Thomas Nabelek
October 25, 2016

Project #2 Report

Objective and Overview

The objective of the second project for this course was to explore and design an architectural robotics system or component for use by children, taking into consideration their needs. To meet these requirements, my partner Adam and I developed a bedroom environment that would help a child to develop a sense of time progression throughout the day and associate various times of the day with certain activities, and to help a child fall asleep and wake in a soothing manner.

Time is the primary motivator behind the actions that the system takes. Throughout a 24-hour period, the LED lights around the room transition through set colors. These colors always appear at the same time of day and so help a child to be aware of the passage of time. The light configuration could be changed, but as an example, the lights could be yellow in the morning when the child wakes, red around lunch time, cyan when it's time to get ready for bed, and blue at night, with other colors in between.

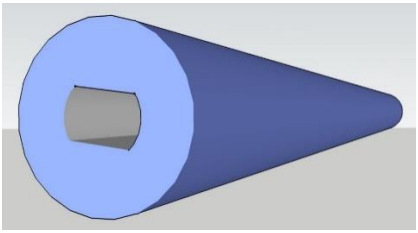
During the setup process, the parent user uses a remote to set up an appropriate number of sleep cycles for their child, e.g. two sleep periods: one for a nap time and one for sleeping at night. When the sleep time for the first sleep period is reached, the lights automatically deactivate, the curtain lowers, and a soothing sound plays (*Brahm's Lullaby* for our demonstration). When the wake time is reached, a soothing wake sound plays (*House of the Rising Sun* for our demonstration), the curtain rises, and the lights reactivate to the appropriate color.

Design, Solutions, Code Logic

The Arduino C code for the project is given in Appendix A. The general functionality and logic of the code is given as follows:

We made use of three Arduinos to complete this project. The system components are shown in the figure below. We decided it was better to have multiple Arduinos because of the lack of multithreading support on the Arduino platform. The System/Time Control Arduino (A1) manages the setup process and keeps track of time throughout the day. For demonstration purposes we simulated 24 hours in about six minutes. A1 signals the LED Control Arduino (A2) to synchronize their clocks and to inform A2 that setup is complete and the LED color sequence should be started. A1 also signals A2 when a sleep period begins or ends so that the LEDs can be deactivated or reactivated as necessary. A1 signals A3 at the beginning of a sleep period to activate the motor and lower the curtain, and to play the sleep sound. A1 signals A3 at the end of a sleep period to activate the motor and raise the curtain, and to play the wake sound.

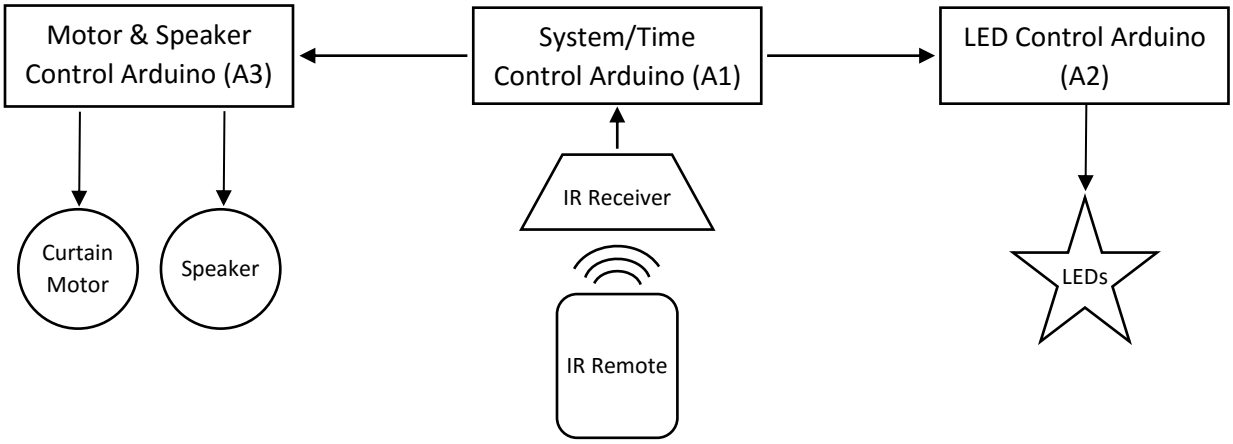
A stepper motor was used to raise and lower the curtain. The curtain rod was 3D modeled and 3D printed to suit our purpose:



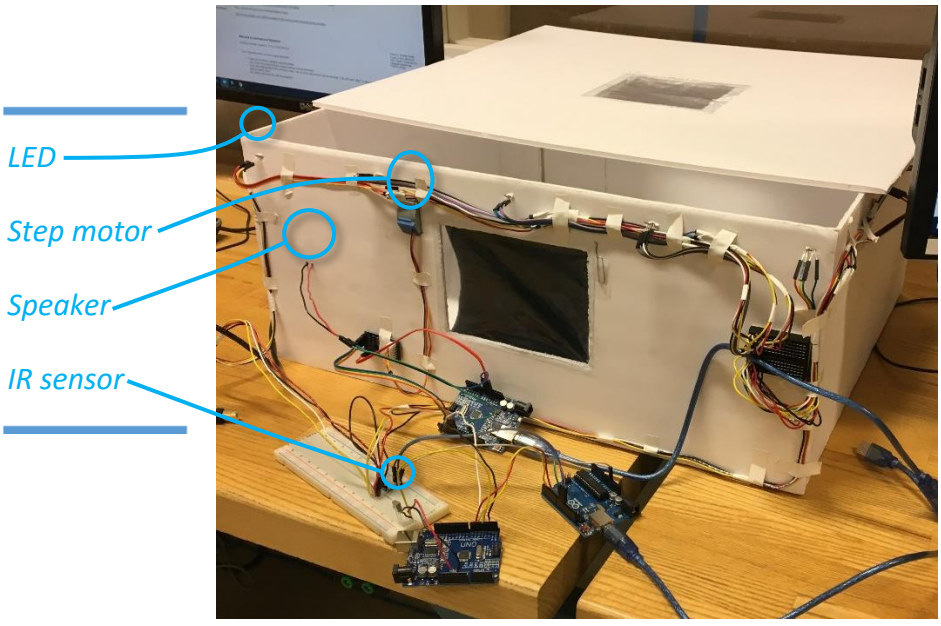
Thomas Nabelek
October 25, 2016

Project #2 Report

A basic speaker was used to play the sounds. The melodies used were written with individual frequencies (from a header file using `#define` statements, e.g.: `#define NOTE_C4 262`) and durations. Fourteen common anode diffused RGB LEDs were used and placed around the room near the ceiling. These allowed us to create custom colors by setting different values for each of the red, green, and blue LED leads. A simple IR remote control and receiver were used to add the setup functionality.



System Component Diagram



Outside of the bedroom

Thomas Nabelek
October 25, 2016

Project #2 Report

Inside of the bedroom

A video demonstration is available at youtube.com/watch?v=51Vg4IkWz3g.

One component we tried to include but were not able to get working was an LCD that would be used in the setup process and show the current time. We were almost able to use the LCD for our demonstration, but it would occasionally print incorrect characters or cause the Arduino to crash. Because of this, we defaulted to using the serial monitor on a desktop computer.

Additionally, we had planned to fade the LEDs at the beginning and end of wake periods, but there was an issue in the code that prevented this from working correctly. For the demonstration we had the LEDs simply deactivate and reactivate (on or off, no fading).

Future Improvements

In addition to the few things that weren't completed as planned and are simply a matter of playing with the code a little bit, such as dimming/raising the lights subtly rather than having them switch off/on suddenly, there are some other improvements and features that would be implemented in a product version of the system.

Ideally the curtain motor would run at the same time that the melody is played by the speaker. This would either require another Arduino, a task management system, or multithreading which is not available on an Arduino. Other synchronization features could also be incorporated if a task management system or multithreading were used. We would consider using a Raspberry Pi or something similar to accomplish this.

