VICC

Introduction

Virtual Incident Command Center (VICC) aims to provide a system that allows a large-scale event to be monitored. This system includes tools that allow people who are running the event to communicate, coordinate emergency vehicles, monitor the event through cameras, and use social media mining to gather useful information about the event. There are many components to this system including a solution to track the location of an emergency vehicle with a corresponding map, a camera that takes pictures of the event, an Android app that participants can use to coordinate, and a web app that gives a command center view of the event. In order to connect all of these reliably, we use IBM's Bluemix.

Approach and Results

A. IBM Bluemix

IBM Bluemix is a cloud service that allows developers to design applications that are easy to deploy and manage on the cloud. Bluemix offers many services and API's that can be easily integrated into an app. Three primary services that we used are the Internet of Things Foundation (IoTF), List databases, and Node-Red.

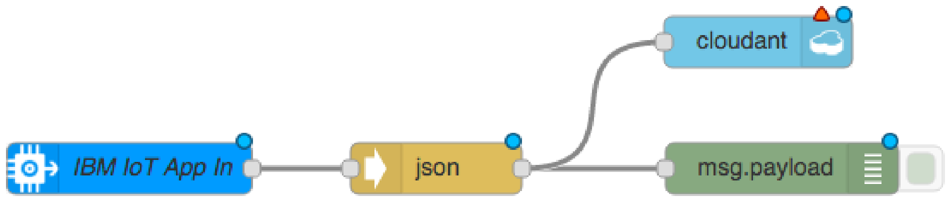
i. Internet of Things Foundation

One of the services that is an essential part of this project is the Internet of Things Foundation (IoTF). This service deploys a MQTT server for your application that allows devices to connect and publish/subscribe to content. Each device that connects to the IoTF must be registered on Bluemix and be assigned an API key. The key, along with a unique identifier that is assigned by the developer (i.e. a member of VICC), are used for authentication for the IoTF's MQTT server.

ii. Node-Red

The centerpiece that ties everything together is a Node.js application that is editable through a Node-Red interface. Node-Red provides a way to plug together various applications, API's, and network interfaces into an easy to use flow. Each entity that processes data is referred to as a "node". Each node has specific parameters that manipulate the data, then the data is passed on to the next node. The data is passed in a JSON format. Figure 1 represents an example flow. -- Go through and label each node.

iii. ClearDB MySQL Database



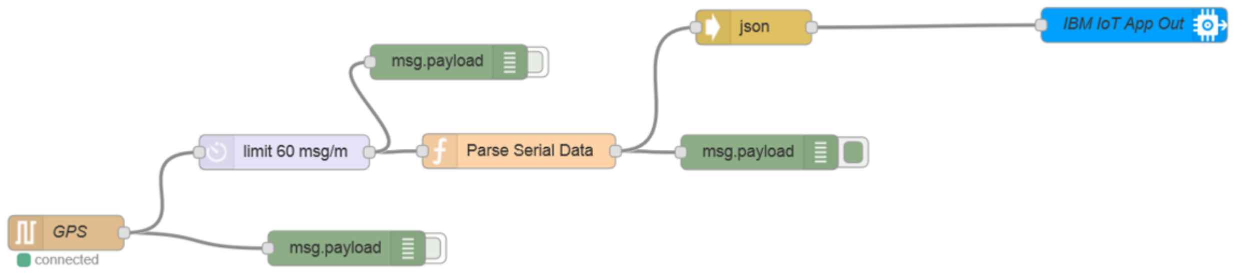
The ClearDB MySQL Database is service that is associated with Node-Red web application and is accessible via a MySQL node within the Node-Red flow. Using Node-Red, queries and can be sent to the database to search for, update, insert, or delete data. HTTP GET requests nodes also handles JSON encoded input from the VICC Android application and send update queries to the MySQL node. Database responses from the node are handled by helper functions and can either be sent to the Map node or output as JSON encoded HTTP responses to the Android application.

