CSCE 315 Project 1: People Counter Hand-Off Documentation

**Description of Project and Methodologies**

For this project our team developed three systems that interlock to allow our customer to count the number of people entering a building with 2% accuracy (based on a sample size of 5 test runs of 100 people walking through the sensor). The HCSR04 Ultrasonic ranger1 was used in conjunction with an Arduino to detect a difference in range. Readings from the Arduino were communicated across a Serial Socket2 using a custom protocol that is defined in the later portion of this document. This protocol and the Arduino/Sensor was abstracted into the Python class SensorApp. The DataFilter class then calls on the SensorApp to get data and then analyzes it to determine the number of people that have walked by. The ServerApp calls on the DataFilter once every 15 minutes to find the number of people that have passed and updates the database with that value.

Our focus was on creating a seamless, well-timed user experience that abstracted the pains of dealing with the MySQL database as well as the Arduino serial interface while still allowing the user to be able to customize the movement of that communication. The Arduino can be configured to handle up to four ultrasonic sensors as well as have a varying update rate and communication rate. Similarly, the ServerApp can be connected to any database of the users choosing without issue. However, for the DataFilter, we took a much more strict approach implementing only the use of two ranging sensors. This streamlines the proof of concept and was eventually shown to provide adequate accuracy for the product.

The user is encouraged to examine the accompanying documentation to see more specific use cases and issues.

**Dependencies**

Arduino

1. Compiled using Platformio on Windows 10 using Cygwin3
   1. In the Setup portion, we will be walking you through utilizing the Arduino IDE to compile and download the code. Both methods for deploying the program have been tested, but the original compiler should be noted.
2. HC-SR04 Hardware
   1. If you do not use the HC-SR04 ranging sensor, you will need to change the method for configuration. This protocol was written specifically for the HC SR04 and any changes to that hardware will mean that it does not work.

Python

1. The PySerial module was used for communication with the Arduino. You can find it using pip or easy\_install. That will be discussed more in setup.
2. Schedule is used by the ServerApp to schedule the database updates.
3. \_mysql is used for communicating with the database.

PHP

1. To develop this we used PHP Storm to actually write and run a local server for the website and it is recommended that you use the same one
2. Additionally to use the test cases that are provided with these functions you will need to use the PHP Unit functions that are apart of a package associated with PHP Storm. You can learn how to install them [here](https://www.jetbrains.com/help/phpstorm/enabling-php-unit-support.html)

**Setup**

This section will walk you through the setup of the Arduino, the python code, and then the PHP server.

Database

1. See MySQL \_Table\_Setup.txt for instructions on how to setup the database inside of the PHPMyAdmin.
2. Verify that the table is setup correctly by pushing some data into and out of it.

PHP

1. Copy PHP files into a development server (this can be build.tamu.edu/projects.cse.tamu.edu or whatever) or use PHPStorm as a development server.
2. This will act as your front end to show the website.
3. It is recommended that you use the local server built into PHP Storm to run the website for development and the PHP Unit for doing all of the test cases. A link on how to install both of those is provided in the PHP dependencies section.

Arduino

1. Either clone the provided github.tamu.edu link or unzip the provided folder onto your computer.
2. Open the Arduino IDE (if you have not downloaded it you can find it [here](https://www.arduino.cc/en/Main/Software).
3. Inside the IDE go to the top menu bar and enter File🡪 Open.
4. Navigate the Open menu to the directory where you unzipped the provided code.
5. Inside the directory go to [SOURCE DIRECTORY NAME] 🡪 Arduino\_ID\_Build🡪csce\_315\_project\_firmware and click on the file named: “csce\_315\_project\_firmware.ino”
6. Press Open.
7. Now the file should be open inside of the IDE. If you couldn’t find the file, then contact the Professor or try unzipping the file again and follow the above steps.
8. With the file open press the checkmark button in the upper left-hand corner. This is the “Verify” button.
9. The file should compile without issue. If there is a compile error, check your dependencies and then figure out who futzed with my code.
10. Connect the Arduino Uno board to the computer using the USB to Serial cable provided. Make sure the IDE has the right “COM” port selected.
11. Once the project is compiling, press the 🡪 button in the right-hand corner next to the “Verify” checkmark. This will deploy the code to the Arduino. You should get a success message in the console on the bottom.
12. You should now be ready to interface with the Arduino board.
13. You also need to make sure you wired up your ultrasonic sensors and take note of the pins that you put them on. You will need to modify DataFilter.py to reflect those changes.

