Model HHIxD 400

Set-Up Guide for Open-Source Smart Hand Hygiene Dispensers

Johanna Blaak, Reilly Baggs, Rachel DiMaio, Dr. John Conly

W21C

Includes instructions on:

1. Hardware Configuration
2. Network and Message Transmission
3. Visualization and Interaction Design
4. Data Management and Web Interface

Version 1.0

Contents

Introduction

This guide is still in active development and improvements are scheduled to be published in Q4 2019. Any questions regarding different aspects are welcome on the GitHub page of the project.

This guide will provide you with the all the information needed for the integration of the internet of things (IoT) hand hygiene innovation. This guide goes hand in hand with the GitHub repository listed below, in this repository you will find all the code for the different aspects.

https://github.com/W21C

Project History

This projects started at the Emily Carr University of Art + Design (ECUAD) as an initiative to bring interaction design to hand hygiene through an open-source solution, with an implementation at Vancouver General Hospital. Through time the project has evolved, and W21C has since taken the lead in the project. In 2017 the project was implemented in a unit at Foothills Medical Centre in Calgary and in 2019 the current version of hardware was implemented in Alberta Children's Hospital.

W21C

W21C is a research and innovation initiative based in the University of Calgary (UCalgary) and the Calgary Zone of Alberta Health Services (AHS). W21C conducts health systems research with the overarching mandate to improve patient safety and quality of care. Through this strategic research and innovation agenda, we hope to **make care better** for our communities, both now and in the future.

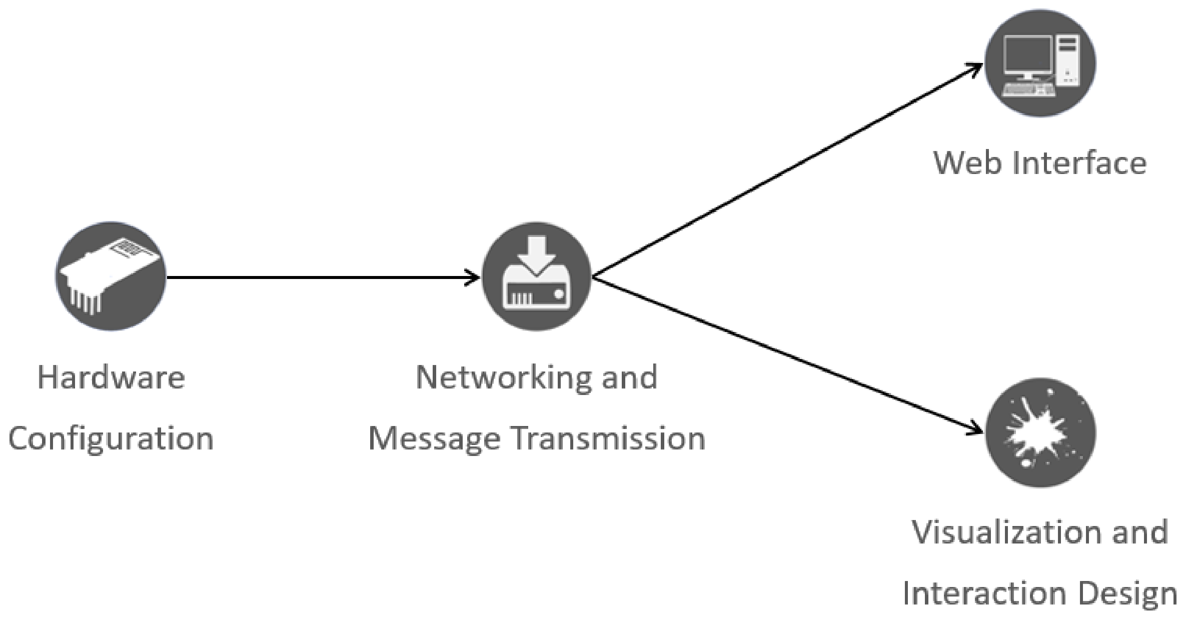
https://www.w21c.org/

Overview

Understanding the system

The system consist out of four categories:

- **Hardware configuration:** the sensor module that is installed in the ARB dispensers and send messages to a central server.
- **Networking and message transmission:** the server that receives all messages, logs them and acts as a broker in order for other parts of the system to be able to receive messages.
- **Visualization and interaction design:** the interactive display that shows usage of the connected ARB dispensers in real-time.
- **Data management and web interface:** an administrative dashboard, allowing users to view use over time and current battery status of the different hardware modules.


