Red Light Green Light

New Mexico

Supercomputing Challenge

Final Report

April 1st, 2015

Team 132

School of Dreams Academy

Team Members:

Victoria Troyer

Clara Sims

Sponsor:

Eric Brown

Mentor:

Creighton Edington

Table of contents

Background

Last year we were both a part of the School of Dreams Academy's Lemelson-MIT InvenTeam. This team invented "Police ALERT", a monitoring device for police officers that detects potential threats within a 40 ft. 360° radius around their patrol vehicle[1]. A device with a camera looking up at a conical mirror is placed on the roof of the patrol vehicle. The police officer in the vehicle is warned via an in-cab indicator when motion is detected within the stated parameters[1]. The in-cab indicator includes both audio and visual cues of where the motion is coming from along with an integrated user interface to control which areas the device is monitoring[1]. Police ALERT is unlike other existing technology for police officers. It protects against approaching threats they were previously blind to and reduces the likelihood of an ambush while in their patrol vehicle[1]. The alpha prototype cost $500 to fabricate. We were the authors of the patent application for this device. The inspiration for the project Red Light Green Light came from exploring other practical applications for this device.

Introduction

The very first traffic light system installed was on the corner of Euclid Avenue and East 105th Street in Cleveland, Ohio[2]. The year was 1914 and this electric traffic signal was wired to a manually operated switch inside a control booth[2]. Since then driving nations have evolved with traffic lights, not only being used worldwide as the primary form of traffic regulation, but have also come a long way in their technology.

Right now the most public traffic systems use time, detection, or four-way-stops (stop signs). The most commonly used traffic light system is the timed system, which uses traffic

coordination. This means traffic analysts calculate through data analysis of many different intersections, the most efficient times to set for each light[3]. Every city has different times set for their traffic lights because each has their unique traffic flow and employ different analysts, however, the average stop light is set to operate with a red phase of 100 seconds, a green phase of 60 seconds, and a yellow phase lasting roughly 10 seconds[3]. The detection system uses an inductive loop[4]. This is a loop of wire that is imbedded into the surface of the pavement leading up to the traffic light in order to detect vehicles and change the light according to how many vehicles are waiting on either side[4]. The four way stop is a common intersection where four stop signs conduct traffic to stop and let other vehicles go accordingly to who arrived at the intersection first. This system is used on roads with relatively light traffic flow.

After researching the current methods of traffic regulation it is clear that these methods are effective and efficient. They would have to be with an exponentially increasing population of over 217 million licensed drivers in America in 2014[5]. However, we believe they could, and need to be more efficient. Not because, yes, it is annoying to sit at a red light when there are no other vehicles in sight, but because the use of diesel and fuel in transportation is the second largest output of carbon dioxide in the U.S. today, accounting for 32% of annual carbon dioxide emissions[6]. One way to target this problem would be to drive less. However, in a country that relies heavily on vehicles for transportation, much more than other industrialized countries who take advantage of high-speed rail, we need to find a way to decrease carbon dioxide emissions without lowering drive time.

The purpose of our project is to decrease carbon dioxide emissions from idling vehicles. The average American spends 16 minutes a day or more idling in their vehicles. This equates to 16 ounces of carbon dioxide being released into the atmosphere per day per vehicle[6]. With 190

million licensed drivers and 217 million registered vehicles in America in 2014, that is roughly over 1 billion pounds of carbon dioxide emissions per year[5]. If our motion light system works effectively, we will decrease the idle time at traffic lights and therefore we will significantly reduce carbon dioxide emissions from vehicles.

The objective of our project is to develop a new traffic light system that decreases vehicle idle time at traffic lights, and costs less than imbedding wires into the road when installing an inductive loop. The system we propose would use motion detection, and would ultimately be applied to a device similar to that of our "Police ALERT" model. We decided that rather than focusing on modifying the actual device we would test our motion light system in a pre-existing NetLogo program. This way we could compare the motion system to the timed, detection, and four way stop system in order to determine its effectiveness.

Program/Code

The pre-existing NetLogo program that we used for our project was set up as a model of traffic moving through a city traffic grid. It allowed users to control traffic through global variables, such as speed limit and the number of vehicles. We kept these, along with the grid size x and y which controls how many intersections there are, and added the three other light systems (detector, stop-sign, and motion).

Hypotheses

1. If we simulate a traffic program that uses the motion light-system, then the traffic flow will be more efficient.
2. If we simulate a traffic program that uses the motion light-system, then the traffic flow will not change.
3. If we simulate a traffic program that uses the motion light-system, then the traffic flow will be less efficient.

