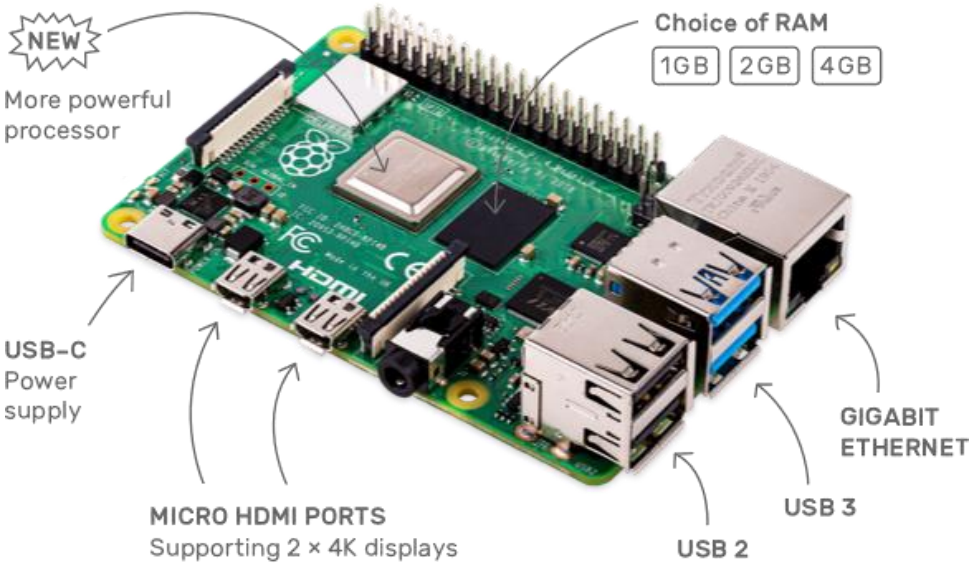
EECS 388 Lab #9

Board-to-Board Communication

In this lab, you will establish UART based communication channels between the Pi 4 and the HiFive1 board.



Part 1: Setup the UART connections

In this part, we will connect the HiFive1 and the Raspberry Pi 4 boards via two UART channels.

While the Pi 4 has 4 UARTs, we will only be using two of them today (uart2 and uart3). Examine the file `/boot/config.txt` for the following two lines enabling uart2 and uart3, **add them to the file if they are not present.**

If the following two lines are already present in the config.txt file, then don't need to write or edit anything in the file.

```
dtoverlay=uart2,115200
dtoverlay=uart3,115200
```

If you needed to add these two lines, reboot the Raspberry Pi. After rebooting the system, `/dev/ttyAMA1` and `/dev/ttyAMA2` will be created. (**Again, if the lines are already in the file, don't do anything**)

Connect HiFive's UART1 RX (pin7) to Raspberry Pi 4's UART2 TX (pin 27). This is the main communication line between the Pi 4 and the HiFive1. From the Pi 4, you can access the channel via `/dev/ttyAMA1`.

For debugging of HiFive 1, connect HiFive1's UART0 TX (pin1) to Pi 4's UART3 RX (pin 29). From the Pi 4, it can be accessed via /dev/ttyAMA2.

In summary, you will be able to access the following two files from the Pi 4.

```
/dev/ttyAMA1      Pi 4 → HiFive1: Send steering angle to HiFive1 (uart1).
/dev/ttyAMA2      HiFive1 → Pi 4: Receive HiFive1's console (uart0) output
```

Part 2: Programming the HiFive1

In this part of the lab, you will program the HiFive1 to receive data from the Pi 4.

On your PC (not Pi 4), download the starter code.

Sign in to the Canvas account and go to the EEC 388 Embedded Systems section. Inside the "Files" section, go to the "Source Codes" folder.Click on the **Lab09.tar.gz** file to download the source code for lab9. Extract the file inside a folder of your PC.After that, add this folder to the VS Code workspace (like the previous lab)

Your task is to receive the data from HiFive1's UART1 channel and send the received data to UART0 channel **as a null-terminated string**. The following is rough **pseudo code** for the task.

```
while (1) {
  if (is UART1 ready?) {
    data = read from UART1.
    print data to UART0.
  }
}
```

To implement the task, you should use the provided serial API shown in the following. Note that `devid` is 0 to access UART0, while it is 1 to access UART1. For this lab, the two functions `ser_printline` and `ser_readline` will be especially helpful (since you need to send the data as a string).

```
void ser_setup(int devid);
int  ser_isready(int devid);
void ser_write(int devid, char c);
void ser_printline(int devid, char *str);
char ser_read(int devid);
int  ser_readline(int devid, int n, char *str);
```

In particular, you may need to use `ser_isready()` function to check whether a given UART channel has pending data to read. To better understand what a specific function is doing, you can always check both `eecs388_lib.h` and `eecs388_lib.c` files.

```
int  ser_isready(int devid)
{
  uint32_t regval = *(volatile uint32_t *)(UART_ADDR(devid) + UART_IP);
  return regval;
}
```

How to test your HiFive code: Once you finish programming the HiFive1, **switch to the Raspberry Pi 4** and open two terminals: one for sending data to the HiFive1, and one for seeing the debug message output from the HiFive1.

Sender's terminal (terminal 1):

```
$ screen /dev/ttyAMA1 115200
```

Debug terminal (terminal 2):

```
$ screen /dev/ttyAMA2 115200
```

Now, type any string on the 'terminal 1'. If you programmed your HiFive 1 correct, you should see the message coming out from the 'terminal 2' terminal.

If you see error message like "Screen is terminating", you should kill the terminals and then again re-open the terminal.

Today's Task:

Show your work to your TA for demo.

Submit your modified C code file (HiFive part) at Canvas.

Screenshots/PDF/Text files will NOT be allowed/graded as submission. You need to submit modified eecs388_interrupt.c file only.

Appendix

GPIO mapping of Pi 4.

Function	Pin Number	Pin Number	Function
3V3	1	2	5V
SPI3 MOSI/SDA3	3	4	5V
SPI3 SCLK/SCL3	5	6	GND
SPI4 CE0 N/SDA 3	7	8	TXD1/SPI5 MOSI
GND	9	10	RXD1/SPI5 SCLK
	11	12	SPI6 CEO N
SPI6 CE1 N	13	14	GND
SDA6	15	16	SCL6
3V3	17	18	SPI3 CE1 N
SDA5	19	20	GND
RXD4/SCL4	21	22	SPI4 CE1 N
SCL5	23	24	SDA4/TXD4
GND	25	26	SCL4/SPI4 SCLK
SPI3 CE0 N/TXD2/SDA6	27	28	SPI3 MISO/SCL6/RXD2
SPI4 MISO/RXD3/SCL3	29	30	GND
SPI4 MOSI/SDA4	31	32	SDA5/SPI5 CEO N/TXD5
SPI5 MISO/RXD5/SCL5	33	34	GND
SPI6 MISO	35	36	SPI1 CE2 N
SPI5 CE1 N	37	38	SPI6 MOSI
GND	39	40	SPI6 SCLK
I2C			Ground
UART			5V Power
SPI			3V3 Power

Source: https://learn.pi-supply.com/make/raspberry-pi-4-pinout