

Software Development

Denver, Colorado

2016

Software Development

Denver, Colorado

2016

2072-1

Table of Contents

Research.....	4
Project Description.....	5-6
Work Log.....	7
Software Documentation.....	8-19
<i>-Project Requirements</i>	8
<i>-High Level Software Design</i>	9-10
<i>-High Level Architecture</i>	11
<i>-Database Diagram</i>	12
<i>-Testing</i>	13-16
<i>-End-User Product Documentation</i>	17-19
Work Evaluation / Future Idea.....	20
References.....	21

Research

Technology in the modern day is constantly evolving and being improved. A fascinating, and relatively new, field of technology is the Internet of Things. The Internet of Things, abbreviated IoT, is when computing devices are embedded in every day objects so that they can communicate and share data to make a “smarter” world. When given the task of creating a program that has some social value we instantly thought of emergency response and, more specifically, dispatch. Even though we have incredible technology today, dispatch still remains the same, a person, sitting behind a computer using different books, trying to juggle the different tasks thrown at them. In the words of Jeff Hewitt, a former dispatch worker, “When we’re short-staffed, there’s nothing to do but stay at your station.” We believe that by implementing IoT into 911 dispatch we could shift the demands of the job from the workers to computers.

A large majority of our research revolved around how we could demonstrate the potential of a vehicle crash response program with relative ease and manipulate different scenarios inside of that system. We tossed around different ideas, such as creating a visual computer program with streets and lights to show where the crash happens and the affect, or to build a real model on a small scale using micro computers, like Arduinos and Raspberry Pi. We looked into each one and found that using the former would let us build a realistic internet of things system that communicates over a real internet connection as opposed to a pre-set program. It was decided that our Raspberry Pi would be the “brains” of our system because, when we looked into the capabilities of the micro computer, it would be able to host a database and we could send the information using the Python programming language.

We researched what it is like working at dispatch, and looked at videos and pictures of dispatch in operation. This studying re-affirmed our opinion that dispatch is behind with their technology, they need a user interface that only takes one screen (as opposed to their current six monitor set ups), and we didn’t want them to flip through books of information any more, we needed a program that could do all that for them and more. Despite being familiar with the java programming language, we found that Microsoft Visual Studio works very well with a MySQL database and is one of the best IDE’s for creating user interfaces, so we decided to use C#. We found a large amount of information on connecting a C# program to a MySQL database on stack overflow and the official visual studio forum and previous experience with using XAML came in useful as well for designing the user interface. After doing all of this research, we began creating our project with the needs of dispatch in mind, using the most applicable technologies for our internet of things system, and the goal of saving lives in the future.

Project Description

With the task of creating software that has some social or educational value we had to think through different modern scenarios that use technology but seemed inefficient or outdated in how they operate. We decided that, as high schoolers, most of what we know involves education, so we wanted to push ourselves to go outside of our everyday scenarios and into society to get a real world development experience. We spent several weeks listing out different societal events that happen daily and have drastic results due to the inability of the technologies involved, eventually we reached the conclusion that car accidents and dispatch response were the perfect candidates for the problem we wanted to tackle.

According to the Association for Safe International Road Travel, 3,287 people die each day due to the result of car accidents. This number shocked us and made us wonder why we have such modern technology yet so many people still die as a result of a car crash. We agreed that the best way to overcome this problem was to make emergency response more efficient, cutting back on the amount of time it takes for emergency response to reach the crash site, saving lives in the long run. Currently, a single dispatch worker handles everything relating to a 911 call, including sending police, firetrucks, and ambulances, trying to keep the client calm, and organizing the response to the best of their ability with the data the caller gives them. We set out to create a program that would give dispatch more information on the crash and create a more automated response to take the educated guessing out of their job and replace it with facts based off of real data, and the best means of doing it was to implementing the internet of things.

The internet of things is a new and growing technology that puts computing technologies in every day objects so that they can interact and communicate data over the internet to make a process more efficient without human interaction. Our project took this idea and put it into a small scale using micro controllers and micro computers to replicate a car crash, demonstrate the magnitude of the capabilities of the internet of things, and provide a platform for future development. When we had this idea we had to write out the complex interaction of the system and separate what would be handled by the program and what had to be handled by the dispatch employee. Once we knew exactly how each iteration fit together we were able to begin assembling our system.

