzoori@epfl.ch)*
Zeynep Özge Orhan (zeynep.orhan@epfl.ch)
Evgenia Roussinova (evgenia.roussinova@epfl.ch)

1 Installation of the requisite software

1.1 MATLAB

Install MATLAB R2016b or later. To use the MATLAB interface for communicating with the haptic paddle (which is usually not required), the only necessary toolbox is the "Instrument Control Toolbox". For lab 2, Simulink will also be necessary.

1.2 Microcontroller firmware development environment

1. Download and install Java Runtime Environment from Oracle's website (scroll down to “Java SE 8”):
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
   - For Windows/Linux, “JRE” is sufficient, 64-bit version if possible.
   - For macOS, you need “JDK”.
2. Download and install the System Workbench IDE from the links below[1]. It is highly recommended to install it in the default folder suggested by the installer, but if you want to change it, make sure to select an empty folder. The installer might permanently delete any previously existing files in the installation folder.
   - For Windows: https://drive.switch.ch/index.php/s/dyM0IEzTqf6GDU6/download
   - For macOS**: https://drive.switch.ch/index.php/s/fIR21mPi8k35n0u/download
     If macOS prevents you from installing it, you can go to System Preferences > Security & Privacy > General and push the "Open anyway" button.
   - For Linux: https://drive.switch.ch/index.php/s/tIgLPdHH7HGqjwn/download

   In case you encounter problems during installation, especially on macOS and Linux, check the warnings section at the end of this webpage. Also note that the provided installation files are for 64-bit versions. In case you are using a 32-bit OS, you need to download the installation file from this same webpage.
3. Connect the ST-LINK programmer to the computer. In case it is not recognized (check in the device manager), download[2] and install the driver. This step is not required with macOS and Linux.
   https://drive.switch.ch/index.php/s/4KdK1xet5HHmVqO/download

[1] If needed, the latest version of System Workbench can be downloaded from (registration required): http://www.openstm32.org/System%2BWorkbench%2Bfor%2BSTM32
[2] If needed, the latest version of the ST-LINK driver can be downloaded from (registration required): http://www.st.com/en/embedded-software/stsw-link009.html
**If you cannot directly execute the *.run* open a terminal in its folder and change the permissions to execute using `sudo chmod +x name_of_the_file`. Then install by calling `.\name_of_the_file.run`

4. Download, extract and install the USB drivers for the USB-to-UART chip of the paddle electronics board from Moodle ("USB-UART driver for Windows/MAC OS" under General).
5. Start System Workbench.
   When starting, it will ask for a workspace folder location. You can you choose any location, but the full path should not have any space characters, and should be a permanent location. Then, tick the checkbox "do not ask again" and press OK.
   If Windows asks for granting this application to communicate through the firewall, make sure to tick both checkboxes before validating.
   Wait until everything is setup, then close the "Welcome" tab.
6. Download the source code from Moodle ("hhri-software-v2.7.zip" under Lab 0) and unzip the archive to the directory of your choice, but again, avoid spaces in the path.
   In System Workbench, go to File > Import... > General > Existing project into workspace > [Enter the path of the source code folder] > OK. You can also just double-click on the ".project" file that is in the Firmware directory.

1.3 C++ Development Environment (optional)

Note: This step is only required if you want to edit or view the source code of the Graphical User Interface (GUI) application that is used to communicate with the paddle while running. For this course you wdo not need to do this, you can simply use the compiled executable version. If you want to install the development environment anyway, since it is a heavy program and the installation takes a long time, please do this outside of the class hours.

Go to https://www.qt.io/download, choose the open source version, download and install the Qt development environment for Desktop PC.
When in the "Select Components" page, choose only the following items:

- Qt 5.10 (or a later one if available):
  - MinGW 5.3.0 32 bit (or a later one if available).
  - Qt Charts
- Tools:
  - MinGW 5.3.0 (the one that matches your Qt version).

2 First steps with the board

The goal of this exercise is to get the serial communication working, and have the board and the development environment ready for the next (real) exercises.

2.1 Source code structure

The given source code archive contains several sub-projects.

