Haptic Human-Robot Interfaces: Tele-Operation Specialization Lab

Assistant: *Ali Reza Manzoori (ali.manzoori@epfl.ch)*

1 Hardware Setup

1.1 Extension connector and the breakout board

In this specialization lab, we will use a UART communication to exchange information between two haptic paddles. In order to access the UART pins of the microcontroller, we will use an extension board (introduced as the "digital breakout board" in the paddle hardware documentation) that is connected to the digital extension connector of the paddle's mainboard. The digital extension connector is the double-row female pin header connector (2×10 pins) that is located between the microcontroller and the programmer connector (Figure 1).

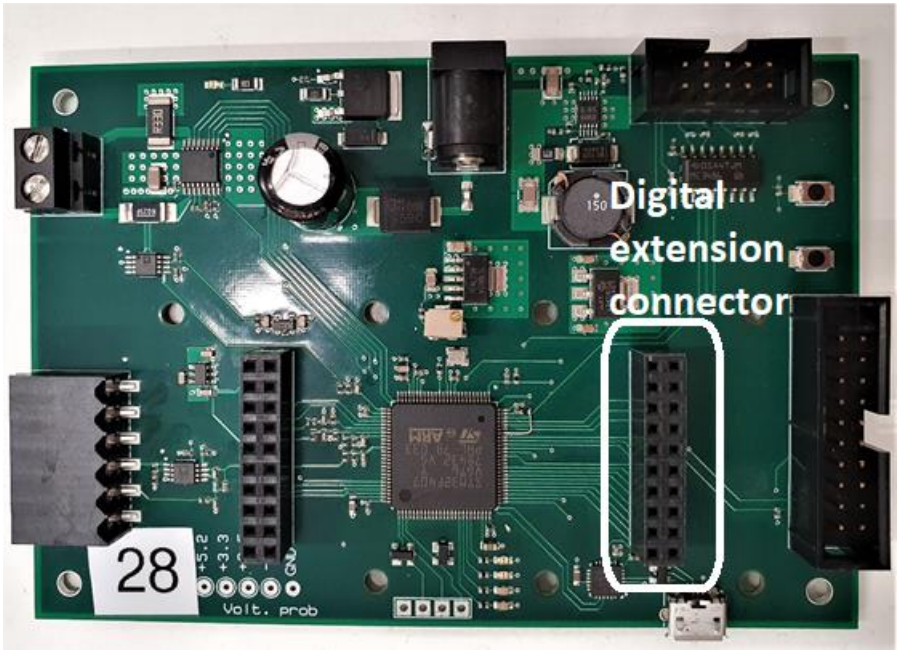

Figure 1 - The digital extension connector (marked with a white rectangle) sits between the microcontroller and the programmer connector.

The digital breakout board has two female connectors on top, and a male connector on the bottom. The male connector is plugged into the digital extension connector on the paddle mainboard. The two female connectors on top provide access to several of the digital pins of the microcontroller, which can be used for different purposes (GPIO, UART, I2C, etc.). For this project we only need three of the pins on the J3 connector: Pins 11 or 12 (UART RX), 13 or 14 (UART TX), and 19 or 20 (both GND). These pins are marked in Figure 2 (note that the white dots on the upper right corner of each connector on the board mark the pin #1 of the connectors J2 and J3).

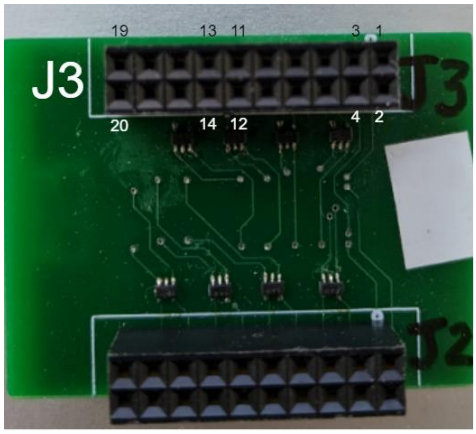

Figure 2 - The digital breakout board and its connector J3.

1.2 Connecting the breakout board to the extension connector

Important: Please unplug the paddles from the electric outlet to power them off before proceeding with connecting the breakout boards.

To connect the digital breakout board to the mainboard of the paddle, first identify the digital extension connector on the mainboard. Then, orient the digital breakout board such that the connector J2 is placed near the programming connector, and the connector J3 is near the microcontroller (see Figure 3). Then, align the male connector on the bottom of the digital breakout board with the digital extension connector of the mainboard, and then insert the digital breakout board into the extension connector by pushing it down gently. Please pay careful attention to the orientation of the breakout board and the alignment of the male and female connectors; mistakes in connecting the extension board to the mainboard can permanently damage the mainboard!