In our model, we used a button press to trigger the example crash sequence, a GPS module is used to get the location of the “accident”, and a custom built phone module to connect the people involved with dispatch automatically. Clearly, our project is based off of trends we are seeing in current car models, because as technology evolves vehicles are becoming more like driving computers than metal frames with engines, so incorporating things such as GPS modules, phone capabilities, force sensors, and speed trackers is not very far behind. When the crash is initiated, all of the readings from the sensors are collected and sent straight to the screen of dispatch and appear in list form for easy accessibility. At the same moment, certified first responders in the area are identified and notified by the program through a vibrating skin patch (we hope to implement a mobile app in a future release). The vehicle’s phone module, when it

receives notification of the accident, instantly and automatically connects the people involved in the crash with 911. Dispatch can take the coordinates that the GPS module sent and tell the emergency response teams exactly where to go, and can see a view from the street camera nearest to the coordinates (in our demonstration this is a video stream from a micro computer). All of the processes described above take only a matter of seconds to play out, shifting the work from a single person to a multitude of information based operations and letting emergency vehicles leave the garage with more information, faster than ever before. With this simplicity, all dispatch has to do is ensure the program is operating correctly and talk to the person on the line.

The benefit of condensing our program into a small scale model is that we can test and implement different scenarios and ideas with ease and use it as a demonstration for what it would look like when used in a real world accident. Our final small scale internet of things system works alongside user interface geared towards dispatch that displays all of the sensor's data and a direct livestream from our model street camera, and is capable of incorporating more sensors and data. This project is an incredible platform that anticipates where car technology is going and takes advantage for internet of things concepts so that it can be incorporated when the time comes. In our next version, we hope to archive car crash data so that, when used in the real world, it could be used for analysis, calculating expected response times, for insurance purposes, and even in court.

Work Log

Date	Task	Time Involved	Team Members
11-13-16	Research / created program plan	1.5 hours	Both team members
11-20-16	Created presentation / Arduino bluetooth communication / prepared IoT demo	2 hours	Both team members
11-29-16	TSA Regionals Runoff Event	6 hours	Both team members
12-10-16	Created MySql server / set up Raspberry Pi hub	3 hours	Both team member
12-21-16	Set up artificial crash sequence	30 minutes	Both team member
1-7-17	Began creating dispatch GUI / set up sending and receiving from C# program	3 hours	Both team member
1-11-17	Continued working on GUI / set up webcam to stream to Raspberry Pi / began documentation	4.5 hours	Both team member
1-22-17	Finished GUI / completed crash sequence demo	1 hour	Both team member
1-23-17	Finished documentation	1 hour	Both team member

Software Development Documentation

Project Requirements

Languages:

1. MySQL
2. C#
3. XAML
4. Python
5. C/C++ (Arduino)

Database:

1. SoftwareDev (Database)
2. CrashData (Table)
3. IDCrashData (Column)
4. LatitudeCrashData (Column)
5. LongitudeCrashData(Column)
6. More columns can be added if more sensors are incorporated

Software:

1. Microsoft Visual Studio
2. Raspbian
3. MySQL Workbench
4. Google Drive
5. Python Libraries (Serial Time MySQL)
6. Creatly
7. Arduino IDE

Materials:

1. Arduino Pro Mini
2. Arduino Uno
3. Adafruit FONA Shield
4. Raspberry Pi Model B Version 3
5. Personal Network Router
6. Xbees
7. Xbees sheild
8. Xbee Break-out board
9. Vibration Motor
10. Computer
11. Power cables for Arduino's and Raspberry Pi

High-level software design

For our project, we chose to avoid the waterfall design technique, where we design, code, test, and then release, and go with increments and iterations. Using this technique allowed us to break our project down into chunks, perfect each small section, test them, and then combine them with the other increments of our program. The iteration part comes after the increment and is used to change what doesn't fit or optimize anything that needs optimization. When designing, we broke our project down into three simple increments and used different iterations of each before combining them to make our program.

