

## LCD <> ChibiOS communication (Sprint 3)

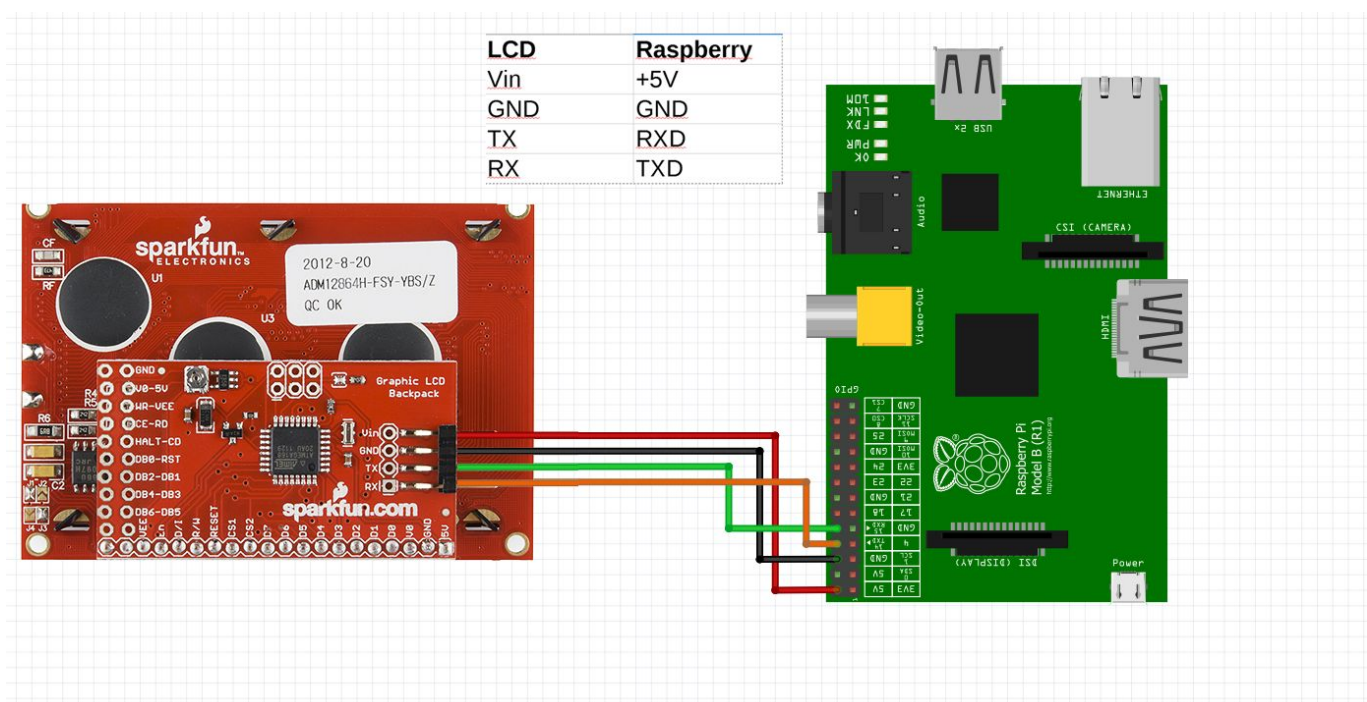
MINF UDL 20-21

Ubiquitous and embedded systems

Team 1

**Objective:** Make the chibiOS works with the LCD Screen, because this feature will be needed in the final sprint (data representation).

