Lab4 *Arduino Workshop*

Let's Talk!!![1]

Objectives

Implement communication between two Arduinos using different technologies. We shall also compare which technology is suitable under which requirement.
Alongside, we shall understand some common message passing methods between controllers.

General Instructions:

For this lab, you'll require two Arduinos and communicate between them in different ways, using different technologies and/or protocols.

Problem 4.1: UART

Arduino software provides us with *Serial* library for implementing UART based communication. Following are a few important functions provided by the library

1. ***begin***: Initializes the UART functionality with the provided argument as baud rate. [Usage: Serial.*begin(valid_baud_rate)*]
2. ***print:*** sends or transmits the provided argument to function. The argument could be an integer, float or a string. [Usage: Serial.*print(10) or* Serial.*print("hello")*]
3. ***println:*** Similar function to print, except that it appends a <new_line> character ('\n') to the transmitted data. [Usage: Serial.*println(10)*]
4. ***available****:* returns the number of bytes available in received buffer of serial communication.
5. ***read:*** reads a byte from receive buffer.

You might want to see arduino reference page and examples to get a better understanding of these functionalities.

Problem 4.1.1: *Greetings!!!*

Send some data to computer using Serial communication. Use the serial monitor of Arduino IDE to display the received data.

Now, send greeting to arduino 2. Display the greetings received by arduino 2 on computer. Do you encounter any problem? If yes, why?

---

[1] Author: Abhishek N. Kulkarni (mailto:abhishek.kulkarni12@vit.edu)

Problem 4.1.2: A*dvanced greetings!*

Well, it appears we cannot confirm if we received the correct greetings as transmitted by Arduino 1 to Arduino 2. To check if we did receive the complete string, we blink an led (use on board LED) once we receive the new-line character. (So you'll have to use *println* function while transmitting).

To validate if we received a correct string (say "hello" is what we agree to send/receive), we compare the received string with expected string. If there is agreement, we keep the led ON. Else after blinking to indicate reception of data, we switch it OFF.

Problem 4.1.3: *Remote control…*

So let's come to real deal! Generally, we want communication to convey messages to **do** something. In this problem, you should write a simple application which turns an led on pin 13 ON/OFF based on what command is received via UART.

Let's say command to turn ON led is "on" (in lower case) and to turn off led is "off" (again in lower case). Further, each command is terminated by a new-line character.

So implement the system so as to turn on LED if we receive "on" and turn off the LED if we receive "off". [Hint: refer to SerialEvent example in *File → Examples → Communications*]

Problem 4.1.4: *Complex commands (optional)*

Well, in last problem you took some actions based upon the received commands. However, many times we require to transmit some parameters along with command. Say in an application where we would like to control the brightness of one of lights connected to a PWM pin. Now, say we have a "brightness" command. But with this we would also require a number between 0-255 which sets the duty cycle of PWM resulting in brightness control. In short we would like to have a command which may look like "brightness 50" or "brightness 255".

Can you think of how could such (command, parameter) set be transmitted. (one method is suggested above). Develop an application which implements your idea and discuss with your instructor.

Problem 4.2: *I2C*

I2C as we discussed in theory session, is targeted for Inter IC Communication. Arduino exposes this functionality via Wire library. (Refer to documentation of this library on Arduino Reference Page).

Problem 4.2.1: *Do it all over again!!*

Well, repeat problems 4.1.1 to 4.1.3 using I2C communication. Notice the difference that, you may now use Serial monitor on Arduino IDE to see what's happening! So modify the problem statements of 4.1.1 accordingly.

Problem 4.2.2: *Which is better?*

A natural question arises about which one of I2C and Serial communications is better? One way to find this out may be by comparing the time it takes to transmit a byte. (Can we use time required for receiving a byte instead of transmitting? Why? Discuss with your instructor)

In this problem, we attempt to estimate the time for transmitting 1 byte for Serial communication, under different baud rates, and with I2C channel.

To do this, add a line: *time = micros();* just before function which transmits a byte and add a line: *time = micros() – time;* just after the function. Then the "time" variable shall contain the time it took to transmit a byte in microsecond.

Problem 4.3: *Xbee Communication*

Xbee provides a very large range of functionalities and practically it is impossible to cover all possible configurations in this problem statements. Hence, we shall stick to implementation of basic communication using Xbee.

Problem 4.3.1: *Wireless greetings…*

You must be bored sending greeting now. But let's do it one last time.

But now, attach a Xbee and do it! You'll need to configure the Xbee. (We'll watch nice video before we attempt this problem! Remind your instructor to play it!)

Problem 4.3.2: *Wireless data acquisition.*

So let's get down to real problem now. Use LDR board and transmit live data to computer and plot it in real-time. <Provide a program in python/matlab for real time plotting>.

The link should be something like:

LDR board → Arduino → XBee1))) XBee 2 → Laptop/PC → plotter

Problem 4.4: *IR Communication*

Design the problem.

Problem 4.5: *GSM Communication*

Design the problem.

Have fun…

Keep Learning!