PROJECT 2: VISIBLE LIGHT-BASED DIGITAL COMMUNICATION USING ON/OFF KEYING

(TRANSMITTER/RECEIVER DESIGN)

(Groups of 2 – DUE 12/12/2018, worth 30% of your grade)

Description & Desiderata:

In this project, each group is to construct a wireless transmitter/receiver pair, capable of transmitting digital information across few centimeters using light. Specifically, it should employ **on-off signalling**, which is the simplest digital transmission technique, and encode the information (sentences) using ASCII codes. In order to compensate for possible timing errors / offsets / jitter which often occur in communication systems, Manchester coding will also be implemented. The **choice of the data signaling rate** and various implementation choices involved in sychronization, such as preambles or end-of-transmission blocks (if you require) are **left to each group.**

Performance criteria: We will be testing each pair using a “sentence” as the digital information. For sentences, encoding each character via ASCII tables, transmitting/receiving each bit sequentially, and decoding each character (hence the sentence) will be required. (Turkish characters are not necessary)

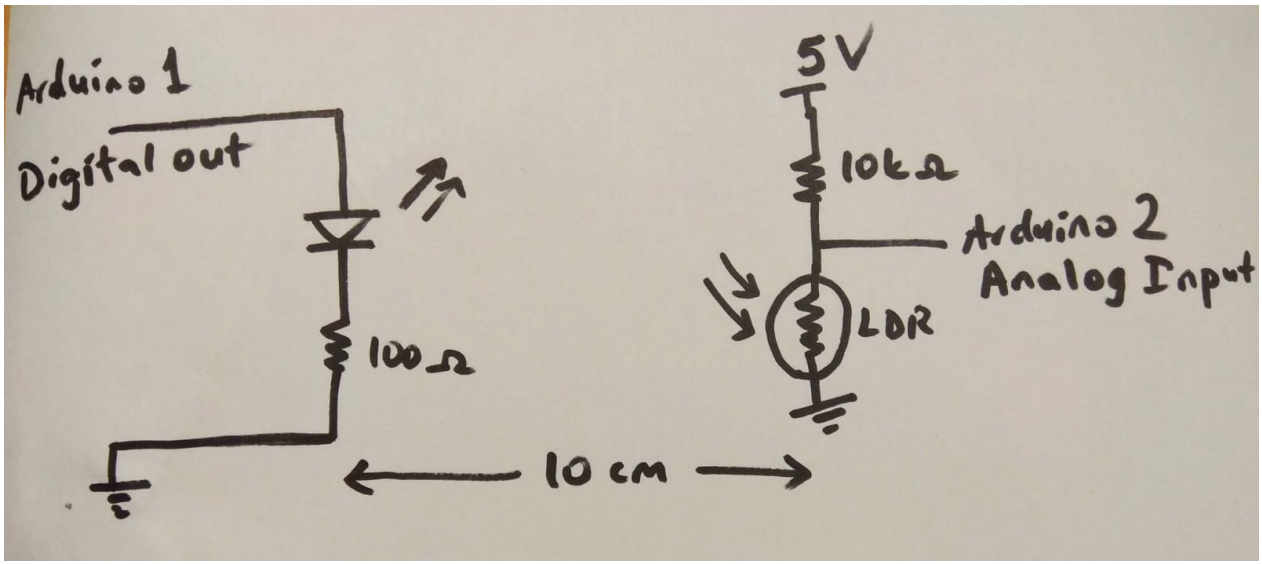
Obviously, **faster** (higher data rate) and **more robust** (fewer bit/character errors under suboptimal conditions) communication systems are desired. You will test your systems using different data speeds, different frequency offsets and random clock delays of different variances.

Transmitter: You will use one of your Arduino boards to encode the letters into a bit stream and turn your transmitter on and off accordingly. You may use the digitalWrite() and delay functions of Arduino.

Receiver: You may employ the analogRead and serial.write functions from the first project to dump your data to Matlab in real-time, and do the decoding there.

Equipment & Components :

Each group has been given two Arduino Uno’s, a white LED, a light-dependent resistor, 2x100 ohm and a 1x10k ohm resistor. You are free to use the equipment in the lab (oscilloscopes, etc.) by appoinment (please email alicangok@gmail.com or can.gursoy@boun.edu.tr beforehand). The overall circuit schematic is as follows:



Reports & Test Cases:

Each group will write a project report, and attach 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 decoding ASCII codes 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). A good decoding algorithm can be found in Section 4.1 of http://ww1.microchip.com/downloads/en/AppNotes/Atmel-9164-Manchester-Coding-Basics_Application-Note.pdf

Specifically, you will test and report on the following:

1) ASCII Code

a) Test **how fast you can transmit/decode** information correctly: Decrease the pulse durations in the transmitter (Arduino) until you start to get errors in the decoding. Note that we assume the receiver (Arduino/Matlab) and the transmitter are synchronized. This means that the receiver knows the data rate of the transmitted signal.

b) 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 trasmitting 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 start to occur?

c) Intentionally introduce **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?

(Please note that while these frequency/clock offsets/delays will naturally occur due to unmatched oscillators or multi-path effects in actual fast wireless communication systems, we are intentionally insterting these errors in our simple and slow systems to gain insight into their undesired effects and how we can mitigate them)

2) ASCII + Manchester Code

Repeat 1a, 1b, 1c for ASCII + Manchester codes. Is your system more robust than using ASCII code by itself; did Manchester Code help in synchronization? Comment on your results.

Having done these tests, you can discuss the advantages/disadvantages of these techniques based on their performances. Also include a couple of plots of sampled received signals that you obtained.