Simulation of North-South and East- West Traffic Lights Using Arduino Uno and atmega328p

Tom Dion
Western New England University
Springfield MA 01119
td591325@wne.edu

Abstract: In today's world, traffic is a major part of everyday life that we cannot avoid. Traffic is often controlled by some sort of device and in many instances it is a traffic light. This work focuses on the simulation of two traffic lights, one controlling north-south traffic and the other controlling east-west traffic. The simulation was implemented using the atmega328p microprocessor connected to an Arduino Uno microcontroller which we connected to a breadboard. The programming platform we used for the simulation was done using the Assembly programming language. The bread board consisted of light emitting diodes (LEDs) and multiple wires.

I. Introduction

Having efficient and safe traffic flow is essential in today's world and the implementation of traffic lights is a way to improve that flow. Traffic lights are a way to control traffic through the entire day and year effortlessly by the use of electronics.

Traffic becomes an issue in dense areas with many roadways and more often than not, traffic flow is controlled by a traffic light. Traffic lights can be used multiple ways such as being hung across an intersection or simply a pole with a traffic light on the top of it.

There are three standard colors used in traffic lights, green, yellow, and red. Each of these colors tells the operator of a vehicle how they should operate their vehicle.

1. A green light implies that they should continue on their current path and that it is safe to go through the intersection because the flow to the left and right is stopped.

2. A yellow light indicates that the light is about to become red. Meaning the operator should slow and stop their vehicle at the intersection.

3. A red light indicates that that operator should have their vehicle come to a complete stop at the intersection.

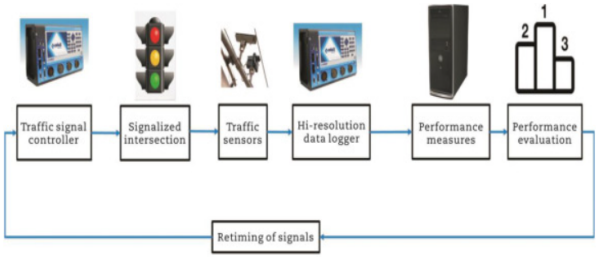
Overall, traffic lights are an effective wait to control traffic to create an efficient and safe flow. Furthermore they are simple to implement and reliable in many conditions throughout all seasons.

II. Related Work

Source [1] covers how The New York City department of transportation (NYC DOT) decides whether or not to implement a traffic

light In New York City there are 13,543 intersections that are controlled by traffic lights as of March 2022. 2,862 in Manhattan, 1,768 in the Bronx, 4,848 in Brooklyn, 3,432 in Queens and 633 in Staten Island. NYC DOT uses a process called an intersection control study to decide if a traffic signal or multi-way stop signs are needed or not. This study includes NYC DOT inspectors to check all agency records such as sign orders, pavement marking orders, and school maps for the location. NYC DOT inspectors also do a field investigation to create a condition diagram of the location. What this diagram does is show the street and sidewalk widths, location geometry, street directions, location and conditions of NYC DOT signs and markings, land use, street furniture, distance to the nearest traffic control device, and other necessary information. NYC DOT also does a Field Observation Report which includes information on drivers' compliance with existing controls, geometric or sight distance issues, and violations of the speed limit. Inspectors must also do manual counts of the number of vehicles and pedestrians in that intersection/area. They often do this in the morning/evening rushes. They also do this during non-peak hours.

Source [2] focuses on the performance of traffic signals. It mentions that each state or local transportation agency in the United States uses citizens' complaints and occasional checkups when checking if a traffic light's timing needs to be adjusted. It concludes that these methods are slow and inefficient. It suggests that agencies should implement a signal performance evaluation framework that will observe the effects of the retiming efforts. This framework requires an immense amount of data that will check the conditions on the roads. These evaluations will prioritize which traffic signals need retiming which will create a feedback loop of improvement of a traffic control system. Figure 1 shows the process flow of traffic signal performance evaluation.



III. Methodology

In this work we created a simulation of a traffic light system that controls traffic flow for north-south traffic flow and east to west traffic flow. We programmed an atmega328p microprocessor using assembly programming language which was integrated into an Arduino Uno microcontroller used to send electrical signals to a breadboard where we replicated two traffic signals that would work in opposite parity.

