|  |
| --- |
| FRC #503 |
| 2017 Frog Force Control Systems |
| White Paper |

|  |
| --- |
| Author: FRC 503 – Frog Force  3/6/2017  Version: 1.0 |

**Contents**

[1 Introduction 4](#_Toc477876835)

[2 Motion Profiling 4](#_Toc477876836)

[2.1 Introduction 4](#_Toc477876837)

[2.2 Generating Motion Profiles 5](#_Toc477876838)

[2.2.1 Introduction 5](#_Toc477876839)

[2.2.2 Generating a smooth path 5](#_Toc477876840)

[2.2.3 Generating Velocities 6](#_Toc477876841)

[2.2.4 Generate Positional values 6](#_Toc477876842)

[2.2.5 Merge positional and velocity values into a runnable profile 7](#_Toc477876843)

[2.3 Motion Profile Execution 7](#_Toc477876844)

[2.3.1 Introduction 7](#_Toc477876845)

[2.3.2 Creating a Motion Profiling Thread 7](#_Toc477876846)

[2.3.3 Motion Profile Execution Control Loop 8](#_Toc477876847)

[3 Vision Processing 9](#_Toc477876848)

[3.1 Image Filtering 9](#_Toc477876849)

[3.2 Math on the Finished Image 10](#_Toc477876850)

[3.3 Using a Raspberry Pi for Vision 11](#_Toc477876851)

[3.4 Speed of Processing on a Pi 11](#_Toc477876852)

[3.5 Turret Alignment toward goal 12](#_Toc477876853)

[3.6 Gear alignment toward peg 12](#_Toc477876854)

[4 Scouting System 13](#_Toc477876855)

[4.1 Scouting Tablets 13](#_Toc477876856)

[4.2 The Collector 16](#_Toc477876857)

[4.2.1 Data transfer 16](#_Toc477876858)

[4.2.2 FIRST Match Data 16](#_Toc477876859)

[4.2.3 MySQL Data Storage 17](#_Toc477876860)

[4.3 Match Strategy 17](#_Toc477876861)

[5 The Orange Alliance.org 17](#_Toc477876862)

[6 Frog Force Website 21](#_Toc477876863)

[Appendices 25](#_Toc477876864)

[Appendix A – New Technology Introduced this season 25](#_Toc477876865)

# Introduction

This year, Team 503 greatly increased the goals for the programming team. Not only did we need to write the software to drive our competition robot, but we also needed to develop a website for FTC teams in Michigan (TheOrangeAlliance.com), a new scouting application using tablet computers, and a new team website!

To accomplish these goals, we needed to learn a ton of new software and programming techniques to develop our applications. This was made especially challenging as we have a large percentage of new freshman students that had never written software before!

We also wanted to greatly expand our robot capabilities. These included Motion Profiling for precisely driving our robot in AUTON. Multiple Cameras to help place our gears and align our shooting turret to the goal. Raspberry Pi’s to offload our image processing from the RoboRio to separate processors. Ultrasonic and infrared sensors to accurately measure distance.

# Motion Profiling

## Introduction

Frog Force uses an advanced control system to control the drivetrain of the robot. Motion profiling defines a set of position and velocity set points to target over certain time intervals. This is implemented by having a two dimensional double array with position in encoder counts, velocity in feet-per-second and time in seconds that is fed into the Talon motor controller. The motor controller then executes these points with the assistance of a PID controller. The motor controller can only hold about 128 points at any instant. As Frog Force uses a 50ms time interval, the robot must keep filling the buffer with more and more points as it executes them as fast as it can. By generating these points in the right way, Frog Force is able to simply input a positional coordinate point for the robot to travel to and the robot is able to generate the correct motion profile points to achieve those points.

**FROG FORCE MOTION PROFILING METHOD (Fig. 1)**

## Generating Motion Profiles

### Introduction

The motion profile generation algorithm uses coordinate points, total time, and a Boolean to reverse the direction of the robot as its inputs. It outputs two motion profiles, one for the left side of the drivetrain and one for the right side of the drivetrain. To do this the robot must first generate a smooth path to drive along, then generate the necessary velocities to drive along that path, then generate positions to target from those velocities.

**FROG FORCE MOTION PROFILE GENERATION METHOD (Fig. 2)**

### Generating a smooth path

The algorithm first generates a smooth path based off the coordinate points fed into it. First the algorithm breaks the path up into coordinate points made up only of direction changing nodes and the starting and ending point. This means that if the input points were [(0,0),(4,0),(8,0)], the (4,0) would be discounted as it is not a direction changing node. **Note: our coordinate system assumes that the x coordinate goes up as the robot travels forward and the y goes up as the robot moves laterally.** The algorithm then calculates the number of points that it must inject into the path for the robot to meet the time constraint. This is only necessary for driving along non straight line paths as the robot needs to determine properties of the curve it must drive along. The algorithm then iteratively calculates the points that it must inject to smooth the path using gradient descent and runs in BigO: Nx , where x is the number of loops required to bring the error below the tolerance. Once this is finished the algorithm uses the wheel base of the robot to translate the path into two different paths for the two different sides of the robot.

**FROG FORCE SMOOTH PATH GENERATION METHOD (Fig. 3)**

### Generating Velocities

The algorithm now must take the smooth positional coordinate point path it has generated and generate the required velocities that will fulfill this path. Using a simple derivative function, an original velocity array is generated and in technicality this array could be used on its own, however the robot’s motions will be extremely jerky and therefor inaccurate. The algorithm goes on to smoothen these velocities using gradient descent and then it fixes small errors in these velocities.

**FROG FORCE VELOCITY GENERATION METHOD (Fig. 4)**

### Generate Positional values

Next the algorithm must generate positional values from the velocity values previously calculated. Using integration is a fairly accurate way of accomplishing this. The algorithm converts feet-per- second to encoder counts-per-time interval using basic dimensional analysis. To integrate this velocity array into a positional array, the algorithm iterates over the loop and sets the value of the position array at each index to the value of the sum of all values until the index of the value it is setting. For example, the first value of the positional array is simply the same as the first value in the velocity array however the second value in the positional array is the sum of the first and second value in the velocity array and so on.

**FROG FORCE POSITIONAL ARRAY GENERATION METHOD (Fig. 5)**

### Merge positional and velocity values into a runnable profile

Finally, the algorithm has all of the pieces that the robot needs to run along a path, now it must merge the velocities and positions which are stored in double arrays into a two dimensional double array with position velocity and time interval. This is done by simply iterating over the arrays and pushing the values into the two dimensional array the algorithm returns. The motion profile generation is now complete.

