Drew Borneman

Jim Patrizi

EECS 303

Group 4

Assignment 5 Report

**I. Assignment**

The purpose of this lab is to implement a traffic controller for use in a one way intersection with and without pedestrian crossings. The purpose of this assignment is to :

1. Describe the traffic controller functionality using well documented flow diagrams.

2. Design and code arduino sketches (codings) to implement the controller.

3. Verify, test and demo your design.

4. Your assignment report should provide a brief but complete narrative of your traffic controller design.

and

1. Validate intuitively (not formally) that the above pedestrian crossing rules are safe.

2. Describe the modified controller by timing diagrams which include traffic on streets A and B, and the pedestrian crossings.

3. Modify your Arduino sketches (programs) to implement the upgraded traffic controller

4. Verify and test your design.

This was all done on the Arduino Rapid Prototyping Platform with the Arduino IDE.

**II. Materials**

**2.1. Hardware**

To interface with the Arduino, we used a Raspberry Pi model 3. Other materials include a breadboard, wires, and LEDs for the assignment.

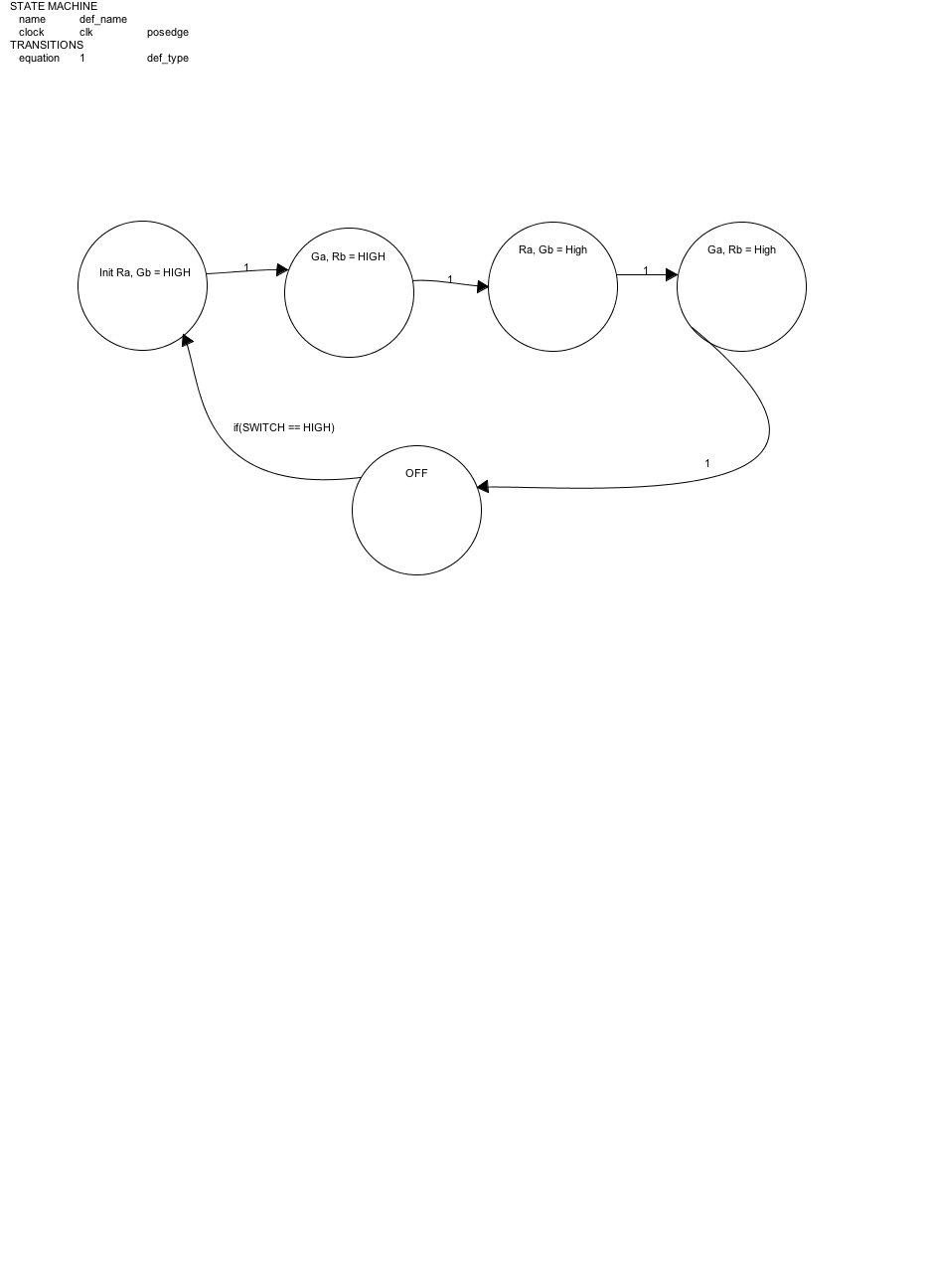
