PROJECT 5: VISIBLE LIGHT-BASED DIGITAL COMMUNICATION (TRANSMITTER / RECEIVER DESIGN)

Due January 18, 9:00 AM

DESCRIPTION

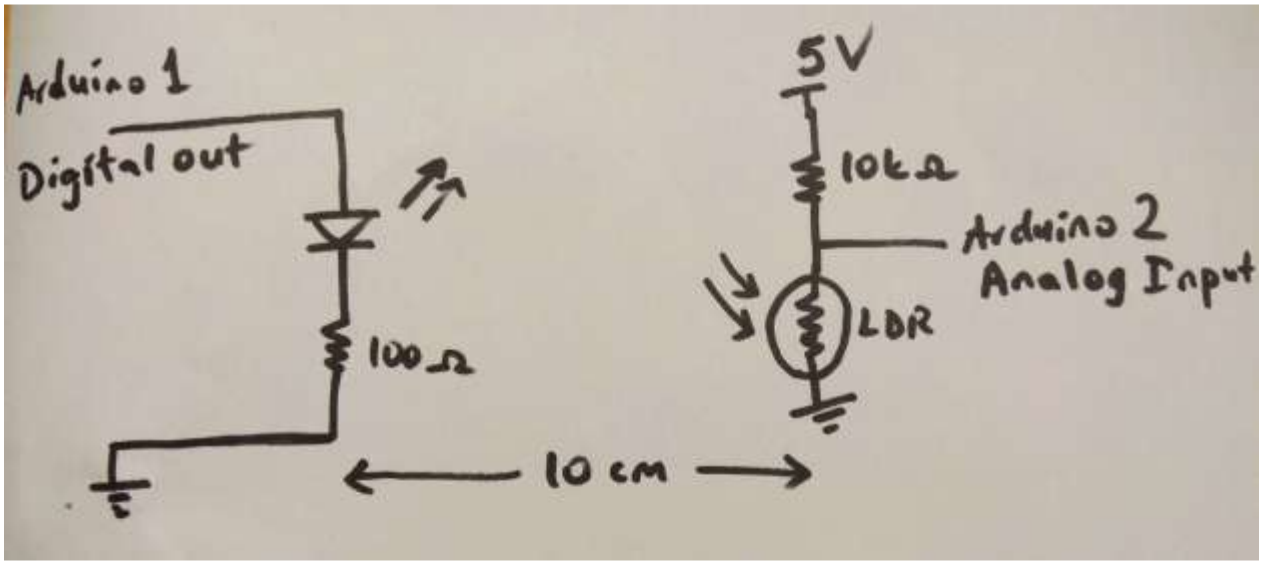
In this project, you will construct a wireless transmitter/receiver pair, capable of transmitting digital information across few centimeters using light. You are expected to implement two signal models.

1) **On-Off Signalling**: Each bit is transmitted by either a high voltage or a low voltage.
2) **Manchester Coding:** Each bit is transmitted by either a voltage drop or a voltage increase.

The **choice of the data signaling rate** and various implementation choices involved in sychronization, such as preambles or end-of-transmission blocks (if you need) are **left to you.**

EQUIPMENT & COMPONENTS:

Arduino Uno's (x2), a white LED, a light-dependent resistor, 100 ohm resistor (x2), 10k ohm resistor. You may find the following sketch useful for your design.



REQUIREMENTS

1) Your communication system should be able to transmit and receive "characters".
2) The transmitter must convert each character to bit sequences via ASCII encoding. Turkish characters are not necessary. You may use the digitalWrite() and delay functions of Arduino.

3) The receiver must translate the received bit stream to sentences sequentially via ASCII decoding. You may employ the analogRead and serial.write functions dump your data to Matlab in real-time, and do the decoding there.
4) Your design should provide real-time communication.
5) Obviously, **faster** (higher data rate) and **more robust** (fewer bit/character errors under suboptimal conditions) communication systems are desired. You will test your systems using different data speeds, different frequency offsets and random clock delays of different variances, explained as below.
6) **ASCII + On-Off Signalling:**

a) Data Rate: Test **how fast you can decode** the received information correctly: Decrease the pulse durations in the transmitter (Arduino) until you start to get errors in the decoding. In this part, the receiver and the transmitter are synchronized. This means that the receiver knows the data rate of the transmitted signal.

b) Unknown Frequency Offset: Intentionally disrupt the sychronization by introducing a frequency offset. For example, let us assume that you set each bit duration to be 100 ms long, in which case the receiver expects signals of length 100 ms for each bit. Now, instead of transmitting bits of length 100ms, transmit with a 10% frequency offset. That is, transmit bits of length 110 ms constantly (while the receiver still expects bits of length 100 ms) Try with 1%, 5%, 10%, 20% offset, etc. What percentage of frequency offsets can your system handle, before decoding errors starts to occur?

c) Random clock delays: Instead of transmitting bits of length 100 ms, transmit with uniformly distributed random clock delays up to 10%. For example let us say you are sending the letter S, whose ASCII codeword is 01010011, normally to be transmitted using:

"100 ms of silence - 100 ms of signal – 100 ms of silence – 100 ms of signal – 200 ms of silence – 200 ms of signal"

Instead of the above, transmit this by:

"randi(90,110) ms of silence - randi(90,110) ms of signal - randi(90,110) ms of silence - randi(90,110) ms of signal - randi(90,110) ms of silence - randi(90,110) ms of silence - randi(90,110) ms of signal - randi(90,110) ms of signal" (in Matlab notation)

One such realization could be "93 ms silence – 106 ms signal – 97 ms silence – 100ms signal...", another could be "95-91-99-106". The idea is that each pulse will feature a random clock delay. Try with random clock delays of ± 1%, ± 5%, ± 20%, ± 30%... What percentage of random clock delays can your system handle, before decoding errors start to occur?

7) **ASCII + Manchester Coding**

Repeat 6a, 6b, 6c for Manchester coding. A decent decoding algorithm can be found in Section 4.1 at http://ww1.microchip.com/downloads/en/AppNotes/Atmel-9164-Manchester-Coding-Basics_Application-Note.pdf

REPORT & DEMONSTRATION

You will write a project report, where you embed the commented Arduino/Matlab codes.

The report should include short background information on Ascii and Manchester codes and explain your design choices in detail (thresholds, signal processing, detection techniques, etc.). The ASCII decoding should be straightforward, but for Manchester coding, you can either come up with your own method or implement existing techniques in literature/white papers, in which case you need to **cite** your source.

In the report, compare the maximum data rate, maximum frequency offset and maximum average clock delays obtained by both systems. Which system is more robust, on-off signalling or Manchester coding? Explain. Having done the tests, discuss the advantages/disadvantages of both techniques based on their performances. Also include a couple of plots of sampled received signals.

You must demonstrate that both of your designs work properly according to the specifications given in the "Requirements" section. For both of your designs, you must show the maximum tolerable frequency offset and maximum tolerable average clock delay.