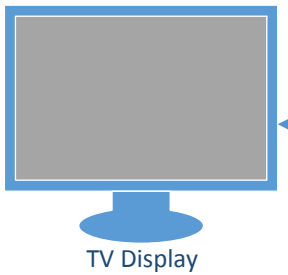
Functional Decomposition
Of Wireless Communication & Display

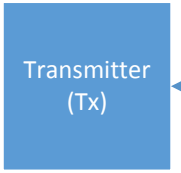
Version: 2.0 (release candidate)
Last Updated: 18th Apr 2016

Prepared by:
Manas Bhatnagar, MHSc
Based on work done by
Health Design Lab at the Emily Carr Design University

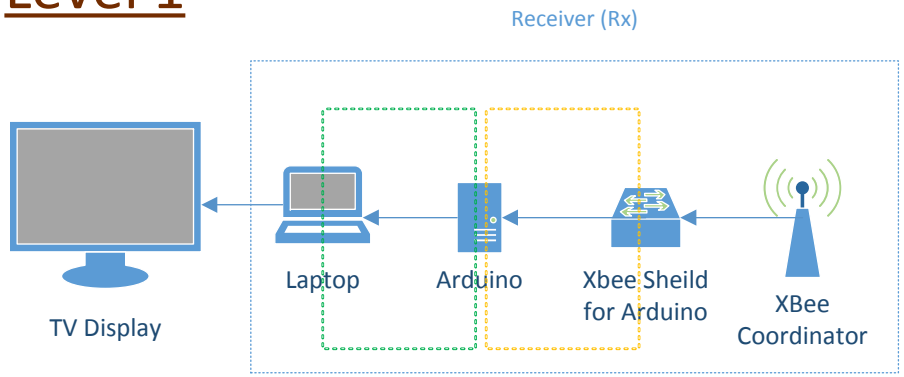
Concept Level

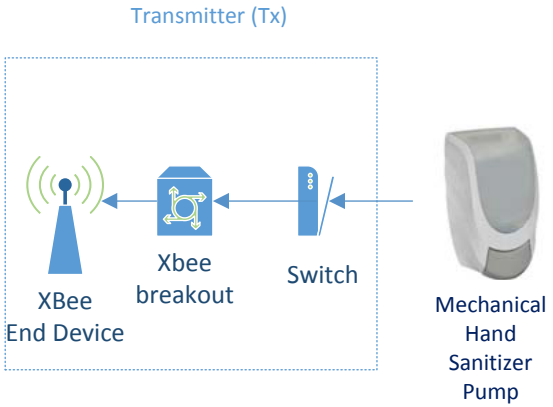
Level 0





Level 1





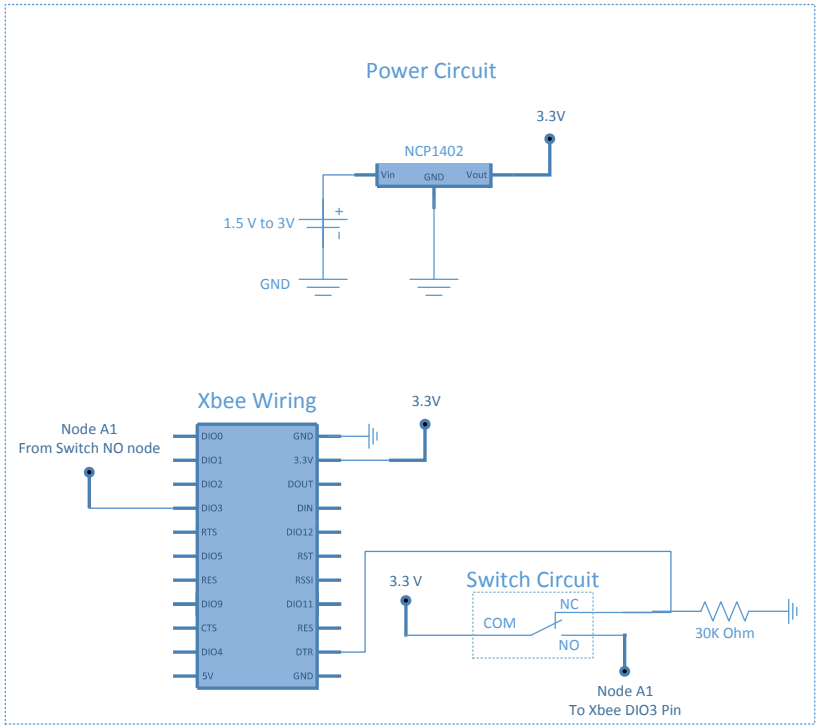
Level 2 – Detailed Design

Xbee to Arduino Communication

- Sketch “XbeeToArduino.ino” runs on the Arduino and reads the Xbee Coordinator output
- Needs two libraries to run – “SerialSoftware.h” and “Xbee.h”

Arduino to Laptop (visualization) Communication

- Sketch “XbeeToArduino.ino” running on the Arduino writes Serial.println “keys” which Processing.js code written by Emily Carr collaborators reads and creates visualization.



