Mechatronic Project 234

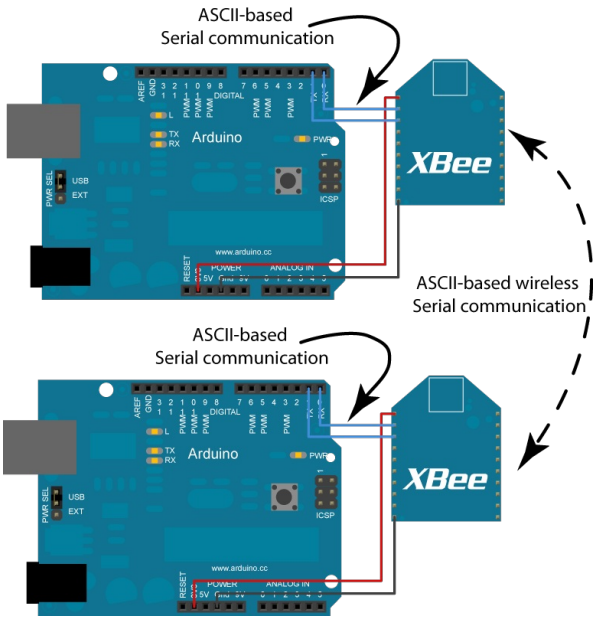
Laboratory G: Serial Communications

Before the lab:

- Ensure you have read the lecture notes on serial communication, PWM and servomotors
- Review the instructions given in the Lecture session
- Review the Atmega datasheet on timers and PWM
- Review the XBee datasheet

Task:

- A controller Arduino will be provided, equipped with an LCD display, two analogue joysticks and an XBee communication unit.
  Schematic: https://github.com/mxeng/mcp-docs/blob/master/labs/controller_schematic.pdf,
  Wiring Diagram: https://github.com/mxeng/mcp-docs/blob/master/labs/controller_wiring.pdf
- You should establish wireless communication between two the controller unit and your robot Arduino unit equipped with a second XBee.
- Transmit instructions for two servos (as per lab F) from the control unit to the robot unit. **(G1)**
- Transmit sensor readings back from the robot unit to the control unit. **(G2)**
- Signal transmission should be simultaneous, smooth, and with little latency or delay.



Note:

- Show the design of your communication protocol in your lab book **(G3)**. Make it clear how your instructions are structured, and how the instructions are delimited.
- Discuss any considerations for ensuring adequate data flow: i.e ensuring that buffer overflows do not occur, and that latency is minimised. **(G4)**
- Include schematics for both of your circuits in your lab book. **(G5)**

Multibyte Sending:

```
//main loop
while(1)
{
    current_ms = milliseconds;

    //sending section
    if(current_ms-last_send_ms >= 100) //sending rate controlled here
    {
        last_send_ms = current_ms;
```

```
        serial2_write_byte(255); //send start byte
        serial2_write_byte(1); //send first parameter
        serial2_write_byte(2); //send second parameter
        serial2_write_byte(3); //send third parameter
        serial2_write_byte(4); //send fourth parameter
        serial2_write_byte(5); //send fifth parameter
        serial2_write_byte(254); //send stop byte
    }

}
```

Multibyte Recieving:

Don't forget to initialise your serial subsystems and to declare and initialise your variables. (An example is here: https://github.com/mxeng/mcp-docs/blob/master/labs/initialisation_example_for_coms.md)

```
//main loop
while(1)
{
    if(UCSR2A&(1<<RXC2)) //if new serial byte has arrived: refer to page 238 of datasheet. Single bit flag indicate
    {
        serial_byte_in = UDR2; //move serial byte into variable

        switch(serial_fsm_state) //switch by the current state
        {
        case 0:
        //do nothing, if check after switch case will find start byte and set serial_fsm_state to 1
        break;

        case 1: //waiting for first parameter
            temp_parameter_in_1 = serial_byte_in;
            serial_fsm_state++;
        break;

        case 2: //waiting for second parameter
            temp_parameter_in_2 = serial_byte_in;
            serial_fsm_state++;
        break;

        case 3: //waiting for third parameter
            temp_parameter_in_3 = serial_byte_in;
            serial_fsm_state++;
        break;

        case 4: //waiting for third parameter
            temp_parameter_in_4 = serial_byte_in;
            serial_fsm_state++;
        break;

        case 5: //waiting for third parameter
            temp_parameter_in_5 = serial_byte_in;
            serial_fsm_state++;
        break;

        case 6: //waiting for stop byte
            if(serial_byte_in == 254) //stop byte
            {
                // now that the stop byte has been received, we can process the whole message
                parameter_in_1 = temp_parameter_in_1;
                parameter_in_2 = temp_parameter_in_2;
                parameter_in_3 = temp_parameter_in_3;
                parameter_in_4 = temp_parameter_in_4;
                parameter_in_5 = temp_parameter_in_5;

                sprintf(serial_string, "1:%d, 2:%d, 3:%d\n", parameter_in_1, parameter_in_2, parameter_in_3);
                serial0_print_string(serial_string);  // this is just debugging, printing to the USB serial to make

                //put your code to send back to the controller here if you want to do two way comms
            } // if the stop byte is not received, there is an error, so no commands are implemented
            serial_fsm_state = 0; //do nothing next time except check for start byte (below)
```

```
        break;
        }

        if(serial_byte_in == 255) //if start byte is received
        {
            serial_fsm_state=1; //reset to waiting for first parameter state on 255
        }
    }
}
```

To program your controller you will need different external tool commands. (because it's an ATMega1280 with a different bootloader). You will also have to press the reset button just before you program it.

Title: `Flash ATmega1280 on COM14`

Command: `"C:\Users\Public\Documents\Microcontroller Project\avrdude.exe"`

Arguments: `-C"C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -F -v -p atmega1280 -c arduino -P\\.\COM14 -b57600 -D -Uflash:w:"$(ProjectDir)Release\$(ItemFileName).hex":i`