## Motion Profile Execution

### Introduction

Once the robot has generated the motion profiles needed to drive the robot, the next step is to run those motion profiles. To do this in the most efficient way possible, the robot creates a separate thread to run the motion profiling algorithm in. This is done as the robot constantly needs to refill the buffer in the TALON with more points as they are executed since it can only hold 128 points at any given time.

**FROG FORCE MOTION PROFILE EXECUTION METHOD (Fig. 6)**

### Creating a Motion Profiling Thread

In order to increase efficiency and speed up processes, Frog Force uses separate threads throughout the project to multitask. Motion profiling is a fairly intensive process as it involves constantly communicating with and pushing data to the TALONs over the CAN bus. The methods inside the drivetrain subsystem (which the thread calls upon) are all protected and synchronized to avoid any issues. The thread is started from within the drivetrain subsystem and calls upon two separate motion profiling objects, one for the left side and one for the right side.(**Fig. 7).**

**FROG FORCE MOTION PROFILING: THREAD BASED CALL HIERARCHY (Fig. 7)**

### Motion Profile Execution Control Loop

The control loop within the motion profiling object is the lowest level code Frog Force writes to execute the motion profiles. First, the loop checks the status of the TALON so that it can make conclusions from that information later in the loop. If the loop has timed out, meaning something isn’t quite right (the TALON is in the wrong mode), then the loop will stop wasting power trying to do the impossible and will simply give up and throw an error. If more points remain in the profile and TALON has space, the algorithm will fill the TALON with those points. (**Fig. 8**) Motion profile set values are either enable disable, or hold. The control loop is called repeatedly form within the motion profiling thread. This completes the Frog Force motion profiling drivetrain control method.

**FROG FORCE MOTION PROFILE EXECUTION CONTROL LOOP (Fig. 8)**

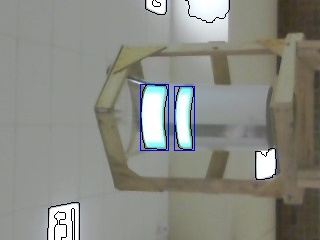
# Vision Processing

Using a Logitech C270 USB webcam to dynamically identify targets using the GRIP/OpenCV vision processing algorithm on a Raspberry Pi 3 off-board processor. GRIP overlays the target image with filters to isolate and identify the target and is exported as a java file to be integrated with other code.

## Image Filtering

A graphical interface for OpenCV called GRIP is used on an external laptop to create and edit filters for the image taken by the camera. The target image (fig. 9) is overlaid with both RGB (light color) filters and HSL (light intensity) filters. These filters help to isolate the reflective tape of the target in the image. The image is then put through a find contours filter that highlights the edges of the reflective targets. After the filters are identifying the target, the process is adjusted for other possible distances or conditions to create the most reliable overall filter. The final filter is exported as a file written in Java that Frog Force used as the base for the program that went on the Raspberry Pi. Inside the output program that GRIP generated, Frog Force wrote additional OpenCV code to take the contours generated by GRIP and turn them into bounding rectangles that enclose the target. The bounding rectangles are then filtered by location in the image in relation to each other and by width and length to remove any extraneous contours that may have been detected by accident. At the end of this filtering process, the only area identified on the image is the intended target (fig. 10).

Fig. 9 Fig. 10

Original Image. Highlighted blue area is the final area identified as the target.

## Math on the Finished Image

The post process image is then used to calculate the offset from the center of the image used for turret targeting, as well as the angles and distances to the target (fig. 11). The center of the identified target in pixels is taken and subtracted from the image width in pixels to find the offset in pixels that the target is from the center of the camera’s field of view. This pixel value can then be translated to an exact angle degree using a constant ratio that is calculated for the camera (real world degrees per pixel). The angle offset is then uploaded to WPILIB network tables to allow the robot to access the value.

The distance math is entirely based on triangle trigonometry, finding the hypotenuse to the target. To properly calculate this value the constants for width of the image in pixels, the FOV of the camera in degrees, and a value “k” that is used to tune the calculation. The other value needed, the width of the target in pixels (apparent width) is calculated using the bounding rectangle’s width in pixels. The output of this math is a distance away (in feet) from the target that the camera is located which is then uploaded to the same network table that the offset value is in.

Fig. 11

FOV = 60°

w

α

d

l

Target

## Using a Raspberry Pi for Vision

Because a Raspberry Pi is an ARM Linux processor, default external libraries used for Windows, Mac or Linux will not work and Frog Force had to compile special network tables and OpenCV libraries on the Pi itself specifically for the ARM architecture. Once these libraries are compiled, the .jar files can be transferred to any other development environment to compile code specifically for the Pi. The entire process code is then compiled with the ARM libraries and can be transferred to the Pi using FTP over wireless or an Ethernet connection. Once the compiled code is on the Raspberry Pi, a shell script was created that runs the java file using the specific libraries compiled for the Pi. To enable the vision processing code to run on startup of the Pi, a system daemon was edited to execute the shell start script every time the Raspberry Pi is powered on. To combat possible loss of power or connection issues, a heartbeat value was added that incrementally counts by 0.001 if the code is running. This allows the robot to check if the Pi is running properly or not. Similarly, a Boolean “discard image” value is sent from the robot to inhibit the vision processing while the robot is moving, preventing junk values due to motion blur.

## Speed of Processing on a Pi

Speed on a Raspberry Pi is a major concern due to the 1.2 GHz processor. This poses a serious risk when attempting to do vision targeting as by the time a value is transmitted, the robot could have moved or the turret could overshoot the desired angle. To help ensure the program runs at the max speed possible Frog Force lowered the image size to 320x240 reducing the number of pixels that must be searched through when finding contours. Also, disabling the graphical UI and other unnecessary system processes leaves the java program as the only other non-system process running, drastically speeding up the execution time.

## Turret Alignment toward goal

Identify what is in scope, out-of-scope and any unresolved Turret moves using one motor and has 2 limit switches. The limit switches are important so the turret doesn't move past the switches and have its wires pulled. The turret runs on its own separate thread. A camera that's mounted on the turret is used to give the turret the angle it needs to move to align with the goal. The turret allows us to shoot without having to align the chassis with the goal. This gives us more time to shoot while the robot is doing other things such as intake or placing a gear. In AUTON, the turret will always run parallel to placing the gear and dumping the balls.

## Gear alignment toward peg

We have three possible autonomous programs in our AUTON selector, each for a specific starting position. For the center peg we are just using a motion profile to align with the peg and a Gear Placer Command to place the gear. For the left and right peg, we also start by using the motion profile to drive towards the peg. Then the camera is used to align the robot parallel to peg and the Gear Placer Command places the gear on the peg. There is also a Drive Endgame that uses the ultrasonic sensors to fix the alignment. One the distances that each ultrasonic returns is equal, then we will know that the robot is parallel with the peg.