In the first increment, we built the brain of the internet of things system, the Raspberry Pi. For this development category we set up the Raspberry Pi, hosted a MySQL server on the Raspberry Pi using Python, connected to the MySQL server from the MySQL workbench on a separate machine, added a button event that triggers the crash sequence, connected a GPS module, built a working phone, and prepared the vibrating skin patches for first responders. We had two iterations of this first increment, in the first one we set up the skeleton of the program and tested to make sure each component (such as the server, GPS, phone, and skin patch) was individually reliable and functional, and in the second we added the usability so that when the button is pressed the Raspberry Pi is notified, the crash information (crash ID, latitude, and longitude) is automatically committed to the MySQL server into a row, a phone call is automatically made to a test number, and local responders are notified through the vibrating skin patch. The completed and detailed information on the series of events can be found in the sequence diagram below.

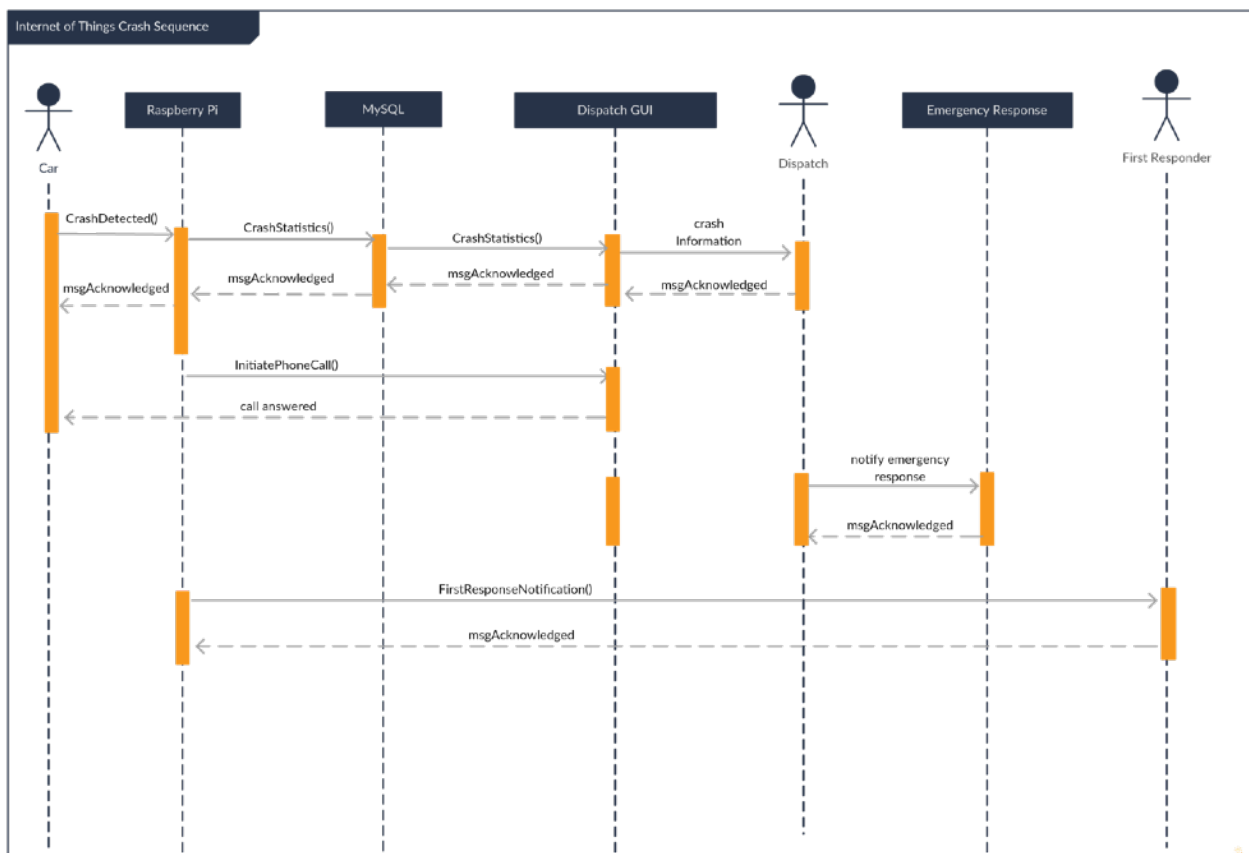
In our second increment, we worked on the C# client program and the GUI for dispatch in XAML. This increment was key because it needed to have a great interface that made dispatch's operations simple, so going in we knew it would take multiple iterations. We created a method that opens a MySQL connection to the server, gets the data from the table, and then closes the connection. After that we used Visual Studio's Windows Presentation Foundation (WPF) to create a window that enters fullscreen automatically, has a web view (for the next increments web based video stream), a button to close the program, a data grid for listing the MySQL information, and six labels for the information from the database. In the first iteration we simply put these items into the WPF application but in the second iteration we figured out the best location for them and arranged them accordingly. Additionally, we discovered that when the database is not running, a MySQL exception is thrown and unhandled so the program can't compile, this was solved in a third iteration by creating a handler (see testing section) for when the server is not running or there is an error in the connection string.

