Scouting 2016

FRC 4048, Redshift, Westborough MA  
V 1.0, 2017-Feb-02

Table of Contents

* License
* Executive Summary
* White Paper
  + Why We Did This
  + Pros and Cons of This Implementation
* How to Use
  + Dependencies
  + How to Use with Minimal Changes
  + How to Run (collecting data, transferring to laptop)
* Future features

Scouting 2016

**License**

This software is free to run, distribute, modify, and redistribute, so long as the original copyright is given proper credit. Do not commercialize this software. Any modifications to this software are made such that it retains this same license. The author and any contributors are not responsible for any damage to people or devices caused by use, modification, or reuse of this software.

**Executive Summary**

This project is meant to supercharge scouting in FRC teams by getting rid of paper and pen and move on to tablets and computers, so that useful statistics and data-analysis may come out of scouting. The project consists of 6 Android machines (one per scout, per robot, per match) which send scouting forms to a master PC, which compiles and organizes all of the information into a database. This document, besides giving an explanation as to how our team came to create this project, also serves as a guide to port this project with minimal changes. Bear in mind that even minimal changes will require experience with basic SQL syntax, Java, and Android development.

**White Paper**

Why We Did This

Scouting is a crucial part of any FRC team in making good alliance decisions, but it can be difficult to make scouting useful. Our team (Redshift 4048) used paper and pen to scout prior to the creation of this software, which usually resulted in most of the scouting forms going unviewed and ultimately useless. After all, there is not enough time to review hundreds of pages of scouting, and compare that information to the opinions of the drive team, all in the 20 minutes or so prior to alliance selection. Scouting thus became infamous for being the most boring and useless role on the team that practically everyone had to do.

The idea of scouting showed a lot of promise. If all forms could be reviewed and all information compiled into a summary, the information could turn out to be useful. But no matter how we ran scouting in our team, it was either dependent on a few very dedicated souls who reviewed most of the forms in a rush, or simply useless. Once we realized this, we realized that only a computerized system could ever review all of these forms, and compile the data into a summary quickly enough. That moment marked the beginning of the scouting project.

There were many different tentative designs before the official infrastructure was settled upon. All of them revolved around the same basic infrastructure:

Six scouts would cycle through each match, each scout would handle a single robot per match, thus scouting all 6 robots in every match. Six data-collecting machines would cycle through the scouts so that every match, one machine collected information on one robot. At some point, the information contained locally in the data-collecting machines would be pooled into a single master pc, which would summarize the data in some useful way for the drive team.

This basic infrastructure was retained throughout the design process and became the basis for the final design. Along the way, many different suggestions came forth regarding implementation details, such as using Windows tablets for data collectors, a USB stick to pool scouting info to a master pc, and creating our own data visualization application in C#. The designs quickly became over-complicated and unfeasible, mostly due to the three major limitations on this project: Wi-Fi is not allowed at competitions, wired Ethernet is messy, and we didn’t want to use individual’s cell phone data bandwidth.

Eventually we settled on our final design: 6 Android machines have a data collector app that periodically sends files via Bluetooth to a master Windows laptop. The Windows laptop has a java application that breaks down the contents of the tablet files and stores the information in a MySQL database. Come alliance selections, the Windows laptop uses third party visualization software Tableau to summarize the contents of the MySQL database.

This was found to be the most cost and time-efficient design, as opposed to cycling a USB stick, buying Windows machines for data-collection, or creating our own data visualization software. Unlike Wi-Fi, Bluetooth is allowed in competition venues and tablets have decent battery life. Only the Windows laptop would need to some attention to make sure it has power for the entire day.

Scouts have found the use of a touchscreen much more user-friendly than paper-and-pen, and the drive team has found that the alliance suggestions based on the scouting data have been much more helpful and in agreement with the drive team’s opinions this year.

Pros and Cons of This Implementation

The use of Bluetooth can be sketchy, which does require a few backup procedures and workarounds to ensure no data is lost. For one, Bluetooth is not guaranteed every time to successfully transfer a file (at least not in the environment of this project), which led us to create a “hidden button” in the form in order to resend a file. However, as previously stated, Bluetooth does not require a physical drive to cycle through each data collector and master pc; instead, a Bluetooth connection to the master pc is cycled through each tablet.

