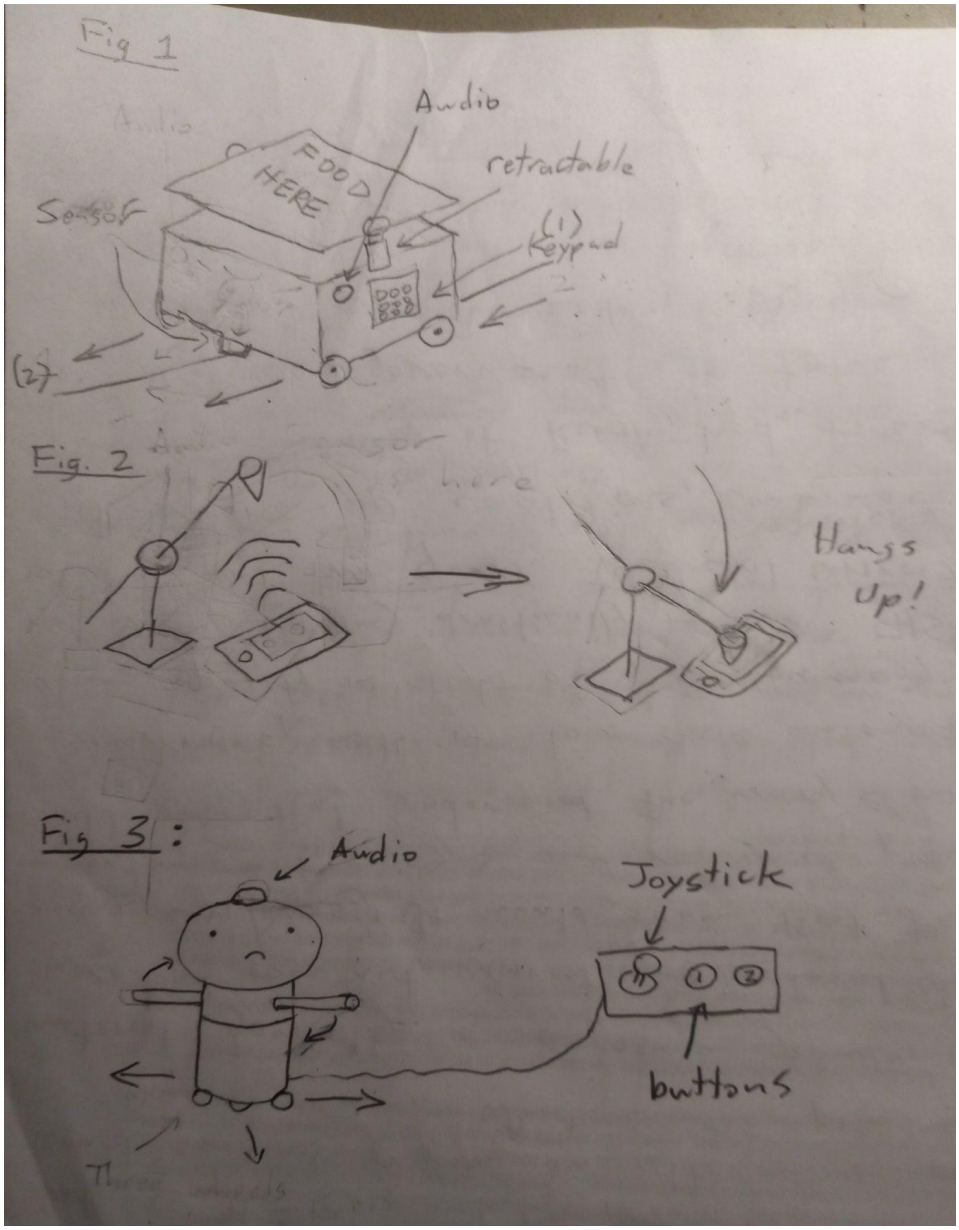


Fig 1: Robot is set on a graph of points. Each point is assigned a number. The user types the number they want to send the robot to and places food on its "table". The robot retracts the table down, follows the line to the destination, and moves the table up again when it stops moving.

