Introduction

Statement of Purpose

As sleep deprived college students waking up on time in the morning is a big problem for us. We intend to make a better alarm that would use light to wake us up and use a system to make sure we actually get out of bed. In addition to the alarm this system would also be used to manage the lighting in the room according to how bright it is outside. In essence, our project is a light level controller working in sync with an alarm clock, with a pressure sensor as another input.

In the past semester we worked on tackling the issue of difficulty for sleep deprived college students waking up on time in the morning with a better alarm that would use light to wake up students along with managing lightning in the room it was in. This semester we hope to improve on our design with new features and refining of features we implemented previously, along with ironing out some technical difficulties we encountered in the previous iteration.

Features and Benefits

The main feature of our project is the automatically controlled light source, which responds to changes in the environment's light level. The benefit of this feature is, through it maintaining an appropriate light level, allowing your "internal clock" to more easily fall asleep at night and get up in the morning.

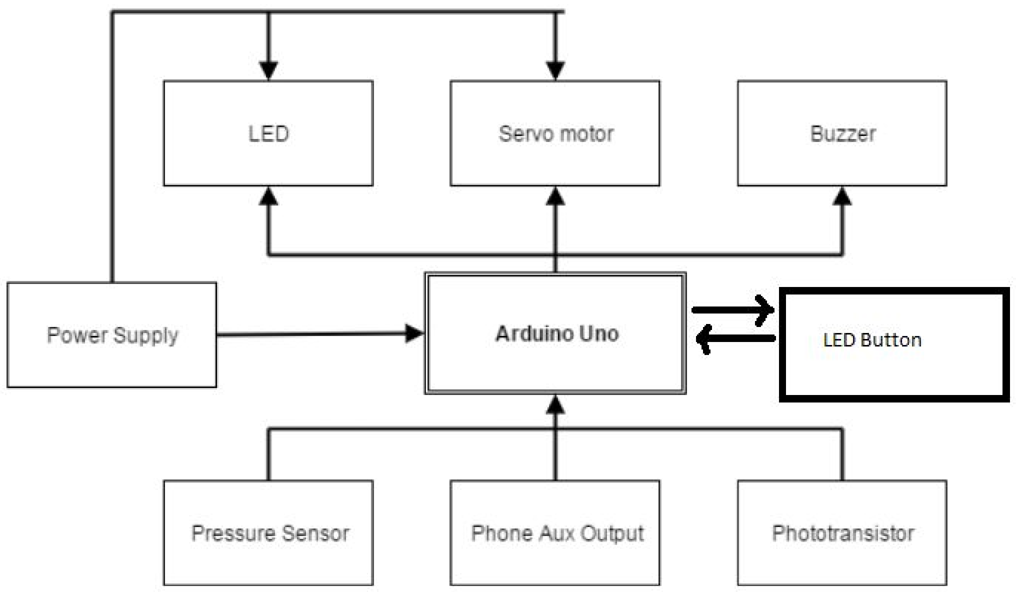
Another feature is the alarm clock functionality, which has all the usual benefits most alarms do, along with the benefit of working in sync with the light systems. A notable part of this feature is that by stripping and utilizing a phone aux cable, we allow the arduino to detect the phone alarm going off and sound its alarm, letting the system be based on your phone clock and alarm without needing much in the way of special arduino modules.

The system is further augmented by a servo feature, intended to control blinds to allow further control of a room's light levels, as well as a pressure sensor intended to be stepped on as you get out of bed, telling the alarm to stop and the light systems to come on. Overall this has the benefit of working very seamlessly to make your time waking up and being in a room more pleasant and healthy for your sleep cycle.

Furthermore, we added an LED button, a button which itself contains an LED, controlling whether the light system is on or off. The LED allows the button to be lit even while the rest of the system is not, making it easy to find in the dark. We went with a blue button LED so that the light was not as annoying in the dark room while one is trying to sleep.