B. Raspberry Pi with GPS and Camera

To track the location of vehicles and to have a view of the event, we incorporated a GPS and Camera with the Raspberry Pi 2 Model B. There will be modules that are solely GPS enabled and modules that use the Camera and the GPS, so the pictures taken from the camera can be geo-tagged. The camera is a Raspberry Pi Camera Board and the GPS is Adafruit's Ultimate GPS Breakout Board.

i. GPS Flow

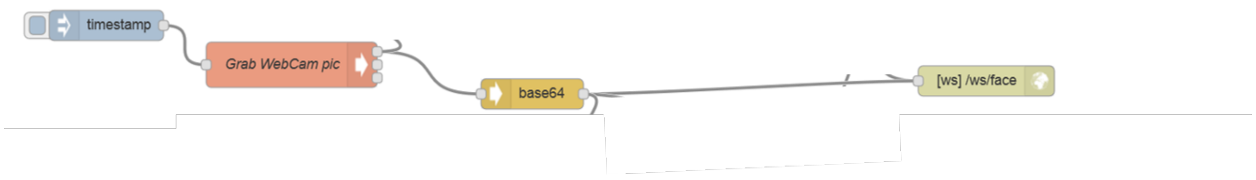
In order to make each unit consistent with IBM's Bluemix, an instance of Node-Red is installed on the Raspberry Pi. Node-Red on the Pi is very similar to a regular instance of Node-Red, accept that it allows for memory-usage control. This is a very attractive feature for a memory constrained device like the Pi. After the instance of Node-Red is started, it can be accessed through a browser on the same network as the Pi, or on the Pi itself. [Please see the Starter-Kit for more information]. The flow that is used to gather data from the GPS is shown in figure x.



The flow takes the GPS data straight from the serial port and then this data is sent to a javascript function. The javascript function parses the data and converts it into a JSON document. The parsing function focuses on the GPGGA parameter in the GPS data stream (the GPS uses the NMEA protocol). The GPGGA parameter offers latitude, longitude, and time. The GPS also provides altitude and information about how many sattelites it is using to fix its position, but to get this information you would need to look at the other NMEA parameters. Once the information is parsed and converted into a JSON document, it is then sent to the proper instance of the IoTF where the data can be used from there.

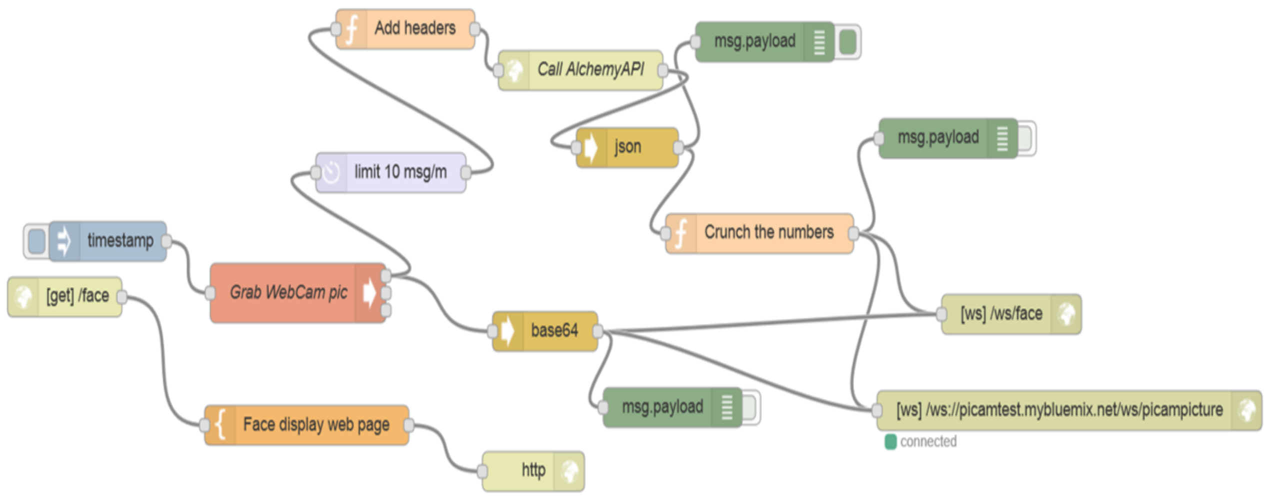
ii. Picture Flow

The Node-red application that has the GPS flow also contains the flow that requests pictures from the PiCam. There are four nodes that take the picture and send it to Bluemix via a websocket. These nodes are shown below in figure x.