Web Interface
Hardware Configuration
Networking and Message Transmission
Visualization and Interaction Design

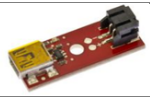
Hardware Configuration

Configuring your sensor hardware and installing it

The primary hardware component that is installed in the ABR dispensers are ESP8266 modules. ESP8266 modules are Wi-Fi enabled microcontrollers that can be programmed in the Arduino IDE. The ESP68266 modules are connected to a button that, when pressed, initiates the sending of a message. The button is installed in the dispenser such that it is pressed whenever the ABR dispenser is used.

This section will show you how to 1) create the simple circuit, 2) how to program the ESP8266 within the Arduino IDE and 3) hardware installation in the ABR Dispensers.

Parts you need

Part	Part Name	Quantity	Description
	NodeMCU	1 per unit	The ESP8266 module that connects to Wi-Fi and the broker to transmit messages
	Lithium Ion or Lithium Polymer Battery	1 per unit	3.7V, 2000mAh power supply
	Switch Button	1 per unit	Simple button to be used for the triggering of message transmission
	Resistor	1 per unit	10K Ohm resistor (used in the circuit as a pull-up resistor to ensure that the reset pin is not accidentally pulled down to ground when the button is not pushed)
	Battery to Wire Connector	1 per unit	A small connector that makes it easier to unplug the battery from the circuit for charging
	Prototyping Board	1 per 4 units	Board onto which the circuit is soldered (2 hole strips)
	Battery Charger	1 per setup	An adaptor that allows the batteries to be charged through a Micro-USB cable
	Spring	1 per unit	
	Plastic Cover	1 per unit	

Tools you need

Tools	Description
Soldering tools	Soldering iron, solder, wire cutter/stripper
Drill	
Micro USB	
Computer	

Building the circuit

The circuit is built using the parts listed above and parts are soldered onto the prototyping board into the configuration shown in Figure 1.

1. Measure the length of the wire needed for the button location and the place the hardware is located in the ABR dispenser. See figure 1.
2. Connect button to ground and to the reset pin on the NodeMCU.
3. Connect the resistor to the 3.3V power, connect the other side to the reset pin.
4. Connect the battery connector, assure the connector aligns with the positive and negative pins upon battery connection.
5. Trim the pins of the NodeMCU with wire clippers to about 3 mm, for easy installation into the ABR dispenser.

The final circuit is displayed in Figure 2

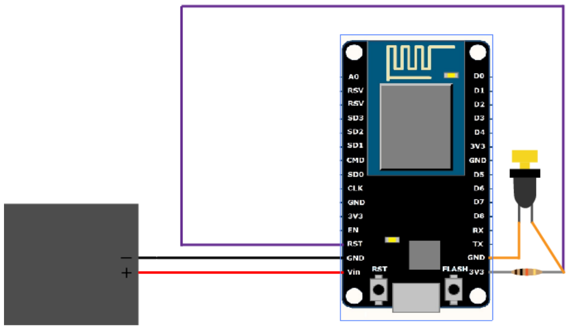


Figure 1 - Wiring Diagram

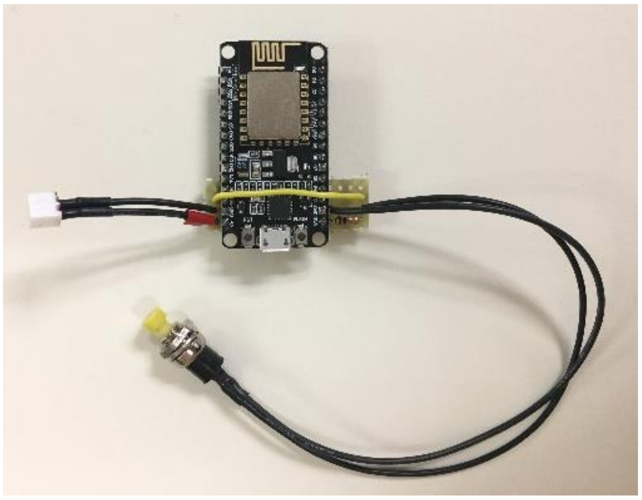


Figure 2 - Final Circuit

Programming the NodeMCU

The code used to program the NodeMCU is written in C/C++ using the Arduino IDE. The Arduino IDE is an open-source development environment that can be downloaded for free from the online Arduino site onto any operating system (Windows, Linux, etc.). If you do not have it downloaded, do so here:

https://www.arduino.cc/en/main/software

In order to upload the code to the NodeMCU, first you need to download and install the ESP8266 Board Manager in Arduino IDE.