Fig 2: Robot listens for a sound. When it hears the phone ring, it "dips" to hang up on the phone immediately. (It subsequently returns to its original position.)

Fig 3: Robot is a cute toy. You can use the joystick to move it around in all directions. Press 1 to spin its arms (slowly but firmly, so that it doesn't hurt anyone). Press 2 to make it cry.

Daylight Alarm Clock Robot (Team #18)

Cam Lipp, Gavin Hsia, Joseph Yu

Purpose - To build an autonomous arduino that dependant on a certain light setting will activate a series of events designed to wake up a sleeping individual.

Components List

- Large Piezo
- Servo mini 180 motor
- RGB LEDs
- Photoresistor
- CPU cooling fan
- 220 Ω resistors
- Pushbutton

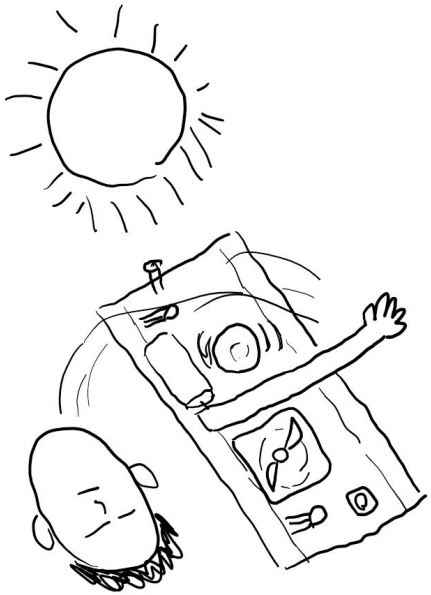
Group Schedule/ Roles

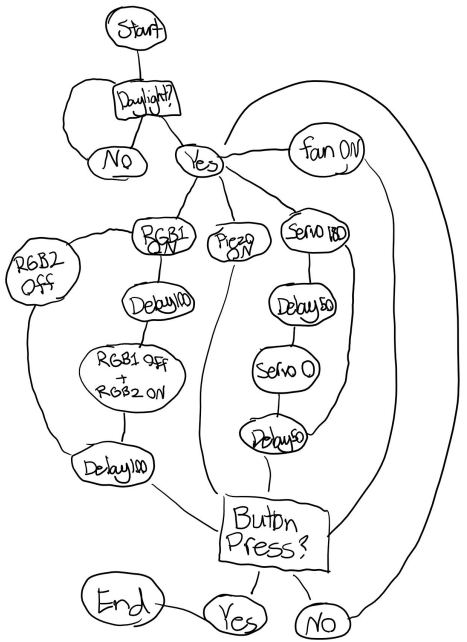
11/16: C,J,G work as a team on code and design

11/20: C work on final material round up

11/21: C,J,G Finalize code, design, and motion

11/23: G,J,C Build final design and test





Pseudocode

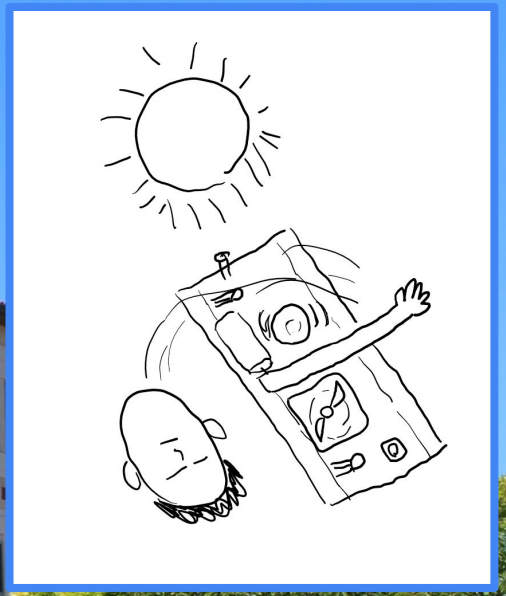
```
func alarm_clock():
        if (daylight detected):
                turn on fan
                sound buzzer
                repeatedly fire LED and motor
                if button pressed, system_exit()
```

