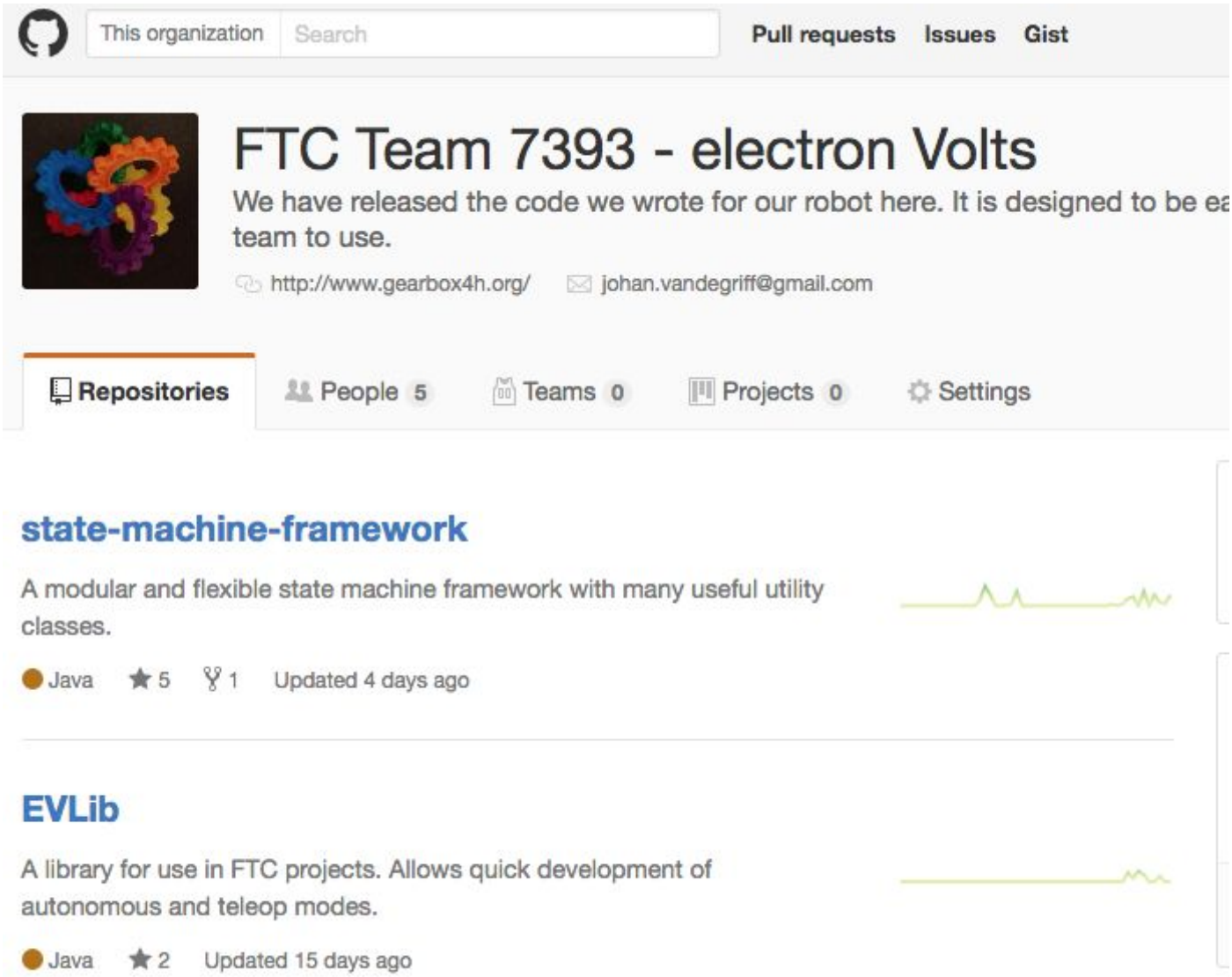
Shared Code from the electronVolts 7393

Our code that is shared on github is separated into 2 layers.The **lowest layer is the state-machine-framework**, which contains a framework for creating a modular state machine, as well as several useful utility classes. The **next layer is EVLib**, which is larger than the state machine framework. It integrates contains code specific to FTC with the state machine framework, including states to control the motors, servos, and sensors, and much more.

The **top layer is the code you write in the robot app.** For us, this includes the robot configurations and opmodes that use both the state-machine-framework and EVLib functionality.

Both of the libraries that are shared on github have a **~30 page wiki** (available on github on the "Wiki" tab in the repository) that explains each of the different features of the library, as well as some sample code showing how to write opmodes with the library. You can find the code on our website (http://www.gearbox4h.org/github/) or on github (https://github.com/FTC7393).

More about each library is on the next page.



state-machine-framework

66 files, 8676 lines (5554 lines of code, 1720 lines of comments, and 1402 blank lines)

The state-machine-framework contains a framework for creating **modular state machines**. It comes with several built-in states, and allows you to create your own for your specific use. It allows for mixing and matching of states (such as drive, turn, or follow line) and end conditions (such as sensors, encoder positions, or time). You can link states together to create branching or loops.

It also contains several **useful utility classes**. One of these, OptionsFile, stores values to a file and retrieves them later. This can be used to set options for your autonomous mode, such as team color and starting delay. It has classes for unit conversions, 2-D and 3-D vectors, digital edge detection, analog scaling, and logging.

The library has a **38 page wiki** (https://github.com/FTC7393/state-machine-framework/wiki) explaining all the different features and how to write a state machine using the framework. This library was designed to have no dependencies. This means it can be used in ANY Java app, whether it is FTC, FRC, Android apps, or even a Minecraft mod!

EVLib

82 files, 10,883 lines (5931 lines of code, 3323 lines of comments, and 1629 blank lines)

EVlib is built on top of the state-machine-framework, and it adds FTC-specific code, including states to control the motors, servos, and sensors, and much more. While this restricts the usage to only FTC teams, it also increases its usefulness by allowing it to be more specific. It has a **35 page wiki** (https://github.com/FTC7393/EVLib/wiki) explaining each of the different features of the library, as well as some sample code showing how to write opmodes with the library.

An example of this is OptionsOp, which is an opmode you can run that uses OptionsFile from the state-machine-framework. When you press certain buttons on the gamepads, OptionsOp will store values to a file that can later be read in autonomous mode. This is very useful in changing the options right before a match when no software changes are allowed.

EVLib includes image processing with Vuforia and OpenCV together, although this requires extra setup for OpenCV (http://www.gearbox4h.org/opencv/ for a tutorial). The wiki has an example of how to use this (https://github.com/FTC7393/EVLib/wiki/Vuforia-and-OpenCV-Beacon-Color-Detection). Check back later as we plan to update the wiki soon. For now, you can visit http://www.gearbox4h.org/opencv/ for a tutorial on how to use OpenCV by itself in the FTC app

EVlib adds many built-in features to every opmode:

- Robot configurations easily used for multiple opmodes
- Joystick scaling and button edge detection
- Simple logging of data in columns in a file on the phone

As well as many features for the different devices on your robot:

- Wrapper class for motors
- i2c Line sensor array reader
- Servo Speed control
- Servo presets that can be changed with the gamepad
- Control of mecanum wheels: Gyro stabilization, Line following, and more
- States and EndConditions that are specific to the motors, sensors, servos, etc.

And other features that don't fit into either category, such as:

- Utility for storing files in a directory tree on the phone
- Global telemetry