Python

1. The major steps below will walk you through setting up the dependencies inside of our project. You should first make sure that you have Python 3.3 or later installed on your computer and included in your PATH variable. Don’t know how to do that? Here’s some references:
   1. Installing Python [here](https://www.python.org/downloads/).
   2. Adding Python to your PATH [here](https://anthonydebarros.com/2015/08/16/setting-up-python-in-windows-10/).
2. Now with Python setup you need to install Pip (Python Package Manager… it makes the rest of this really easy).
   1. How to install pip [here](https://stackoverflow.com/questions/4750806/how-do-i-install-pip-on-windows).
3. Now with your Environment Variables setup and pip install open a Command Prompt (or in Mac a terminal command) .
4. Inside of the Command Prompt enter python and press enter.
5. You should see the IDLE open in your command prompt and that verifies Python’s installation.
6. Type in exit() and press enter.
7. Now we are going to install the dependencies using pip.
   1. Type “pip install schedule” and press enter.
   2. Check that it worked.
   3. Type “pip install pyserial” and press enter.
   4. Check that it worked.
   5. Type “pip install mysqlclient” and press enter.
   6. Check that it worked.
   7. If you have any issues with the above, please consult the gospel of Google.
8. Now you are ready to run the project!

**Running the Project**

To actually run the project you need to make sure that the above two stages have been setup. You also need to have a MySQL database that you have access to with a username, password, database name, host URL, and table name.

1. Navigate to the source directory to which you unzipped the project.
2. Once in the directory open Command Prompt/Terminal and navigate to the python\_src folder.
3. Here you can either modify demo.py with your database information or you can simply create a new python file and include the ServerApp in it and follow the example shown in demo.py.
4. You also need to modify DataFilter.py with the echo/trigger pins that you setup for the HCSR04s in the Arduino setup Section.
5. Once you’ve entered your data and are ready to get going plug the Arduino into the computer, get the port number (this can be found in System -> Device Manager on Windows and opening the Device Manager form the Dashboard on Mac/Ubuntu/Linux.
6. Once you have made sure that the Arduino is connected and you’ve followed the Arduino Setup so the firmware runs properly you can run python [main\_file\_name] and press enter.
7. You should then see a brief pause before SUCCESS prints in the terminal. Now you have a running ServerApp!
8. If you want to see a demo of how to run this project alongside another, you can look at demo.py which runs the ServerApp inside of another Thread.

**Communication Protocol**

Main Communication protocol::

- Three Characters for Command

- Followed by actual data being sent back and forth

- End of command/data flow is delineated by an endline character

Supported commands

- RDY = Ready to be configured (Comes from the Arduino Cleint

- CON = Connected, about to send data (Comes from the Python host)

Followed by updateRate, TRI, triggerPins,ECH, echoPins, EL

Example string “ CON5,TRI,1,2,3,4,ECH,5,6,7,8,\n”

- SUC = Succesfully Configured (Comes from the Arduino Client)

**Additional Notes**

1. The HC SR04 Documentation can be found [here](http://www.micropik.com/PDF/HCSR04.pdf).
2. The Pyserial module was used for communicating over a COM socket at 250 kHz. The baudrate is fixed because it is hard coded into the firmware, but redeploying the firmware with a different baudrate would allow for differences (Documentation for the PySerial Module can be found [here](https://pythonhosted.org/pyserial/)).
3. Cygwin is a bash emulation tool that was used in conjunction with Platformio to simplify the user experience for the programmer…. He hates using that Arduino IDE and was complaining too much so we let him have his fun. You don’t need to use this method. Simply use the Arduino IDE to Verify and Download the code onto the Arduino and everything will be peachy. We have verified that both work so no worries.