XBee

- Many types. We are using the “Xbee ZigBee (2 mW -PCB Antenna)” type.
- Each has a unique factory set ID (Figure 1). We need to note these down and identify the ones we configure as coordinators.
- The pins on the Xbee are too close together for prototyping and so we need to use a breakout board to work with it or read it using a USB. We are using the Sparkfun Breakout board for Xbee Module (Figure 2) for this purpose.
- Before the Xbee’s can be used in a network, we need to configure them -
    - Mount Xbee unit onto Explorer USB and connect to a PC via USB.
    - Start the XCTU application. Click “Discover Devices” button in the top left corner.
    - Click “Next”>”Finish”. Any connect devices should be displayed at this point. Click “Add selected devices”.
    - If this is the **first time** this Xbee is being connected to a computer, then click the “Update firmware” icon. If this Xbee will be used at the receiving end, select “ZigBee Coordinator API”. Else if this Xbee will be used at the transmitting end, select “ZigBee End Device API”. Click “Update”
    - Change “PAN ID” to a unique value. All Xbee’s on a network have the same Pan ID.
    - If you are configuring an Xbee to be used at the transmitting end, then change the following values,
        - Set “JN Join Notification” to Enabled
        - Set “NI Node Identifier” to name this particular Xbee.
        - Set “PL Power Level” to Lowest [0]
        - Set “PM Power Mode” to Boost mode disabled [0]
        - Set “SM Sleep Mode” to Pin Hibernate [1]
        - Set “SP Cyclic Sleep Period” to AF0
        - Set “SN Number of Cycles to power down IO” to FFFF
        - Set “SO Sleep Options” to 4
        - Set “D3 AD3/DIO3 Configuration” to ADC [2]
        - Set “IR IO Sampling Rate” to 190. This makes the Xbee send its state every 400ms. Set it to 3E8 if you want the Xbee to send its state every second. Set it to 1F4 if you want the Xbee to send its state every half second. For other time intervals, use a decimal to hexadecimal calculator to convert time in milliseconds to its hexadecimal representation.
        - Set “V+ Supply Voltage High Threshold” to FFFF
    - If you are configuring an Xbee to be used at the receiving end, then change the following values
        - Set “NI Node Identifier” to name this particular Xbee.
        - Set “AP _____” to 2
        - Set “SP Cyclic sleep period” to AF0
        - Set “SN Number of Cycles to power down IO” to FFFF
- Click “Write Radio Settings” icon to save these setting to the Xbee.

Figure 1

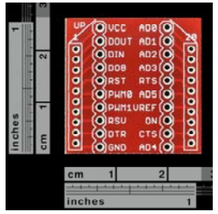
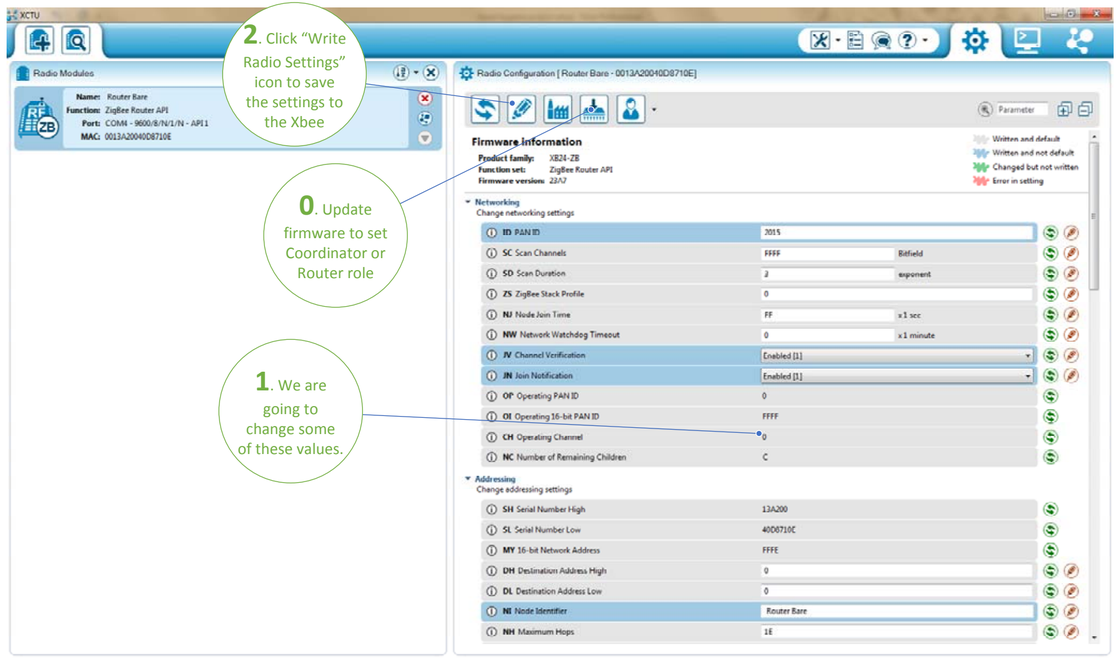