With the third and final increment, we added a video camera to the Raspberry Pi that streams straight to a website. With the web address we were able to use the web view from our previous increment and live stream the footage straight to the WPF application, but with a large delay. We tested different ways of streaming from a Raspberry Pi to a website and found that when it streams in m-jpeg form the streamed footage is so large there is a considerable delay. We

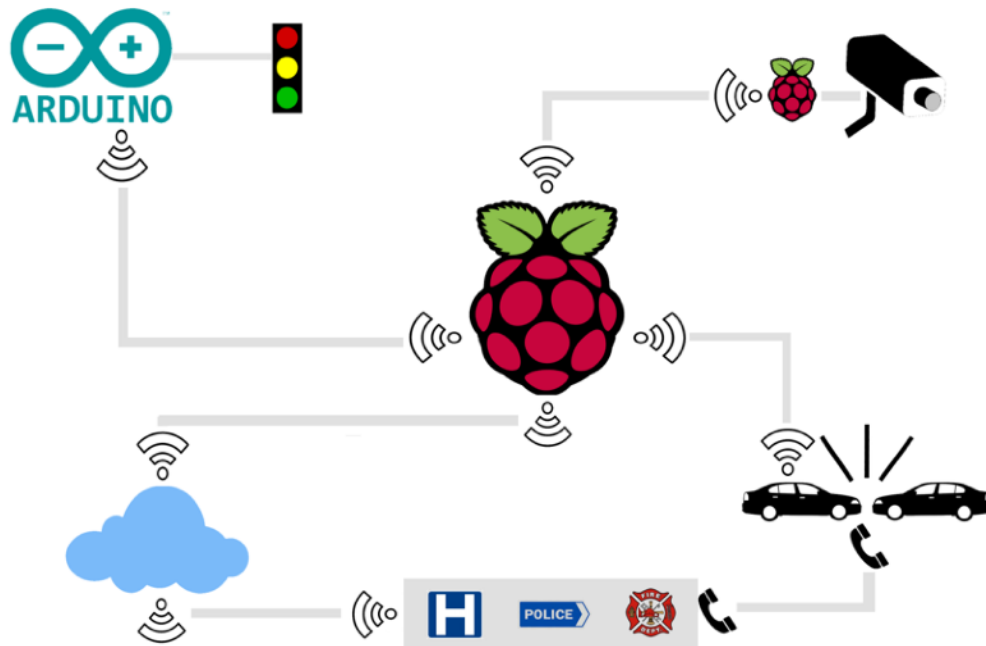
did a second iteration that used a m-jpeg with a smaller resolution and less frames per second to fix the delay problem.