Materials

1. Computer
2. NetLogo program

Experimental Procedure

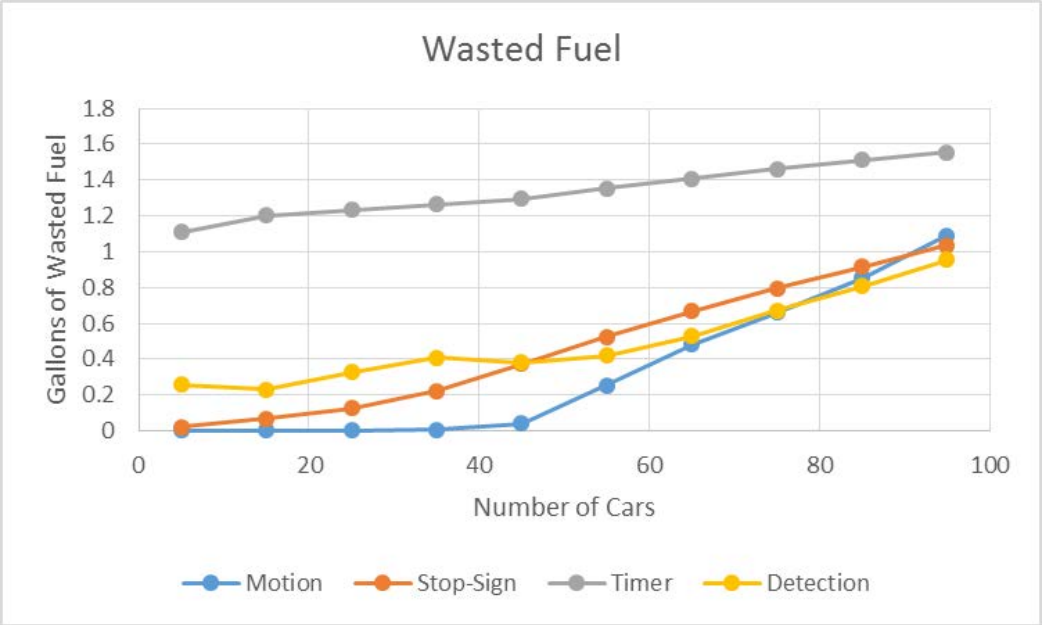
1. The first step for this project was to analyze the pre-existing code in NetLogo
2. Then, keeping many aspects of the pre-existing program including timed system, we modified it to fit our needs by adding the following.
    a. Motion, detection and four way stop
3. Determined ticks-per-cycle would equal 24 hours (43200)
4. We determined 0.0625 pounds of carbon dioxide equaled 1 minute of idling
5. Inserted Extra-CO2 emissions as equal to wait time per vehicle
    a. vehicle idling for 1 minute (1 minute of wait time) = 0.0625 pounds of CO2 emissions
6. Inserted wasted fuel as equal to wait time per-vehicle

    a. vehicle idling for 1 minute = .0025 gallons of wasted fuel

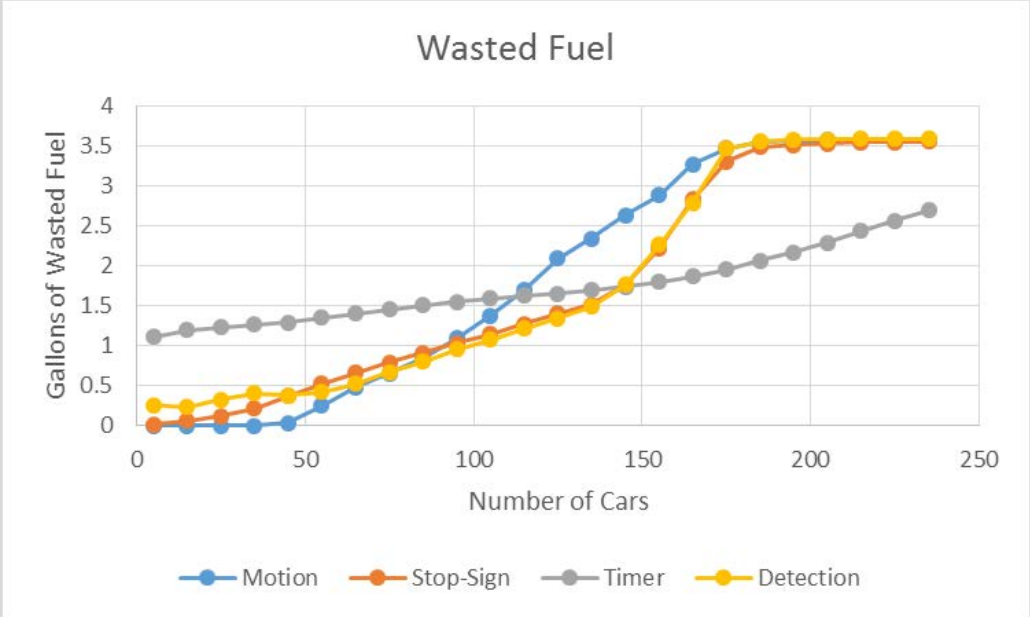
7. We ran each light system for the equivalent of 24 hours (43200- ticks)
    a. 30 ticks = 1 minute
    b. 30(60) = 1800
    c. 1800(24) = 43,200
    d. 43,200 ticks = 1 day
8. Through behavior-space we documented the data from each light system to an Excel spreadsheet
    a. grouping extra carbon dioxide emissions for all four systems
    b. grouping wasted fuel for all four systems
    c. grouping speed for all four systems
    d. grouping wait time for all four systems
9. We then compared, analyzed, and graphed the data.