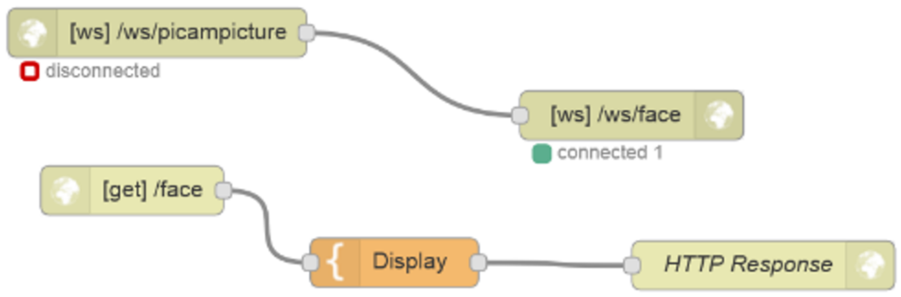
First there is an injection node that triggers a command to take a picture every five seconds. The node it triggers is a system node that uses the command “raspistill -vf -hf -w 640 -h 480 -t 1000 -o” to grab a picture and send it to the next node. The next node encodes the image using base64 and finally sends the node to a websocket on bluemix called “ws://picamtest.mybluemix.net/picampicture”.

There are other components to this flow that are not essential to take a picture, but add information and functionality to the overall system. The first component uses IBM’s Alchemy API to determine if there are faces in a picture, the age range of each person in the picture, and the gender of each person in the picture. Each picture that is taken is sent the Alchemy and the information that is retrieved is put into JSON format and sent to the same websocket the pictures are sent through. The last component of this flow sets up a small webserver on the Raspberry Pi that displays the data retrieved from the camera and the results of sending the picture to the Alchemy API.



C. Displaying Pictures and GPS coordinates

Once the pictures and the GPS coordinates have been retrieved and sent to Bluemix, both need to be displayed. In order to display the GPS coordinates, there is a Node.js application with a Node-Red flow that has a mapping node. This mapping node creates an icon at the location that is represented by the GPS coordinates. A separate Node.js application has a Node-Red flow that displays the pictures on a webpage. Figure x shows the flow that takes an incoming picture and displays it on a webpage. The code inside the “Display” node can be found in the starter kit.

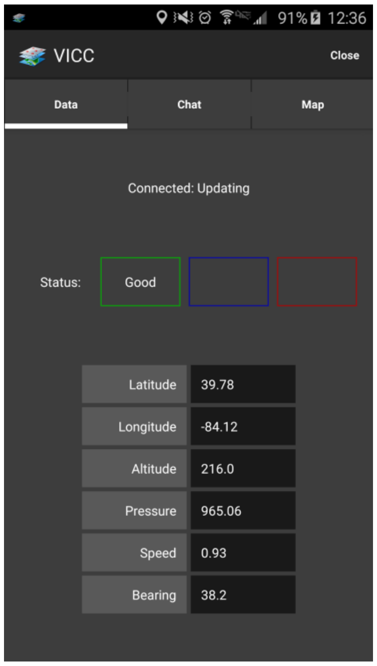


The top left node ("[ws]/ws/picampicture") is an open websocket that takes in the pictures and information from the Alchemy API and transfers it to a new websocket that can be accessed at this address: http://picamtest.mybluemix.net/face.

D. Android Application
For increased usability, a VICC Android application was developed that would push location and sensor data to the web application's database, similar to the Raspberry Pi, as well as allow the user to access the VICC chat room and map to communicate with and view other users. The Android application, named VICC, was developed using Android Studio and contains a single *MainActivity* class, a *DataUpdateService* class, and two helper classes, *JSONParser* and *Variables*.

The *MainActivity* class is an activity class that extends the *FragmentActvitiy* class and implements the *ActionBar TabListener* interface. The *MainActivity* class uses fragments to easily allow the user to switch between three different layouts associated with the activity. The three layouts are the data layout (**Android 1**), chat layout (**Android 2**), and map layout (**Android 3**). The data layout displays the connectivity state of the Android application with the web app, allows the user to select a current status, and shows the real-time sensor data for latitude, longitude, altitude, air pressure, speed, and bearing. The chat layout is simply a webview that points to the VICC Zopim chat URL, the same one used in the iFrame of the VICC Interface. The map layout is also a webview that points to the VICC GPS Map URL and will display all of the same variable data shown on the map application on Bluemix.