Features would be added to the system so that the parent user setting up the system would have more preference control over individual elements such as the light configuration and the melodies that are used. Speakers capable of generating sound from audio files would be used. Finally, a button would be added to raise and lower the curtain on demand, separate from the sleep cycles.

More advanced features that could be included would respond to the child. For example, motion and other sensors and machine learning could be used to determine if the child is restless or out of bed, and

Thomas Nabelek
October 25, 2016

Project #2 Report

a melody could be played or the parents could be alerted as appropriate. If the child is not in the room, the color-changing LEDs could be deactivated.

Finally, a method for resetting the system and accessing the setup mode would be provided.

Thomas Nabelek
October 25, 2016

Project #2 Report

Appendix A: Arduino Code

Setup_Time_Control.ino

```
#include <IRremote.h>
#include <TimeLib.h>

// ------- GLOBAL PIN VARIABLES --------
int music_sig_pin = 2;
int motor_sig_pin = 3;
int led_sys_pin = 5;
int led_status_pin = 6;

// ------- GLOBAL SLEEP CYCLE/IR VARIABLES --------
#define MAX_CYCLES 5
#define LCD_DELAY 10
int RECV_PIN = 4;

const char sleepCycles[] = "Sleep Cycles: ";
const char enterSleepTime[] = "Enter Sleep Time";
const char enterWakeTime[] = "Enter Wake Time";


IRrecv irrecv(RECV_PIN);

decode_results results;

int cycles = 0;
int result = 0;
int setupStage = 1;
int i = 0, currHour = 0, currMin = 0, prevHour = 0, prevMin = 0;;

enum swStatus_ {WAKE, SLEEP};
swStatus_ swStatus = WAKE;

struct SleepWakeTime
{
  int sleepHour;
  int sleepMinute;
  int wakeHour;
  int wakeMinute;
  swStatus_ swStatus;
};

struct SleepWakeTime sleepWakeTime[MAX_CYCLES];


// ------------- BEGIN SETUP --------------
void setup() {

  Serial.begin(9600);
  irrecv.enableIRIn();

  // ---------- Setup Pins -----------
  pinMode(music_sig_pin, OUTPUT);
  pinMode(motor_sig_pin, OUTPUT);
  pinMode(led_sys_pin, OUTPUT);
  pinMode(led_status_pin, OUTPUT);
  pinMode(10, INPUT);
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
  digitalWrite(music_sig_pin, HIGH);
  digitalWrite(motor_sig_pin, HIGH);
  digitalWrite(led_status_pin, HIGH);
  digitalWrite(led_sys_pin, LOW);


  // ------ Setup Sleep Cycles -------

  setupCycles();
  setupTimes();

  digitalWrite(led_sys_pin, HIGH);
  setTime(8,0,0,1,1,2016);

}

// ------------- BEGIN LOOP --------------
void loop() {

  currHour = hour();
  currMin = minute();

  displayTime();

  /*If currently awake check all sleep times against current time */
  if(swStatus == WAKE) {

    for(i = 0; i < cycles; ++i) {
      if(sleepWakeTime[i].sleepHour == currHour
      && sleepWakeTime[i].sleepMinute <= currMin) {

        digitalWrite(music_sig_pin, LOW);
        digitalWrite(motor_sig_pin, LOW);
        digitalWrite(led_status_pin, LOW);
        sleepWakeTime[i].swStatus = SLEEP;
        swStatus = SLEEP;
        break;
      }
    }
  }

  /*If currently sleeping check all wake times against current time */
  else {
    for(i = 0; i < cycles; ++i) {
      if(sleepWakeTime[i].swStatus == SLEEP
      && sleepWakeTime[i].wakeHour == currHour
      && sleepWakeTime[i].wakeMinute <= currMin) {

        digitalWrite(music_sig_pin, HIGH);
        digitalWrite(motor_sig_pin, HIGH);
        digitalWrite(led_status_pin, HIGH);
        sleepWakeTime[i].swStatus = WAKE;
        swStatus = WAKE;
        break;
      }
    }
  }
}

/* Display time every 10 mins */
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
void displayTime() {
  if(currHour != prevHour
  || currMin > (prevMin + 9)){

    if(currHour >= 10) {
      Serial.print(currHour);
    }
    else {
      Serial.print("0");
      Serial.print(currHour);
    }
    Serial.print(":");
    if(currMin >= 10) {
      Serial.print(currMin);
    }
    else {
      Serial.print("0");
      Serial.print(currMin);
    }

    Serial.println();


    prevHour = currHour;
    prevMin = currMin;
  }
}
```