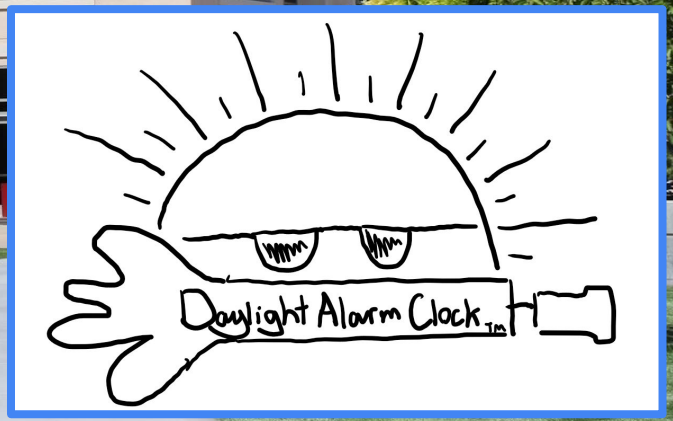
Daylight Alarm Clock

Team 18: Cam Lipp, Joseph Yu, Gavin Hsia

The Daylight Alarm Clock is the perfect robot to get you back on a good sleep schedule. It contains a photoresistor to detect the morning sunlight, two RGB LEDs to flash a plethora of beautiful wake up hues, a piezo to act as your active noise to wake you up, and a state of the art double hand to slap you just in case you need that extra motivation to get up.

"Don't just wake up, kick the sheets and seize the day!"





Daylight Alarm Clock Presentation

Project Purpose-

Good sleep is something that college students know almost nothing about. Late nights go into early mornings and almost every college student needs to take a nap after a couple short college classes. Waking up in a natural way is proven to be healthier and the most natural way to do it is sunlight. Humans used to use sunlight as their alarm clock since the homo sapiens and we have devised the best alarm clock to wake you up in the morning based on sunlight. The Daylight Alarm Clock wakes you up when the sunlight is at the best angle to wake up and it leaves college students like you well rested throughout your whole day.

Target Goal-

Our target is to create a standalone arduino circuit whose actions are dependent on the light level received by a photoresistor. When the light level received is above a certain threshold the circuit will signal for a series of actions to ensue. These actions will include a RGB light display in addition to a hand that will slap the intendented target through a servo motor. There will also be a piezo and a fan to make sure that the target is woken up when intended.

MVP-

Our RGB is working. Our servo motor is working though we do not have the hand designed yet. Our piezo is making noise but we need to further develop the specific ringtone. We don't have a mount for our arduino or motor yet. Fan is not wired in because we need more resistors.

Daylight Alarm Clock: Summary Report

by Cam Lipp, Gavin Hsia, Joseph Yu

Students are unable to maintain a healthy sleep schedule, and many of them have the tendency to oversleep. Therefore, we tasked ourselves to build an alarm clock to force students to wake up as soon as the sun rises, complete with flashing lights, a high-pitch buzzer, and a spinning hand designed to slap their face.

We used a photoresistor to detect incoming sunlight, which would feed back the light level into the Arduino. The MCU would compare this value to a given threshold, which determines whether there is sunlight entering the photoresistor or not. If the MCU "detects" sunlight in this manner (if the level light exceeds the threshold), the motor will start to spin continuously, slapping an elongated hand-shaped piece of plastic onto the person over and over. Meanwhile, the Piezo buzzer will play a very high pitched noise that loops frequencies through a preset interval, based upon the "video game" sound effect from Week 5, though the frequency interval was modified to emit much higher pitch sounds. The RGB LEDs flash colors using a similar method that acts upon the intensity of each of the three colors, creating a relative "rainbow" effect.