Results

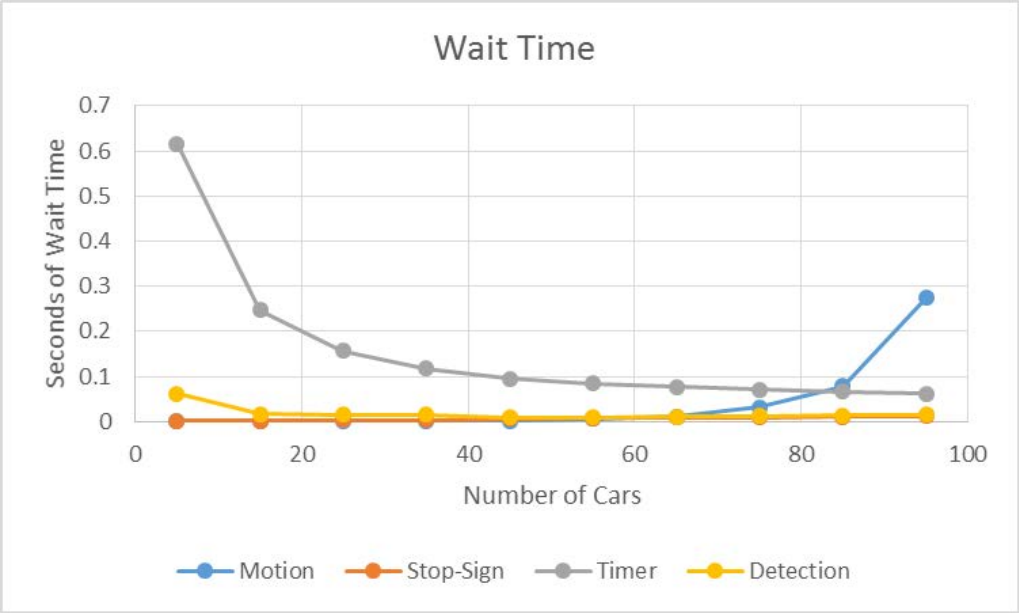
The results of this project are shown through graphs and charts.



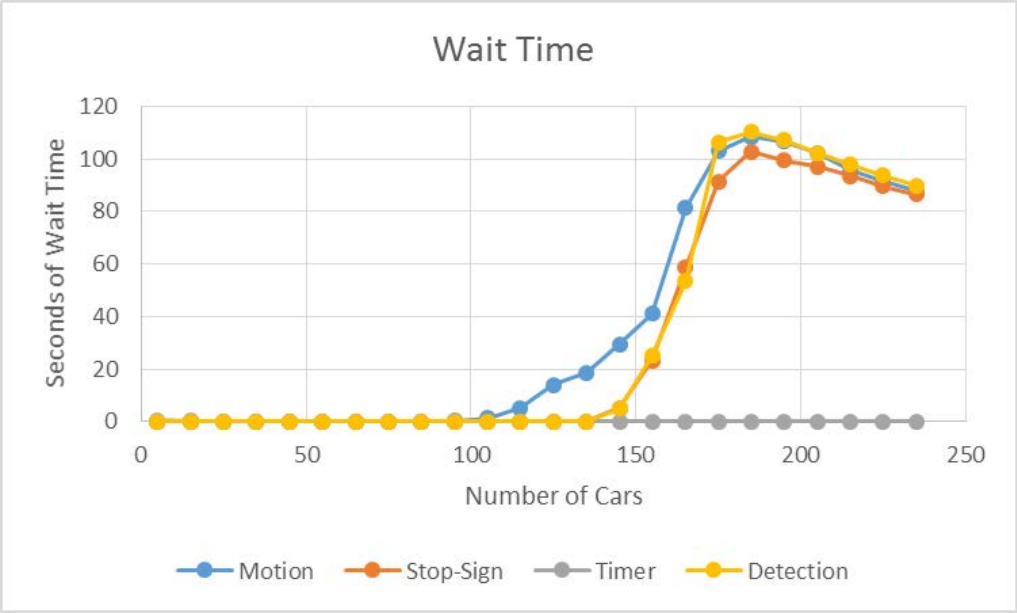
The data above represents the amount of wasted fuel in gallons per seconds of wait time. 95 cars are measured during a 24 hour period.



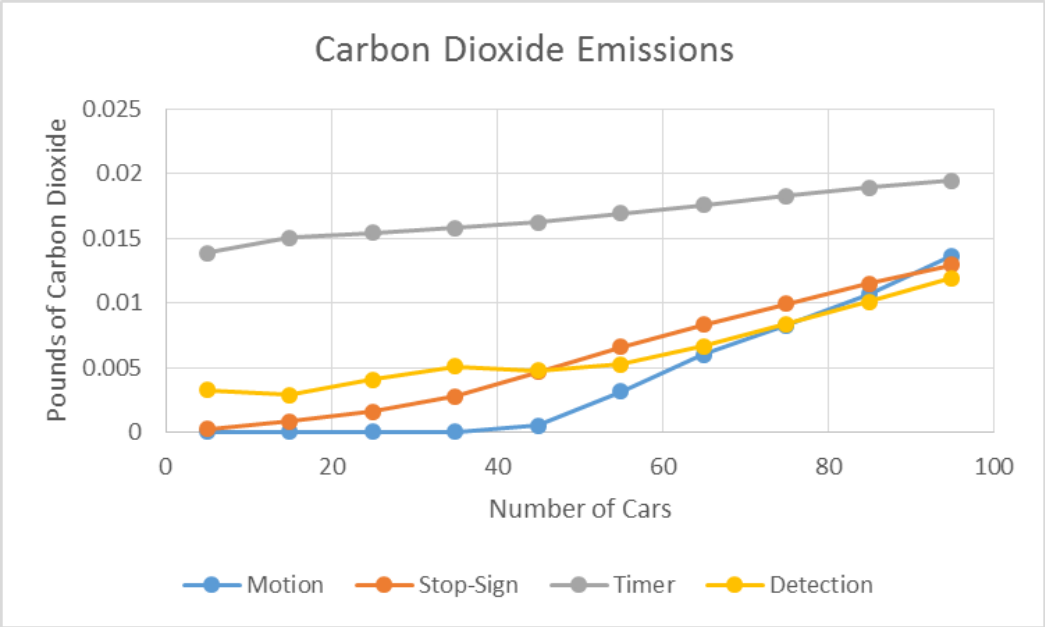
The data above represents the amount of wasted fuel in gallons per seconds of wait time. 235 cars are measured during a 24 hour period.