**2.2. Software**

The Raspberry Pi we used was running the Raspbian distro of Linux, specially created for the various Pi models. We did our coding on the Pi using the included Arduino IDE, and compiled our code with the the included compiler. We also used Github with our project for easy collaboration and version control.

**III. Methods**

**3.1. Part 1**

*Flow Diagram and Narrative of design*



Basically we have 5 separate states that our loop goes through in order to implement part 1 of this lab. This can be seen in the flow diagram above. We have or LED port declarations as Ra=12,Ga=9,Rb=11,Gb=10. Each of these states represent our code in our loop of the program. Each of these pins are setup as output on the pwm capable pins of the arduino uno. We have maintenance signal that is implemented with buttonPin=7, when this is high, we kick off the state machine. As long as the pin is asserted, the state machine runs. Otherwise, the lights will safely turn off after the cycle has completed.

//Init Ra, Gb = HIGH

digitalWrite(Ra, HIGH);

analogWrite(Gb, brightnessHigh);

delay(5000);

analogWrite(Gb, brightnessLow);

delay(1000);

analogWrite(Gb, brightnessOff);

//Ga, Rb = HIGH

digitalWrite(Rb, HIGH);

delay(1000);

digitalWrite(Ra, LOW);

analogWrite(Ga, brightnessHigh);

delay(5000);