Design

System Overview



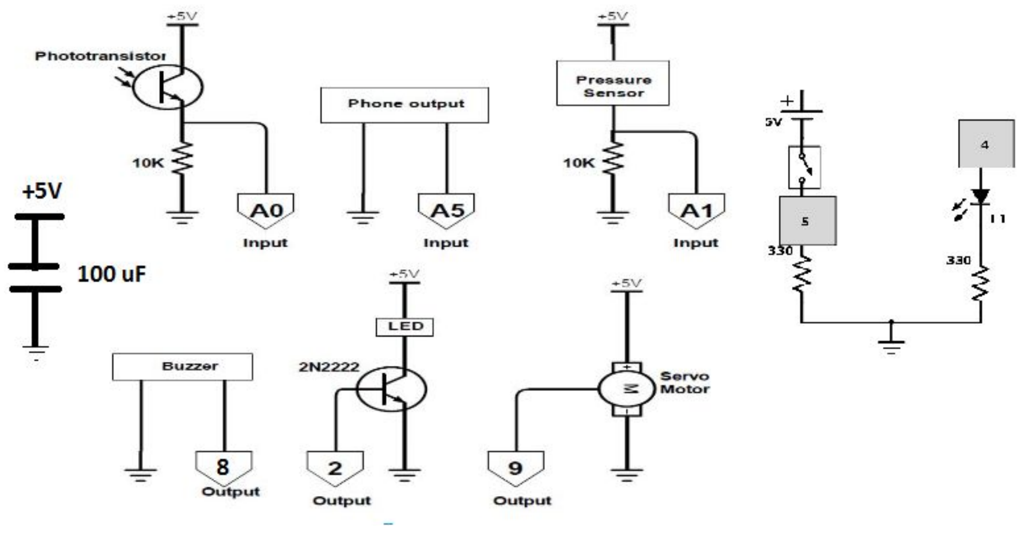
The Arduino receives the output from the phone as the condition to turn on both the alarm clock and the light control system. The Arduino also receives an digital signal controlled by a button, which allow the user to shut off the light control manually. In the light control system, the Arduino receives input from the phototransistor circuits which determines the ambient brightness. The inputs from the light sensor inside the room are used to adjust the brightness of LED strip and the angle of the servo motor as the demo of blind control. Every morning the alarm clock sends a on signal to the Arduino to turn the light control system when it reaches the set time, the Arduino then sends a signal to the servo and LED to adjust the light, and activates the alarm. The alarm is on until Arduino receives a signal from the pressure sensor circuit that indicates the user wakes up and stands on the scale. If the user is using the phone as the alarm signal, he also have to manually turn off the phone output, just to make sure the user is completely awake. The servo and LED system will still be on after the alarm clock is turned off until the user turn it off. The alarm will be on after receiving alarm signal regardless of the status of the light control system. The user can manually shut down the light control system with a button. The LED on the button will be on when the system is shut down so that the user can see the button in low ambient light.

Design Details

In the past semester, the power supply was a laptop connected to the arduino, simply chosen for convenience of pushing code and testing, while getting serial inputs from the arduino.The arduino uno was chosen in conjunction with its use in our regular ECE 110 course. The buzzer, servo motor, and pressure sensor were chosen due to their inclusion in our sparkfun kits, while the specific phototransistor we used was chosen for its detection of ambient light as opposed to just direct light. The LED was an RGB LED to allow us to simulate the dimming and lighting up of a larger lamp scale light source, and the phone aux input was chosen due to the convenience of allowing us to use a phone alarm in design instead of taking apart an alarm clock for incorporation into our circuit.

In the light control system, we apply the Arduino map function to map the input from ambient light sensor to the LED strip brightness and servo position. In addition, we set three different stages so that the change would be more noticeable and reasonable. The first one is extreme darkness. When the light sensor input is below a certain threshold which represent full darkness, the LED strip brightness will be set to highest level and the servo position would be set to the boundary which represents the curtain is closed at night. The second one is extreme darkness. When the light sensor input is above a certain threshold which represent extremely bright environment, the LED strip brightness will be set to 0 and the servo position would be set to the boundary which represents the curtain is fully open, in which case we are using the light outside as main light source. The third one is regular light level, in which case the LED brightness and servo position is mapped to the ambient light sensor input.

This semester, the power supply changed, as we implemented more features and systems that the arduino could not reasonably supply power to on its own, such as replacing our RGB LED with an entire LED strip. As a result, we used the lab power supply for most of the components to spare the arduino the effort. In addition, due to our last servo burning out we got a generally more powerful servo this semester, and the buzzer and light sensor remained from the past semester, along with the force sensor. We also continued to use the phone aux cable.



Result

Ambient light sensor**:**
Voltage across resistor (connected in series with 2K Ohm resistor and 5V power supply)
Darkness 0.03V
Brightness 4.3V
Ambient light 4.1V

Arduino Analog Read

Darkness below 400
Brightness 850-900
Ambient light 780-840

Voltage across Phototransistor
Darkness 4.97
Brightness 0.7V
Ambient light 0.9V

Pressure sensor
When no pressure is being applied to the FSR its resistance will be larger than 1MΩ.
When the FSR is pressed on pretty hard, its resistance will be only 1KΩ.