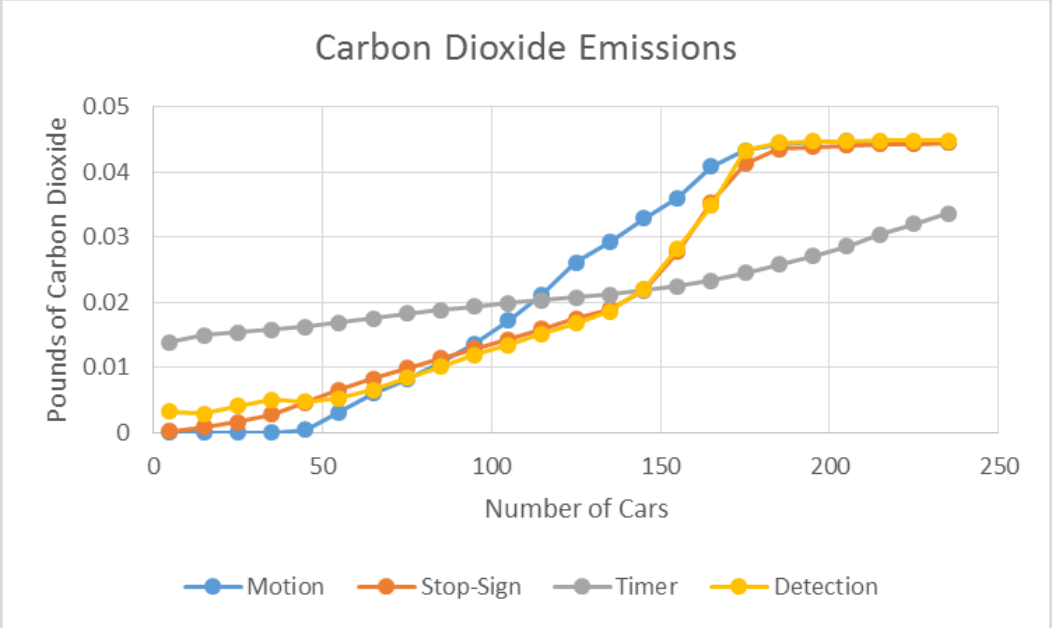
The data above represents the average wait time per car in seconds (0.5 equals 1 second). 95 cars are measured during a 24 hour period.



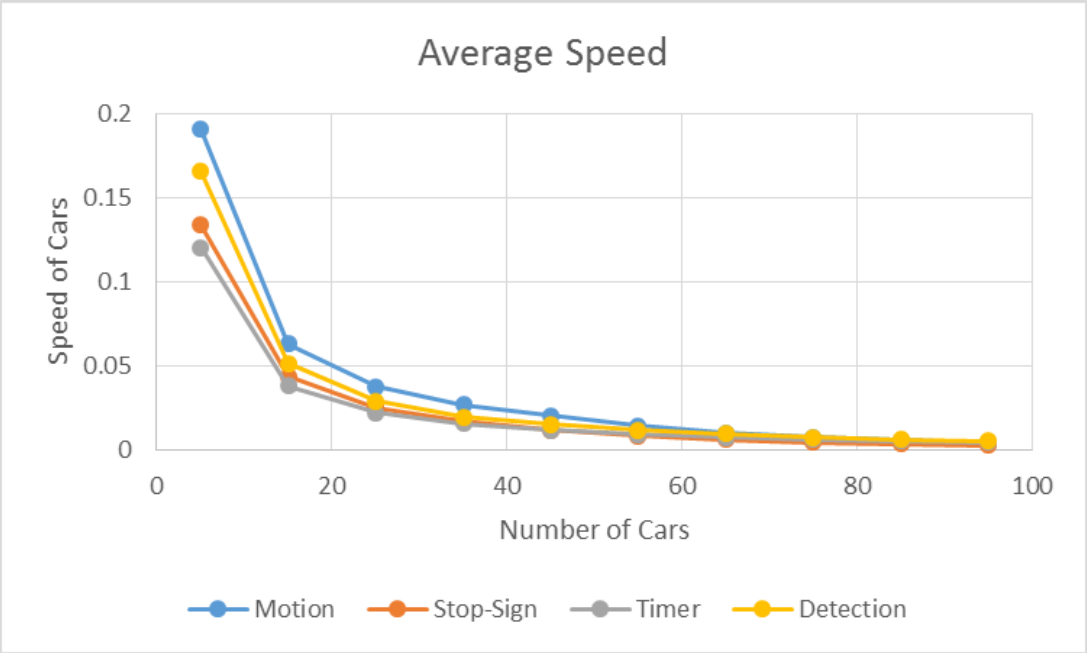
The data above represents the average wait time per car in seconds (0.5 equals 1 second). 235 cars are measured during a 24 hour period.