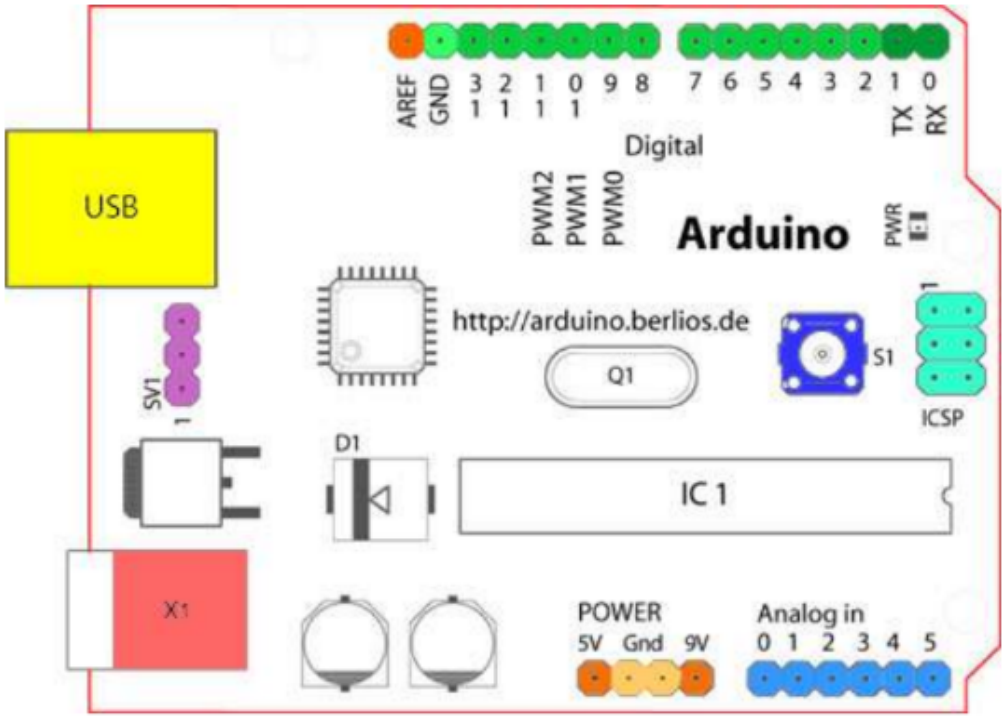


Figure 2: Diagram of an Arduino Uno Board
Source: [3]

Starting clockwise from the top center:

- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (digitalRead and digitalWrite) if you are also using serial communication (e.g. Serial.begin).
  Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green) Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

In addition to the Arduino Board, other tools we used were:

- 2 green, 2, yellow, and 2 red LEDs
- A breadboard
- Connecting wires

IV. System Design

The idea of this was to produce a specification that will make it possible for the controller to maintain precise new system implementation. The concept is based on simulating a functional north-south and east-west traffic control system using the Arduino Uno microcontroller.

IV. I. Input Design

Input to the system is done by sending signals to the Arduino which would be processed by the atmega328p microprocessor. To replicate each light we used connecting wires from port B of the Arduino to the breadboard where they were connected to each LED ordered green, yellow, then red from left to right. Each LED was connected to the ground of the Arduino.

IV. II. Simulation of the Traffic Lights

To simulate a traffic light we created north-south (NS) and east-west (EW) lights where NS green be on for 50 seconds while EW is red for 50 seconds. NS would then turn yellow for 10 seconds while EW stayed red. Then EW would turn green and NS would turn red. EW would turn yellow for 10 seconds while NS stayed red then would loop.

We used Atmel Studio 6.2 to create assembly code to simulate the timing of this traffic light and control which LEDs would light up. See Appendix A for code.

IV. III Results

The simulation successfully replicated the timing and functionality of a traffic light system for both north-south and east-west traffic. The LEDs on the breadboard accurately represented the transitions between green, yellow, and red lights, mimicking the control of traffic flow. The atmega328p microprocessor, programmed using assembly language, effectively processed the signals from the Arduino Uno and displayed the desired light patterns.

V. Conclusion

Overall, this work demonstrated the successful simulation of a simplified traffic light system using an atmega328p microprocessor and Arduino Uno. The study contributed to the understanding of traffic light systems and an approach using microprocessors and micro controllers. This work also shows the importance of electronic control for efficient traffic control. Using the assembly programming language and the Arduino Uno represents the importance and use of microprocessors and microcontrollers in the real world.

References

[1] https://www.nyc.gov/html/dot/html/infrastructure/signals.shtml

[2] https://www.sciencedirect.com/science/article/pii/S2095756422000605#fig1

[3] https://docs.arduino.cc/tutorials/uno-rev3/intro-to-board

APPENDIX

A.

```
.include "m328pdef.inc"
.org 0
rjmp main


main:
ldi r16, low(RAMEND) ; set up stack com3
out spl,r16
ldi r16, high (RAMEND)
out sph,r16
ldi r16, 0xFF ;
out DDRB,r16 ; I/O mapped (see
m328pdef.inc)

L1: ldi r16, 0X21 ; ;North-South Green for
50 seconds, East-West Red for 50
Seconds, uses 00100001 in hex
out portB,r16
rcall mydelay

ldi r16,0x22 ; North-South Yellow for 10
seconds, East-West Red for 10 seconds,
uses 00100010 in hex
out portB,r16
rcall mydelay

ldi r16,0X0C ; North-South Red for 50
seconds, East-West Green for 50 seconds,
uses 00001100 in hex

out portB,r16
rcall mydelay

ldi r16,0X14 ; North-South Red for 10
seconds, East-West Yellow for 10 seconds,
uses 00010100 in hex
out portB,r16
rcall mydelay
rjmp L1

mydelay: ;Triple nested loop creates 1
second delay
ldi r17,0x52 ;; outer loop
L2: ldi r18,0xFF ;; inner loop
L3: ldi r19,0xFF ;; inner most loop
L4: dec r19
brne L4
L5: dec r18
brne L3
dec r17
brne L2
ret


shortdelay: ;calls mydelay ten times create
ten seconds of delay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
rcall mydelay
ret


longdelay: ;calls shortdelay five times to
create 50 seconds of delay
rcall shortdelay
rcall shortdelay
rcall shortdelay
rcall shortdelay
rcall shortdelay
ret
```