Christopher Chung A11404827
Will An A11522644
Physics 124
March 24th, 2017

Project Report

Content:

Proposal:

Motivated to design a project that encapsulates autonomous driving and hardware design, as a team we envisioned a car that would use a sensor to traverse through a smart maze. At the time, the type of sensor was still being tested. We were split between using IR sensors to determining distance between openings, or RGB visible light sensors to sense a pathway lit through LEDs. As the project developed we realized that IR sensors were sensitive enough, however the smart maze would be too difficult to produce in order for favorable results. By process of elimination, we went with the RGB visible light sensors and determined that sensing light, whether it be produced from LEDs, would satisfy our goals.

All-in-all, we are creating a car on a Zumo frame that will use an RGB sensor to traverse a smart maze lit through LEDs or other form of light.

Sensor:

The sensor we are using is the SparkFun RGB and Gesture Sensor APDS 9 SEN-12787. In order to make the car a bit more interesting we will be using a servo motor that will have the sensor attached to it, taking it from 0 degrees to 180 degrees (essentially left to right). This will be the “head” of the car reading one side and comparing the light values to the opposite side. The sensor itself is highly sensitive and it is able to determine different wavelengths of light. Due to the high ambient lighting we have in the physics lab, we have gone through several trials of different light sources. The source that is the most promising is a mini incandescent light bulb, although it is not as efficient as an LED it emits a higher intensity in light that can be differentiated from the ambient light.

In order to decipher when to turn into the open gap in each row, we are using a photoresistor to determine when the car is on top of the lightbulb. After receiving a high visible light measurement, the resistance of the photoresistor drops to give a high analog signal. This signal is used to determine when the car should turn to proceed through the gap into the next row.

Implementation:

a. The Car

The car rests on a Zumo kit car chassis using two stepped down motors to produce more torque than speed. This cart is front wheel drive, however it cart turns into a tank-like vehicle utilizing a belt connected to both rear and front wheels. In order to be mobile, the Zumo kit uses 4 double A batteries producing about 6V in series to power the Arduino. The Arduino Uno is attached to a motor shield, which links up to the two DC motors and one servo motor to control the sensor. The code below specifies exactly what the car is doing, but as a quick reference. The car is entering the starting row, sensing both sides (left and right) as it turns its “head” on the servo motor, determines which side is brighter turns to that side, traverses the row till the light is sensed by the photoresistor, then going upwards to the next row.

```
#include <AFMotor.h>
#include <Servo.h>
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
Servo hitec;
AF_DCMotor leftmotor(4);
AF_DCMotor rightmotor(3);
int const degree = 90;
const int SPEED = 255;
const int SenseLight = A0;
//const int balance =  //sensitivity of the sensor readings(threshhold to keep the sensors equal or close to equal)
const int ontop = 20; //sensitivity of the sensor reading while on top of the LED
int u;
int j;
int Le;
int Ri;
int left;
int right;
uint16_t leftlight;
uint16_t rightlight;
uint16_t green_light;
uint16_t blue_light;
uint16_t ambient_light;
const int row0 = 10;
const int row1 = 11;
const int row2 = 12;
const int clockA = 8;  //these may have to be swapped
const int clockB = 13;
const int mazeReader = A3;
```

```
const int readyLight = 2;
int memory[3];

void clk_sig() {
    digitalWrite(clockA, LOW);
    delay(50);
    digitalWrite(clockA, HIGH);
    delay(50);
    digitalWrite(clockA, LOW);
    delay(50);

    digitalWrite(clockB, LOW);
    delay(50);
    digitalWrite(clockB, HIGH);
    delay(50);
    digitalWrite(clockB, LOW);
    delay(50);
}

void flush_buffer() {
  digitalWrite(row0, LOW);
  digitalWrite(row1, LOW);
  digitalWrite(row2, LOW);
  for (int i=0; i!=4; ++i) {
    clk_sig();
    delay(300);
  }
}
void test() {
  digitalWrite(row0, HIGH);
  digitalWrite(row1, HIGH);
  digitalWrite(row2, HIGH);
  clk_sig();
  delay(300);
  flush_buffer();
}

void lightShow_1() {
  digitalWrite(row0, HIGH);
  digitalWrite(row1, LOW);
  digitalWrite(row2, LOW);
  clk_sig();
  delay(300);
```

```
  digitalWrite(row0, LOW);
  digitalWrite(row1, HIGH);
  digitalWrite(row2, LOW);
  clk_sig();
  delay(300);

  digitalWrite(row0, LOW);
  digitalWrite(row1, LOW);
  digitalWrite(row2, HIGH);
  clk_sig();
  delay(300);

  digitalWrite(row0, LOW);
  digitalWrite(row1, HIGH);
  digitalWrite(row2, LOW);
  clk_sig();
  delay(300);
}

void lightShow_2() {
  digitalWrite(row0, HIGH);
  digitalWrite(row1, LOW);
  digitalWrite(row2, HIGH);
  clk_sig();
  delay(300);

  digitalWrite(row0, LOW);
  digitalWrite(row1, HIGH);
  digitalWrite(row2, LOW);
  clk_sig();
  delay(300);
}

void lightShow_3() {
  digitalWrite(row0, HIGH);
  digitalWrite(row1, LOW);
  digitalWrite(row2, HIGH);
  clk_sig();
  delay(300);

  digitalWrite(row0, HIGH);
  digitalWrite(row1, HIGH);
```

```
  digitalWrite(row2, LOW);
  clk_sig();
  delay(300);

  digitalWrite(row0, LOW);
  digitalWrite(row1, HIGH);
  digitalWrite(row2, HIGH);
  clk_sig();
  delay(300);
}
void read_row(int row) {
  digitalWrite(row+10, HIGH);
  delay(100);
  int readVal = analogRead(mazeReader);
  Serial.println(readVal);
  if (readVal>40 && readVal<50)                    // adjust
    memory[row] = 0;
  else if (readVal>85 && readVal<=95)
    memory[row] = 1;
  else if (readVal>100 && readVal<=113)
    memory[row] = 2;
  else if(readVal>=114 && readVal<=140)
    memory[row] = 3;
    //Serial.println(memory[row]);
  digitalWrite(row+10, LOW);
  delay(100);
}

void maze_read() {
  for (int i=10; i!=13; ++i) {
    read_row(i-10);
    delay(100);
  }
}

void light_maze() {
  for (int t=0; t!=4; ++t) {
    for (int r=0; r!=3; ++r) {
      if (t==(3-memory[r]))
        digitalWrite(r+10, HIGH);
    }
    delay(250);
    clk_sig();
```

```
    for (int r=0; r!=3; ++r)
      digitalWrite(r+10,LOW);
    delay(250);
  }
}

void NextRow(){
  leftmotor.run(FORWARD);
  rightmotor.run(FORWARD);
  delay(2600); // how long does it take to travel to the next row, still have to measure
  leftmotor.run(RELEASE);
  rightmotor.run(RELEASE);
}
void TurnLeft(){
delay(100);