Just incase the camera or motion profile is not able to align with the peg, we use a NavX Gyro and the two ultrasonic sensors as a Plan B. We first had to tune the Gyro PIDs in order to make the turns more accurate. The Gyro P, I, and D are constants in the Robot Hardware of the Competition Bot. There is also a gyro tolerance so that there is a range of error for the gyro turn, this is also a constant. Using the distance from the ultrasonic sensors, a program with a mathematical formula is able to calculate the angle the gyro needs to turn. The formula is the tan inverse of the left ultrasonic distance minus the right ultrasonic distance divided by the distance between the sensors. The Gyro Command then uses this angle to tell the NavX Gyro to turn in order to be aligned with the gear.

We drew up a plan for three AUTONS for the respective sides that you could start the robot on. They are called Left Peg AUTON, Right Peg AUTON, and Center Peg AUTON. All the AUTONS begin with a Motion Profile leading them to the peg. After that, there is a method called the Drive End Game. The Drive End Game uses the camera to “steer” the robot toward the peg. The program calculates the angle toward the goal. We then divide the angle by how much distance is left to travel using the Ultrasonic sensors.

For the Left Peg Auton, the team developed a special process to dump the bin and shoot as much fuel as possible based on how much time is in AUTON.

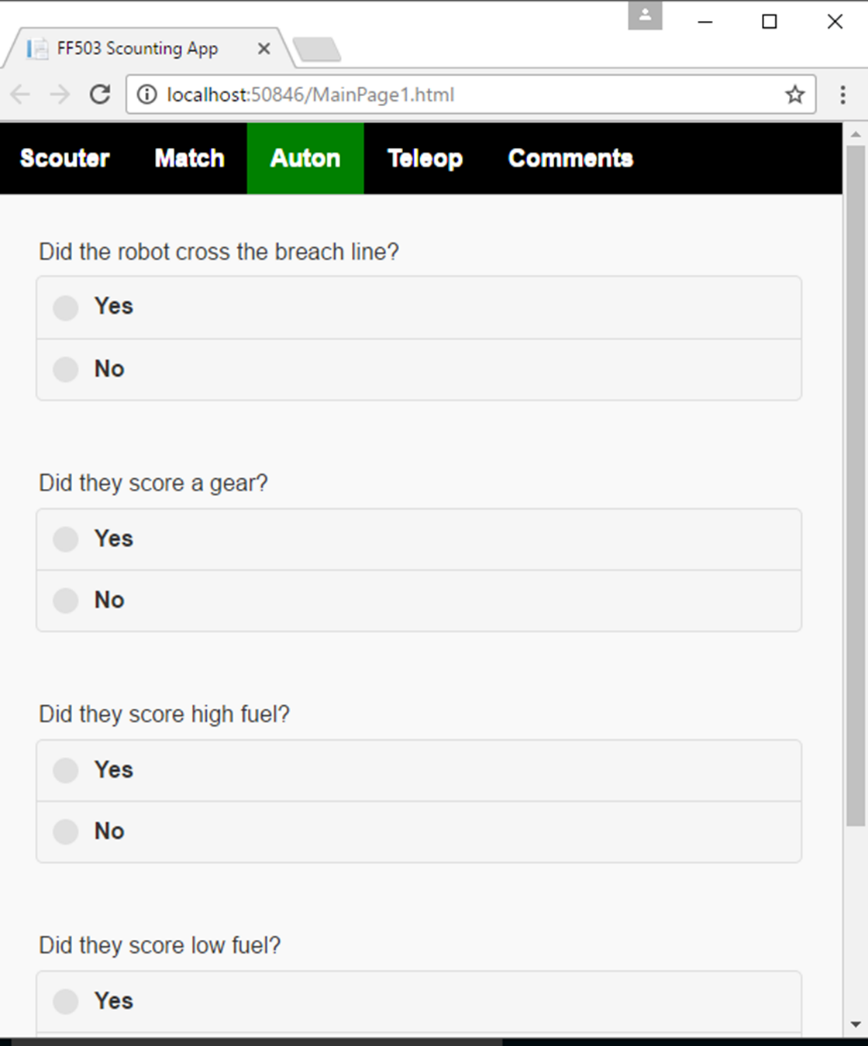
# Scouting System

Last year the team relied on a paper-based scouting system to identify future alliance partners for final matches. Scouts would observe qualification matches and write down their observations on preprinted forms and submit them for data entry. One student would sit in the stands with a laptop computer and type-in all the forms into a single giant spreadsheet. The spreadsheet contained our statistical analysis models to identify each teams’ strengths and weaknesses and rank our perfect alliance partners. We then made a copy of this spreadsheet on a USB stick and ran the results down to the drive team in the PITS to help determine the match strategy. While we are very proud of the scoring system and the great results it produced, it was very labor intensive and prone to data entry errors.

This year, the team developed a tablet-based electronic scouting system. The system is really comprised of three separate components: The scouting tablets, the PC-based collector, and the PC-based strategy dashboard as follows:

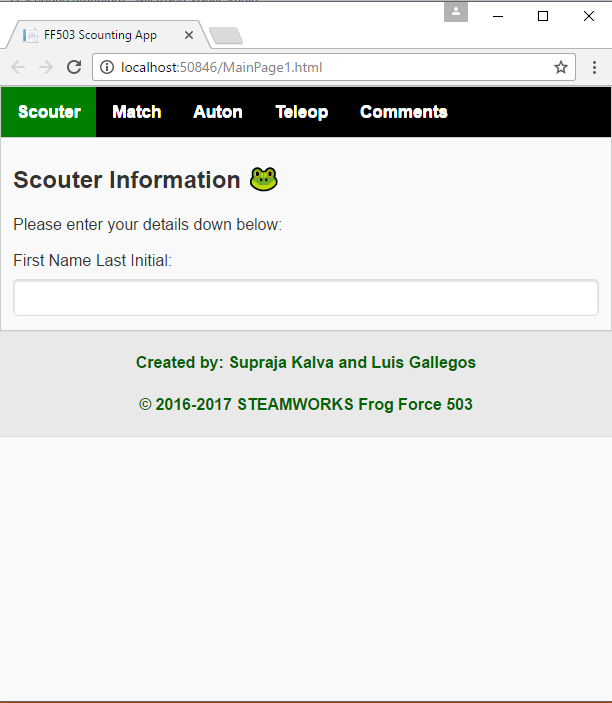
## Scouting Tablets

The scouting system utilizes Dell Venue 8-inch tablets for the team scouts to observe qualification matches and record their findings. The tablets run a browser-based web page to ask the scouts questions relating to the match. Asking specific questions allows the team to use any team member as a scout-i.e. you do not need to be specifically trained to use the system, simply answer the questions.