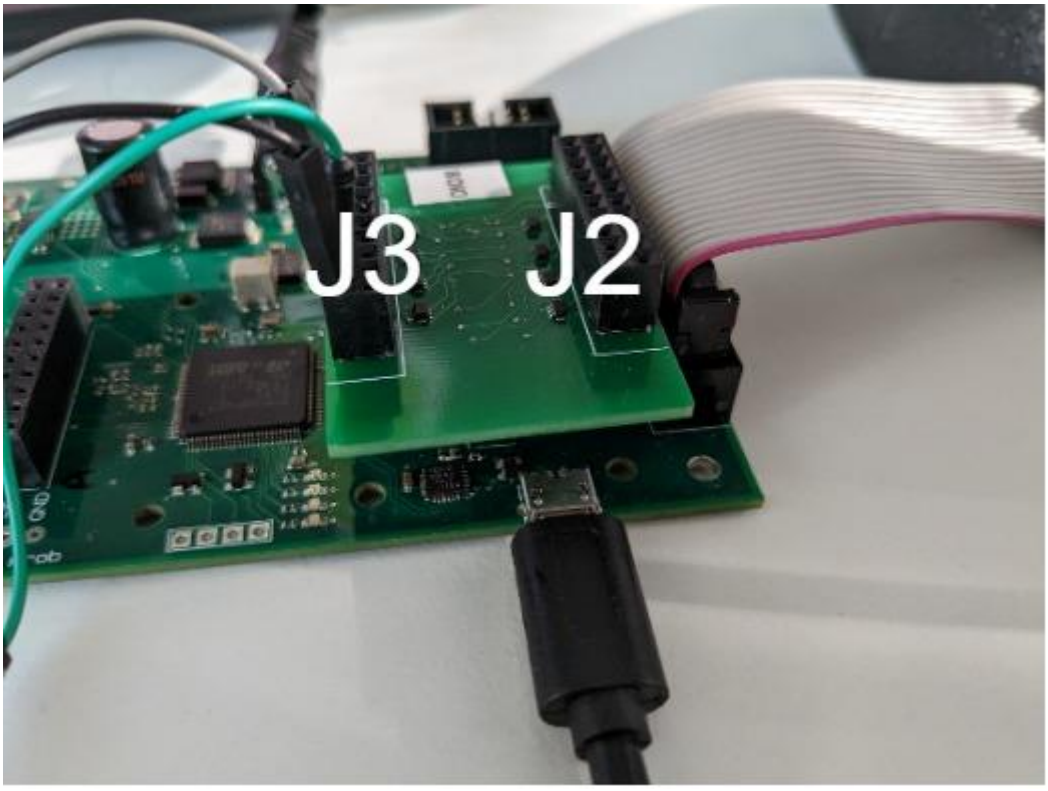

Figure 3 - The correct orientation of the digital breakout board when connecting to the digital extension connector of the haptic paddle mainboard.

1.3 Wiring the two paddles for communication

As mentioned before, the two paddles will communicate over the UART protocol. For a quick introduction to the UART protocol, check out this webpage. This communication therefore requires the two paddles to be connected to each other. Firstly, the UART TX (transmit) pin of the first paddle should be connected to the UART RX (receive) pin of the second paddle. Similarly, the UART RX of the first paddle should be connected to the UART TX pin of the second one. Finally, in order for the digital signals between the two paddles to have the same reference, the GNDs should also be connected together.

Keep the two paddles unplugged from the electric plug while doing the wiring. After you are done with this step, check your setup with the assistant before powering on the paddles.

2 Software Setup

To use the extension UART port of the microcontroller, there is a driver with ready-to-use functions in the firmware. This driver is located under "`Firmware\src\drivers\ext_uart.h`". To be able to use the driver, you need to include it in your code (`haptic_controller.c`) by adding this include line to it:

```
#include "drivers/ext_uart.h"
```

Then, you can call any of the following functions provided in this driver.