leftmotor.run(BACKWARD);
rightmotor.run(FORWARD);
delay(2350);
leftmotor.run(RELEASE);
rightmotor.run(RELEASE);
delay(200);
}
void TurnRight(){
delay(100);
leftmotor.run(FORWARD);
rightmotor.run(BACKWARD);
delay(2350);
leftmotor.run(RELEASE);
rightmotor.run(RELEASE);
delay(200);
}

void LookRight(){
  hitec.write(90);
  delay(1000);
  hitec.write(0);
  delay(1000);
   }
void LookLeft(){
    hitec.write(180);
    delay(1000);
    }
```

```
void setup() {
   leftmotor.setSpeed(SPEED);
   rightmotor.setSpeed(SPEED);
  pinMode(row0, OUTPUT);
  pinMode(row1, OUTPUT);
  pinMode(row2, OUTPUT);
  pinMode(clockA, OUTPUT);
  pinMode(clockB, OUTPUT);
  pinMode(readyLight, OUTPUT);
  pinMode(mazeReader, INPUT);
  Serial.begin(9600);
  pinMode(SenseLight,INPUT);
  hitec.attach(9,500,2400);

  Serial.println();
  Serial.println(F("--------------------------------"));
  Serial.println(F("SparkFun APDS-9960 - ColorSensor"));
  Serial.println(F("--------------------------------"));
   if ( apds.init() ) {
   Serial.println(F("APDS-9960 initialization complete"));
  } else {
   Serial.println(F("Something went wrong during APDS-9960 init!"));
  }

  // Start running the APDS-9960 light sensor (no interrupts)
  if ( apds.enableLightSensor(false) ) {
   Serial.println(F("Light sensor is now running"));
  } else {
   Serial.println(F("Something went wrong during light sensor init!"));
  }

  // Wait for initialization and calibration to finish
  delay(200);

}
void start(){
  flush_buffer();
  test();
  lightShow_1();
  lightShow_1();
```

```
  lightShow_2();
  lightShow_2();

  flush_buffer();
  maze_read();
  light_maze();
  Serial.println(j);
  while(analogRead(mazeReader)<10) {
    digitalWrite(readyLight, HIGH);
  }
  digitalWrite(readyLight, LOW);
  delay(5000);
}

void loop() {
if (u == 0){
  start();
}
    Serial.println("start");
    delay(200);
  if (Le==0 && Ri==0){
   LookRight();
   delay(200);
      if (  !apds.readRedLight(rightlight)||
      !apds.readAmbientLight(ambient_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ){
    Serial.println("Error reading light values");}
    else{
      Serial.print("Right Light value: ");
      Serial.println(rightlight);
      delay(200);
    }
   LookLeft();
   delay(200);
        if (  !apds.readRedLight(leftlight) ||
         !apds.readAmbientLight(ambient_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ){
    Serial.println("Error reading light values");}
    else{
      delay(200);
      Serial.print("Left Light value: ");
```

```
    Serial.println(leftlight);
    hitec.write(90);
    delay(200);
  }
 if (rightlight > leftlight){
  leftmotor.run(FORWARD);
  rightmotor.run(FORWARD);
  delay(1300);
  leftmotor.run(RELEASE);
  rightmotor.run(RELEASE);
  delay(200);
  TurnRight();
   Ri++;
  Serial.print("Turned Right");
  Serial.println(" Next turn is left");
 }
 else{
  leftmotor.run(FORWARD);
  rightmotor.run(FORWARD);
  delay(1100);
  leftmotor.run(RELEASE);
  rightmotor.run(RELEASE);
  delay(200);
  TurnLeft();
  Serial.print("Turned Left");
  Serial.println(" Next turn is Right");
 }
}
 int st0p = analogRead(SenseLight);
 Serial.println("Started to move towards light");
 while(st0p < ontop){
  leftmotor.run(FORWARD);
  rightmotor.run(FORWARD);
  st0p = analogRead(SenseLight);
  //Serial.print(st0p);
 }
 Serial.print("problem");
 Serial.println(Le);
 Serial.println(Ri);
 leftmotor.run(RELEASE);
 rightmotor.run(RELEASE);

 if (Ri == 0){
```

```
    Serial.print("Senselight value is above 10: ");
    Serial.println(st0p);
    leftmotor.run(FORWARD);
    rightmotor.run(FORWARD);
    delay(900);
    leftmotor.run(RELEASE);
    rightmotor.run(RELEASE);
    j++;
    u++;
    TurnRight();
    Serial.println("Second turn is Right");
    delay(200);
  }
 else {
    Serial.print("Senselight value is above 10: ");
    Serial.println(st0p);
    leftmotor.run(FORWARD);
    rightmotor.run(FORWARD);
    delay(900);
    leftmotor.run(RELEASE);
    rightmotor.run(RELEASE);
    j++;
    u++;
    Ri--;
    TurnLeft();
    Serial.println("Second turn is Left");
    delay(200);
  }
  NextRow();
  Serial.println("move up to next row");
  Serial.println(j);
  delay(200);
  Serial.println("problem2");
  Serial.println(Le);
  Serial.println(Ri);
  if (j==3){
    leftmotor.run(FORWARD);
    rightmotor.run(FORWARD);
    delay(1500);
    leftmotor.run(RELEASE);
    rightmotor.run(RELEASE);
    TurnLeft();
    TurnLeft();
```