Having completed all three increments and tested each one individually and extensively, we combined them for final testing before completion. Each part worked together quite well and seemed very efficient, the only problem we faced was getting the MySQL information to appear in the correct formatting on the data grid, but this was fixed once we researched how to add different columns and rows automatically based off of the database's columns and rows. We then put the final system through extensive testing to make sure everything was running as expected

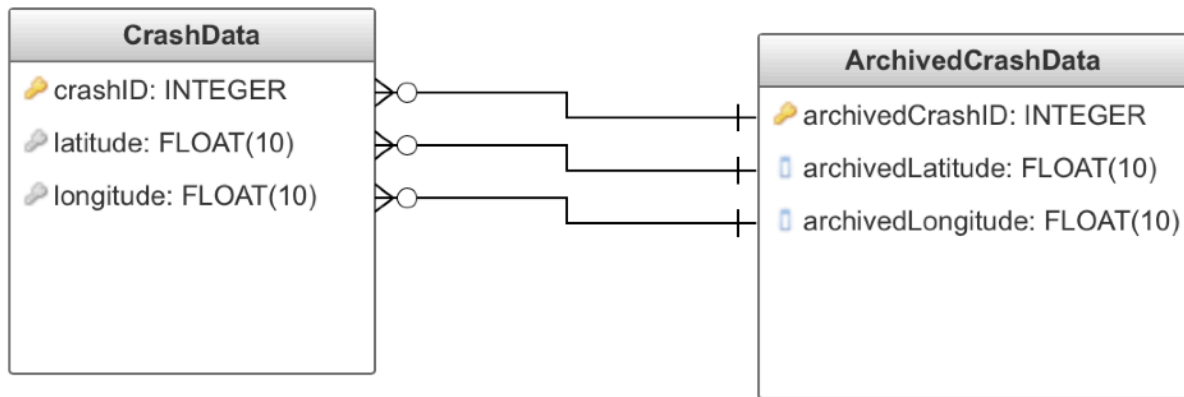
(Below) A flow chart of our program that shows the sequence of events and how it works programmatically



High Level Architecture



Database Diagram



This is our data table from our MySQL server hosted on the Raspberry Pi. It has three columns currently but new columns can easily be added as more sensors are incorporated into the internet of things system. In the future we hope to incorporate an archive table, so that the data can be moved from the current **CrashData** to **ArchivedCrashData** for future analysis.

Testing

In our testing, we found that if there was an error in the connection string, or the server was not running the program would crash and we would get an error that said it had an unhandled exception. Below is the process of how we tested different scenarios for connecting to our database in our PullData() method.

Original Method	<pre>public void PullData() { var table = new DataTable(); using (var da = new MySqlDataAdapter("SELECT * FROM CrashData", "Server= 192.168.2.20; Database=SoftwareDev;UID=root;Password=raspberry")) { da.Fill(table); dataGrid.ItemsSource = table.DefaultView; dataGrid.IsReadOnly = true; dataGrid.CanUserAddRows = false; } }</pre>
Analyze Output	An exception of type 'MySql.Data.MySqlClient.MySqlException' occurred in MySql.Data.dll but was not handled by user code.

Add catch and handler	<pre> public void PullData() { try { var table = new DataTable(); using (var da = new MySqlDataAdapter("SELECT * FROM CrashData", "Server= 192.168.2.20; Database=SoftwareDev;UID=root;Password=raspberry")) { da.Fill(table); dataGrid.ItemsSource = table.DefaultView; dataGrid.IsReadOnly = true; dataGrid.CanUserAddRows = false; } } catch (MySqlException ex) { Console.WriteLine("Caught MySqlException. Exception: "+ ex.Number); switch (ex.Number) { case 0: Console.WriteLine("Can not connect to server. Check address. Exception: " + ex.Number); this.Close(); break; case 1045: Console.WriteLine("Invalid username/password. Exception: " + ex.Number); this.Close(); break; } } catch (Exception e) { Console.WriteLine("Unhandled exception "+e.StackTrace); } } </pre>
Analyze Output	<p>When we put in the wrong address it output: "Can not connect to server. Check address. Exception: 0"</p> <p>When we put in the wrong username/password it output: "Invalid username/password. Exception: 1045"</p> <p>When the server was not running we got an unhandled exception "An exception of type 'MySql.Data.MySqlClient.MySqlException' occurred in MySql.Data.dll but was not handled by user code"</p>