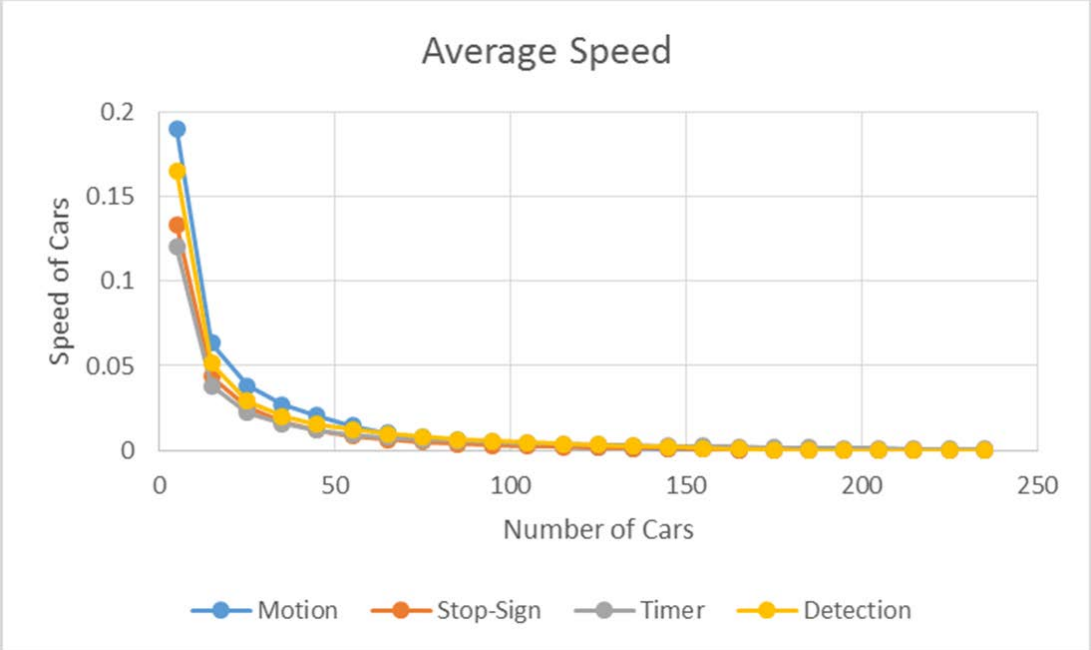
The data above represents the amount of carbon dioxide emitted per seconds of wait time. 95 cars are measured during a 24 hour period.



The data above represents the amount of carbon dioxide emitted per seconds of wait time. 235 cars are measured during a 24 hour period.



The data above represents the average speed of 95 cars measured during a 24 hour period.



The data above represents the average speed of 235 cars measured during a 24 hour period.

Discussion

While conducting the experiment for Red Light Green Light, we determined that the motion light system was more effective than the other three systems (time, detector, stop sign), but only when there was less traffic. When the amount of vehicles was set equal to or below 95 there was less wait time, less wasted fuel, and lower carbon dioxide emissions than the other three systems. However, anywhere above that number of vehicles a major grid-lock occurred. We speculate that this could be from insufficient detail in our code. For example, in a grid-lock situation, a vehicle will enter the intersection and then stop in the middle of it. When this happens the motion system simply stops working because it does not know what light to change when there is no motion being detected.

Our stop sign and detector light systems became more efficient when the number of vehicles was between 100 and 145. Still, like the motion system, these two systems also became congested, almost to the point of grid-locked, soon after exceeding 145 vehicles. The timed system was the least efficient up until 145 vehicles. After that it was significantly more efficient up until our maximum of 235 vehicles. We did not run the system with more than 235 vehicles, but the fact that the timed system did not appear to become grid-locked at any point led us to exciting questions about its effectiveness in evacuation or natural disaster situations. More

research and experimentation would need to be done to prove this but it is a possibility for future projects.

The traffic grid model is a very simplified model. Traffic only goes in one direction, and there are no turning lanes. As a result of this we are not sure how these things would affects any of the systems, especially the motion and timer light systems. Thinking about the practical applications of the device, we speculate that it would be hard to mount the motion detection traffic light in large intersections. Because of the need for the motion device to be mounted in the middle of the intersection, we are not sure if it would works with more than two lanes going in each direction. The timer system may become grid-locked with more than two lanes going in either direction, also. This is speculation and further testing needs to be done for conclusive and reliable evidence to support these claims.

The fact that the motion system was the best light system with 96 vehicles and below supports our idea that this would be an effective alternative to the detection and timer systems in rural, small town, and relatively low traffic areas.