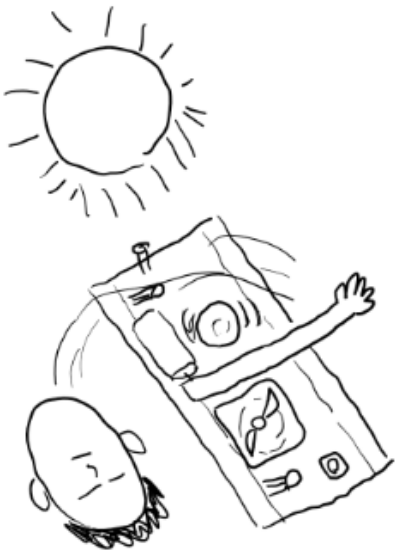
When the student is finally awake, they can press on the button, which would stop all functions, including the photoresistor. In a sense, the button turns the alarm clock "off," and it would need to be pressed again to essentially "arm" the alarm clock for the next morning.

The working criteria is as follows:

- All other functions of the alarm clock respond to the button and the photoresistor as described above.
    - The motor indeed spins and slaps the person at the correct times
    - The LEDs flash different colors at regular set intervals at the correct times
    - The Piezo buzzer produces sound at the correct times
- The appearance of the alarm clock does not look too clunky, and most of the wiring is hidden within a cardboard box.
- The alarm itself, produced by the functional parts (i.e. the motor, the buzzer, and the LEDs), is as obnoxious and ostentatious as it needs to be to remind students of the terrific suffering that constitutes existence and the fact that they cannot escape life's problems by sleeping.

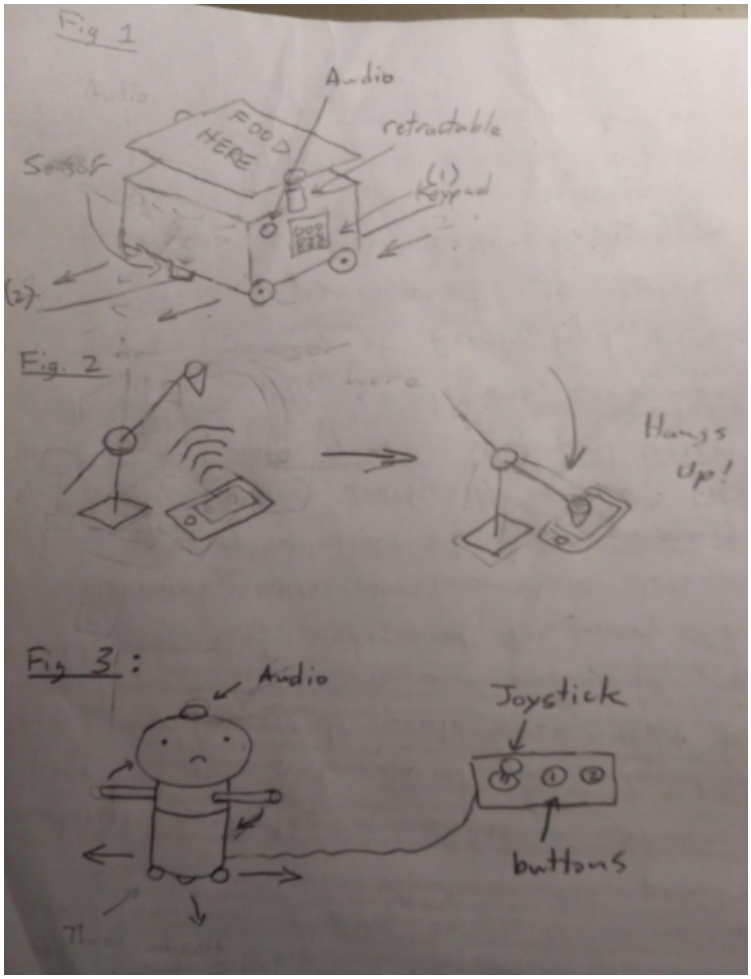
(Next Page)

We were originally going to include a fan within the final design of the alarm clock (see image on the right), but eventually scrapped the ideal due to issues with the wiring. We believe that the current product displayed before you today is sufficient to meet the criteria.