- **Firmware/:** source code of the firmware of the HRI board. Most of the work during the lab sessions will be performed in the file Firmware/src/haptic_controller.c.
  You will use System Workbench IDE to work with this project. The code documentation can be found in Firmware/doc/firmware_documentation.html.

- **CPP/:** contains a ready-to-use GUI to interact with the board, by reading and setting remotely some selected variables of the board firmware. You need Qt Creator to work with this project.
  A custom C++ library (CPP/HriBoardLib) can be used to develop your own programs. This can be useful for some specialization projects.
  An example project (HriExampleProgram) shows how to make a custom user interface using this library.
  The code documentation can be found in CPP/doc/cpp_interface_documentation.html.
- **MATLAB/:** contains a MATLAB library to communicate with the board. This allows to write MATLAB scripts that can interact with the paddle board. A basic GUI is also available, but it is recommended to use the CPP version of the GUI.

2.2 Programming the board and connecting to it

If you have installed the development environment correctly, you should already be able to plot some standard variables with the GUI. To do so:

- Connect the board via the ST-Link V2 to the PC (used to program the microcontroller).
- Connect the board via the USART-USB link to the PC (used to communicate with the running program on the microcontroller: e.g. log variables states, change parameters).
- Connect the power supply first to the board and then to the power plug.
- In System Workbench, select the project in the tree on the left (Figure 1 - 1), then compile the code using Project > Build all (or press , Figure 1 - 2). This step is optional.
- Then, you can load the program on the board by using Run > Run (or press , Figure 1 - 3) and selecting "Ac6 STM32 C/C++ Application".

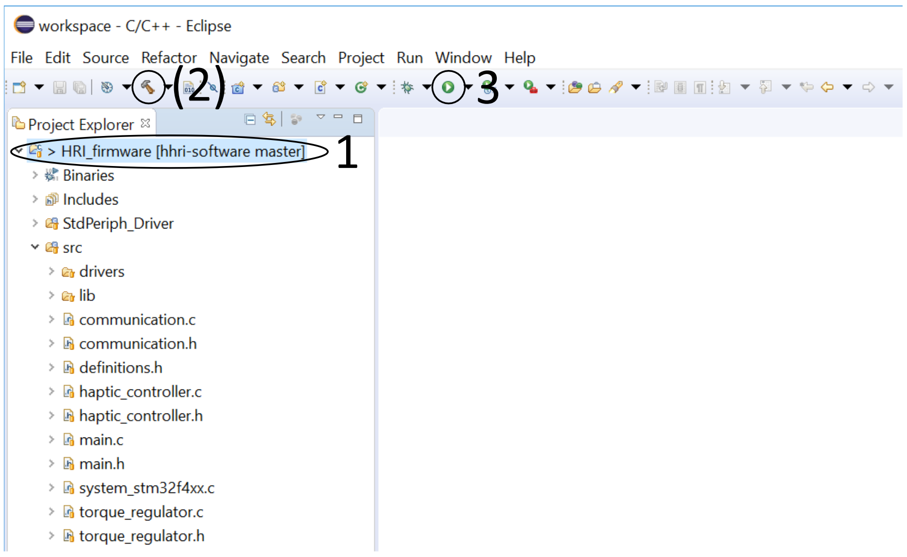
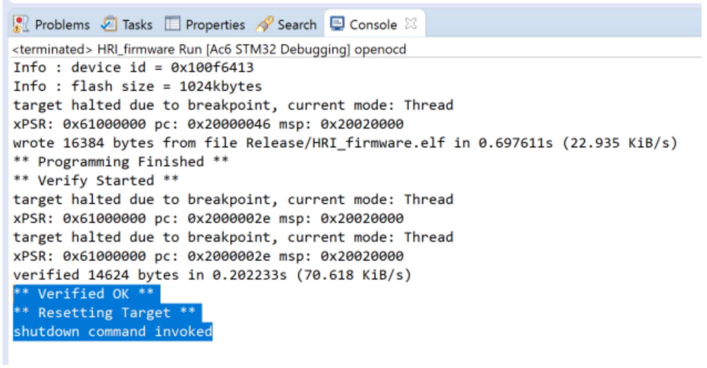