### 1. Wiring connection:



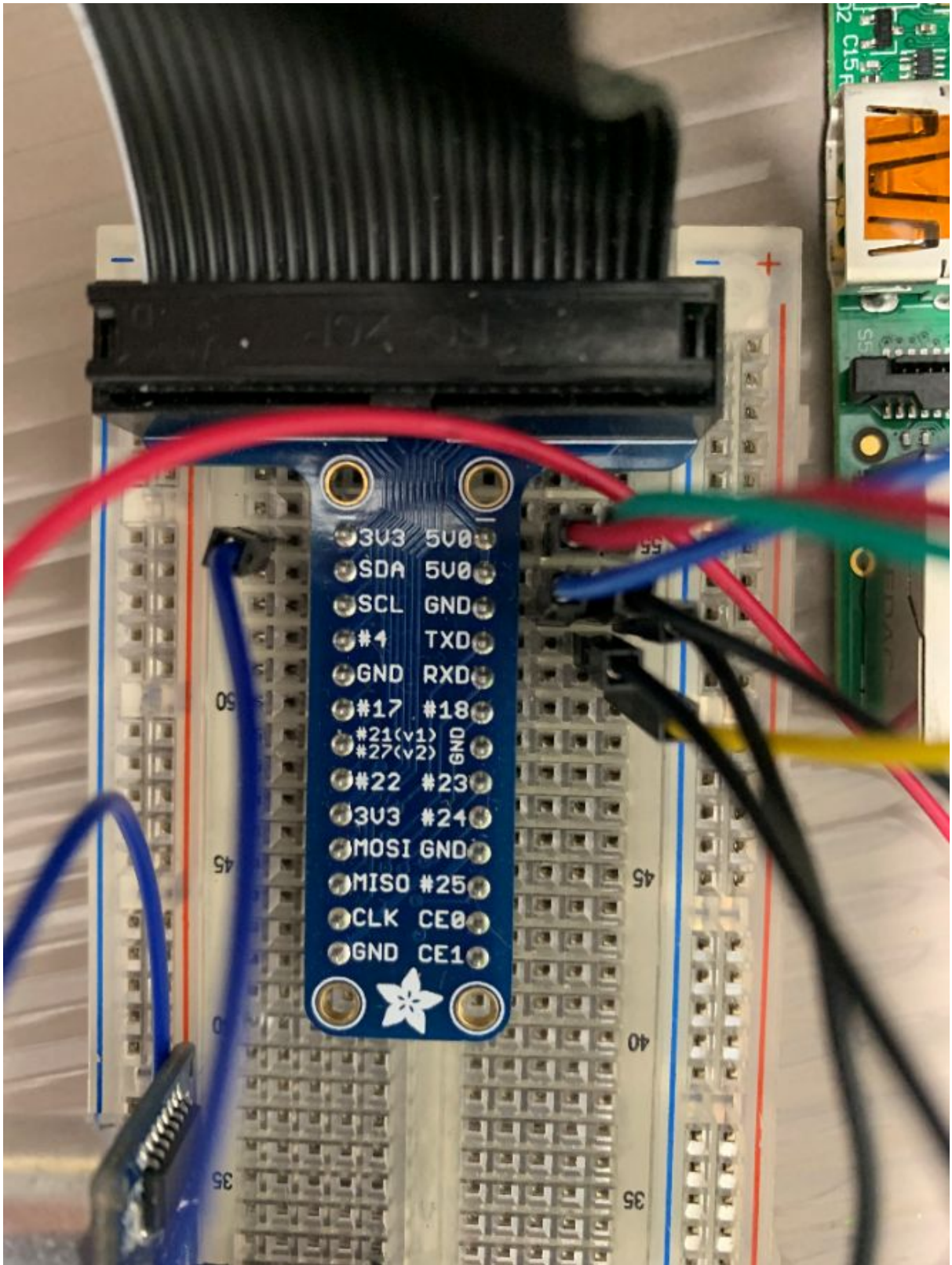
Connect the pins carefully to not invert the TX and RX.

## 2. It should look like this:

- a. (In my case the Data Producer 1 is also connected to the Raspberry, please ignore it)







### 3. Code:

- a. In this case, we are using a code example as base to build our final code, this code will print some strings to the LCD and also an iteration.
- b. The communication goes like this: We are using the driver SerialDriver of the ChibiOS to communicate with the LCD Screen, using the SD1 which means the UART pins of Raspberry (In our case, is the TXD and RXD pins).
- c. In the future, we will implement in this code the feature to print in the LCD Screen the numbers we have received via i2c.

```
#include "ch.h"
#include "hal.h"
#include "chprintf.h"

static WORKING_AREA(waThread_LCD, 128);
static msg_t Thread_LCD(void *p)
{
    (void)p;
    chRegSetThreadName("SerialPrint");
    uint16_t iteration = 0;
    while (TRUE)
    {
        sdPut(&SD1, (uint8_t)0x7C);
        sdPut(&SD1, (uint8_t)0x18);
        sdPut(&SD1, (uint8_t)0x20);
        chThdSleepMilliseconds(10);

        sdPut(&SD1, (uint8_t)0x7C);
        sdPut(&SD1, (uint8_t)0x19);
        sdPut(&SD1, (uint8_t)0x20);
        chThdSleepMilliseconds(10);

        chprintf((BaseSequentialStream *)&SD1, "Iter.: %u", iteration);
        iteration++;
    }
}
```

```

    chThdSleepMilliseconds(2000);
}
return 0;
}

/*
 * Application entry point.
 */
int main(void)
{
    halInit();
    chSysInit();

    char txbuf[] = "Hello Chibi-World - UDL-MINF";

    // Initialize Serial Port
    sdStart(&SD1, NULL);

    // First Message
    chprintf((BaseSequentialStream *)&SD1, "Main (SD1 started)");

    // Coordinates
    sdPut(&SD1, (uint8_t)0x7C);
    sdPut(&SD1, (uint8_t)0x18);
    sdPut(&SD1, (uint8_t)0x00);
    chThdSleepMilliseconds(10);

    sdPut(&SD1, (uint8_t)0x7C);
    sdPut(&SD1, (uint8_t)0x19);
    sdPut(&SD1, (uint8_t)0x0A);
    chThdSleepMilliseconds(10);

    // Second message
    chprintf((BaseSequentialStream *)&SD1, txbuf);




    // Start thread
    chThdCreateStatic(waThread_LCD, sizeof(waThread_LCD), HIGHPRIO,
Thread_LCD, NULL);

    /*
     * Events servicing loop.
     */

```

```
chThdWait(chThdSelf());  
  
return 0;  
}
```

4. Put the code on the SD card (there's already an example done in the Github, under /documentation/ called boot\_LCD.rar
  - a. Remember, the SD should look like this:

Name	Date modified	Type	Size
 bootcode.bin	13/09/2019 17:33	BIN File	51 KB
 kernel	30/10/2020 17:15	Disc Image File	8 KB
 start.elf	13/09/2019 17:32	ELF File	2.781 KB

5. Put the SD on the Raspberry, turn it on and test!