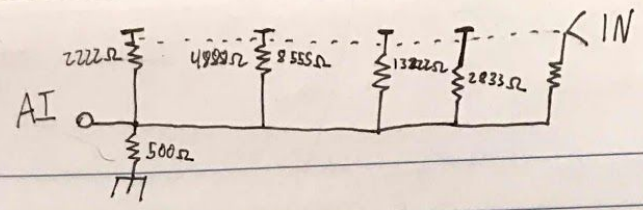
```
 }
}
```

B. The Track

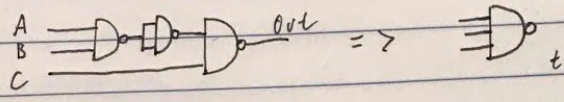
The track consists of three rows of metal posts and light bulbs, that run through an interface circuit that facilitates our desired interaction between maze and car. In each row, a digital output signal is run through the right side, and depending on how many posts are linked by wire, a varying amount of resistance leads to a varying voltage, which is read by an analog input pin on the car, which then stores the associated row opening based on the voltage read. All three rows feed into that the same analog input, so the digital output from the car to row is run one at a time. The car now knows the configuration of the maze.

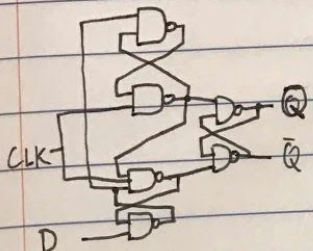
Each row of lights is controlled by a single digital output on the car, as well as a clock signal that is shared by all of the rows. The digital interfacing circuit for each row consists of four, positive edge triggered D-flip-flops. Each row is fed four clock signals, and depending on the value in memory associated with the row, a HIGH signal is put into its data line at the right time to light up the proper light. The maze is now properly lit, and the car is ready to navigate.

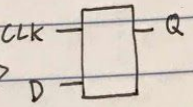
Sensing



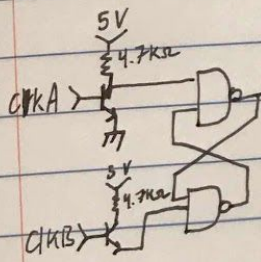
Controlling

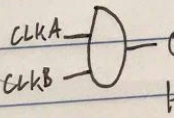






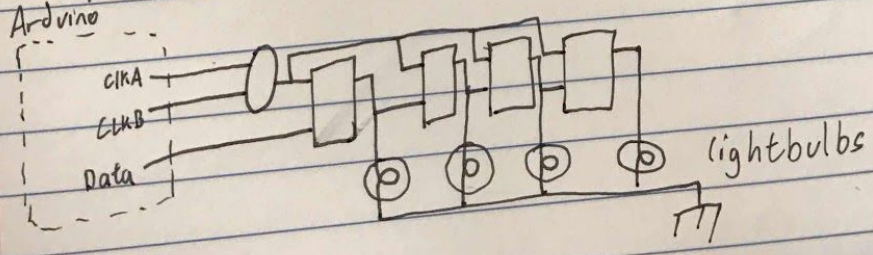
positive-edge triggered
D - Flip - Flop





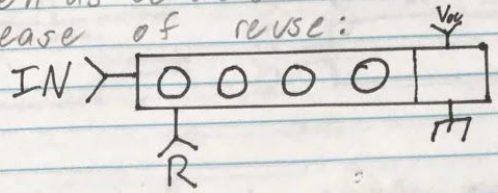
Hardware debounce

High level Diagram for Arduino circuit interface:



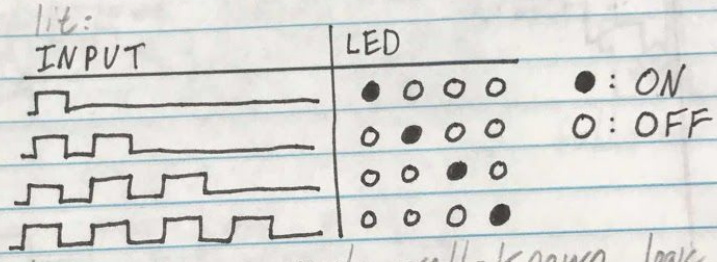
Abstract

Purpose: Be able to control which LED in a series is lit, through one digital input, as well as be able to reset the configuration for ease of reuse:



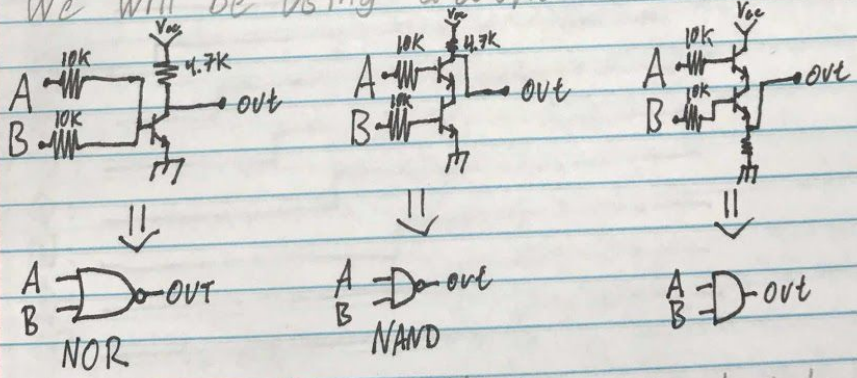
Functional

Our design will work by counting the # of rising edges at the input to determine which LED is lit:



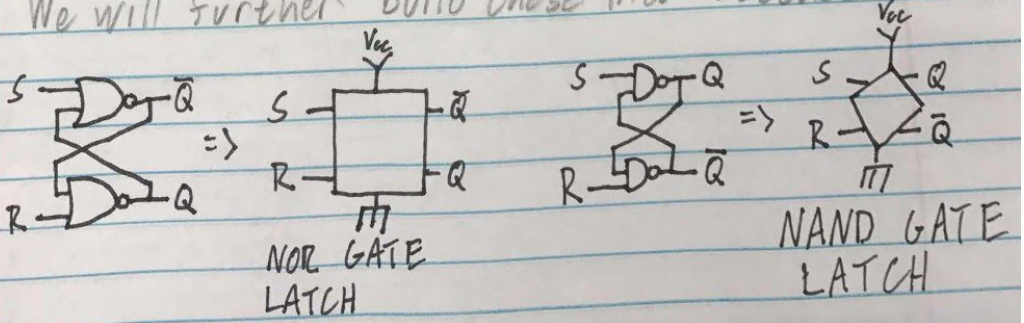
Components 1

We will be using a couple well-known logic gates:



Components 2

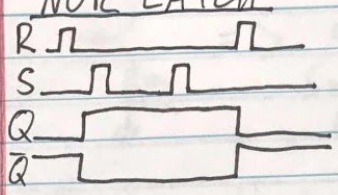
We will further build these into latches:



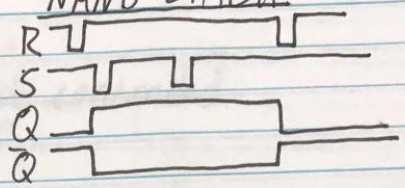
Functional

These latches behave like:

NOR LATCH

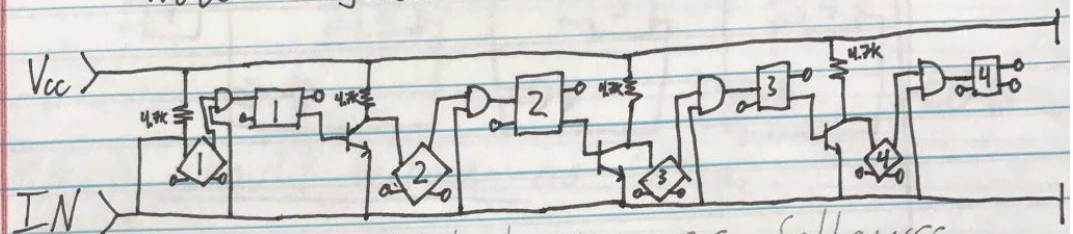


NAND LATCH



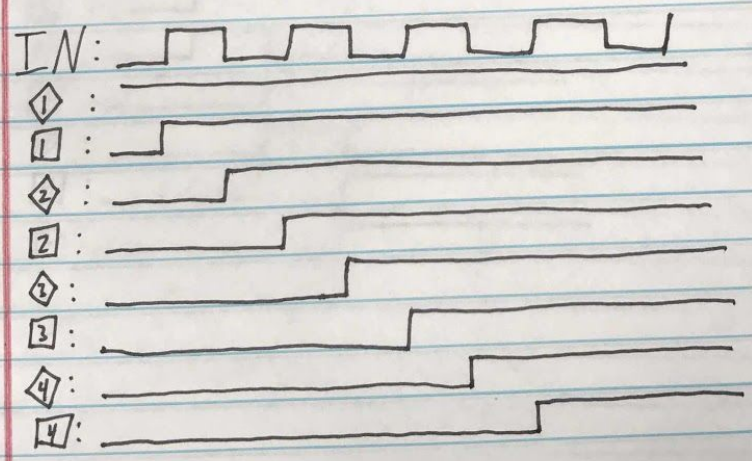
Note: all gates + latches must attach to a V_{cc} + GND

ayer α



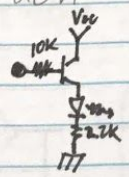
Their Q outputs behaves as follows:

unctional



Components'