Using an HTML page allows us to use web-page graphics which makes the system very easy to use. It also allows us to run our scouting app on many different platforms, anything that will support a browser and we have even tested it on cell phones! We will never again have to write an application for a specific hardware device and then re-write it for different devices!

Next year it will be very easy to adapt our scouting application to next year’s competition, we simply change the HTML page and the questions the scouts want to ask. Even better is when this hardware fails, we will simply move the scouting to the students’ cell phones-anything that will support a browser!

**

We run the browser “headless” which means we never connect to a web-server. We utilize Google Chrome for our tablet browser and simply run it in offline mode. Offline mode allows us to run the tablets without Wi-Fi during a competition-the tablets run entirely standalone.

For recording the data we use CouchDB/PouchDB, which are Apache Open Software Foundation products. CouchDB utilizes a Non-SQL Indexed database within the browser and allows our code to be transportable between browsers. Because CouchDB is an indexed database changed rows can be automatically identified. This is even better when you setup CouchDB to automatically synchronize data between the tablet database and the Collector PC database! Specifically, PouchDB is the JavaScript version of CouchDB. So PouchDB is installed on the tablets and CouchDB is installed on the Collector PC. The development team simply adds JavaScript commands to the HTML page to store the data. This means that our scouting applications is written entirely in standard HTML and JavaScript, what could be more transportable?



## The Collector

The scouting system Collector is a Visual Basic application that provides three key functions:

* It transfers the data from the tablet and the PC.
* It periodically polls the First Servers to obtain match scoring data
* IT updates a local MySQL database with all the data it has collected.

### Data transfer

To transfer the tablet data, the system provides for three separate options:

1. We can connect the tablet to the PC via the micro-USB connection then through a USB to Ethernet adapter and then into the PC Ethernet port. This method automatically synchronizes the data between CouchDB databases on the devices.
2. We can export a CSV file and then transfer the file using BlueTooth.
3. We can display a QR barcode on the tablet and then use the PC camera to read the barcode and save the data on the PC.

### FIRST Match Data

For this year’s competition-FIRST Steamworks- we quickly determined that the scouts could not physically count the fuel that was shot into the boilers through observation alone. Another solution had to be found! We originally contacted the **Blue Alliance** to connect to their API and obtain match scoring data. Based on our work to develop the Orange Alliance website for FTC teams, we were able to work with FIRST in Michigan to access the FIRST developer API’s that the Blue Alliance uses to obtain their data directly from FIRST- so another solution was born!

The team developed a VB program to periodically wake up and contact the FIRST server and download any match data for specific events. Now we can get the exact scoring data for each team. Not only for the event that we are scouting, but all matches that every robot at the event has attended. This data is invaluable for determining the correct strategy to compete with. And allows us to use our scouts for observations and not quantifiable data that we can get elsewhere.

### MySQL Data Storage

Once we have collected the scouting data and the match scoring data we need to organize it so that we can analyze the results. To accomplish this, we wrote a VB program to insert the collected data into a MySQL database that resides on the collector laptop. Once into the database we can export it into a spreadsheet or send it down to the drive team in the PIT.

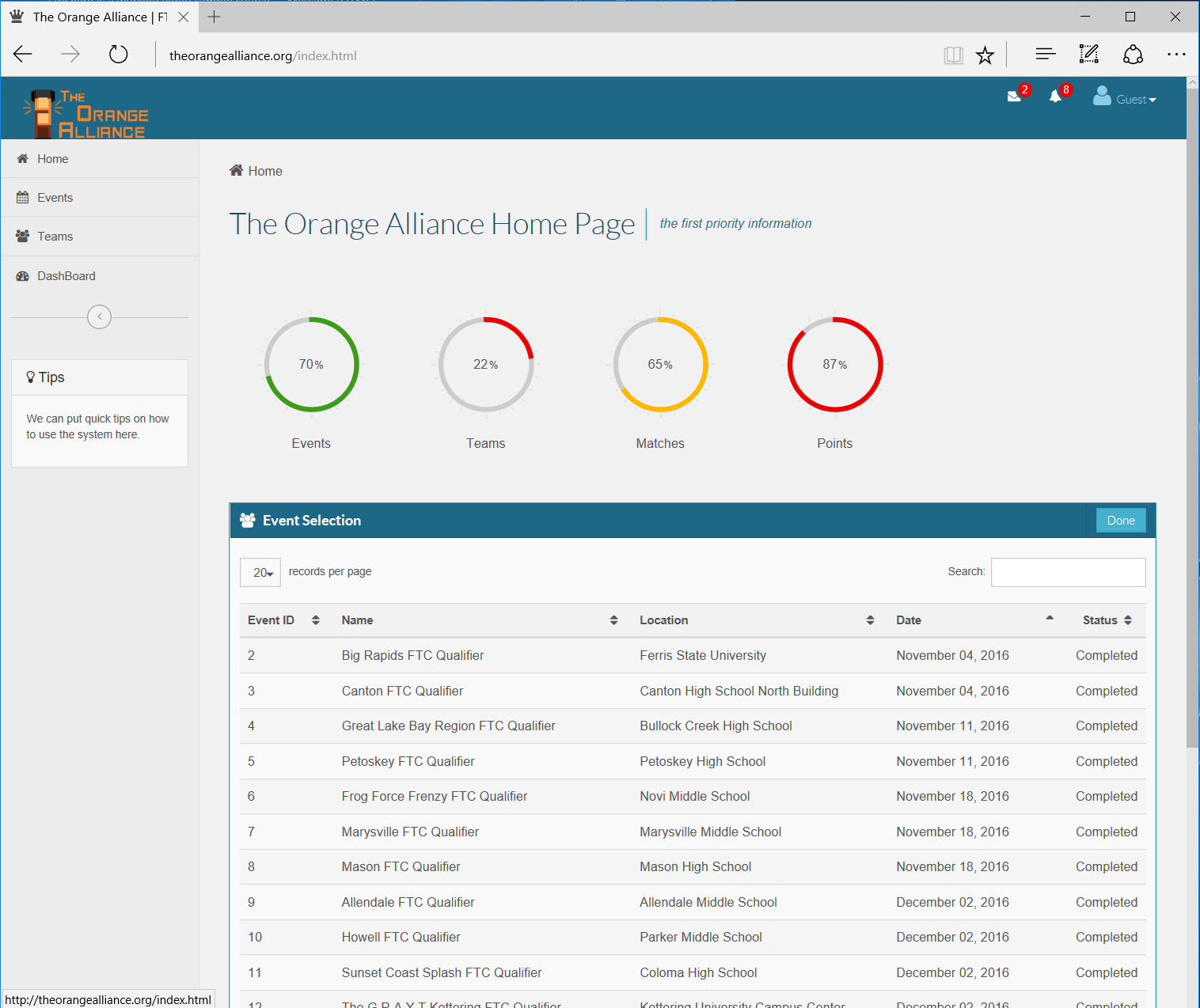
## Match Strategy

All the data we have collected is useless if we can’t get it to the drive team to make good match strategy decisions. The Match Strategy is an HTML page that runs on a Microsoft Internet Information web server that is running on the Strategy laptop. The laptop has a cellular data connection to reach the Collector PC in the stands. Once the connection is established, the MYSQL database is replicated onto the match strategy laptop. If a cellular connection can not be established, the laptop can be synchronized manually using a USB stick and a runner between the machines.