Setup_Functions.ino

```
void setupCycles() {

  int  remoteVal;

  Serial.print(sleepCycles);
  Serial.println(cycles);

  while(setupStage == 1) {
    //Set Number of Sleep Cycles
    if (irrecv.decode(&results)) {
      remoteVal = decodeRemote(results.value);

      if(remoteVal == 10) { //Go to next stage
        setupStage = 2;
      }

      else if(remoteVal > MAX_CYCLES) {
        cycles = 0;
        Serial.print(sleepCycles);
        Serial.println(cycles);

      }

      else if(remoteVal != -1){
        cycles = remoteVal;
        Serial.print(sleepCycles);
        Serial.println(cycles);
      }
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
     irrecv.resume();     //Restart the receiver
    }
  }
}

void setupTimes() {

  int remoteVal, currCycle = 0, cycleCounter = 0, prevVal = 0, cursorPos = 0,
sleepWake = 0, currentTime[4] = {0,0,0,0};

  sleepWake = initializeSleepDisplay(sleepWake);

  while(cycleCounter < (cycles *2)) {

    if (irrecv.decode(&results)) {
      remoteVal = decodeRemote(results.value);

      //If Valid value from remote
      if(remoteVal < 10 && remoteVal != -1){
        if(checkTimeValue(cursorPos,remoteVal,currentTime)){

          Serial.print(remoteVal);

          if(cursorPos == 1) {
            Serial.print(":");
          }
          currentTime[cursorPos] = remoteVal;

          ++cursorPos;

          prevVal = remoteVal;
          if(cursorPos == 4) {
            cursorPos = 0;
            Serial.println("");

            //Set times in global structure for use in main loop
            if(sleepWake == 1) { //Just set sleep time
              sleepWakeTime[currCycle].sleepHour = (currentTime[0] * 10) +
currentTime[1];
              sleepWakeTime[currCycle].sleepMinute = (currentTime[2] * 10) +
currentTime[3];
            }

            else { //Just set wake time
              sleepWakeTime[currCycle].wakeHour = (currentTime[0] * 10) +
currentTime[1];
              sleepWakeTime[currCycle].wakeMinute = (currentTime[2] * 10) +
currentTime[3];
              ++currCycle;
            }

            ++cycleCounter;
            for(int i = 0; i < 4; ++i) {
              currentTime[i] = 0;
            }
            if(cycleCounter < (cycles * 2))
              sleepWake = initializeSleepDisplay(sleepWake);
          }
        }
      }
```

```
      irrecv.resume();     //Restart the receiver
    }

    delay(25);
  }
}

int initializeSleepDisplay(int sleepWake) {

  int returnVal = 0;

  //Initialize display

  if(sleepWake == 0) {
    Serial.println(enterSleepTime);
    returnVal = 1;
  }
  else {
    Serial.println(enterWakeTime);
    returnVal = 0;
  }

  return returnVal;
}

boolean checkTimeValue(int cursorPos, int value, int currentTime[]) {

  //Do error checing on the input time to ensure it is valid
  if(cursorPos == 0) {
    if(value > 2) {
      return false;
    }
    else if(currentTime[1] <= 4) {
      return true;
    }
    else {
      return false;
    }
  }
  else if(cursorPos == 1) {
    if(value > 4 && currentTime[0] == 2) {
      return false;
    }
    else {
      return true;
    }
  }
  else if(cursorPos == 3) {
    if(value > 5) {
      return false;
    }
    else {
      return true;
    }
  }
  else{
    return true;
  }
}
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
int decodeRemote(unsigned long value) {
  int returnVal = 0;
  switch(value) {
    case 16738455 :
      returnVal = 0;
      break;

    case 16724175 :
      returnVal = 1;
      break;

    case 16718055 :
      returnVal = 2;
      break;

    case 16743045 :
      returnVal = 3;
      break;

    case 16716015 :
      returnVal = 4;
      break;

    case 16726215 :
      returnVal = 5;
      break;

    case 16734885 :
      returnVal = 6;
      break;

    case 16728165 :
      returnVal = 7;
      break;

    case 16730805 :
      returnVal = 8;
      break;

    case 16732845 :
      returnVal = 9;
      break;

    case 16712445 :
      returnVal = 10; //Next
      break;

    case 16720605 :
      returnVal = 11; //Prev
      break;

    default :
      returnVal = -1;
      break;
  }

  return returnVal;

}
```