Conclusion

In conclusion our first hypothesis (If we simulate a traffic program that uses the motion light-system, then the traffic flow will be more efficient) was correct. In areas with less traffic i.e. small towns and suburban areas the motion light system would be beneficial and an improvement in cost, efficiency and decreasing negative environmental impact. It would reduce CO2 emissions from idling cars, waste less fuel, and there would be a decrease in travel time for individual cars. Even in traffic situations where the detection system and motion system are close in their effectiveness the motion system is more cost efficient and easier to install then installing

and maintaining the inductive loop of the detector traffic control system. As stated in the introduction, the motion system may only cut back on wait time by what seems to be a slim margin, but when trying to reduce carbon dioxide emissions and waste less fuel, every little bit counts.

Practical Application

With development and further testing**,** our project is the beginning step to a practical real world solution to more efficient and environmentally friendly traffic regulation. For practical application our code would have to be applied to a device similar to that of Police ALERT, the device discussed in the background.

Future Expansion

- Can our program be applied to the device previously used for Police ALERT or one similar to it?
- Will be it be as effective in the actual monitoring of traffic systems as it was shown to be in our code?
- Does the timer light-system ever become grid-locked?
- If the timer light-system doesn't ever become completely grid-locked could it be used as a backup for every traffic light in an emergency or evacuation situation?
- How will adding turn lanes, and lanes going in two directions affect each light system?

Acknowledgements

Thank you to everyone who made our research and experimentation possible. We couldn't have done it without your help!

Bibliography

1. Grubb, Chloe. "Lemelson-MIT Program." *School of Dreams Academy InvenTeam*. 1 Jan. 2014. Web. 10 Mar. 2015.

2. Staff, History.com. "First Electric Traffic Signal Installed." *History.com*. A&E Television Networks, 1 Jan. 2009. Web. 4 Mar. 2015.

3. "Mathematical Modelling in Traffic Flows." Web. 9 Mar. 2015.

4. "How Does a Traffic Light Detect That a Car Has Pulled Up? - HowStuffWorks." *HowStuffWorks*. HowStuffWorks.com, 1 Apr. 2000. Web. 9 Mar. 2015.

5. "Blueprint for Safety - Risk Control Coverage Guide." *Blueprint for Safety - Risk Control Coverage Guide*. Web. 9 Mar. 2015.

6. "Carbon Dioxide Emissions." *EPA*. Environmental Protection Agency, 2 July 2014. Web. 8 Feb. 2015.