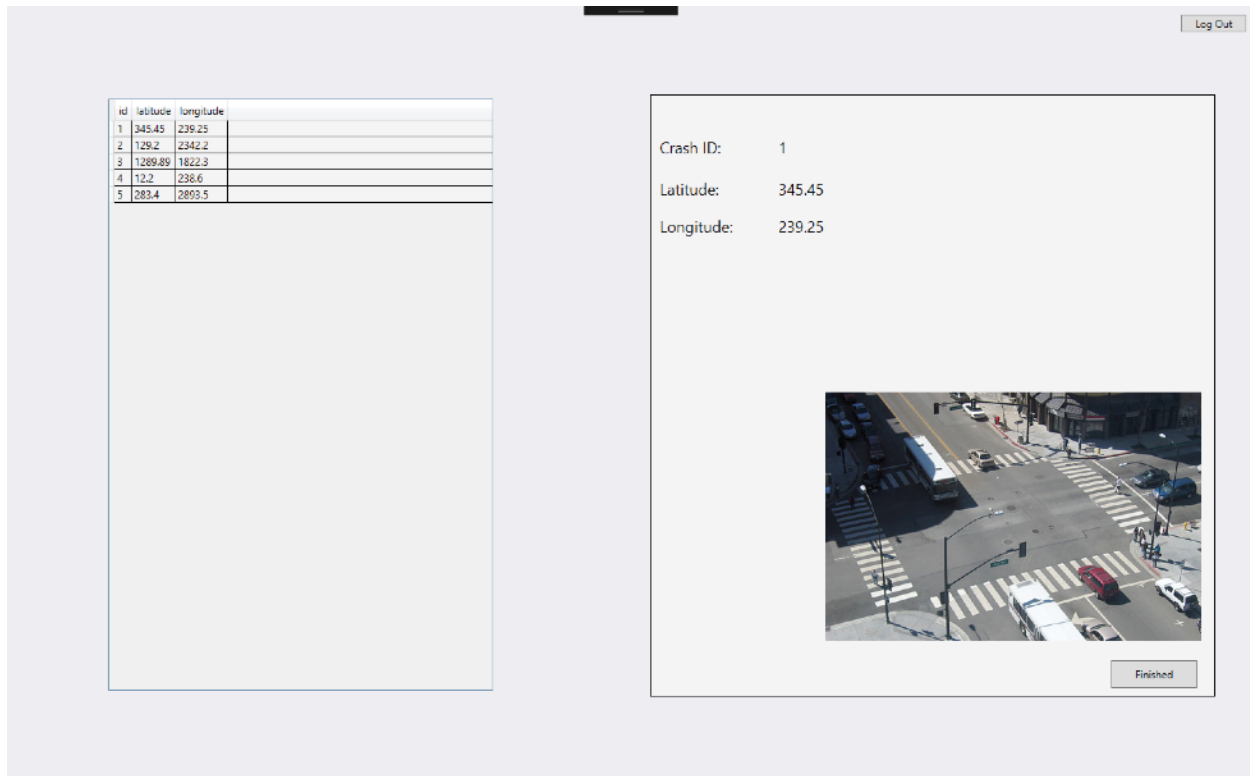
Fix final
unhandled
exception

```
public void PullData()
{
    try
    {
        var table = new DataTable();
        using (var da = new MySqlDataAdapter("SELECT * FROM CrashData",
"Server= 192.168.2.20; Database=SoftwareDev;UID=root;Password=raspberry"))
        {
            da.Fill(table);
            dataGrid.ItemsSource = table.DefaultView;
            dataGrid.IsReadOnly = true;
            dataGrid.CanUserAddRows = false;
        }
    }
    catch (MySqlException ex)
    {
        Console.WriteLine("Caught MySqlException. Exception: "+ ex.Number);
        switch (ex.Number)
        {
            case 0:
                Console.WriteLine("Can not connect to server. Check address.
Exception: " + ex.Number);
                this.Close();
                break;
            case 1045:
                Console.WriteLine("Invalid username/password. Exception: " +
ex.Number);
                this.Close();
                break;
            default:
                Console.WriteLine("Unhandled Exception: "+ex.Number);
                this.Close();
                break;
        }
    } catch (Exception e)
    {
        Console.WriteLine("Unhandled exception "+e.StackTrace);
    }
}
```