In the end we managed to develop a working code and a demo circuit. Used a servo to simulate the blind control, an LED strip that might be used in controlling a room's lighting, a button control for turning the system on and off, a phone to send signal as the alarm clock, and a pressure sensor to simulate the user standing up to stop the alarm clock. In the demo session, we managed to make the light control system work as coded. When the ambient light is bright, the LED strip would produce little light; when the surrounding is dark the LEDs would be at full light; in common room light, the LEDs would produce a medium amount of light. Since we changed from a simple RGB LED to a LED strip we were able to vary actual light levels instead of just colors. In addition, the system is turned on when it received signal from phone, and the pressure sensor is able to shut down just the alarm circuit so that the light control system would still be on after the buzzer stop. Additionally the button, when pressed, can turn the light system off.

Problems and Challenges

One of the main challenges we faced was learning to properly control and operate the LED strip we implemented. The documentation was a bit lacking in places for out specific version, but with the help of the TAs we eventually figured it out. Another challenge came from managing the multitude of features we implemented in each arduino cycle. The programing solution to this came in the form of many conditional statements managing each piece of the system. However, this resulted in a “flickering” of the components such as servos and LEDs, which we were ultimately unable to fully solve, simply because they were connected to a highly varying room light level.

Future Plans

In terms of future work on the project, we would mostly look to improve it from project to product, and work to set it up for realistic operation. For example, we would mount our LED strip in a room structure to observe its capability to have significant impact on the light level of its environment. We would also properly mount the servo mechanism so that it controlled actual blinds, calibrating it appropriately. In addition to that, we would ideally, in a finished project, have multiple phototransistors in different locations to more accurately read the light level of a room and how to change it. For example, with a light sensor specifically looking outside, we can tell how much the blinds should be open to brighten the room a certain amount. Through calibrating these, we could have a very impactful and well controlled control over the light of a room. Additionally, while our current prototype operates on a phone alarm system, we might look to make the product more self sufficient with its own built in alarm systems.

In addition to these, we would design the system to take into account the feedback loop it might produce in increasing a room’s light level, but shining on its own phototransistors. Overall, we would look to keep the light level more stable an ambient, and less “flickery.” Furthermore, while our current system is good for essentially keeping a general brightness level all day, we might want to implement more of a connection to the actual time of day, and implement a f.lux like system of light being more orange as the day progresses, as well as being still a bit dimmer towards night so that it does not feel unnatural to people.

Lastly, this semester we had hoped to design and have printed a PCB of our circuit in EAGLE, but the time it took to rework our circuit and implement and refine features resulted in it being too late to do that. In the future we would hope to print and implement a PCB of our circuit.

References

[1]"Self-adjusting Study Light", Jacob Taylor, Thomas McCarthy, Karl Mulnik, 2015. [Online]. Available:https://wiki.illinois.edu/wiki/display/ECE110HLSF15/Self-adjusting+Study+Light?preview=/560271196/583206214/Final_Report_Desk_Lamp.pdf.

[2]"f.lux: sleep research". [Online]. Available:https://justgetflux.com/research.html.

[3]Stephen E. Blackman, "Lamp and alarm clock with gradually increasing light or sounds", U.S. Patent 6236622 B1 issued May 22, 2001. Available:https://www.google.com/patents/US6236622.

[4]Harisrikanth, Keshav, and Tingrui Guan. "Automatic Room Light Controller and Alarm Clock - ECE 110/120 Honors Lab Section - Illinois Wiki." *Automatic Room Light Controller and Alarm Clock - ECE 110/120 Honors Lab Section - Illinois Wiki*. IllinoisWiki, 12 Dec. 2016. Web. 10 Feb. 2017