Code

```
; Modified for Supercomputing Challenge (SODA - Sims and Troyer)
; Prediction - an "ALERT"-based traffic control system will develop into a CAS

globals
[
  grid-x-inc               ;; the amount of patches in between two roads in the x direction
  grid-y-inc               ;; the amount of patches in between two roads in the y direction
  acceleration             ;; the constant that controls how much a car speeds up or slows
down by if
                     ;; it is to accelerate or decelerate

  num-cars-stopped         ;; the number of cars that are stopped during a single pass thru
the go procedure

  ;; patch agentsets
  intersections            ;; agentset containing the patches that are intersections
  roads                  ;; agentset containing the patches that are roads
  light-timer              ;;

  max-patch-motion         ;; sets the maximum value for motion in an intersections radius
based on grid size
  extra-CO2-emissions
  mean-mean-speed-of-turtles ;; The average of the average of the speed of cars
  temp-mean-speed-value
  wasted-fuel              ;; How much fuel waisted from cars stopped/Slow
  mean-wait-time-of-turtles;; Average of the average wait time of cars
  temp-wait-time-of-turtles
]

turtles-own
[
  speed     ;; the speed of the turtle
```

```
  up-car?   ;; true if the turtle moves downwards and false if it moves to the right
  wait-time ;; the amount of time since the last time a turtle has moved
]

patches-own
[
  intersection?   ;; true if the patch is at the intersection of two roads
  green-light-up? ;; true if the green light is above the intersection.  otherwise, false.
                  ;; false for a non-intersection patches.
  my-row          ;; the row of the intersection counting from the upper left corner of the
                  ;; world.  -1 for non-intersection patches.
  my-column       ;; the column of the intersection counting from the upper left corner of
the
                  ;; world.  -1 for non-intersection patches.
  my-timer        ;; the phase for the intersection.  -1 for non-intersection patches.
  num-north-cars
  num-west-cars
]



;;;;;;;;;;;;;;;;;;;;;;
;; Setup Procedures ;;
;;;;;;;;;;;;;;;;;;;;;;

;; Initialize the display by giving the global and patch variables initial values.
;; Create num-cars of turtles if there are enough road patches for one turtle to
;; be created per road patch. Set up the plots.
to setup
  clear-all
  setup-globals

  ;; First we ask the patches to draw themselves and set up a few variables
  setup-patches

  set-default-shape turtles "car"

  if (num-cars > count roads)
  [
    user-message (word "There are too many cars for the amount of "
                       "road.  Either increase the amount of roads "
                       "by increasing the GRID-SIZE-X or "
                       "GRID-SIZE-Y sliders, or decrease the "
                       "number of cars by lowering the NUMBER slider.\n"
```

```
                    "The setup has stopped.")
    stop
  ]

  ;; Now create the turtles and have each created turtle call the functions setup-cars and
set-car-color
  crt num-cars
  [
    setup-cars
    set-car-color
    record-data
  ]

  ;; give the turtles an initial speed
  ask turtles [ set-car-speed ]

  reset-ticks
end

;; Initialize the global variables to appropriate values
to setup-globals
  set num-cars-stopped 0
  set grid-x-inc world-width / ( grid-size-x )
  set grid-y-inc world-height / ( grid-size-y )

  ;; sets max vaule for motion detection
  ifelse grid-x-inc >= grid-y-inc
  [
    set max-patch-motion grid-y-inc  / 3
  ]
  [
    set max-patch-motion grid-x-inc  / 3
  ]

  ;; don't make acceleration 0.1 since we could get a rounding error and end up on a patch
boundary
  set acceleration 0.099
end

;; Make the patches have appropriate colors, set up the roads and intersections agentsets,
;; and initialize the traffic lights to one setting
to setup-patches
  ;; initialize the patch-owned variables and color the patches to a base-color
```

```
  ask patches
  [
    set intersection? false

    set green-light-up? true
    set my-row -1
    set my-column -1
    set my-timer -1
    set pcolor brown + 3
  ]

  ;; initialize the global variables that hold patch agentsets
  set roads patches with
    [(floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0) or
    (floor((pycor + max-pycor) mod grid-y-inc) = 0)]
  set intersections roads with
    [(floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0) and
    (floor((pycor + max-pycor) mod grid-y-inc) = 0)]

  ask roads [ set pcolor white ]
  setup-intersections
end

;; Give the intersections appropriate values for the intersection?, my-row, and my-column
;; patch variables.  Make all the traffic lights start off so that the lights are red
;; horizontally and green vertically.
to setup-intersections
  ask intersections
  [
    set intersection? true
    set green-light-up? true
    set my-timer 0

    set my-row floor((pycor + max-pycor)  / grid-y-inc)
    set my-column floor((pxcor + max-pxcor) / grid-x-inc)
    set-signal-colors
  ]
end

;; Initialize the turtle variables to appropriate values and place the turtle on an empty road
patch.
to setup-cars  ;; turtle procedure
  set speed 0
```

```
  set wait-time 0
  put-on-empty-road
  ifelse intersection?
  [
    ifelse random 2 = 0
    [ set up-car? true ]
    [ set up-car? false ]
  ]
  [
    ; if the turtle is on a vertical road (rather than a horizontal one)
    ifelse (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0)
    [ set up-car? true ]
    [ set up-car? false ]
  ]
  ifelse up-car?
  [ set heading 180 ]
  [ set heading 90 ]
end

;; Find a road patch without any turtles on it and place the turtle there.
to put-on-empty-road  ;; turtle procedure
  move-to one-of roads with [not any? turtles-on self]
end



;;;;;;;;;;;;;;;;;;;;;;;;
;; Runtime Procedures ;;
;;;;;;;;;;;;;;;;;;;;;;;;

;; Run the simulation
to go


  update-intersections-timer



  if light-system = "timer"
  [
    ;; have the intersections change their color based on timing
    set-signals-timer
    set num-cars-stopped 0
  ]
```

```
  if light-system = "motion"
  [
    ;; have the intersections change their color based on motion
    set-signals-motion
    set num-cars-stopped 0
  ]

if light-system = "detector"
  [
    ;; have the intersections change their color based on motion
    set-signals-detector
    set num-cars-stopped 0
  ]


if light-system = "stop-sign"
  [
    ;; have the intersections change their color based on motion
    set-signals-stop-sign
    set num-cars-stopped 0
  ]