Thomas Nabelek
October 25, 2016

Project #2 Report

sound_header.h.ino

```
#include "pitches.h"

#define sound_pin 13
#define PAUSE_BETWEEN_NOTES 1.10 // the note's duration + 30% seems to work well

struct note {
  int pitch;
  double duration;
};

// notes in the melody, durations: 4 = quarter note, 8 = eighth note, 3 = dotted
quarter

// House of the Rising Sun
const struct note notes_wake[] = {
  {NOTE_A3,  8}, // There
  {NOTE_A3,  3}, {REST,     4}, {NOTE_B3,  8}, {NOTE_C4,  3}, {REST,     4}, {NOTE_E4,
8}, // is a house in
  {NOTE_D4,  8}, {NOTE_A3,  8}, {NOTE_A3,  4}, {REST,     4}, {REST,     4}, {REST,
6}, {NOTE_A4,  8}, // New Orleans they
  {NOTE_A4,  1.75}, {NOTE_A4,  16}, {NOTE_G4,  1.85}, {NOTE_E4,  8}, // call the
Rising
  {NOTE_E4,  2} // Sun
};

// Brahm's Lullaby
const struct note notes_sleep[] = {
  {NOTE_E4,  8}, {NOTE_E4,  8}, {NOTE_G4, 2}, {NOTE_E4, 8}, {NOTE_E4, 8},
  {NOTE_G4, 2}, {NOTE_E4, 8}, {NOTE_G4, 8}, {NOTE_C5, 4}, {NOTE_B4, 4}, {NOTE_A4, 4},
  {NOTE_A4, 4}, {NOTE_G4, 4}, {NOTE_D4, 8}, {NOTE_E4, 8}, {NOTE_F4, 4}, {NOTE_D4, 4},
{NOTE_D4, 8}, {NOTE_E4, 8},
  {NOTE_F4, 2}, {NOTE_D4, 8}, {NOTE_F4, 8}, {NOTE_B4, 8}, {NOTE_A4, 8}, {NOTE_G4, 4},
{NOTE_B4, 4},
  {NOTE_C5, 2}, {NOTE_C4, 8}, {NOTE_C4, 8}, {NOTE_C5, 2}, {NOTE_A4, 8}, {NOTE_F4, 8},
//  {NOTE_G4, 2}, {NOTE_E4,  8}, {NOTE_C4,  8}, {NOTE_F4, 4}, {NOTE_G4, 4}, {NOTE_A4,
4},
//  {NOTE_G4, 2}, {NOTE_C4, 8}, {NOTE_C4, 8}, {NOTE_C5, 2}, {NOTE_A4, 8}, {NOTE_F4,
8},
//  {NOTE_G4, 2}, {NOTE_E4, 8}, {NOTE_C4, 8}, {NOTE_F4, 4}, {NOTE_E4, 4}, {NOTE_D4,
4}, {NOTE_C4, 2}
};

// Send 1 for sleep music and 2 for wake music
void play_music(int music_selection) {
  if (music_selection == 1) {
    int whole_note_duration = 1000;
    // iterate over the notes of the melody
    for (int thisNote = 0; thisNote < sizeof(notes_sleep)/sizeof(notes_sleep[0]);
thisNote++) {

      double note_duration = whole_note_duration / notes_sleep[thisNote].duration;
      tone(sound_pin, notes_sleep[thisNote].pitch, note_duration); //e.g. quarter note
= WHOLE_NOTE_DURATION / 4, eighth note = WHOLE_NOTE_DURATION / 8, etc.
      delay(note_duration * PAUSE_BETWEEN_NOTES);
      noTone(sound_pin); // Stop tone
    }
  }
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
  else {
    int whole_note_duration = 1700;
    // iterate over the notes of the melody
    for (int thisNote = 0; thisNote < sizeof(notes_wake)/sizeof(notes_wake[0]);
thisNote++) {

      double note_duration = whole_note_duration / notes_wake[thisNote].duration;
      tone(sound_pin, notes_wake[thisNote].pitch, note_duration); //e.g. quarter note
= WHOLE_NOTE_DURATION / 4, eighth note = WHOLE_NOTE_DURATION / 8, etc.
      delay(note_duration * PAUSE_BETWEEN_NOTES);
      noTone(sound_pin); // Stop tone
    }
  }

}
```