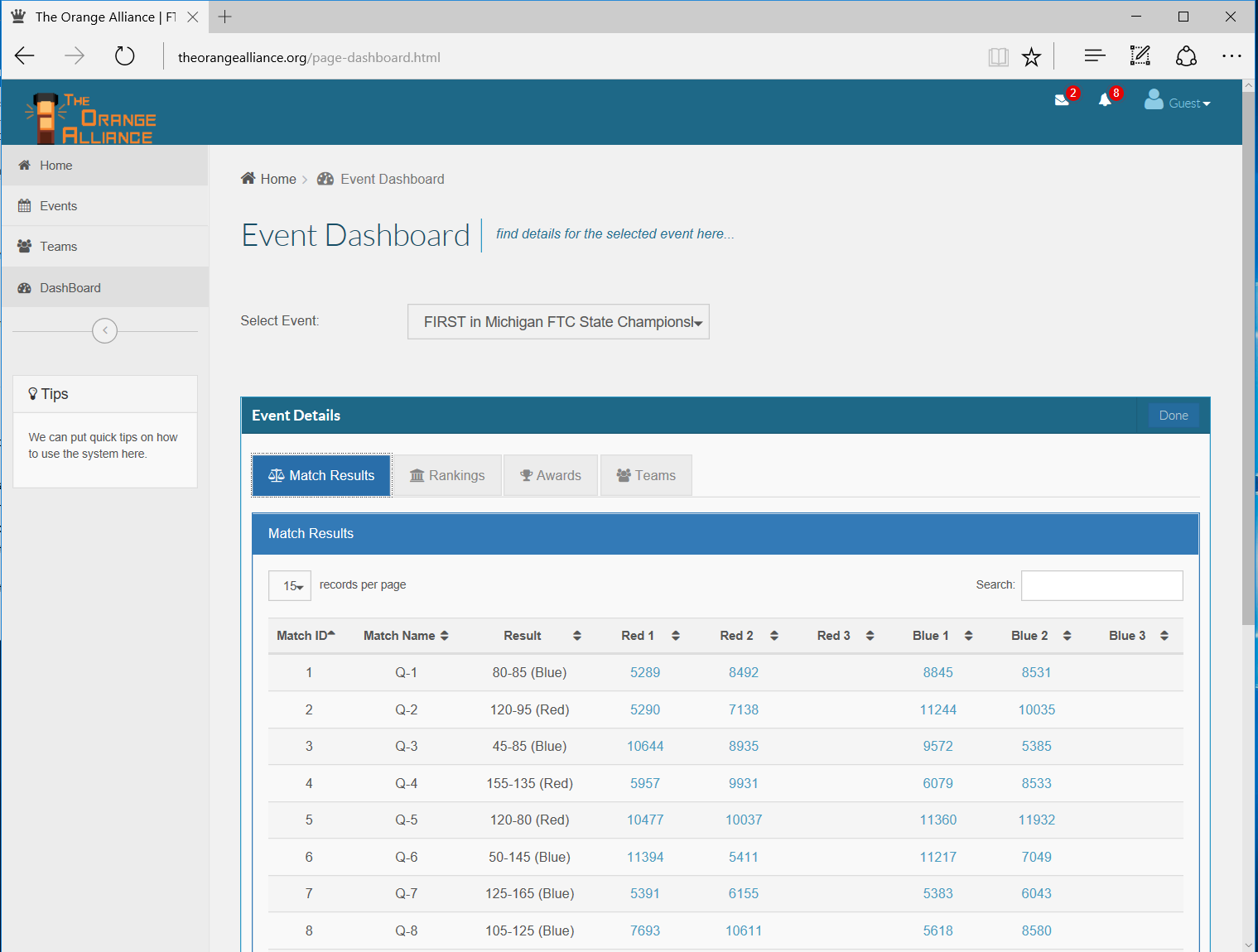
# The Orange Alliance.org

Frog Force dedicates a large part of its off-season time to mentoring FTC teams in the metropolitan Detroit area. This includes volunteering at a number of FTC events. Through our volunteer efforts, it became very clear that the events lacked a scoring system similar to what the Blue Alliance does for FRC teams. To fill this need Frog Force developed the Orange Alliance.