1. Open the Arduino application and go to File > Preferences
2. Under 'Additional Boards Manager URLs' enter the URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json and press OK
3. Navigate to Tools > Boards Manager and search for ESP8266, and install this board manager
4. To confirm it is installed, you should be able to choose NodeMCU under the Tools tab.

The code is available in the folder hardware folder on the GitHub page. The following modifications should be made to the code before you upload it to the NodeMCU.

Inserting the correct Wi-Fi information and MQTT.

1. Networking and Message Transmission. Find the following section in the code:

```
//WIFI  and MQTT Credentials


const char* ssid = "Wi-Fi name";

const char* password = "Wi-Fi password";


const char* mqtt_server = "Server IP";

const int mqtt_port = 1883/Port;

const char* mqtt_username = "MQTT username";

const char* mqtt_password = "MQTT password";
```

2. First change the blue highlighted sections to the correct information. Server IP you can find, on the machine that is hosting the Mosquitto broker by opening cmd and typing ipconfig (Windows), ifconfig (Linux). Use the IPv4 (e.g., 192.168.0.1). Unless you changed the port configuration in your Mosquitto broker, use the port 1883.
3. A standard install of the Mosquitto broker will not come with a password or username. In this case you can comment the lines with MQTT username and password out, by adding // in front of those lines.

The changes you just made are for each pump the exact same, and you do not have to worry about configuring this again. The following change you will have to make for every single pump as this is the unique pump identifier (or payload that will be send).

1. Look for the following strings of code:

```
reconnect();

 client.publish("pumpID", "00"); //The number sent here identifies 
the pump

 Serial.println("Published: 00");

 delay(100);
```

2. Change the 00 in the code to the unique number you want the pump to have, any number between 00 and 99 works (e.g, 04).

Uploading the code and testing it:

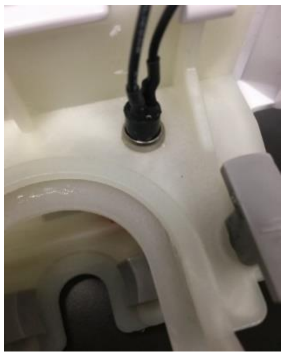
1. Connect the NodeMCU to the computer using a microUSB cable.
2. Check if the connection is recognized, you will see in the bar underneath a com port listed indicating a connection has been made.
3. Verify and upload the code by clicking the upload button.
4. After it is done uploading the code to the NodeMCU, open the serial monitor (**Tools → Serial Monitor** or **Ctrl+Shift+M**), click the button and check if he registered the button click and is able to connect to the broker
   Note: make sure the broker is already configured, otherwise you will only see the button click appear in the serial monitor.

Installing the hardware

The ABR dispenser must be minimally modified to accommodate the button, NodeMCU and battery.

These modifications are not universal, the basic mechanism is similar to all pumps but for any dimensions or exact/perfect location for the button measure the pump type used.

Additional photos and illustrations of these steps will be presented in the next version of this document.



1. For the placement of the button drill a small hole in the lower piece of the ABR using a drill bit suitable for the button you purchased. Then place the button through the hole and secure it with the button's nut.

2. A spring must be added to connect the button to the lower platform so that it triggers when the pump is used. A 27 mm spring with a diameter of 11 mm was used and hot glued to the button. Assure you use the right size spring for your ABR pump.

3. To hold the NodeMCU and battery in place and protect the electronics from potential leaks of dispenser fluid, a plastic pouch was attached to the back-plate. In this case, a plastic card cover was attached to the back of the pump interior. However, any similar plastic covering could be fashioned with available materials.

Troubleshooting

NodeMCU is not recognized by computer

Try using a different USB port or manually select the port: **Tools → Port**

No libraries found

Ensure the libraries are installed: **Tools → Manage Libraries…** or **Ctrl + Shift + I**. Install the following libraries: <ESP8266WiFi.h> <PubSubClient.h>

Networking and Message Transmission

Setting up your MQTT Broker

The networking component of the project uses an open-source version of the MQTT messaging protocol called Mosquitto. A Mosquitto broker can be set-up on any device with the following capabilities:

- If using an established Wi-Fi network, connect to Wi-Fi.
- If unable to use an existing Wi-Fi network, able to create a hotspot.