  ;; set the turtles speed for this time thru the procedure, move them forward their speed,
  ;; record data for plotting, and set the color of the turtles to an appropriate color
  ;; based on their speed
  ask turtles
  [
    set-car-speed
    fd speed
    record-data
    set-car-color
  ]

  let current-gas-wasted ( num-cars-stopped * 0.0025 ) /  30 ;; stop for 1min = .0025
gallons of gas wasted
  set wasted-fuel wasted-fuel + current-gas-wasted ;; in gallons


  let current-poll ( num-cars-stopped * 0.0625 ) / 30 ;; stop for 1min = 1oz of CO2 and
1oz = .0625 pounds
  set extra-CO2-emissions extra-CO2-emissions + current-poll
```

```
  set temp-mean-speed-value temp-mean-speed-value + mean [speed] of turtles
  set mean-mean-speed-of-turtles ( temp-mean-speed-value / ( ticks + 1 ) )

   set temp-wait-time-of-turtles temp-wait-time-of-turtles + mean [wait-time] of turtles
  set mean-wait-time-of-turtles ( temp-wait-time-of-turtles / ( ticks + 1 ) )



  ;; update the phase and the global clock
  tick
end

;; have the traffic lights change color if phase equals each intersections' my-phase
to set-signals-timer
  ask intersections
  [
    if my-timer >= ticks-per-cycle
    [
      set green-light-up? (not green-light-up?)
      set-signal-colors
    ]
  ]
end

;; This procedure checks the variable green-light-up? at each intersection and sets the
;; traffic lights to have the green light up or the green light to the left.
to set-signal-colors  ;; intersection (patch) procedure
  ifelse green-light-up?
    [
      ask patch-at -1 0 [ set pcolor red ]
      ask patch-at 0 1 [ set pcolor green ]
    ]
    [
      ask patch-at -1 0 [ set pcolor green ]
      ask patch-at 0 1 [ set pcolor red ]
    ]
end

;; set the turtles' speed based on whether they are at a red traffic light or the speed of the
;; turtle (if any) on the patch in front of them
to set-car-speed  ;; turtle procedure
  if pcolor = ( white - 1 )
```

```
  [ set speed 0.0000001 ]
  ifelse pcolor = red
  [ set speed 0 ]
  [
    ifelse up-car?
    [ set-speed 0 -1 ]
    [ set-speed 1 0 ]
  ]
end

;; set the speed variable of the car to an appropriate value (not exceeding the
;; speed limit) based on whether there are cars on the patch in front of the car
to set-speed [ delta-x delta-y ]  ;; turtle procedure
  ;; get the turtles on the patch in front of the turtle
  let turtles-ahead turtles-at delta-x delta-y

  ;; if there are turtles in front of the turtle, slow down
  ;; otherwise, speed up
  ifelse any? turtles-ahead
  [
    ifelse any? (turtles-ahead with [ up-car? != [up-car?] of myself ])
    [
      set speed 0
    ]
    [
      set speed [speed] of one-of turtles-ahead
      slow-down
    ]
  ]
  [ speed-up ]
end

;; decrease the speed of the turtle
to slow-down  ;; turtle procedure
  ifelse speed <= 0  ;;if speed < 0
  [ set speed 0 ]
  [ set speed speed - acceleration - random-float .05 ]

end

;; increase the speed of the turtle
to speed-up  ;; turtle procedure
  ifelse speed > speed-limit  ;; slows down car if traveling faster than speed limit
```

```
  [ set speed speed-limit ]
  [ set speed speed + acceleration + random-float .03 ]
end

;; set the color of the turtle to a different color based on how fast the turtle is moving
to set-car-color  ;; turtle procedure
  ifelse speed < (speed-limit / 2)
  [ set color blue ]
  [ set color cyan - 2 ]
end

;; keep track of the number of stopped turtles and the amount of time a turtle has been
stopped
;; if its speed is 0
to record-data  ;; turtle procedure
  ifelse speed = 0
  [
    set num-cars-stopped num-cars-stopped + 1
    set wait-time wait-time + 1
  ]
  [ set wait-time 0 ]
end

;; have the traffic lights change color based on motion of cars
to set-signals-motion
  update-intersections-motion
  ask intersections
  [
     if num-north-cars > num-west-cars and green-light-up? = false
    [
      set green-light-up? (not green-light-up?)
      set-signal-colors
    ]

    if num-north-cars < num-west-cars and green-light-up? = true
    [
      set green-light-up? (not green-light-up?)
      set-signal-colors
    ]
      ]
end
```

```
;; have the traffic lights change color based on cars stopped at red light
to set-signals-detector
  update-intersections-detector
  ask intersections
  [

     if num-north-cars > 0 and ( ticks-per-cycle / 2 ) > my-timer and green-light-up? =
false
    [
      set green-light-up? (not green-light-up?)
      set-signal-colors
      set my-timer  ( ticks-per-cycle / 2 + 1 )
    ]

    if num-west-cars  > 0 and ( ticks-per-cycle / 2 ) > my-timer and green-light-up? = true
    [
      set green-light-up? (not green-light-up?)
      set-signal-colors
      set my-timer ( ticks-per-cycle / 2 + 1 )
    ]
  ]
end




;; have cars take turns at intersections (stop signs)
to set-signals-stop-sign
  update-intersections-stop-sign
  ask intersections
  [
    ask patch-at -1 0 [ set pcolor ( white - 1 ) ]
    ask patch-at 0 1 [ set pcolor ( white - 1 ) ]

     if num-north-cars > 0
    [
      set green-light-up? (not green-light-up?)
    ]

    if num-west-cars  > 0
    [
      set green-light-up? (not green-light-up?)
```

```
    ]
  ]
end



to update-intersections-timer
  ask intersections
  [
    if my-timer >= ticks-per-cycle
    [
     set my-timer 0
    ]

    set my-timer my-timer + 1
  ]
end

to update-intersections-motion
  ask intersections
  [
    let my-pxcor pxcor
    let my-pycor pycor
    set num-north-cars count turtles in-radius motion-detection with [ heading = 180 and
ycor >= my-pycor ]
    set num-west-cars count turtles in-radius motion-detection with [ heading = 90 and
xcor <= my-pxcor ]
     ;let closest-car min-one-of turtles [distance myself]

  ]
end

to update-intersections-detector
  ask intersections
  [
    let my-pxcor pxcor
    let my-pycor pycor
    set num-north-cars count turtles in-radius 1.6 with [ heading = 180 and ycor > my-
pycor and speed <= 0.01 ]
    set num-west-cars count turtles in-radius  1.6 with [ heading = 90 and xcor < my-pxcor
and speed <= 0.01 ]
     ;let closest-car min-one-of turtles [distance myself]
```

```
  ]
end


to update-intersections-stop-sign
  ask intersections
  [
    let my-pxcor pxcor
    let my-pycor pycor
    set num-north-cars count turtles in-radius 1.6 with [ heading = 180 and ycor > my-
pycor and speed <= 0.001 ]
    set num-west-cars count turtles in-radius  1.6 with [ heading = 90 and xcor < my-pxcor
and speed <= 0.001 ]
     ;let closest-car min-one-of turtles [distance myself]
  ]
end


; Copyright 2003 Uri Wilensky.
; See Info tab for full copyright and license.
; NOTE - used within the agreement specified by Wilensky (Edington 10NOV14)
```