Homework #6: Arduino Wireless Sensor Network

In this assignment, your team’s task is to implement a wireless sensor network (WSN) using three Arduino Unos with sensors, a base station computer, and four XBee radios (plus accessories). At a high level, your WSN will collect time-stamped, synchronized data from three wireless sensor nodes and transmit the data wirelessly to the base station computer.

Each of the three Arduino Unos in your WSN must be connected to:

- a sensor of your choosing,
- an XBee radio shield, and
- an XBee 802.15.4 (Series 1) radio

Additionally, each wireless node must meet the following requirements:

- have a unique, hardcoded ID number (i.e., 1, 2, or 3)
- when powered on, node will remain idle and wait for commands from base station
- upon receiving ‘R’, node will begin sampling sensor at 10 Hz
  - node must toggle built-in LED 13 during each sample
- while sampling, node must periodically transmit data in following format: ID,t,data
  - ID := the node’s ID
  - t := timestamp of data (i.e., 0, 1, 2, 3, ...)
  - d := data for given timestamp
- upon receiving ‘S’, node will synchronize
- upon receiving ‘T’, node will stop sampling and wait for commands

As a whole, your WSN must:

- Implement synchronization
  - When the base station transmits an ‘R’ or ‘S’, the nodes must sample and toggle LED 13 in unison
- Have MINIMAL data loss
  - E.g., If the WSN samples for 30 seconds, then there should be 300 samples from each sensor (with time stamps 0, 1, 2, 3, ..., 299)
- Avoid wireless collisions
  - Nodes must never transmit at the same time

When you are finished, submit a compressed file with your Arduino code base and be prepared to demo your solution in class.
**IMPORTANT!!!**

- Your team must **configure** all XBee radios before using them
  - Four radios: 1 base station, 3 wireless nodes

- You must use free software from the radio manufacturer called **XCTU**

- At a minimum, you must configure the following 5 XBee radio parameters:
  - **CH** - channel
  - **ID** - PAN (Personal Area Network) ID
  - **MY** - 16-bit source address
  - **DL** - 16-bit destination address
  - **BD** - baud rate

- **How to configure each parameter:**
  - **CH** is the radio channel (e.g., F).
    - **All radios** in your WSN must have the **same channel**
  - **ID** is the PAN ID (e.g., 3332).
    - Unique for your team
    - **All radios** in your WSN must have the **same PAN ID**
  - **MY** is the 16-bit **source** address for each radio.
    - Each radio in your WSN must have a **different** MY address, e.g.
      - BACE for your base station radio
      - 1 for wireless node #1
      - 2 for wireless node #2
      - 3 for wireless node #3
  - **DL** is the 16-bit **destination** address for each radio
    - Base station must **broadcast** (DL = 0xFFFF)
    - Wireless nodes must use **base station’s MY** address (e.g., DL = BACE)
  - **BD** is the radio’s **baud rate**
    - **All radios** in your WSN must have **same** baud rate
    - Make sure baud rate matches Arduino’s Serial.begin (e.g., 57600)

- The **XBee shield** has a small switch:
  - **DLINE** – allows you to **program** and **test** the Arduino Uno via USB cable
  - **UART** – transmits / receives Serial data via attached XBee Radio
EXAMPLE
- Using Arduino’s built-in serial monitor:
  - Entered ‘R’ (not shown)
    - Nodes begin:
      - sampling at 10 Hz in unison
      - toggling LED 13 in unison (during each sample)
      - periodically transmitting data
  - Entered ‘S’ (not shown)
    - Nodes synchronize
  - Entered ‘S’ (not shown)
    - Nodes synchronize
  - Entered ‘S’ (not shown)
    - Nodes synchronize
  - ...
  - Entered ‘T’ (not shown)
    - Nodes stop:
      - sampling
      - toggling LED 13
      - transmitting

HINTS
- MsTimer2
- Streaming
- delay( ... )
  - give radio extra time to transmit
- double buffering
  - one buffer filled while other buffer is transmitted

- start with a single node
  - receive commands
  - collect data
  - toggle LED
  - transmit data
  - stop on command

- when working on multiple wireless nodes:
  - start with synchronization
    - make sure all nodes toggle LED 13 in unison
  - use “time slots” to transmit data buffers

- work incrementally and test often!!