In the case of this right up we are using a small computing device called a NUC. The NUC also acts as a wireless router, providing Wi-Fi for the various clients that make use of the MQTT protocol; the three clients represent the three other components of the project including the hardware, visualization code and user interface. The hardware clients send messages which are passed on to the other two clients.

Additional information on the messaging protocol and the open source broker can be found here:

https://mosquitto.org/

A recommended and extensive documentation on MQTT and how it works can be found here:

http://www.steves-internet-guide.com/mqtt/

Configuring your broker

This is by far the best and most comprehensive guide we have found in order for you to set-up a Mosquitto Broker. The following URL is for a Windows install, but provides a link to a guide for a Linux install.

http://www.steves-internet-guide.com/install-mosquitto-broker/

Our recommendation is to use a Linux OS as these seem to be more reliable.

Visualization and Interaction Design

Setting up the visualization

The Interaction Design provides immediate visual feedback upon use of the ABR dispensers. The display screen shows an incremental count of the number of hands cleaned that day and a themed graphic appears with each use, created using Processing code.

This section will show you how to 1) install the visualization software, 2) navigate the visualization.

Parts you need

Part	Quantity	Description
Display Screen	1 per setup	TV Screen, computer monitor, or other large display screen
Computer/ Raspberry Pi*	1 per setup	Computer should be capable of connecting to the network on which the pumps are hosted. Raspberry Pi 3 B+ (such as a Starter Kit sold by CanaKit on Amazon with a case, power cord, SD card etc.)
HDMI cable	1 per setup	To connect Raspberry Pi to display screen, depending on your screen a different type of cable might be required
Keyboard	1 per setup	To navigate the visualization display by connecting to the Raspberry Pi

*In case no computer is available, a good cost alternative is the use of a Raspberry Pi. A Raspberry Pi is a single board computer capable of running the visualization software without issues. Turning an old TV or display into a smart TV.

Installing visualization software

1. If you are using a Raspberry Pi, use a memory stick to upload a version of the Raspbian operating system with Processing 3 included onto the Raspberry Pi. Once this is done, the Processing code can be immediately transferred and run on it. For the online download of Raspbian, and more resources on how to use a Raspberry Pi, check out:

   https://www.raspberrypi.org/downloads/

2. Make sure your Raspberry Pi or other computer is connected to the right Wifi network.

3. Download the Process Visualization Code folder from the Github page and do not change the file structure. This includes the software for processing and the code needed to run.

4. Open the code by going to **Process Visualization Code → Visualization code → Handwashing → Handwashing.pde** and change the following line of code for your specific MQTT login and password, and server IP-address.

   If you have multiple visualizations running, change the name of client. Change “processing1” to “processingx” to ensure a unique identifier.

[…]

```
//MQTT IMPLEMENTATION

client = new MQTTClient(this);

client.connect("mqtt://MQTT_Username:MQTT_Password@Broker_IP",
"processing1");

client.subscribe("pumpID");
```

5. After you change the code, ensure everything is running smoothly by running the code.

6. Export the code to .exe by navigating to **File → Export Application**. Make sure to export the code on the same operating system (OS) that the final code will run on (due to dependency differences on OS). If you have exported the code on a computer that is not the intended execution computer, test that the code runs on the intended computer.

7. Set up an auto-restart on your computer to avoid the processing code slowing down the computer. Processing tends to occupy memory and drive capacity. To set up auto-restart on Windows 10:
    a. Open the task scheduler by typing taskschd.msc into the start search
    b. In the right panel, click Create Basic Task
    c. Assign a name and a description if wanted and click next.
    d. Select when you want the task to start (ex. We chose daily) and click next.
    e. Select the start date and time (ex. Is it easier for your computer to restart overnight) and how often you want the recurrence (ex. Every 2 days).
    f. On the action page, select Start a Program and click next
    g. On the Program/script space type shutdown and in the Add arguments box type /s/f/t 0.
    h. Click next to review all and finally click finish. Your computer will now automatically restart at the designated day and time.

8. Set up an automatic start of the visualization code upon computer start. To do this in Windows 10:

    a. In the address bar in file explorer, enter %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup.
    b. Place a shortcut to the processing application into this folder.
    c. Restart the visualization computer and the program should launch automatically.

9. Connections: connect raspberry pi to display screen with HDMI cable (or other?). Connect a keyboard to your Raspberry Pi to navigate the visualization with keystrokes.

Navigating the visualization

When the processing code is running, different keystrokes can control functioning or themes as outlined in tables X and Y.