Analyze Output	<p>When we put in the wrong address it output: “Can not connect to server. Check address. Exception: 0”</p> <p>When we put in the wrong username/password it output: “Invalid username/password. Exception: 1045”</p> <p>When the server was not running we got an unhandled exception “Unhandled Exception: 1042”</p>
-------------------	---

End-user Product Documentation

Since our program is targeting users who may not have technological backgrounds, we created a set of instructions that describe the layout to potential users. When the program is launched, this is what the user sees right away. We are still perfecting this interface and we hope to make it more professional in time, but for now this gets the data and displays it as promised.



On the left, the user will see a list of all of the accidents that have been automatically recorded and sent to the Raspberry Pi, in order from oldest crash (needing urgent attention) to newest crash (least likely to need instant attention), similar to waiting in line on the phone.

id	latitude	longitude
1	345.45	239.25
2	129.2	2342.2
3	1289.89	1822.3
4	12.2	238.6
5	283.4	2893.5

To look deeper into the data, the user can click on the list item and the detailed view on the right will display all of the relevant information and video stream from the nearest street camera.

id	latitude	longitude
1	345.45	239.25
2	129.2	2342.2
3	1289.89	1822.3
4	12.2	238.6
5	283.4	2893.5


Log Out

Detailed View

Crash ID: 1

Latitude: 345.45

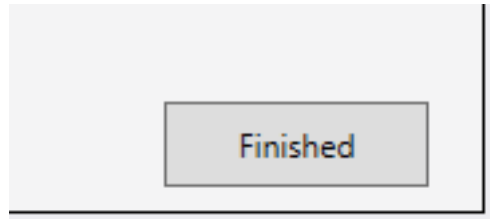
Longitude: 239.25



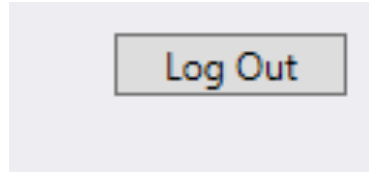
Finished

id	latitude	longitude
1	345.45	239.25
2	129.2	2342.2
3	1289.89	1822.3
4	12.2	238.6
5	283.4	2893.5

At the bottom of the detailed view, the finish button can be used to close the current accident that the user has open, dispose of it from the list (this is when we will archive it in the future), and automatically open the next urgent accident on the list.



When the dispatch session is over, the user can click on the “log out” button to close the program.



Work Evaluation and Future Concepts

Our work is very impressive because it is ahead of the game in the field of technology, using cutting edge ideas to show what the future of technology might look like. The internet of things system we built, even though it is small scale, shows how different devices can communicate to accumulate data and produce real world results. The project is very future oriented so we have tons of ideas for future designs. Firstly, we would like to archive the data collected by the IoT system into our MySQL server for evaluation and comparison, and secondly we want to incorporate a mobile app to notify first responders alongside the vibrating skin patches we made. This project can and will evolve as more sensors are put into cars and road systems, which, in the end, will provide dispatch with even more data than before, and hopefully, one day, run and execute emergency response decisions without the help of a person.

We agreed that data is just data unless it is analyzed and used. Building an archive of previous crashes could let dispatch compare a current crash with a past one to know exactly how long it will take for emergency vehicles to reach the site, for example. Additionally, with the data collected from the system a comparison to the lives saved using the internet of things versus a single dispatch person could be compared and, we believe, yield positive results. Clearly, the next step in the program is to discuss with local authorities how to implement it into our town with the sensors that cars have now and to move it from a small scale test to something larger. While there might be a lot of work ahead, we are confident that our software is not only a good platform for development, but a great demonstration of an internet of things world.