Appendix

Arduino Code 1.0

```
//Using button to simulate the alarm signal.
#include <Adafruit_NeoPixel.h>
#define PIN 3
#define LED_COUNT 30

#include <Servo.h>
Servo servo1;

// Create an instance of the Adafruit_NeoPixel class called "leds".
// That'll be what we refer to from here on...
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB +
NEO_KHZ800);

boolean button = false;
boolean light = false;
boolean alarm = false;

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 servo1.attach(9);  //pin9-servo

 pinMode(A0,INPUT);//light sensor
 pinMode(A1,INPUT);//pressure sensor
 pinMode(5,INPUT);//button
 pinMode(4,OUTPUT);//button LED
 pinMode(8,OUTPUT);//buzzer
 pinMode(10,INPUT);//alarm signal
 Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB +
NEO_KHZ800);
}

void loop() {
 if(light==false){
   lightControl();
 }
 if(light==true){
     leds.setBrightness(0);
     leds.show();
     if(servo1.read()!=0){
       servo1.write(0);
     }
 }

 button = digitalRead(5);

 if(button && !light){
   leds.setBrightness(0);
   light = true;
   digitalWrite(4,1);
   while(button){
     button = digitalRead(5);
   }
```

```
  }

 if(button && light){
     leds.setBrightness(0);
     light = false;
     digitalWrite(4,0);
     while(button){
      button = digitalRead(5);
     }
  }

  alarm = digitalRead(10);
  if(alarm){
   light = false;
   alarm = false;
   digitalWrite(4,0);
   int read = analogRead(A1);
   while(read<10){
     tone(8,3600,300);
     lightControl();
     read = analogRead(A1);
   }
  }
}

void lightControl(){
    for(int i = 0; i < LED_COUNT; i++){
    leds.setPixelColor(i, 0xFF00FF);    // Set fourth LED to full red, no green, full blue
    leds.setPixelColor(i, 0xFF, 0x00, 0xFF);
    leds.begin();
  }
  // put your main code here, to run repeatedly:
  int sensor = analogRead(A0);
   int out = map (sensor, 800, 900, 20, 80);
   //Serial.print(sensor);
   //Serial.print("     ");
   //Serial.println(out);
   if(sensor<400){
    leds.setBrightness(100);
    leds.show();
    if(servo1.read()!=0){
      servo1.write(0);
    }
   }else if(sensor>860){
    leds.setBrightness(out);
    leds.show();
    int servoposition = map(sensor,860,950, 30, 140);
    servoposition = constrain(servoposition, 0, 180);
    if(servo1.read()!=servoposition){
      servo1.write(servoposition);
    }
   }else{
    leds.setBrightness(out);
    leds.show();
    int servoposition = map(sensor,400,860, 60, 140);
    servoposition = constrain(servoposition, 0, 180);
```

```
    if(servo1.read()!=servoposition){
      servo1.write(servoposition);
    }
   }
}
```

Arduino Code 2.0

```
//using headphone wire to receive signal from an actual phone
#include <Adafruit_NeoPixel.h>
#define PIN 3
#define LED_COUNT 30

#include <Servo.h>
Servo servo1;

// Create an instance of the Adafruit_NeoPixel class called "leds".
// That'll be what we refer to from here on...
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB +
NEO_KHZ800);

boolean button = false;
boolean light = false;
boolean alarm = false;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  servo1.attach(9);  //pin9-servo


  pinMode(A0,INPUT);//light sensor
  pinMode(A1,INPUT);//pressure sensor
  pinMode(5,INPUT);//button
  pinMode(4,OUTPUT);//button LED
  pinMode(8,OUTPUT);//buzzer
  pinMode(A5,INPUT);//alarm signal
  Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB +
NEO_KHZ800);
}

void loop() {
  if(light==false){
    lightControl();
  }
  if(light==true){
      leds.setBrightness(0);
      leds.show();
      if(servo1.read()!=0){
        servo1.write(0);
      }
  }

  button = digitalRead(5);

  if(button && !light){
    leds.setBrightness(0);
    light = true;
    digitalWrite(4,1);
    while(button){
      button = digitalRead(5);
    }
   }

  if(button && light){
      leds.setBrightness(0);
      light = false;
      digitalWrite(4,0);
      while(button){
       button = digitalRead(5);
      }
   }

   //alarm = digitalRead(10);
   int phone = analogRead(A5);
   Serial.print("phone ");
   Serial.println(phone);
   if(phone > 15 && phone < 20){
    alarm = true;
   }else{
    alarm = false;
   }

   if(alarm){
    light = false;
```

```
    alarm = false;
    digitalWrite(4,0);
    int read = analogRead(A1);
    while(read<10){
      tone(8,3600,300);
      lightControl();
      read = analogRead(A1);
    }
   }
}

void lightControl(){
    for(int i = 0; i < LED_COUNT; i++){
    leds.setPixelColor(i, 0xFF00FF);    // Set fourth LED to full red, no green, full blue
    leds.setPixelColor(i, 0xFF, 0x00, 0xFF);
    leds.begin();
  }
  // put your main code here, to run repeatedly:
  int sensor = analogRead(A0);
   int out = map (sensor, 800, 900, 20, 80);
   //Serial.print(sensor);
   //Serial.print("     ");
   //Serial.println(out);
   if(sensor<400){
    leds.setBrightness(100);
    leds.show();
    if(servo1.read()!=0){
      servo1.write(0);
    }
   }else if(sensor>860){
    leds.setBrightness(out);
    leds.show();
    int servoposition = map(sensor,860,950, 30, 140);
    servoposition = constrain(servoposition, 0, 180);
    if(servo1.read()!=servoposition){
      servo1.write(servoposition);
    }
   }else{
    leds.setBrightness(out);
    leds.show();
    int servoposition = map(sensor,400,860, 60, 140);
    servoposition = constrain(servoposition, 0, 180);
    if(servo1.read()!=servoposition){
      servo1.write(servoposition);
    }
   }
}
```