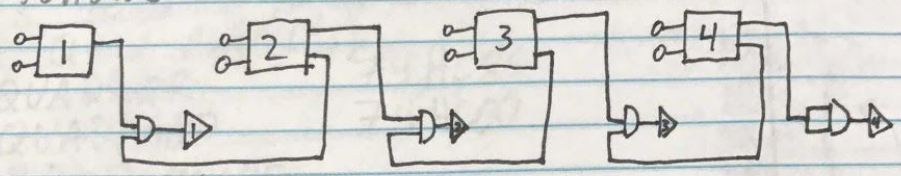
Each LED will be controlled by a switch:



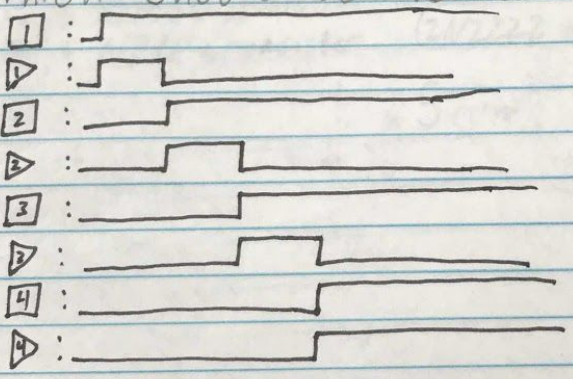
=> Switch controlled LED

Which will be attached to our counter as follows:

Layer β

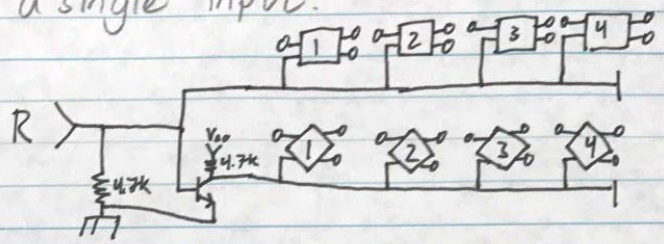


Which should behave as follows:



Layer 8

Finally, we will tie all of the resets to a single input:



Parts required per row (using Quad NOR's and Quad NAND's):

2 × QUAD NOR	74HC86	10 ×
2 × QUAD NAND	74HC00	10 ×
14 × 4.7kΩ resister		70 ×
20 × 10kΩ resister		100 ×
20 × NPN transister	(2N2222 or 2N3904)	100 ×

× 5 rows

Error/Risk:

Like any project a lot of problems came up when actually implementing our design concept. A large problem was the sensitivity and ambient lighting. The problem was that the sensor could not pick up the small amounts of light that the lightbulbs would output. A solution we came up with was to use a black backdrop on each side, filtering out more ambient light. Earlier we mentioned we used mini incandescent lights instead of LED's because they're brighter, however the problem with the lightbulbs was their resistance. Since the lightbulbs are hooked up in series, each light bulb subsequent to the previous one gets dimmer and dimmer. The sensitivity again was the problem. If we could re-do the lab I think working on a schedule would have foreshadowed this problem a lot sooner than it did, giving us time to order brighter LEDs. Another problem was the sizing of the track. The track we created is very tight relative to the vehicle. This creates a problem, because when we try navigating through the track it, it demands a very critical and robust code to accommodate for the tight turns. Again this problem could have been avoided if we started earlier, giving us more time to work with different materials rather than force the situation on a small track.

The track's light control interface required several iterations to get write. During the experimentation phase, I tried many different concepts, including a combinations of analog and digital components. After developing an extremely complex and unreliable circuit, involving the use of capacitors and diodes, I realized that the analog components were introducing unreliability into the design, and made the transition to an all digital circuit. Early attempts at building logic gates out of transistors were extremely frustrating and unfruitful, as they were also unreliable and had a tendency to leak signal. Finally, I decided that all gates had to be built out of quad gate chips, and designed and created the first working iteration (diagram can be seen in the proposal). It involved the use of 5 quad nands and 4 quad nors, the signal also has to be hardware debounced through a latched nand. With a working version at hand, Professor Murphy happened upon what I was creating, and came back with a single chip, a positive edge triggered octal D-flip-flop, functional enough to control two of my rows. After some painful deliberation, I realized it was best to abandon my design rather than build it twice more, and instead recreated four of the D-flip-flops, which required 8 quad nand chips. This very painful first real experience designing a digital circuit has taught me to truly appreciate the software side of things, where things are reliable and behavior is truly consistent.

Conclusion:

Our project was a learning experience for both of us. It gave us insight on the importance of creating a work schedule and also shooting for realistic goals that were also fulfilling.

In the future we recommend that partners create a work schedule that accommodates both partners. Also create a project that both of you are interested in and make clear your motives and direction within the project.