Other designs that we came up with include:

- A machine that follows a traced line and serves food at its endpoints.
- A small machine that hangs up the phone for you when it rings (see image below).
- A kind of battle-toy that moves according to a joystick, spins its arms, and produces sounds (see image below)

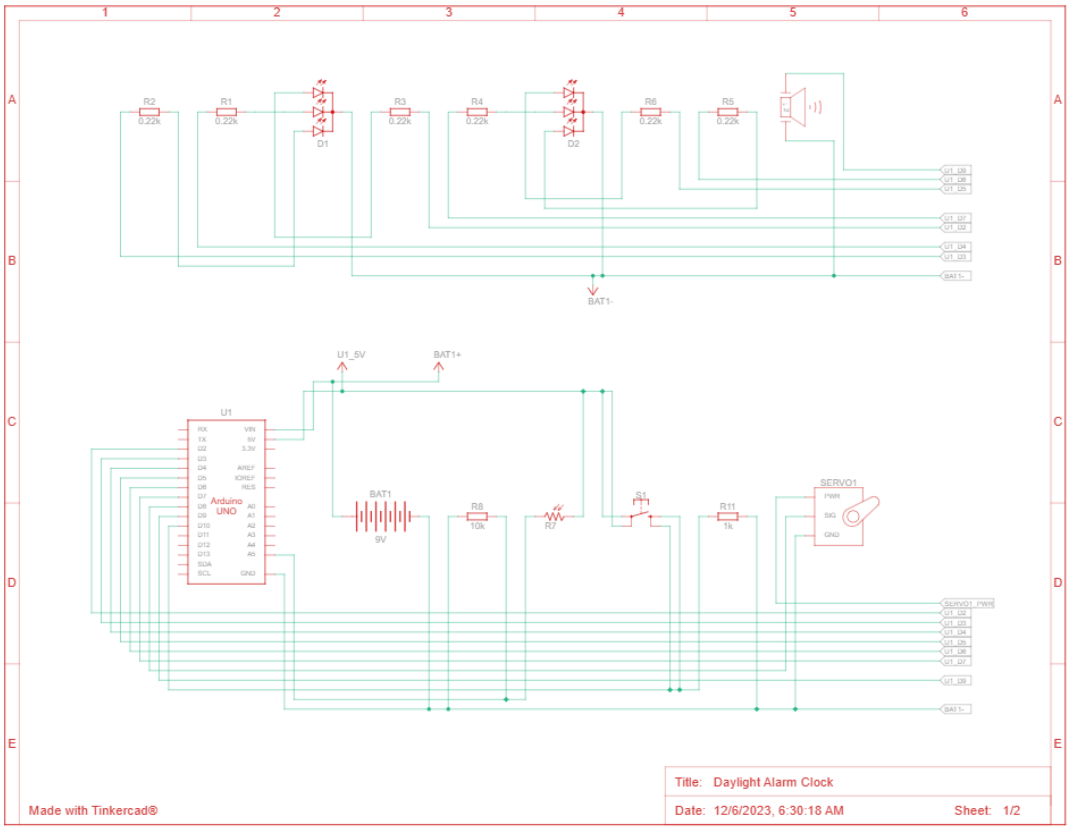


We decided to go for the alarm clock after acknowledging that the other designs were frankly too complicated and stupid.

The rest of the document consists of the various required tables, images, and code.

(Next Page)

Electronic schematic of the robot project



Team member project time log
(Note: we spent way too much time on this)

Team Member	**Time spent on project (include individual and team time)**	
	In-class (any lab periods attended)	Out-of-class (non-lab periods)
Gavin Hsia	6 hrs	6 hrs.
Cam Lipp	6 hrs	6 hrs.
Joseph Yu	6 hrs	6 hrs.