Figure 2



Arduino

- Many types. We are using the “Arduino Uno SMD Rev3” type.
- In addition to the Arduino IDE, you’ll need to ensure that the following libraries are installed in the Arduino IDE –
    - SoftwareSerial.h - https://www.arduino.cc/en/Reference/SoftwareSerial
    - XBee.h - https://github.com/andrewrapp/xbee-arduino

- Arduino code (“sketch”) XbeeToArduino the code that will need to be changes if any changes or additions are required.
- Once the code is changed, click “Verify” icon to ensure there are no errors.
- If the code successfully compiles, click the “Upload” icon to download the code to the arduino board. This code will run in a loop whenever the arduino has power.

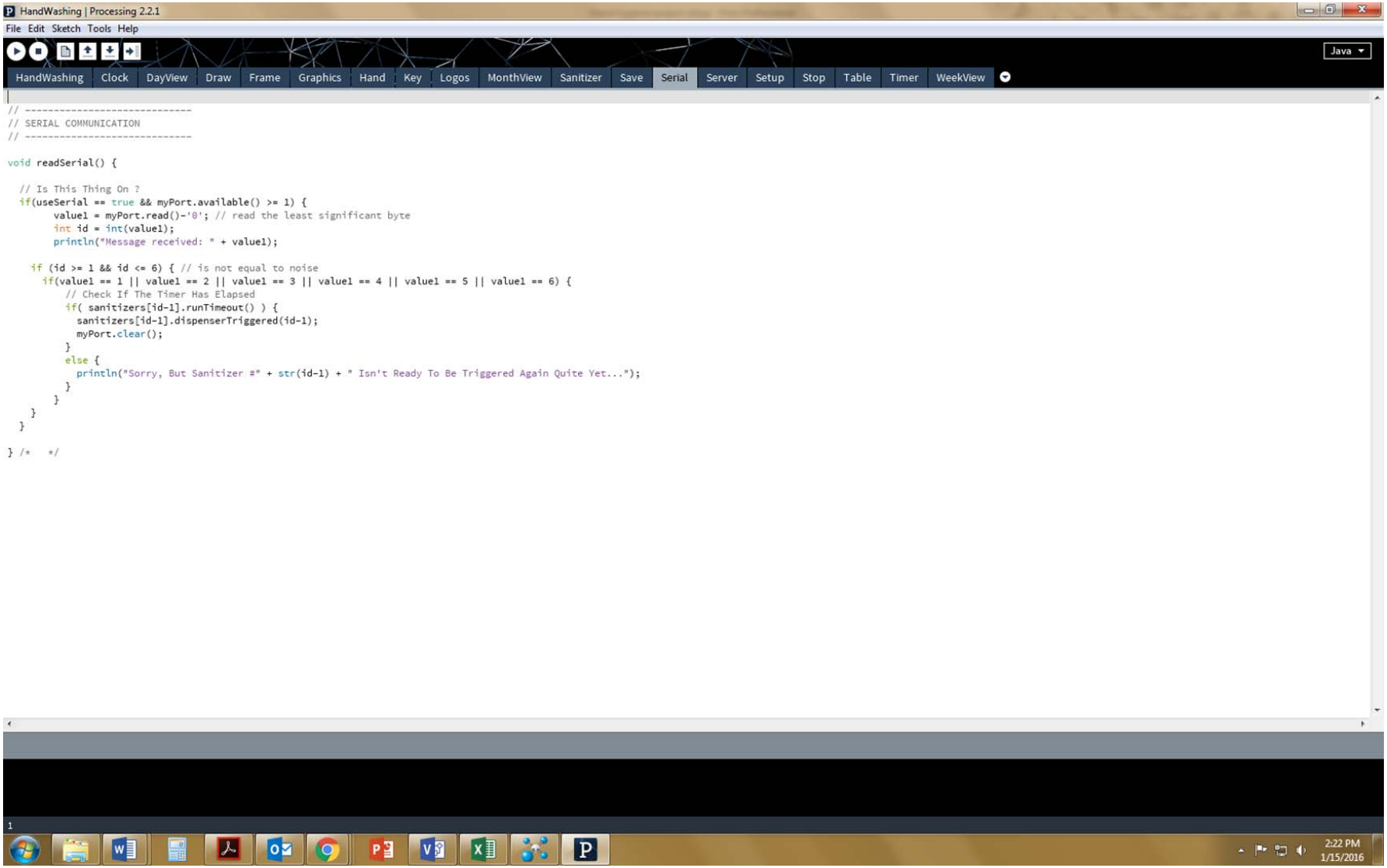
- For troubleshooting, a lot of the commented code can be uncommented and its output can be viewed on the “Serial Monitor” in the Arduino IDE.