References

- Hong, Dennis, Robert Rosati, M. Peter Storino, Soren Bowie, and Tom Reimann. "5 Terrible Things I Learned Working as a 911 Dispatcher." *Cracked.com*. N.p., n.d. Web. 23 Jan. 2017.
- Mohammadi, Mohsen, Ashkan Nasiripour Amir, Mahmood Fakhri, Ahad Bakhtiari, Samad Azari, Arash Akbarzadeh, Ali Goli, and Mohammad Mahboubi. "The Evaluation of Time Performance in the Emergency Response Center to Provide Pre-Hospital Emergency Services in Kermanshah." *Global Journal of Health Science*. Canadian Center of Science and Education, Jan. 2015. Web. 23 Jan. 2017."Road
- Crash Statistics." Road Crash Statistics. N.p., n.d. Web. 25 Jan. 2017.

STUDENT COPYRIGHT CHECKLIST

(for students to complete and advisors to verify)

- 1) Does your solution to the competitive event integrate any music? YES _____ NO X

If NO, go to question 2.

If YES, is the music copyrighted? YES _____ NO _____

If YES, move to question 1A. If NO, move to question 1B.

1A) Have you asked for author permission to use the music in your solution and included that permission (letter/form) in your documentation? If YES, move to question 2. If NO, ask for permission (OR use royalty free/your own original music) and if permission is granted, include the permission in your documentation.

1B) Is the music royalty free, or did you create the music yourself? If YES, cite the royalty free music OR your original music properly in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any music into his/her competitive event solution.

I, _____ (chapter advisor), have checked my student's solution and confirm that the use of music is done so with proper permission and is cited correctly in the student's documentation.

- 2) Does your solution to the competitive event integrate any graphics? YES _____ NO X

If NO, go to question 3.

If YES, is the graphic copyrighted, registered and/or trademarked? YES _____ NO _____

If YES, move to question 2A. If NO, move to question 2B.

2A) Have you asked for author permission to use the graphic in your solution and included that permission (letter/form) in your documentation? If YES, move to question 3. If NO, ask for permission (OR use royalty free/your own original graphic) and if permission is granted, include the permission in your documentation.

2B) Is the graphic royalty free, or did you create your own graphic? If YES, cite the royalty free graphic OR your own original graphic properly in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any graphics into his/her competitive event solution.

I, _____ (chapter advisor), have checked my student's solution and confirm that the use of graphics is done so with proper permission and is cited correctly in the student's documentation.

- 3) Does your solution to the competitive event use another's thoughts or research? YES X NO _____

If NO, this is the end of the checklist.

If YES, have you properly cited other's thoughts or research in your documentation? If YES, this is the end of the checklist.

If NO, properly cite the thoughts/research of others in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any thoughts/research of others into his/her competitive event solution.

I, Bauer (chapter advisor), have checked my student's solution and confirm that the use of the thoughts/research of others is done so with proper permission and is cited correctly in the student's documentation.



TSA LEAP LEADERSHIP RESUME – TEAM EVENT

The resume must be typed using 11pt Arial or Calibri font. For information about how to complete the resume, visit this link: (<http://www.tsaweb.org/LEAP-competition-engagement>)

TEAM IDENTIFICATION

Team ID: 2072-1

Competitive event: Software Development

Level: High School

LEADERSHIP EXPERIENCES (specific to a competitive event)

- All team members read and comprehended project rubric
- All team members used modern IDEs such as Visual Studio, Arduino IDE, and Python IDE
- Team members met with last year's team to understand project requirements and techniques
- Went to runoffs for event
- Met weekly and sometimes daily to prepare for events
- Participated in software development courses at school
- Read various materials on software development techniques, programming languages, and databases

LEADERSHIP EXPERIENCES (connected to one or more of these categories: *Leadership Roles; Community Service/Volunteer Experiences; Leadership Development/Training; College/Career Planning*)

- Went to multiple TSA meetings for information on events and opportunities (Do)
- Took leadership class offered at the run-off competition (Know)
- Became teacher's assistant in general physics class (Do)
- Followed chapter rules and event guidelines (Be)
- Helped introduce freshman to TSA (Do)
- Helped promote computer science and TSA in school elective fair (Know)
- Listened to industry professionals on their experiences (Know)
- Attended TSA meetings on time to be good upper class role models (Be)

TSA LEAP Program

Be. Know. Do.