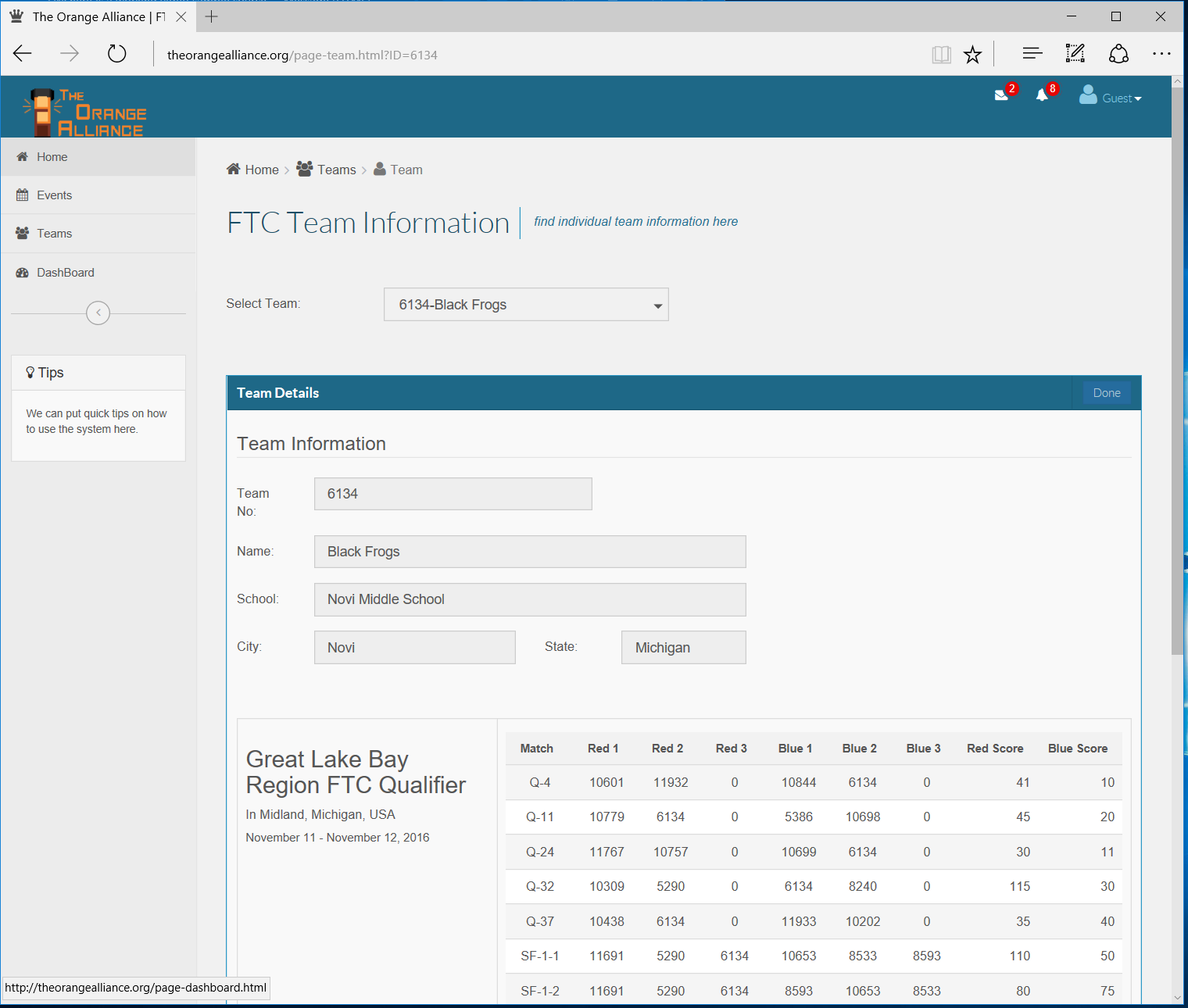
The Orange Alliance is a website hosted by 503. It is an HTML 5 application build on top of Bootstrap, Data Tables, and JavaScript. It is entirely asynchronous and very responsive. On the server side it is built on top of Microsoft Internet Information Server and a MySQL database.



The key technology is a laptop that connected to the FIRST API’s at each event in real time and obtains all match data including scoring. This data is uploaded into the MySQL database and is available for queries. Anyone in the audience at the event can bring up the orange alliance on the cell phone to obtain current information including rankings.



We have loaded all the data for all the FIRST in Michigan events so that each team can review how they did across the whole season.



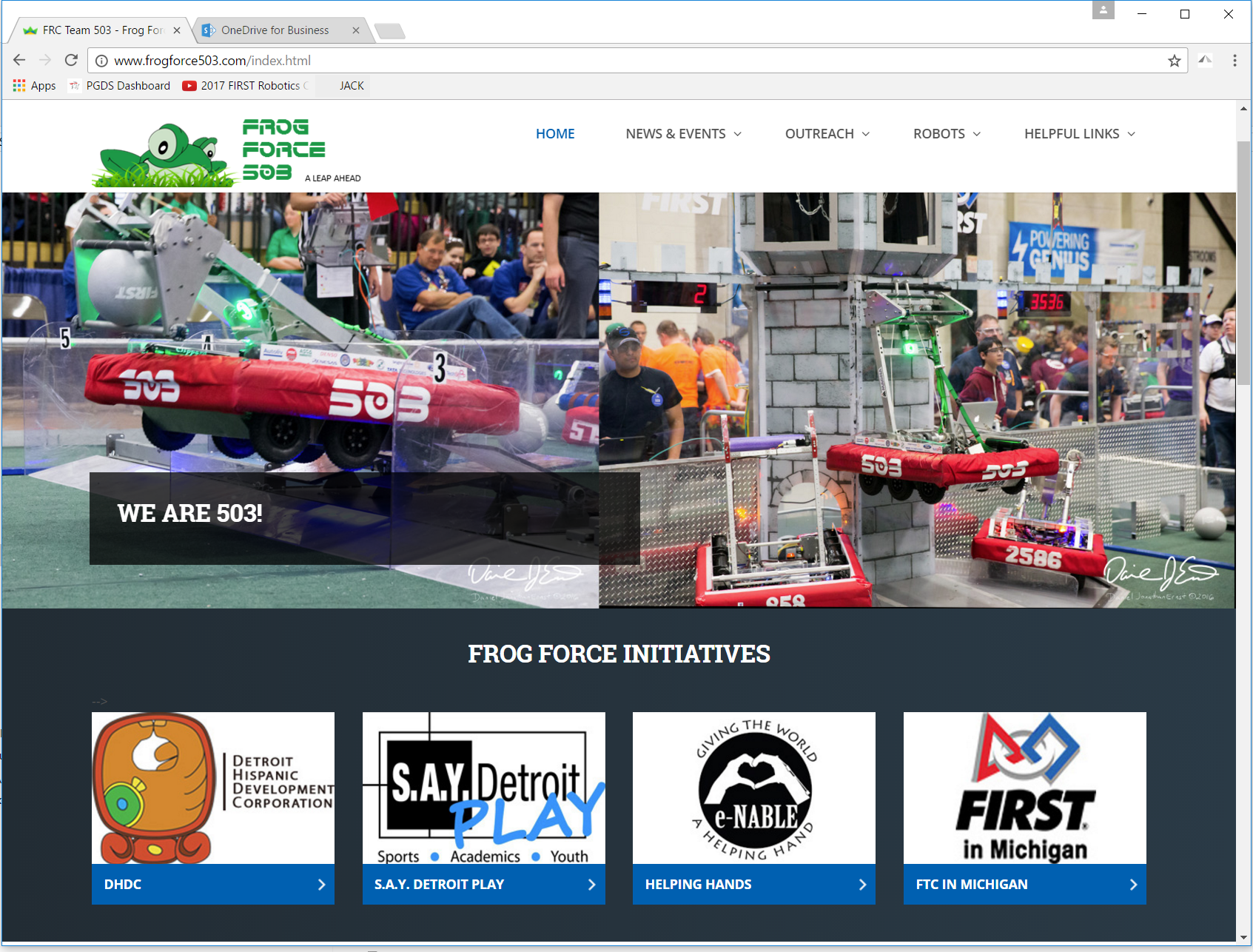
# Frog Force Website

The team was looking to upgrade its website to reflect more news articles and upcoming event information. In addition, we wanted to develop a site that did not require programmers to publish documents. And finally, we needed a website that would connect to our Intranet site in Microsoft Office 365. To meet these requirements the programming team built a native HTML-5 and JavaScript website.