Adding a new Xbee to Arduino code so visualization can occur

- After the Xbee is configured appropriately (See section on Xbee), convert the last four digits of the Xbee’s address to decimal format: http://www.binaryhexconverter.com/hex-to-decimal-converter
- Enter this decimal number into the swtich statement as a new case.
- *Serial.println(“X”)* ← ‘X’ can range from 1 to 6 depending on what colour of visualization you’d like.
- Compile the code. If Compilation is successful, upload the code to the Arduino.

Processing

- Processing.js creates the visualization. Our code runs on the Processing version 2.2.1. Newer versions are available, but I'm not confident that we will not face backwards compatibility issues.
- To change any aspect of the visualization, this is the code that must be changed.
- When any of the code is opened in the Processing IDE, many tabs are created. Relevant tabs might be "Serial" and "Handwashing".
- Maintain the files/folder structure.
- The original Emily Carr code also does table creation and backend utilization counts. But my recommendation is to recreate all research tracking from scratch in the Arduino code.



Steps to set-up code on a new computer

- Install Arduino IDE from: https://www.arduino.cc/en/Main/Software
  - If asked for permissions to install drivers, say yes.

- Once installation is over, start the Arduino IDE. Click Sketch>Include Library>Manage Libraries..
  - In the search bar type “Xbee” and install the Xbee-Arduino Library by Andrew Rapp (You will need to be connected to the internet for this step)
  - Next search for “SoftwareSerial” and ensure that it is installed

- Plug in the Arduino Board.
- Windows might look for device drivers if this is the first time an Arduino is being connected to the machine. Allow it to do so. It may take several minutes.
- Once device drivers are installed Windows will display a COM port associated with the Arduino. Ex: COM1, COM3, COM6 etc.
  - If this is not shown, you can find this information by going to the Control Panel>Hardware and Sound>Device Manager
  - Click ‘Ports’ and note the COM# associated with the Arduino board

- Obtain a bright green USB named HndHygInstl. It contains most of the files needed to make the code run on another machine.
- Copy all the files from this USB on to the computer.
- Open folder “processing-2.2.1” and run the processing application. If prompted to upgrade version, say no.
- Once the processing application has started (an empty sketch). Close it.
- Go to folder labeled ProcessingCode>Processing>HandWashing. Open file called Setup. (You might have to maximize the window size of the Processing IDE and find the Setup tab)
- In the “if(useSerial == true)” statement, in line 37, change the value of COM# to match the value you noted down from the device manager in previous steps.
- In the ‘Handwashing’ tab, change the Network_Name variable to the correct value, i.e. Network_201#
- Save these changes by clicking File>Save.
- Ensure the Arduino is plugged in. Then turn on a transmitter unit. And Run the Processing code by clicking the play button near the top left corner.
- If all is well, then you should see the black screen of the application start and you can test the system by using the transmitter to create a hand print on the screen

Versioning Convention

- 0 for alpha (status)
- 1 for beta (status)
- 2 for release candidate
- 3 for (final) release.

To-do List

- ~~GitHub source control for all code~~
- ~~Add flag for low battery~~
- ~~Add code to track usage~~
- ~~Add software filter to reduce false positives~~
- Periodic polling to ensure all devices are functioning