A few drawbacks of using Fire tablets for data collection include the processing power and screen size of the tablet. Some users have reported that the screen size is slightly uncomfortable for people who have bigger hands. Although not necessarily a drawback, the amount of processing power in the tablet does not allow for any sort of heavy data analytics on the data collectors that could be possible with more capable laptops. However, Amazon Fires are certainly the best balance of cost and reliability, at about $50 each and armed with quad-core processors.

The use of Tableau software makes our software reliant on a free license made available to FIRST teams only, which is never guaranteed to be available the next year. Besides, a couple of join statements in a sql query are enough to cause Tableau to require heavy-lifting from the computer in order to produce graphs, which means the project is reliant on a powerful master pc. However, Tableau is wonderful at presenting results and not very taxing on the hardware as far as data analytics software goes. Plus, the use of Tableau saved us a lot of time as we did not need to make our own data analysis application.

Java is likely not the most conventional choice for a bridge application (used to transfer the forms from the tablets into the database), as this is more commonly the role of a python or ruby script. Plus, the java app is largely unfinished, with a limited user interface. But at least it provides the minimum functionality required to run the full project successfully, and java’s WatchService included in the java api made it easier to write this app in java.

There are not many drawbacks from using MySQL with this project. SQL Server or Oracle Database Express may be more widely used in the corporate world, but MySQL is very popular, free, and open-source, with user-friendly additional programs that allow users to learn how to use and maintain MySQL Server quickly and easily.

**How to Use**

Dependencies

Part of this project was meant to be flexible enough to adapt through different FIRST challenges. However, most of the data-collection application was specifically made for 2016 FIRST STRONGHOLD. That said, full implementation of this project will require modifications to the data-collection app (android/java), minor modifications to the java app, and modification of the contents of the MySQL database. No changes to the java app or database schema (structure) are required to implement this project. However, use of the database will require proficiency with (at least) universal SQL syntax.

This project requires (without modifications) the use of:

* 6 Android tablets (at least 1000x640 screen resolution) for data collection.
* Laptop running Windows 7 or later (Windows 10 is recommended).
* Laptop with at least an Intel Core i3 processor (at least an i5 is recommended).
* Laptop with at least 4GB of RAM.
* Laptop with any version of Tableau 9.0 or later.
* Laptop with Bluetooth 4.0 adapter (built-in or external).
* Laptop running Java 7.0 or later (8.0 is recommended).
* Laptop running any version of MySQL server 5.0 or later. At least one user.
* Laptop with the latest version of MySQL JDBC Connector (in the same folder as the java app).
* Laptop with a version of Eclipse Luna or later.
* It is recommended that this laptop also has a version of MySQL Workbench 6.0 or later, or some version of MySQL Utilities Console, for fast querying.

The data collectors MUST be Android machines if minimal changes are to be made to this project. Due to variations in resolution and pixel density, the absolute layout of the components of the scouting form are not guaranteed to be perfect of consistent across different models. Use of the java app and MySQL database may work on a Mac/Linux machine, though the project has never been tested in these environments. Do so at your own risk. The master laptop must be powerful to some degree in order to run Tableau successfully.

How to Use with Minimal Changes

In order to use this project with minimal changes to the source code, there are a few required changes that must be made prior to compiling and running:

* The STRONGHOLD-specific components of the scouting form must be replaced with relevant components.
* The list of scout names must be replaced with scout names from your respective team.
* The paired device to search for via Bluetooth must be set to the name of the master PC to be used (tablet app).
* The folder to monitor for changes must be changed to the desired folder in the master PC (java app).
* The connection to the database from the java app must be changed to the appropriate connection, username and password.

After completing these items, and after meeting all of the requirements / recommendations previously stated, it is recommended that the following steps are taken in the order they are given.

We will begin by setting up the database. These steps will require proficiency with basic SQL syntax.

The database has four tables in total: datatype, item, record, report. The datatype table contains all of the datatypes that will be represented by the database. A datatype has merely an id and a name.

The item table contains all of the items of the form; an item is defined as a piece/question on the form that requires a singular piece of information from the scout. This means a team name, a checkbox for shooting, and a group of radio buttons for crossing defenses are all examples of a form item. Think of an item as a class definition, not the instance of the class. An item has an id, a string name, a foreign key to the database table, and a boolean indicating whether the item is active or not.

