# SE 101 Project Proposal: *Traffix*

*Navraj Singh Chhina (nschhina), Brooke Mackenzie Dolny (bmdolny), Wesley Alexander Ting-Zhang Leung (watleung)*
*Thursday, October 12th, 2017*

## Background

Traffic optimization has been an increasingly common problem in major cities around the world. A variety of systems are currently in place to tackle this issue, including pre-timed systems (such as Toronto) that change based on the time of day, and coordinated control systems that synchronize traffic lights to ensure drivers encounter a series of green lights, and real-time and dynamic control systems that adjust signal timing based on sensors that monitor the traffic.

## Project Overview

Our project will involve simulating traffic in a city, while also maximizing the flow of traffic by reducing the stopping time at traffic signals. Through the use of an Raspberry Pi controller, each individual traffic light will be controlled based on a number of factors, including the current volume of traffic and the status of other traffic lights. The traffic system will be able to adapt to any layout of roads specified by the user to optimize the traffic. Since there could be a significant number of vehicles in our simulation (exceeding 1000), it would not be feasible to have physical vehicles; vehicles will instead be represented by LEDs on a circuit board.

## Software Components

The software component of our project will involve utilizing optimization algorithms (such as shortest path and maximum flow algorithms) to better maximize the flow of traffic in a city. The Raspberry Pi will be connected to a display for the user to monitor the traffic flow in the entire city. In addition, the display will provide a graphical user interface to allow modifications of certain variables in the simulation. These variables will include, but may not be limited to, the road layout of the city, the volume of traffic in the city, and minor changes to the optimization algorithm (such as how quickly the controller responds to changes in the environment).

## Hardware Components

Our project will require at least the following components:

- **Raspberry Pi:** Programmed with traffic flow efficiency algorithm.
- **Raspberry Pi Real Time Clock Module:** Calculating and keeping track of the time elapsed.
- **LCD Display:** Displays the traffic flow of the entire city. Also provides a graphical interface for the user to set the road layout, or change the level of traffic.
- **LEDs:** Different colours will represent the roads, and cars.
- **Circuit Board:** To mount the LEDs and display the simulation. Should the city be too large to physically display, the circuit board will only display a section of the city.

## Prototype Plan

The prototype will be primarily experimental and probably will be programmed using Python and/or C++. Although many of the algorithms used in the prototype will also be used in the final demo, much of the user interface and the small scale physical model will be redesigned and reimplemented in the final project. The emphasize of the prototype would be on software section of the project and ensuring that the optimization algorithm does not hinder the performance of the simulation.

## Anticipated Challenges

- Synchronizing various traffic lights based on volume can be tedious to compute. Polling and processing the all data based on the environment variables (such as volume of traffic) maybe only be feasible to be collected and computed on certain intervals. Since the many of the algorithms involved have superlinear runtimes, it will be important to optimize our software to ensure that the system's performance is not impacted.
- Keeping the physical circuit board and LCD display in sync with one another may be a challenge given the number of objects that need to be updated; however, given the complexity of the optimization algorithms, this may not be the bottleneck in performance.
- Using LEDs to represent cars maybe not be feasible. Acquiring enough LEDs and circuit boards to have a reasonably sized representation of a city may be too costly.
- It may be difficult to create an intuitive graphical interface to allow the user to modify the environment variables.