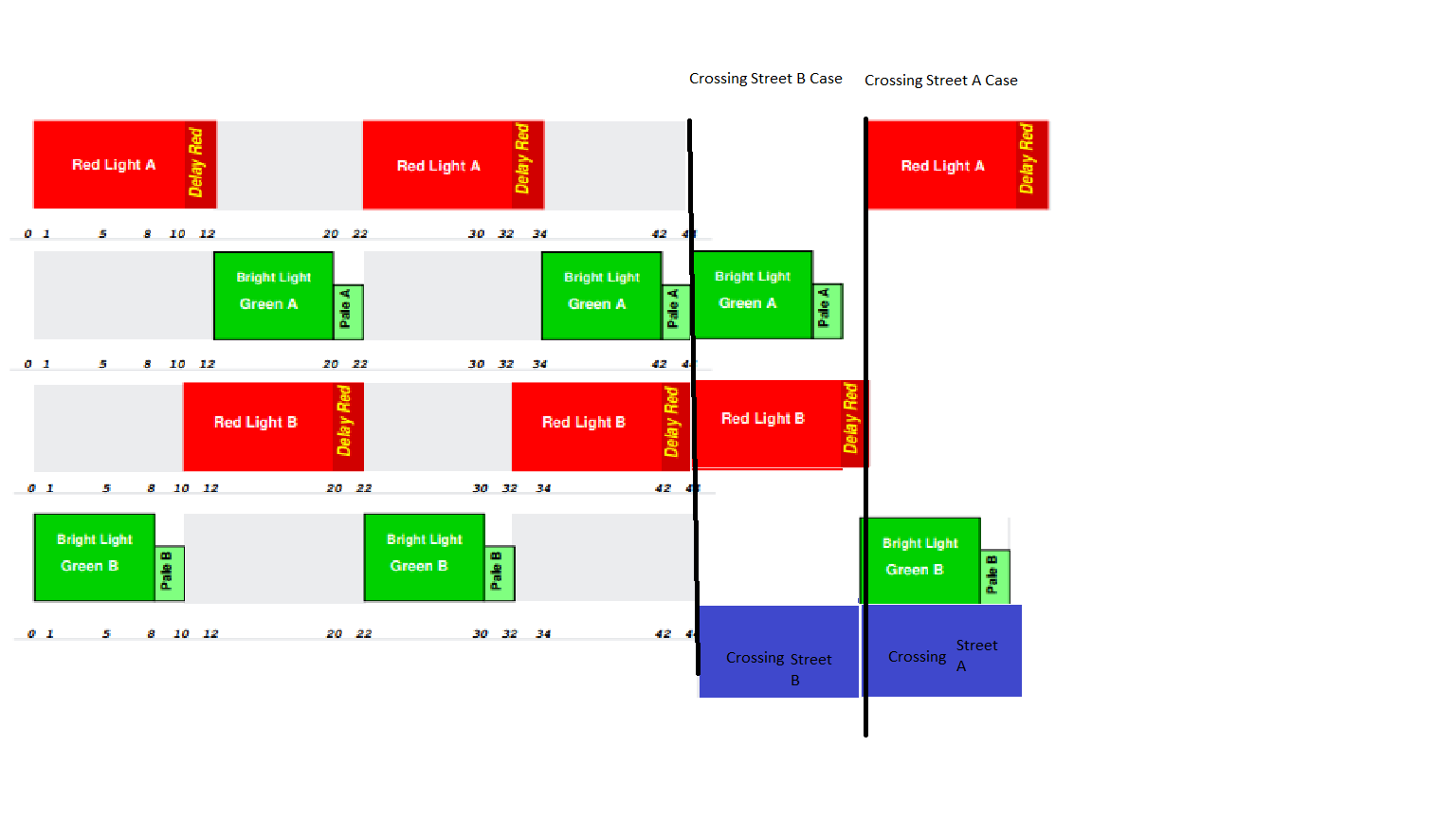
analogWrite(Ga, brightnessLow);

delay(1000);

analogWrite(Ga, brightnessOff);

This is the main logic in the program to implement the timing diagram included with the assignment. **Note** the provided timing diagram was wrong for the red lights. The red lights must have a delay while **both** lights are red from the transition to dim green to red. This is a delay of 10 seconds when both are red but this is **not encapsulated by the diagram. According to the diagram, both red lights are not lit at the same time, but by the instructions listed they should both be on the same delay and lit at the same time.** The basic premise is to light the lights appropriately for the given delay specs. We have the time reduced to accommodate testing and demoing purposes, as this would not be a good traffic control system with the current delays. analogWrite is used to write a value from 0-255 on the pwm pin. 255 means full 100% duty cycle is used. We designated brightnessHigh=255 and brightnessLow=70 in order to get a full power green vs dim green as described in the specs.

**3.1. Part 2 - Pedestrian input**



**IV. Results**

Successfully, we were able to blink the LEDs to implement a traffic control system with the given specs. The Arduino rapid prototyping platform is a great way to test out a general traffic controller.

**V. Comments, Remarks, Conclusions**

~~Assembly is really, really difficult. We can obviously see why higher level languages like C/C++/Java were introduced to abstract the details away from the programmer. Programming is a lot slower this way too. We successfully were able to blink our LED in part 1 for the demo and dim the LED with PWM for part 2, although since this was accomplished using C function calls it did not have the usual assembly benefits of fast, small executables.~~

**VI. Appendix**

Included in this submission are the following files:

* Lab 5 Report.doc - this file
* traffic\_controllerv2.ino- source code for part 1
* README.txt - compilation and run instructions(needs done)