Figure 1: board programming sequence on eclipse

- Verify that the program has been successfully flashed to the board by checking the output in the "Console" tab below the editor (Figure 2); the last 3 lines should be:

```
** Verified OK **
** Resetting Target **
shutdown command invoked
```



```
<terminated> HRI_firmware Run [Ac6 STM32 Debugging] openocd
Info : device id = 0x100f6413
Info : flash size = 1024kbytes
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x20000046 msp: 0x20020000
wrote 16384 bytes from file Release/HRI_firmware.elf in 0.697611s (22.935 KiB/s)
** Programming Finished **
** Verify Started **
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x20020000
target halted due to breakpoint, current mode: Thread
xPSR: 0x61000000 pc: 0x2000002e msp: 0x20020000
verified 14624 bytes in 0.202233s (70.618 KiB/s)
** Verified OK **
** Resetting Target **
shutdown command invoked
```

Figure 2: output of the flashing process in case of success.

- Start the remote-control program[3]. You can download an executable compiled version from Moodle (General/Binaries for HRI-PC-Controller). You can also build and run the CPP/HriPcController project yourself using Qt Creator.

  The board should be recognized, and the GUI will appear.

The board is now connected to the GUI via the USB-UART connection. On the top is a list with all the variables which the microcontroller program allows to read and write from the GUI ("Get" and "Set" buttons). Some of the variables are read-only, so they have no "Set" button. You can also stream some of the variables to the GUI for live visualization. To do so, tick the desired "Stream" checkboxes. You can also log the variables in a file by checking the "Log to CSV file" option. Logging will start as soon as the box is checked, and finishes when you uncheck the box. Then the log file will be saved and you can access it using applications such as Excel or Matlab.

2.3 Interaction with the board

2.3.1 Basic remote control with the GUI

Enable the streaming of the "hall_voltage [V]" variable and you should be able to observe the output of the hall sensor when you move the paddle with your hand. You can do the same with the "encoder_pos [deg]" variable.
You can set the four LEDs intensity by setting the "led_X [0.0-1.0]" variables.

Try to change "motor_torque [N.m]" to value of 0.003 N.m. What happens?
Experiment moving the paddle with your hand. Does it feel like a spring?

2.3.2 haptic_controller.c file

In System Workbench, open HRI_Firmware > src > haptic_controller.c. This is the file you will modify during the lab sessions. Observe its structure.

- Global variables are declared at the top of the file. They have to be here either because they are state variables, or to allow remote monitoring.
- hapt_Init() performs the initialization tasks, and setup a few variables to be remotely controlled later.
- hapt_Update() acquires the sensors data, and computes a motor torque. This function is automatically called periodically at a fixed rate.
- To be able to monitor (read and/or write) a variable from the PC, you should declare it as "shared" in the hapt_Init() function, using the comm_monitor*() functions.

[3] Alternatively, you can also start MATLAB, go to the MATLAB/ folder, and type "hri_gui". However, this interface is less performant.

2.4 Exercises

The goal of today's lab session is to simulate a torsional spring effect with the paddle, then a damper.

2.4.1 Spring effect

1. Start by implementing a motor torque proportional to the angle with the rest position, with the opposite sign. The spring stiffness and rest position should be settable from the computer GUI.
2. Set the stiffness to 0.02 N.m/deg (paddle's torque / angle) and move the paddle over the full angular range. Plot the encoder angle, and the actual motor current. Discuss the hardware limitation.
3. Set the stiffness to 0.01 N.m/deg. Pull and release the paddle such that it oscillates. Record the encoder position and load it into MATLAB. Estimate the paddle inertia from the oscillation frequency. Does it match the value of the hardware documentation?

2.4.2 Damping effect

1. Estimate the angular speed of the paddle by doing the numerical differentiation of the encoder position. Start plotting this speed and move the paddle. Does your speed estimate have a good resolution?
2. Increase the sampling period to 10000 µs, how does the speed resolution change?
3. Implement a controller such that a damping effect (torque proportional to the speed, against the movement) is applied only when the paddle angle is between -10° and 10°. What do you feel when you move the paddle over the full range?