Bill of Materials

Quantity	**Description**	**Cost per item**
1	Arduino UNO R3	N/A - ENGR 1 loan
1	Piezo speaker	N/A - ENGR 1 loan
2	LED RGB	N/A - ENGR 1 loan
9	Resistors > 200 Ω	N/A - ENGR 1 loan

(Next Page)

1	Photoresistor	N/A - ENGR 1 loan
1	Continuous Micro Servo	N/A - ENGR 1 loan
1	Push Button	N/A - ENGR 1 loan
1	Cardboard Box	Trash Materials
1	Plastic Sheet (for the hand cut-out)	Trash Materials
N/A	Various wire components	N/A - ENGR 1 loan
2	9V Battery	$3.50
	Sum of all parts - Total Cost	$3.50

Photo of the Final Product

(Next Page)

Code for the Alarm Clock

```
#include <Servo.h>

// First RGB LED -->  [2-4]
const int redPin1 = 2;
const int greenPin1 = 4;
const int bluePin1 = 3;

// Second RGB LED --> [5-7]
const int redPin2 = 5;
const int greenPin2 = 7;
const int bluePin2 = 6;

// Motor --> [8]; Piezo --> [9]
const int slapPin = 8;
const int buzzerPin = 9;

// Photorest. --> [A5]; Button --> [10]
const int photoresistorPin = A5;
const int buttonPin = 10;

// Create a Servo object
Servo myServo;

void setup() {
  // Set all of the previously stated pins
  pinMode(redPin1, OUTPUT);
  pinMode(greenPin1, OUTPUT);
  pinMode(bluePin1, OUTPUT);

  pinMode(redPin2, OUTPUT);
  pinMode(greenPin2, OUTPUT);
  pinMode(bluePin2, OUTPUT);

  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);

  pinMode(photoresistorPin, INPUT);
  pinMode(buttonPin, INPUT);

  myServo.attach(slapPin);
}

int alarm_state = 1; // 0 = all function
cease
int light_value = 0; // current "brightness"
of LED values
int light_forward = 0; // cond. check if
"brightness" is maxed
int buzzer_freq = 1500; // freq. of buzzer

void loop() {
  // Read the analog values
  int lightLevel =
analogRead(photoresistorPin);
  int button = digitalRead(buttonPin);

  // Check if the light level surpasses the
threshold (e.g., 500)
  if (alarm_state == 1 && lightLevel > 500)
onstate();
  else offstate();
  // Check if button is pressed
  // if yes, switch alarm_state and cause a
three second delay
  if (button == HIGH) {

    if (alarm_state) {
      offstate();
      alarm_state = 0;
      tone(buzzerPin, 4500, 2000);
      delay(2001);
    } else {
      tone(buzzerPin, 4500, 2000);
      delay(2001);
      if (lightLevel > 500) onstate();
      alarm_state = 1;
    }
  }
}

void onstate() {
  // RGB LEDs (loops back and forth between 0
and 255)
  if (light_value == 0) light_forward = 1;
  else if (light_value == 255) light_forward
= 0;
  if (light_forward) light_value += 1;
  else light_value -= 1;

  analogWrite(redPin1, light_value);
  analogWrite(greenPin1, light_value);
  analogWrite(bluePin1, light_value);

  analogWrite(redPin2, light_value);
  analogWrite(greenPin2, light_value);
  analogWrite(bluePin2, light_value);

  // Servo Movement
  myServo.write(180);

  // Buzzer (inspired by Alien sound effect)
  tone(buzzerPin, buzzer_freq, 1);
  buzzer_freq += 1;
  if (buzzer_freq >= 3000) buzzer_freq =
1500;

  // 1 ms delay
  delay(1);
}

void offstate() {
  // Turn everything off
  analogWrite(redPin1, 0);
  analogWrite(greenPin1, 0);
  analogWrite(bluePin1, 0);
  analogWrite(redPin2, 0);
  analogWrite(greenPin2, 0);
  analogWrite(bluePin2, 0);
  myServo.write(95);
}
```