Key	Function
R	Reset the data (restart the count of hands)
1,2,3,4,5,6	Trigger one of the Dispenser objects in the Code (simulate the triggering of the ABR dispenser of a specific ID)
Z	Change the drawstate
Y	Change the subdrawstate (type of logo)
Q	Toggle the complex germ theme on and off (the complex germ theme must be toggled off in order to change the theme to something else)

Key	Theme
H	Hands (original)
A	African Animals
C	Canadian Animals
O	Ocean Creatures
G	Dogs
I	Insects
T	Vehicles (‘t’ for transport)
V	Germs (‘v’ for viruses)
F	Fantasy Creatures
P	Dinosaurs (‘p’ for paleontology)

Troubleshooting

The code reports the error 'libraries not recognized'

1. Make sure the libraries are all in the sketchbook folder:

   >>C\...\Processing Visualization Code\Visualization Code\Handwashing\Libraries

2. If they are not, then a backup of the needed libraries is found in:

   >>C\...\Processing Visualization Code\Libraries

3. In the Processing IDE, go to Preferences → Sketchbook location and check that the sketchbook location is the same as where your code is located:

   >>C\...\Processing Visualization Code\Visualization Code\Handwashing

The visualization is stuck in <u>Germ Squashing</u> mode

1. Use mouse to click interface
2. Press Q
3. Press the key for the wanted theme.

The screen is no longer working but still on (stuck in Windows update or not receiving count).

1. Press windows key on key-board
2. Click power → restart

Screen will restart and automatically launch interface.

The screen is off

1. Make sure the power is plugged.
2. On/off-switch is at the bottom of the screen.

Screen will restart and automatically launch interface.

Data Logging & Web Interface

Web interface set-up and data collection

This section is under development, full release of this section should be published by the end of 2019.

Data management and the web interface that allows you to add pumps and view data has been programmed in Python 3.7. Two different scripts will be running at the same time, one dedicated to logging data and the other dedicated to hosting the web interface.

Data Management

The script dedicated to logging data in a SQL-lite database is integrated with the one that displays the data. Both scripts use the same database. Every single trigger is stored with a unique id, exact time and pump identifier (pumpID). After every trigger the current battery level is logged for the triggered pump in the same database.

In our current version every pump is manually added to the system through the web interface with the unique pumpID.

Web Interface

The purpose of the interface is to display and visualize the ABR use data, allowing users to quickly view frequency of use data. The data is displayed in a graphical format with bar graphs comparing the total use of each dispenser over an hourly, daily, weekly and monthly period as well as with line graphs showing the use of each dispenser over time (in hourly, daily, weekly and monthly intervals as well). This allows users to explore the data in greater depth and identify trends in order to gain insights from it. Custom data intervals can be selected for visualizations.

Additional features in the web interface is the display of battery levels and the ability to add pumps to the system.

Installing the web interface and data logger

This part is still under construction. Below are abbreviated steps for the process. Additional information and explanation will be provided in the next version of this document.

The installation of the python application is using command prompt to install the packages and install the code. You can use different development environments to do this in different ways, but these will not be discussed in this document.

The following steps are set up for a windows computer, if you are using a different computer we recommend you to consult the external sources.

https://www.python.org

1. Install python 3.7 and assure "python" is added as a path variable.

   Adding Python as a path variable in Windows:

   https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/

2. Assure pip, python package manager is installed.

```
py -m pip --version
```

3. And update pip.

```
py -m pip install --upgrade pip
```

4. Virtual environment
   This is an optional part, but allows for a cleaner install of the Python application and is recommended.
   a. Install virtual environment

```
py -m pip install --user virtualenv
```

b. Create virtual environment

```
py -m venv env
```

c. Activate virtual environment

```
.\env\Scripts\activate
```

d. Leaving virtual environment

```
deactivate
```

5. Installing package requirements

```
pip install -r requirements.txt
```

6. Enter the MQTT brokers IP address in the data-logger code.
7. Running the code
   You need to run two different python codes in parallel, one is responsible for logging the data the other is responsible for hosting the webpage.
   a. Running the data logger:

```
python MessageReceiver.py
```

   b. Running the web page:

```
python HH_Data.py
```

Accessing the web interface

The webpage created is hosted on the local network (that which is created by the NUC) and is accessible to any device on the same network. To access the page:

1. Connect to the Wi-Fi network the MQTT Broker is on.
2. Once on the network, go to a web browser and enter the URL http://168.xxx.xxx.xxx