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.

Features and Benefits

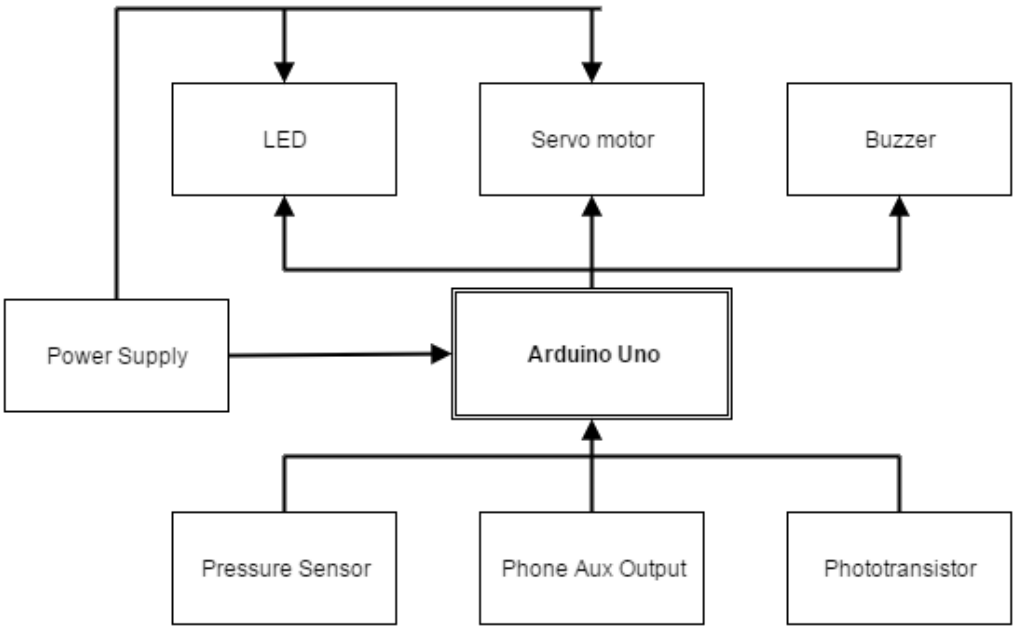
The main feature of our project is the automatically controlled light source, which responds to changes in the environment's light level. The benefit of this feature is, through it maintaining an appropriate light level, allowing your "internal clock" to more easily fall asleep at night and get up in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

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.

Design

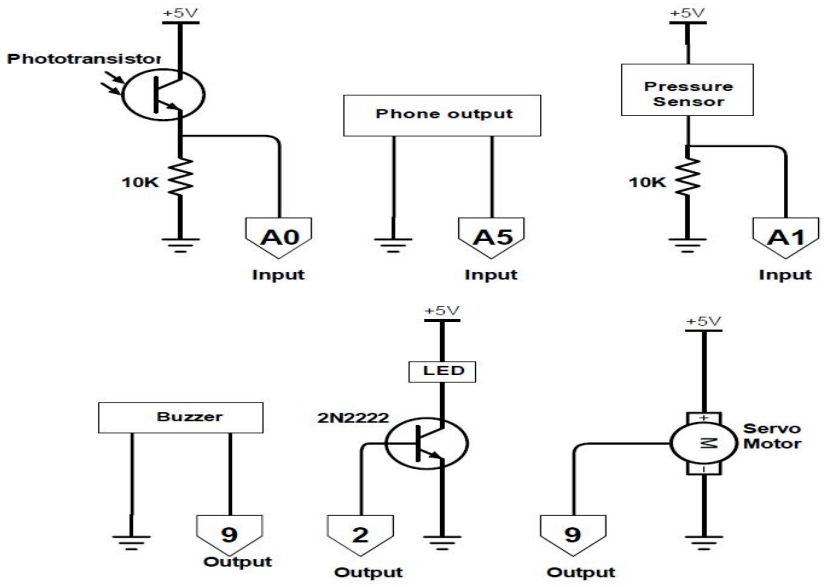
System Overview



The Arduino receives the output from the phone as the condition to turn on both the alarm clock and the light control system. 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 and the color of LEDs 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 system on 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. The servo and LED system will still be on after the alarm clock is turned off. To reset the alarm, the user must manually reset the arduino everyday so the arduino would respond to another input from the phone signal.

Design Details

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.



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

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, two RGB LEDs to simulate the LED panel, 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 dim, the LED would produce no light at all; when the surrounding is dark the LEDs would be at full light; in common room light, the LEDs would produce a little light. Since we are using RGB LEDs, we also make them change color under these three different situation so the difference would be more noticeable. 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.

Problems and Challenges

Some of the biggest problems and challenges we encountered involved powering our setup. Earlier on in the project, due to Amazon pages containing multiple products, we received a much higher power requirement light panel than we intended, which ended up being beyond our capabilities to control and power with our circuit specifications. This was solved by in the end using RGB LEDs to simulate a light panel's dimming and lighting. Additionally, even with the RBG LEDs, our Arduino was strained to power all the systems of our servo, alarm buzzer, photo transistor, and rgb LEDs. This was solved by the use of two arduinos linked up as one, allowig us access to the power necessary to run the circuit. Another issue that came up very late on demo day was the untimely death of our servo, but fortunately we were able to replace it for demoing.

Future Plans

In terms of future work on the project, we would mostly look to improve it from prototype to a more realized project. For example, we would replace our simulatory RGB LEDs with a real light panel that can 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.

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.

Appendix

Arduino Code

```
#include <Servo.h>
Servo servo1;

void setup() {
  // put your setup code here, to run once:
  //clock signal-alarm loop & light control loop
  //break alarm loop until receive signal from pressure sensor

  Serial.begin(9600);
  servo1.attach(9);  //pin9-servo


  pinMode(2, OUTPUT);//led
  pinMode(A5,INPUT);//clock
  pinMode(A0,INPUT);//light sensor
  pinMode(A1,INPUT); //pressures sensor

   //pressure sensor - A0

}

void loop() {
  // put your main code here, to run repeatedly:
  //light & blind control cycle
  boolean SystemOn = false;
  while(1){
    if(analogRead(A5)>30&&analogRead(A5)<150){
       Serial.println(analogRead(A5));
       delay(10);
      SystemOn=true;
      break;
    }
    delay(1);
  }
  while(SystemOn){
    boolean alarmOff = false;
```

```
    while(analogRead(A1)<10){
    digitalWrite(4, HIGH); //buzzerPin
    delay(100);
    digitalWrite(4, LOW);
    delay(100);
    led();
    }
    alarmOff=true;
    while(alarmOff){
    lightControl();
    }
  }

}


void lightControl(){
   // put your main code here, to run repeatedly:
  int sensor = analogRead(A0);

  if(sensor<400){
  digitalWrite(2, LOW);//VARY FROM 0-1
  delay(5);
  }
  if(sensor>650){
    digitalWrite(2, HIGH);
  delay(5);
  digitalWrite(2, LOW);
  delay(1);//VARY FROM 0-1
  }
  if(sensor<650&&sensor>400){
    digitalWrite(2, HIGH);
  delay(1);
  digitalWrite(2, LOW);
  delay(4);//VARY FROM 0-1
  }


  int servoposition = map(sensor,200,600, 30, 150);
```

```
  servoposition = constrain(servoposition, 0, 180);

  servo1.write(servoposition);
}

void led(){
  digitalWrite(2, HIGH);
    delay(1);
  digitalWrite(2, LOW);
  delay(1);//VARY FROM 0-1  //led full brightness
}
```