LED_control.ino

```
#include <TimeLib.h>

#define PIN_SYS_STATUS 2
#define PIN_LED_STATUS 4
#define PIN_RED 9
#define PIN_GREEN 10
#define PIN_BLUE 11
#define delayTime 20
#define fadeDelay 1
#define brightnessMultiplier 1  // 1 is 100% brightness, 0 is off
#define speedMultiplier 0.7
#define NUM_COLORS 7
// delayTime = 20 and speedMultiplier = 1 takes 41.5 seconds
// delayTime = 20 and speedMultiplier = 0.14 takes 4 minutes

struct color {
  int redVal;
  int greenVal;
  int blueVal;
  int change_time; // How long should color stay
};
const struct color colors[NUM_COLORS] = {
  {255, 140,   0,  9}, // yellow 7-9
  {255,  90,   0, 11}, // orange 9-11
  {200,   0,   0, 13}, // red 11-13
  {255,   0, 140, 15}, // purple 13-15
  {  0, 200,   0, 17}, // green 15-17
  {  0, 255, 120, 19}, // cyan 17-19
  {  0,   0, 255,  7},  // blue 19-7
//  { 63,  41, 191,    0},
//  {127,  82, 127,    0},
//  {191, 123, 63,     0}
};
const struct color off = {0, 0, 0, 0};

struct color current;
struct color next;

void setup() {
  Serial.begin(9600);      // open the serial port at 9600 bps

  pinMode(PIN_RED, OUTPUT);
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
  pinMode(PIN_GREEN, OUTPUT);
  pinMode(PIN_BLUE, OUTPUT);
  pinMode(PIN_SYS_STATUS, INPUT);
  pinMode(PIN_LED_STATUS, INPUT);

  // Start with lights off
  current = off;

  analogWrite(PIN_RED, floor(255 - (current.redVal * brightnessMultiplier)));
  analogWrite(PIN_GREEN, floor(255 - (current.greenVal * brightnessMultiplier)));
  analogWrite(PIN_BLUE, floor(255 - (current.blueVal * brightnessMultiplier)));

  // Wait for signal for system to start
  while(digitalRead(PIN_SYS_STATUS) == LOW) {
    Serial.print("waiting to raise lights\n");
    delay(10);
//    String printString = "PIN_SYS_STATUS value: ";
//    printString = printString + digitalRead(PIN_SYS_STATUS);
//    printString = printString + "\n";
//    Serial.print(printString);
  }

  setTime(8, 0, 0, 1, 1, 2016); // Start clock
}

int n;
// One loop is one cycle of colors (24 hours)
void loop() {

  for (n = 0; n < NUM_COLORS; n++) {

    next = colors[n];

    while(abs(current.redVal - next.redVal) > 0 || abs(current.greenVal -
next.greenVal) > 0 || abs(current.blueVal - next.blueVal) > 0) {

      if (next.redVal > current.redVal)
        current.redVal++;
      else if (next.redVal < current.redVal)
        current.redVal--;

      if (next.greenVal > current.greenVal)
        current.greenVal++;
      else if (next.greenVal < current.greenVal)
        current.greenVal--;

      if (next.blueVal > current.blueVal)
        current.blueVal++;
      else if (next.blueVal < current.blueVal)
        current.blueVal--;

      analogWrite(PIN_RED, floor(255 - (current.redVal * brightnessMultiplier)));
      analogWrite(PIN_GREEN, floor(255 - (current.greenVal * brightnessMultiplier)));
      analogWrite(PIN_BLUE, floor(255 - (current.blueVal * brightnessMultiplier)));