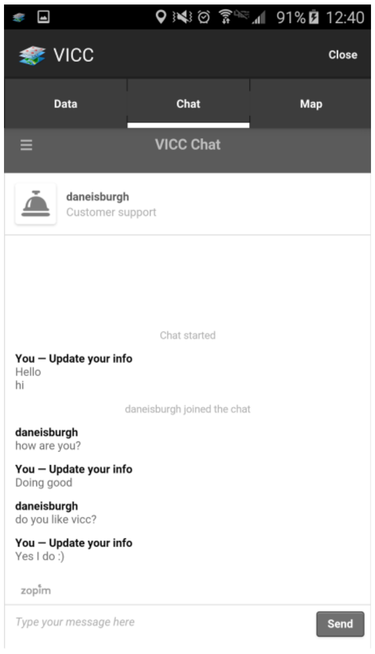
Android 1: Data Layout **Android 2:** Chat Layout **Android 3:** Map Layout



Connected: Updating

Status: Good

Latitude	39.78
Longitude	-84.12
Altitude	216.0
Pressure	965.06
Speed	0.93
Bearing	38.2



daneisburgh
Customer support

Chat started

You — Update your info
Hello
hi

daneisburgh joined the chat

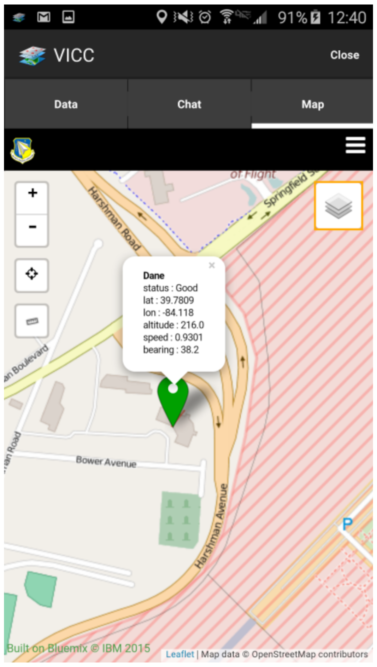
daneisburgh
how are you?

You — Update your info
Doing good

daneisburgh
do you like vicc?

You — Update your info
Yes I do :)

