**User Manual**



Table of Contents

[**Raspberry Pi with Sensors**](#_9u5h1t69kdsw) **2**

[**Java Server, mySQL Database**](#_wkcy2mjo8ld) **3**

[Troubleshooting Tasks](#_tiuw9pvsrf1h) 3

[Maintenance Tasks](#_3hio55tii72h) 3

[**Rescue Dog Tracker Web Page**](#_gt9fh9rljtf3) **5**

# 

# 

# Raspberry Pi with Sensors

# 

# 

# Java Server, mySQL Database

*User:*The end user will not have any direct interaction with the Java server or the mySQL database. This portion of the rescue dog tracker passes information from the rescue dog(s) to the rescue dog tracker web page.

*Admin:*

The admin or technical user requires access to the java server for several tasks. Troubleshooting and administrative tasks will encompass what the admin/tech user will need to accomplish from the java server/mySQL DB console.

### *Troubleshooting Tasks*

* **Observe console/standard output on the server.** This textual output informs the admin of information received by the java server. The console also reports each iteration of an update to the rescue dog tracker web server.
* **Exceptions log file.** Every exception within the java code is captured by the server and written to the log file. The java server is designed to never go down and require no input from an admin to continue functioning. Detailed error reporting is appended to a local log file for analysis.

### *Maintenance Tasks*

* **Provisioning of an additional raspberry pi with sensors to trained dog.** The mySQL database uses MySQLWorkbench as a maintenance interface for the database maintenance tasks. In the event that a new raspberry pi with sensor is deployed with a new dog, the admin user must insert a new record in the DOG table with the following information.
  1. DOG\_ID. This is an integer field that must be passed to the associated Pi device uniquely identifying the dog assigned to the new Pi and harness.
  2. DOG\_NAME. During provisioning, this stores the dog’s name in the database that will be assigned the pi.
  3. DOG\_DESCR. This provides a textual description of the dog that will be assigned the new device.
  4. PI\_SERIAL\_NUM. This field is a character format field intended to store the MAC address of the Raspberry Pi to be assigned to the new dog.
* **Updating information for deployed Pi and Dogs.** The previous steps document adding a new Pi and Dog. In the event that an admin needs to update either the associated Dog and Pi devices pairing, update the DOG table with any necessary fields as documented in the previous task.
* **Delete information for a deployed Pi and Dog.**  It is not recommended to delete any historical information from any of the tables, but in the event of troubleshooting for testing, use the appropriate tools to remove/delete information using MQSQLWorkbench. To remove a Dog/Pi, delete from the DOG table. To remove transactions, delete from the DOG\_TRANSACTIONS table.
* **Addition of sensor events.** The database is configured to accept 4 types of sensor events from the Dog/Pi pair. SOUND, BITE1, BITE2, and PING events. The PING event is configured from the Raspberry Pi to send periodically to keep track of dog movements. The SOUND is configured to indicate a positive detection of a human is found. The BITE1 and BITE2 events indicate a dry contact has been activated by the dog with a trained dog using the two contacts to indicate separate detections. The Matrix Creator device that is attached to the Pi has many other sensors that could be configured to indicate different events. If the admin wishes to add another sensor event, insert the following information into the TRANS\_TYPE table.
  1. Integer indicating a unique sensor event, TRANS\_TYPE, next available number is 5.
  2. TRANS\_NAME is a text field indicating the name of the new sensor event.
  3. TRANS\_DESCR is a text field description for more information about the sensor event.

Any new transactions types must be coordinated with the Raspberry Pi admin so that the Python code can interpret the new sensor data and package the transaction string correctly. Any new transaction types must also update the server code to correctly interpret as a properly formatted string and insert correctly into the DOG\_TRANSACTIONS table.

# 

# Rescue Dog Tracker Web Page

### Administration and Page tweaking

* **Adding More Pi/Dogs**
  + Copy Dog\_ID\_1.html in the Directory \capstoone\Dog\_ID
  + Edit it and change line 94 from /test\_data.js to whatever your dog’s data file is
  + On Index.html, add another row and add the new page
    - Make sure to tell it to open in the index.html’s iframe
* **If you need help with Google API features, here is the main page for the API**
  + <https://developers.google.com/maps/documentation/javascript/tutorial>
* **Update the Refresh rate of the page**:
  + Update Line 68 - PAGE\_RefreshRate = 10 is the current setting
  + Note that the refresh is in Secs
* **To change the Map zoom Level**
  + Line 88 - we defaulted it to: zoom: 17
  + Bigger the number, the closer the zoom is
* **Location of files**
  + Mapped drive on campus: \\anc-web01\students\capstone
  + FTP: webftp.uaa.alaska.edu\students\capstone
  + Main Folder is Capstone
    - Data - Location where you store the data.js files
    - Dog\_ID - Location where you have all of the Dog pages at
    - Images - Location of the map markers and webpage tab
    - Archive - Has various incarnations of the data and pages
* **General Troubleshooting**
  + If the map will not render at al, this usually means that the data.js is corrupt

User interface