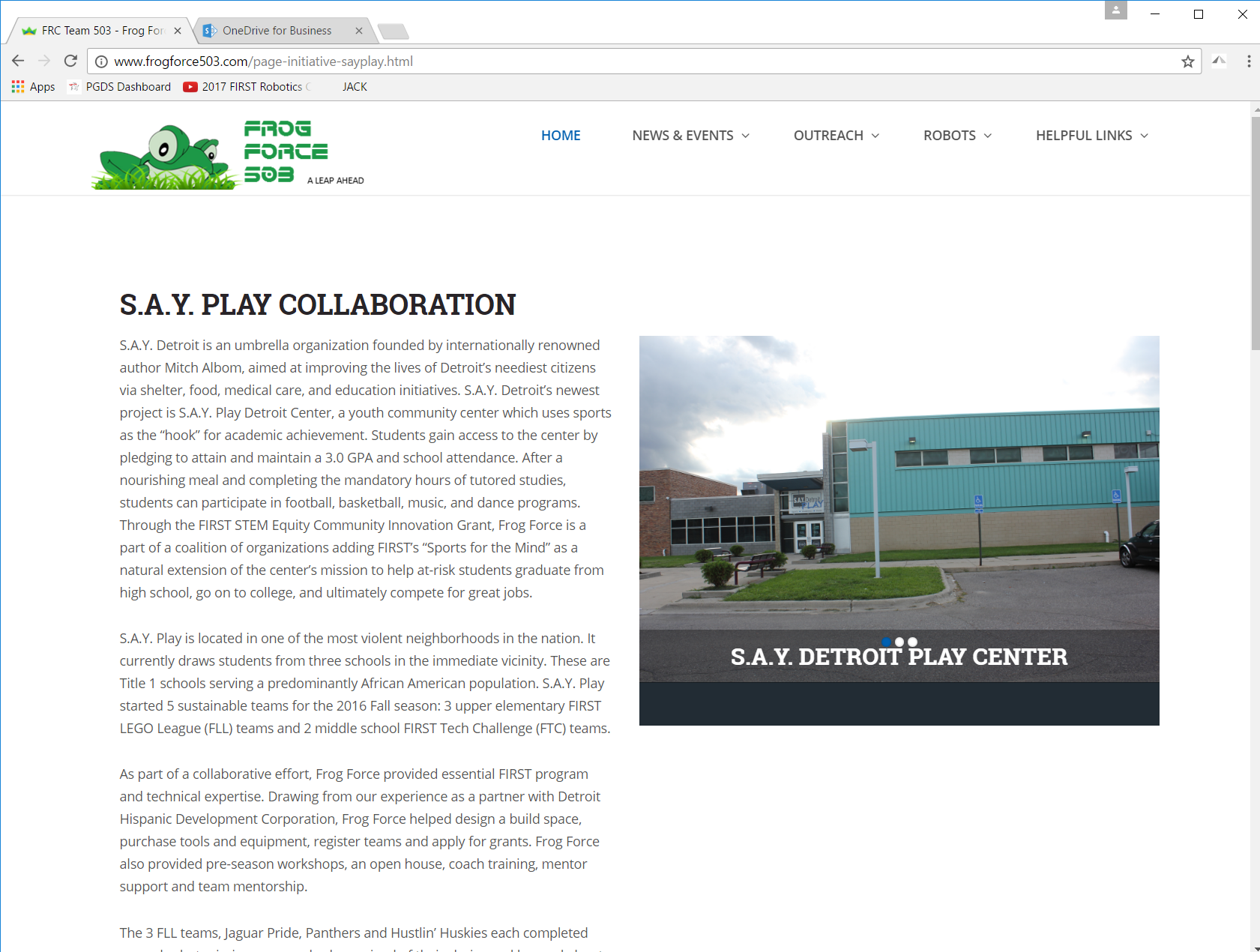
The website is built using a MySQL database to store all the new articles as data-not prebuilt pages. This data is combined with page templates to produce the final page. This is similar in concept to commercial content management systems. However, we could not find one that connected to Office 365 so we ended up developing our own.

The website has a maintenance page to enter new articles and pictures. These are uploaded into the MySQL database and send to a publisher for review and approval. If the publisher approves, the status is change to published and the article immediately is available on the website. No programming involved!

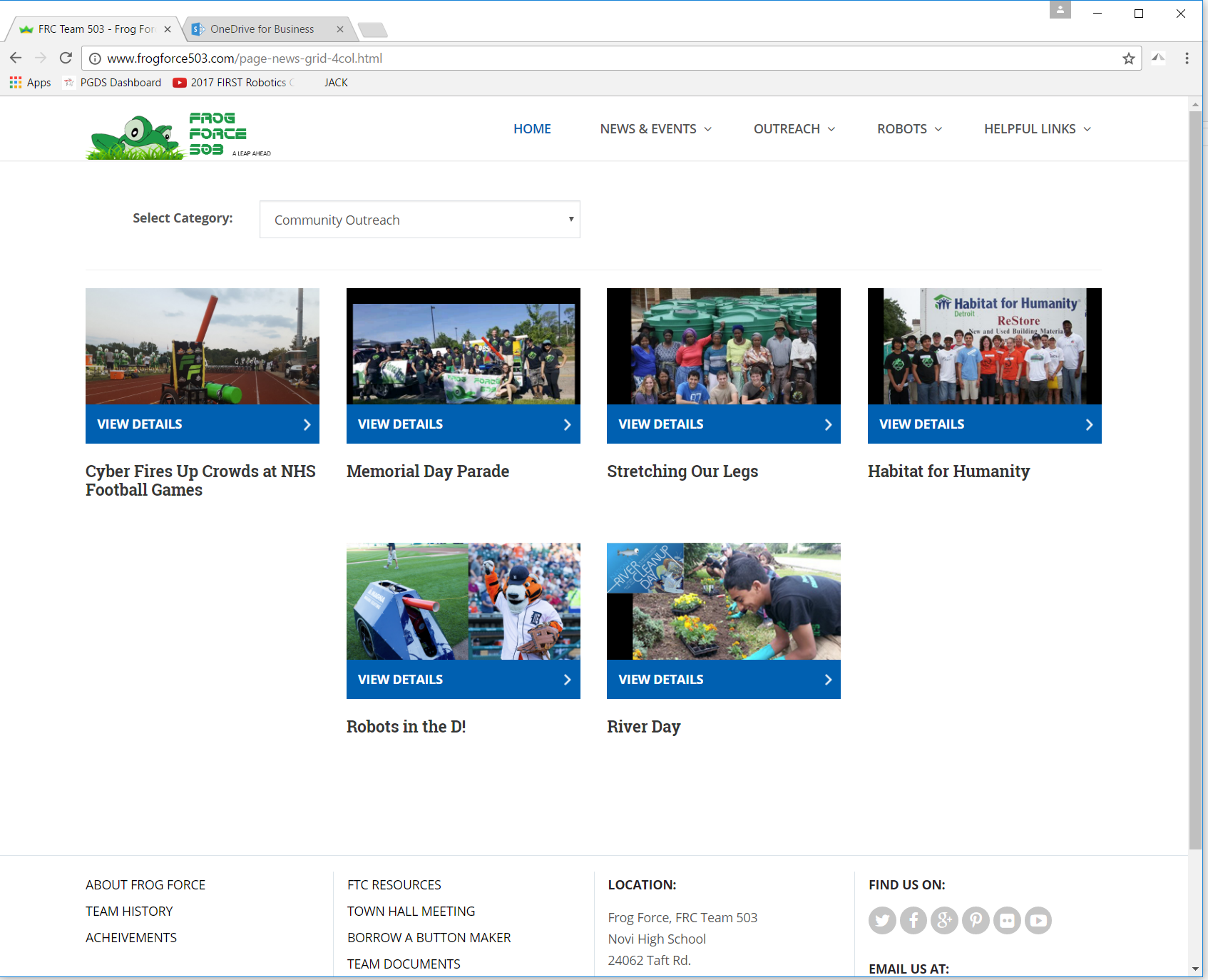
We have a concept of a news carousel that shows large jumbotron pictures of the most recent news. Using the approval date for a publish date, the system automatically identifies the most recent and most important news and adds it dynamically to our news carousel. This is the first thing that any user sees and we wanted to make a big impression.