- **`exuart_Init:`** This function will initialize the external UART port of the microcontroller. Initialization of the UART port is a one-time operation that needs to be done when the controller starts; therefore it must be called in the init function.
- **`exuart_ReceivedBytesCount:`** This function returns the number of bytes that have already been received and are waiting to be read from the buffer. After each byte is read from the buffer, the number returned by this function will decrease by 1 (assuming that no new bytes have been received from the other device in the meantime).
- **`exuart_GetByte:`** Returns the first received byte from the buffer.
- **`exuart_SendByteAsync`** and **`exuart_SendBytesAsync:`** These functions can be used to send one or multiple bytes over the UART port to the other device.

For more information about these functions and the correct syntax for calling them, refer to the comments in `ext_uart.h` and `ext_uart.c` files.

3 Suggested Workflow

The final goal in this lab is to develop a tele-operation framework between the two paddles, so as to explore different modes of tele-operation and the effects of communication on the performance of the tele-operation system. To reach this final goal, we suggest breaking down the development into the following incremental steps, to decouple the different challenges that need to be tackled:

1. **Basic communication development**: Use the functions in the ext_uart driver in your firmware to send and receive one byte between the two paddles.
2. **High-level communication protocol**: Expand the firmware to be able to send and receive variables of type float between the two paddles.

3. **Remotely monitoring a variable in real-time**: Add the functionality of continuously monitoring one of the variables of one paddle by the other in real-time. For example, you can monitor the position of one paddle in the other paddle's GUI.
4. **Simple master-slave configuration (master-to-slave mapping):** Now you can begin working on developing a basic tele-operation framework by assuming one paddle to be the master device and the other to be the slave. By moving the master paddle, the slave paddle should follow. There are different ways of implementing this, start with the simplest idea and then think of other ways!
5. **Bidirectional communication configuration:** Improve the master-slave system by making the communication bidirectional, so that the person handling the master device will have haptic feedback from the slave. The idea is that if the slave device hits an obstacle or experiences some external force, the person moving the master device would feel it. There are different approaches for implementing such a configuration:
    a. One way would be to create a mirrored system, in which both of the paddles are both master and slave. Thus, moving one paddle would lead to the other also being moved, but if the movement of the slave is limited by an external force/object, it will automatically be felt by the master as well.
    b. Another approach would be sending a number proportional to the "resistance" that the slave device is feeling to the master, and then applying this resistance to the user via control of the master device.
6. **Investigating the effects of communication**: Increase the communication delay in your firmware, and see how it affects the quality of the tele-operation. You can also investigate packet loss and study its effects on the quality of the tele-operation. Why are these phenomena important? Can you think of how and why they come into play in a real tele-operated system?

In each of the intermediate steps, make sure to thoroughly test your implementation and make sure it is working well before proceeding to the next step. This will make troubleshooting a lot easier, as you can address the problems one by one and avoid complicated issues due to an interaction between different problems. For example, if your high-level communication protocol for sending and receiving floats (developed in step 2) fails from time to time, this can cause unexpected behavior in steps 4 and onward, and it will be difficult to find out if a malfunction is due to communication issues or due to bugs in your tele-operation code.

4 Report

As boring or tedious as it may appear to be, reporting your methodology and findings is absolutely crucial in science and engineering. The perceived value of your work by other people is highly dependent on good communication and reporting. Learning how to report your work clearly is therefore as important as learning the theoretical and technical aspects. Keep this in mind when writing your reports, and try to be as clear as possible, while keeping your report concise.

Make sure to include the following items in your final report:

- An introduction to tele-operation, its use-cases in the real-world, the importance of haptics for tele-operated systems, and the main technical challenges in terms of control and haptics in these systems
- A summary of your workflow and your main steps.
- A description of what you did in each step, including a discussion about any possible challenges or problems you faced and how you solved it. Appropriate use of block diagrams, flow charts and other

graphical representations to explain your implementation and algorithms clearly is highly recommended.

- Snippets of code showing the most important parts of the communication protocol, the slave device's algorithm, and the master device's algorithm.
- Your findings about the effects of communication on the quality of tele-operation. Try to think of ways to represent your findings objectively and quantitatively as much as possible, rather than only qualitative descriptions.
- Discussion and interpretation of your findings, their implications in real-world applications, and the main limitations of the work. Suggestions for improvement (could be in terms of methodology, software, hardware, etc.) are always welcome. Unleash your critical thinking!
- A short conclusion, summarizing your main steps, what you achieved, what you observed, and the most important take-home messages.

Last but not least: this document serves as a basic guideline for the specialization lab, but is not exhaustive or comprehensive. Feel free to get creative and explore beyond what is suggested in this document. Although it is optional, but any original ideas in your implementation, testing methodology and/or report are greatly encouraged. Enjoy the learning journey!