//      char buffer[30];
//      sprintf(buffer, "%3d %3d, %3d %3d, %3d %3d\n", current.redVal, next.redVal,
current.greenVal, next.greenVal, current.blueVal, next.blueVal);
//      Serial.print(buffer);
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
      if(digitalRead(PIN_LED_STATUS) == LOW) // Check if lights should be lowered
        lower_lights();

      Serial.print("updating color       ");
      Serial.print(hour());
      Serial.print(":");
      Serial.print(minute());
      Serial.print("\n");

      delay( floor(delayTime/speedMultiplier) );
    }

    // Stay on color until set hour
    while(abs(next.change_time - hour()) > 0) {

      Serial.print("difference:");
      Serial.print(abs(next.change_time - hour()));
      if(digitalRead(PIN_LED_STATUS) == LOW) { // Check if lights should be lowered
        lower_lights();
        raise_lights();
        break;
      }
      delay(5);
      Serial.print("       ");
      Serial.print(hour());
      Serial.print(":");
      Serial.print(minute());
      Serial.print("\n");
    }

    current = next;
  }
}

void lower_lights() {

//    while(current.redVal > 0 || current.greenVal > 0 || current.blueVal > 0) {
//      if (current.redVal > 0)
//        current.redVal--;
//      if (next.greenVal > 0)
//        current.greenVal--;
//      if (next.blueVal > 0)
//        current.blueVal--;
//
//      analogWrite(PIN_RED, floor(255 - (current.redVal * brightnessMultiplier)));
//      analogWrite(PIN_GREEN, floor(255 - (current.greenVal *
brightnessMultiplier)));
//      analogWrite(PIN_BLUE, floor(255 - (current.blueVal * brightnessMultiplier)));
//
//      char buffer[30];
//      sprintf(buffer, "%3d %3d, %3d %3d, %3d %3d\n", current.redVal, next.redVal,
current.greenVal, next.greenVal, current.blueVal, next.blueVal);
//      Serial.print(buffer);

//      delay(8);

      Serial.print("lowering lights\n");
      analogWrite(PIN_RED, floor(255 - (0 * brightnessMultiplier)));
      analogWrite(PIN_GREEN, floor(255 - (0 * brightnessMultiplier)));
      analogWrite(PIN_BLUE, floor(255 - (0 * brightnessMultiplier)));
```

Thomas Nabelek
October 25, 2016

Project #2 Report

```
//    }

    // Wait till LED signal goes to high to turn lights back on
    while(digitalRead(PIN_LED_STATUS) == LOW) {
      Serial.print("waiting to raise lights       ");
      Serial.print(hour());
      Serial.print(":");
      Serial.print(minute());
      Serial.print("\n");
      delay(5);

    // Update colors while LEDs are off
      if(abs(next.change_time - hour()) == 0) {
        current = next;
        n++;
        next = colors[n];
      }

    }
}

void raise_lights() {
  Serial.print("raising lights\n");
  analogWrite(PIN_RED, floor(255 - (current.redVal * brightnessMultiplier)));
  analogWrite(PIN_GREEN, floor(255 - (current.greenVal * brightnessMultiplier)));
  analogWrite(PIN_BLUE, floor(255 - (current.blueVal * brightnessMultiplier)));
}
```