Type your message here

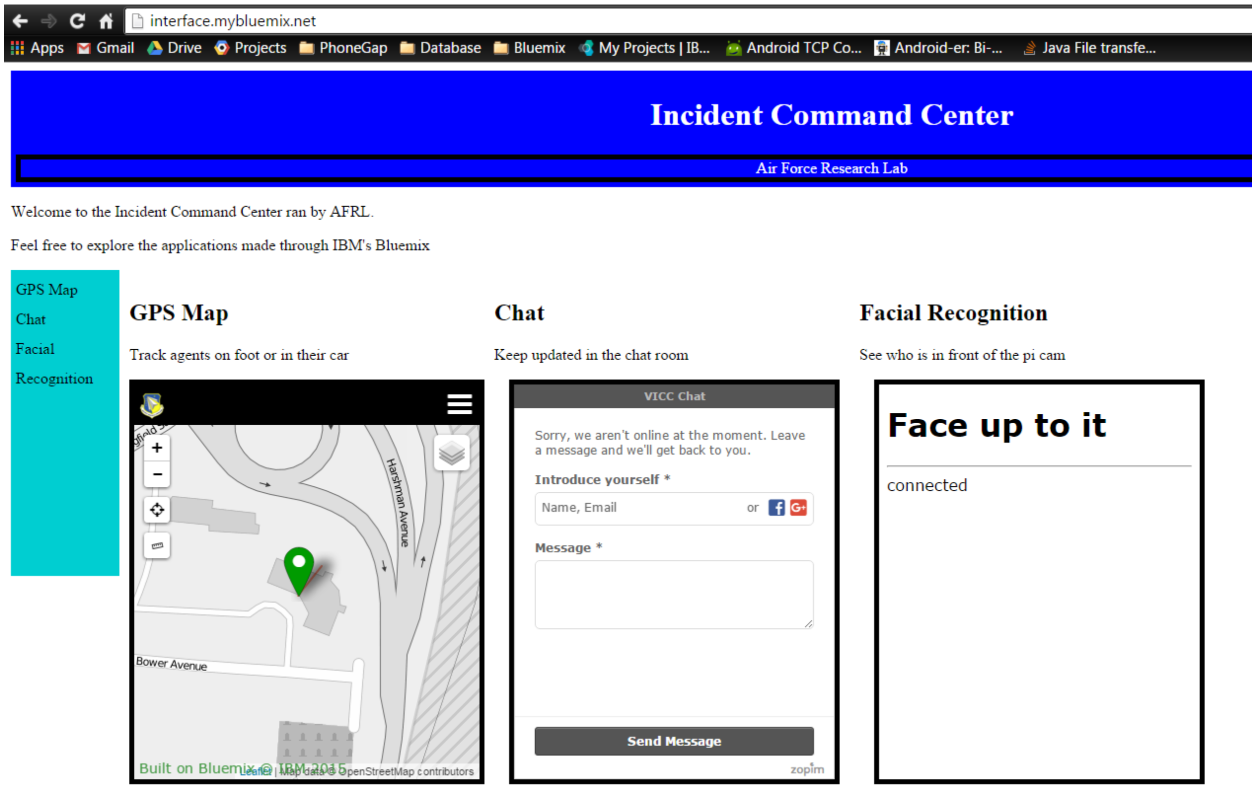


Dane
status : Good
lat : 39.7809
lon : -84.118
altitude : 216.0
speed : 0.9301
bearing : 38.2

The *DataUpdateService* class extends the *Service* class and implements the *LocationListener* and *SensorEventListener* classes. The *DataUpdateService* class is initialized by the start of the *MainActivity* class and, as a service, can run after the user exits the *MainActivity* fragment view but does not completely close the application; this means that the user can open the VICC app and then use other apps while the *DataUpdateService* runs in the background on a separate thread. The *DataUpdateService* class initially accesses the phone's sensors and GPS/Network connectivity and then attempts to connect to the VICC database. If a connection is made and the service is able to access the device's sensor data, the *DataUpdateService* class pushes data to a PHP script via JSON encoded HTTP POST requests. The *DataUpdateService* class first tests both the network and gps connections to make sure it can successfully accesses the web app and push data as well as to retrieve certain location data. First, the network connection is test, if it succeeds it is used to push data as well as retrieve latitude and longitude data BUT NOT speed and bearing data, else if gps succeeds, it is used to push data as well as retrieve latitude, longitude, speed and bearing data. The *DataUpdateService* class also accesses the ambient air pressure sensor data from the phone and uses it to calculate the device's altitude given the standard atmosphere variable constant. All the data is encoded into a JSON object and then, using the *JSONParser* helper class, is pushed to the PHP script's URL on the web application and also retrieves and response message from the PHP script (error or otherwise). The *DataUpdateService* class runs continuously as a service until the VICC application is closed and will attempt to update the VICC database with the latest sensor data every two seconds.

E. Interface

The Interface web application was created to bring together everything that was developed over the semester into a single web application that could easily be displayed for the fall Open House. The Interface web app is hosted on Bluemix and its *index* page (the one the browser initially points to) is simply an HTML page that contains a header, some text information, and three iFrames. The three iFrames display the GPS Map, VICC Chat, and Facial Recognition respectively.


interface.mybluemix.net
Apps Gmail Drive Projects PhoneGap Database Bluemix My Projects | IB... Android TCP Co... Android-er: Bi-... Java File transfe...
Incident Command Center
Air Force Research Lab

Welcome to the Incident Command Center ran by AFRL.

Feel free to explore the applications made through IBM's Bluemix

GPS Map
Chat
Facial Recognition
GPS Map
Track agents on foot or in their car
Chat
Keep updated in the chat room
Facial Recognition
See who is in front of the pi cam
Harshman Avenue
Bower Avenue
Built on Bluemix © IBM 2015
Leaflet | Map data © OpenStreetMap contributors
VICC Chat

Sorry, we aren't online at the moment. Leave a message and we'll get back to you.

Introduce yourself *
Name, Email
or
Message *
Send Message
zopim
Face up to it
connected

Future Work

There are many areas of improvement for VICC including further development on the web and mobile applications. Increased functionality could be added to the map and interface web applications that allowed for user sign in and account creation as well as event planning and management. Increased functionality could also be added to the Android application that allowed users to sign in and/or create an account that then allowed for user disambiguation and would allow user data to be recorded and managed easily by the web app.

The IBM Bluemix easily allows for further development that would possibly make the VICC project a practical tool to incident and event management professionals.