The report table keeps all of the forms in the database. Note however that forms are not stored whole in the database: its items are separated and stored in the record table individually (meant to store instances of an item pertaining to a form). A report has an id, a match number, a tablet number, a team number, a scout name, and a form type. Forms can either be of the present type or not present type. A record has an id, a value, a foreign key to an item, and a foreign key to a report. Though the value field in a record is stored as a string in the database, the foreign key to an item - which leads to an item that has a foreign key to a datatype – tells us the true datatype of the value of said record.

1. Import the schema to the database. This can be done in two ways: by importing Scouting.mwb into MySQL Workbench and forward-engineering the schema script, or by running the sql in Scouting.sql.
2. Plan out the items your team will collect using the scouting forms. No need to think of what they will look like or how they will be implemented yet.
3. Create items (rows in the items table) on the database that will represent the physical form items.

Next, we will set up the java app.

1. Open up Eclipse and create a new project for the java app. Add the java file FileSystemWatcher.java to the project.
2. Add the MySQL Connector to the project as an external jar. Do not simply copy the connector into the project folder.

The java app will be run from the IDE, to be safe in case anything happens. Now, to modify the java app,

1. Go to line 86 in the java app. Change the parameter of *Paths.get* to a path pointing to the folder where tablet forms will be saved.
2. Go to line 126 in the java app. Change the parameter of *Paths.get* to the same folder path + *filename*.
3. Go to line 308 in the java app. Change the parameters of *DriveManager.getConnection* to “jdbc:mysql://[pc name]:[port number]/[schema]?useSSL=false”, followed by the username and password to the database that the program is meant to use.

Finally, we will set up the android app.

1. Design the look and feel of the match scouting form. Keep it simple.
2. Add the folder Scouting0.8 to your AndroidStudioProjects folder.
3. Modify the visual form (just the xml so far) in android studio to match your form ideas. No code yet.
4. In the MainActivity file, change the contents of the inner class ItemIDs (line 63) to reflect the item IDs in the database.
5. Change the form component variables (starting line 105) to reflect the items the form has visually.
6. Change the *TABLET\_NUMBER* variable (line 212).
7. Change the *filename* variable (line 214).
8. Change the variables staring at line 219 to reflect the contents of the form.
9. Change the “secret” in line 267. Be sure to find and replace all references to *lblTeleop* in the code (this is the backup in case the Bluetooth transfer fails. Press this form item 7 times to reset the file to be sent),
10. In line 1367, change the parameter of *device.getName().equals* to the name of the master PC.
11. In order for the code to match your form, you will need to modify the methods *notPresentForm*, *prepareForm*, *resetForm*, *initiateForm*, and *initiateDialog*. Try to maintain the functionality of components that do not pertain specifically to the form, such as *btnSave* or *btnTransfer*.

Now, for the finishing touches:

1. Make sure to pair every tablet to the master PC before use.
2. Make sure to pre-can queries for Tableau so that you don’t have to write those at the venue.

At a venue, make sure that the master PC is waiting for a Bluetooth connection before attempting to transfer tablet forms. If the java app breaks during the venue, it is likely that by counting the number of outputs the java app has displayed on screen (before breaking) the specific form on the file that caused the problem may be located. The simplest solution sometimes is to remove that from the file and save it elsewhere, so that the project may continue to run while a solution to the problem is formulated. It is suggested to pre-can sql queries for use with Tableau and for general use. Some examples of last year’s queries (pertaining to STRONGHOLD) are included in queries.sql.

These steps should ensure that the project runs correctly. However, as our team has never attempted to port this project ourselves, these steps do not guarantee that the project will be working correctly or safely. Remember that the author does not take responsibility for any damage caused by this project.

**Future Features**

There is clearly much room for improvement and additions to this project. That is exactly what will be happening this year (2017). There are improvements which call for a bit more urgency, such as

* Saving forms locally on tablets as opposed to leaving them on RAM.
* Making the java app more robust in general so it does not break occasionally.
* Allowing for Bluetooth transfer without requiring tablets to be previously paired with the PC.

Then there are improvements which will likely come after the ones above, including but not limited to:

* A dual-master-PC system in which two master PCs exist: one for the stands and one for the drive team for quick access to the database.
* A clear distinction between game-specific form items and flexible form items (such as save and transfer).
* A more responsive interface for the java app (that does not simply break when there is an exception, and that indicates more things other than reading a file).
* A greater emphasis on pre-scouting (a second form with possibly a java app meant just for reading pre-scouting forms).
* More general, pre-canned queries that are more easily portable to other years (such as an OPR, a luck rating, etc.).