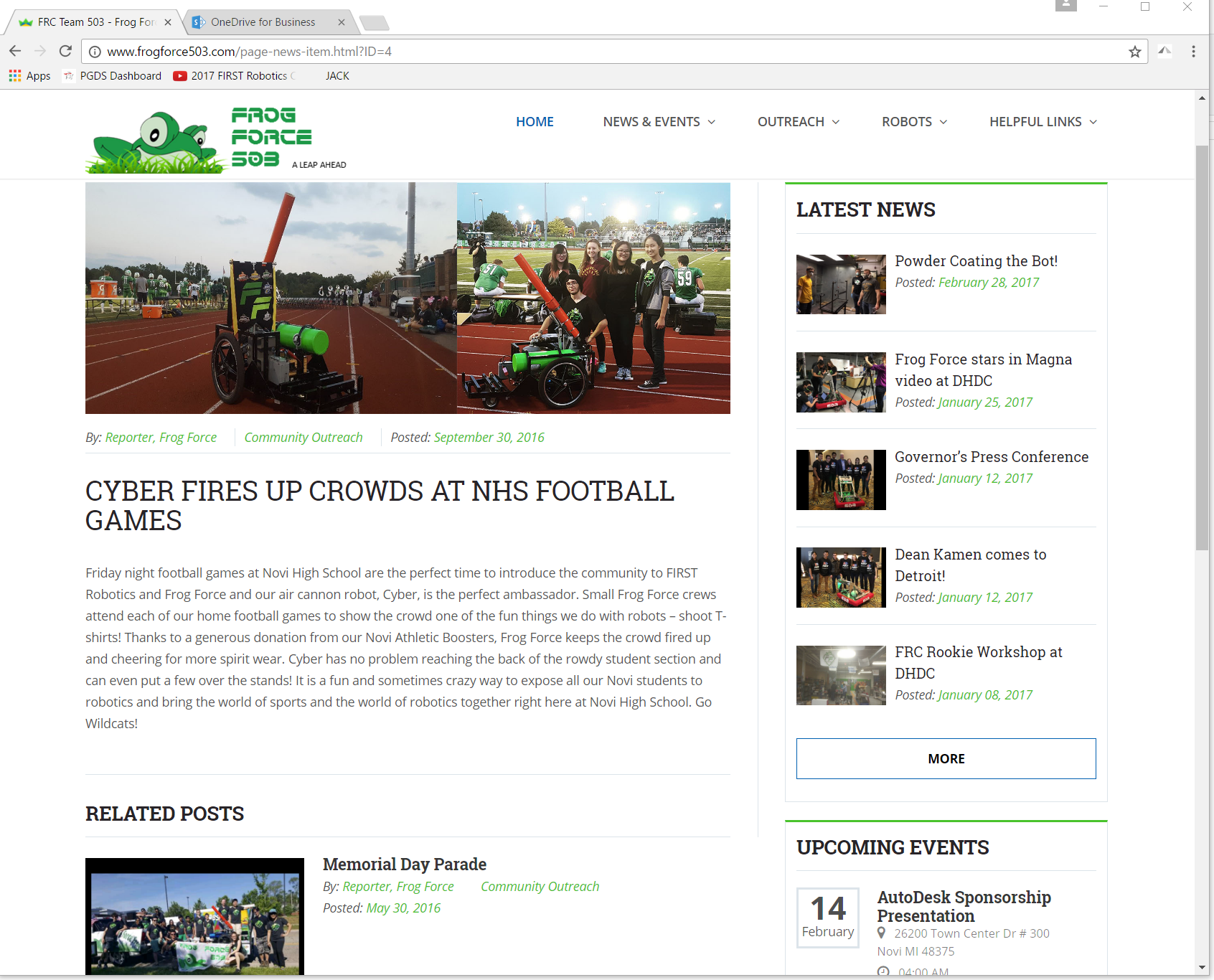
The website also includes dedicated landing pages for each of the teams’ major initiatives. Each landing page contains its own picture carousel to highlight pictures relating to the initiative. In addition, each page includes thumbnail pictures on the most recent news items relating to the initiative. Users can drill down on each of these items to find additional information relating to the initiative.



We also have a news page that shows thumbnails of all news articles contained in the website. These articles are stored by category so that the user can refine their search to only the articles they want to look at .



When you select a news article the news text is combined with a template page to display the webpage specific to the news item. We also show the newest new items, any upcoming events and any other news articles that relate to the main item.



Although our website is now live and ready for use, ew have lots of ideas for expanding its use. And we look toward the off season to finish developing our concepts.

We hope that you like our website. You can check it out at [www.frogofrce503.org](http://www.frogofrce503.org)

Go Frogs!!!

# Appendices

## Appendix A – New Technology Introduced this season

* HTML 5
* CouchDB/PouchDB
* MySQL Database
* QR Code generation
* QR code recognition
* Blue Tooth File Transfers
* Microsoft Visual Basic
* Bootstrap
* Data Tables
* Asynchronous JavaScript
* Microsoft Internet Information Server
* Motion Profiling
* Trajectory Path Planning
* Open CV
* Raspbian
* Java threading