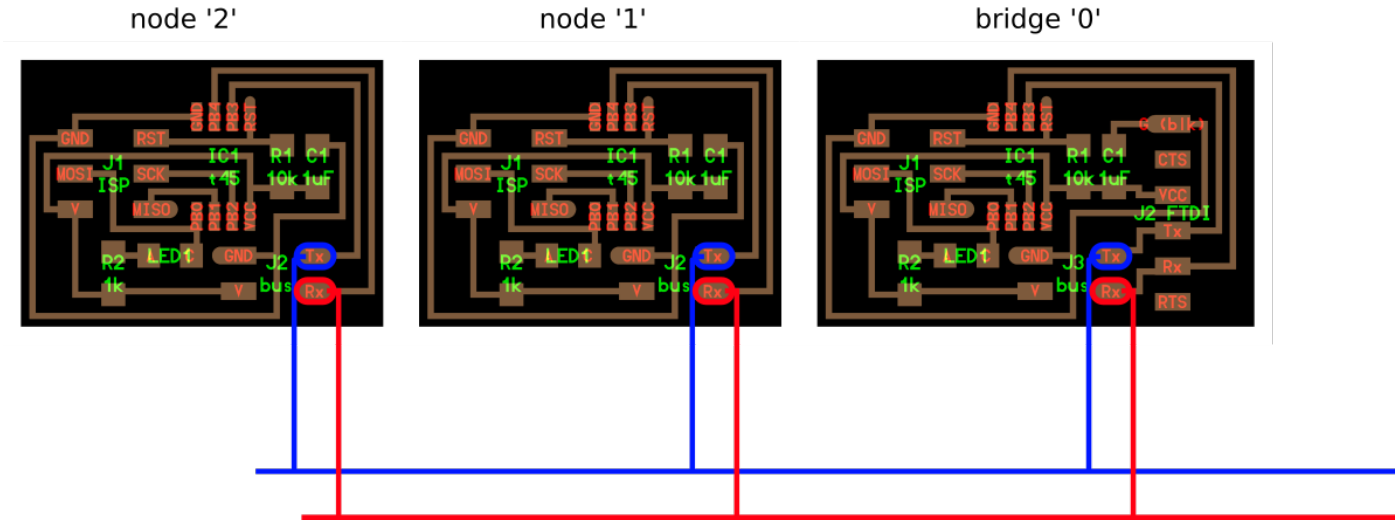
Networking and communications[link]

Serial bus

A serial bus is a set of cables that will carry the communication between devices. It uses serial communication protocol, and all the devices share the same Rx and Tx lines. This means that every time a message is sent over serial, all the devices connected to the bus will receive the same information, so, addressing must be used to identify each device in the network. This setup requires a bridge (bridge between the PC and the network) and one or more nodes. The examples used for this setup can be found here: bridge and node.

This is the setup:



Programming

Remember to burn the appropriate bootloader to your board before uploading any code to it.

Fabacademy examples also include a functional programming codeprogramming code for the boards.

For this example, the code can be divided into smaller part for analysis.

The first part is the address designation. Before uploading the code to each board, remember to give each board a unique name or id. The bridge board can be '0', and then '1', '2', etc. for the nodes.

Basically, the code sets all the pins as inputs so they are " listening". Whenever there is traffic, the led's will flash. If the id matches one of the boards, then that board will switch to sending data and will send its id back and only that board will flash once more, and then the line is set back to "listening".

This id will be the one identifying each board during the communication process.

```
#define node_id '0'
```

Then, the method that reads data is defined with:

```
void get_char(volatile unsigned char *pins, unsigned char pin, char *rxbyte)
```

So, you can use get_char to read data from any of the boards.

Similarly, the method that sends data is defined with:

```
void put_char(volatile unsigned char *port, unsigned char pin, char txchar)
```

You can use put_char to send data from any of the boards.

Then, there is an implementation of data send for a *string* (instead of a *char*). It is done by multiple calls to the **put_char** method. It reads a string one character at a time and while there is a character to be read, the **put_char** method is called. When there are no more characters to read on the *string*, it stops.

```
void put_string(volatile unsigned char *port, unsigned char pin, PGM_P str)
{
    //
    // send character in txchar on port pin
    // assumes line driver (inverts bits)
    //
    static char chr;
    static int index;
    index = 0;
    do
    {
        chr = pgm_read_byte(&(str[index]));
        put_char(&serial_port, serial_pin_out, chr);
        ++index;
    }
    while (chr != 0);
}
```

There is also a method to flash the led on each board during communication.

```
void flash()
{
    //
    // LED flash delay
    //
    clear(led_port, led_pin);
    led_delay();
    set(led_port, led_pin);
}
```

Finally, the main routine

```
while (1)
{
    get_char(&serial_pins, serial_pin_in, &chr);//read from the Tx line
    flash()//all the boards flash the led;
    if (chr == node_id)//if the read char matches the board id
    {

        output(serial_direction, serial_pin_out);//changes pin from input to output
        static const char message[] PROGMEM = "node ";//creates the string with "node "
        put_string(&serial_port, serial_pin_out, (PGM_P) message);//sends the string
        put_char(&serial_port, serial_pin_out, chr);//sends the data read from Tx (the id)
        put_char(&serial_port, serial_pin_out, 10);//sends 'new line' character
        // new line
        led_delay();//delay
        flash();//only the called board flashes the led
        input(serial_direction, serial_pin_out);//sets the pin back to listening (input)
    }
}
```

This simple code can be modified to perform other tasks ans send other messages over serial bus.

To test the code, simply upload it to the board and verify it is responding as expected.

Resources and recommendations

- https://www.maximintegrated.com/en/app-notes/index.mvp/id/723
- https://www.arduino.cc/reference/en/language/functions/communication/serial/
- https://www.ladyada.net/learn/arduino/lesson4.html
- https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi
- https://electronics.stackexchange.com/questions/37085/multiple-arduino-communication-1-master-n-slaves
- https://hackaday.com/2012/12/03/inventing-networking-protocols-for-dozens-of-arduinos/