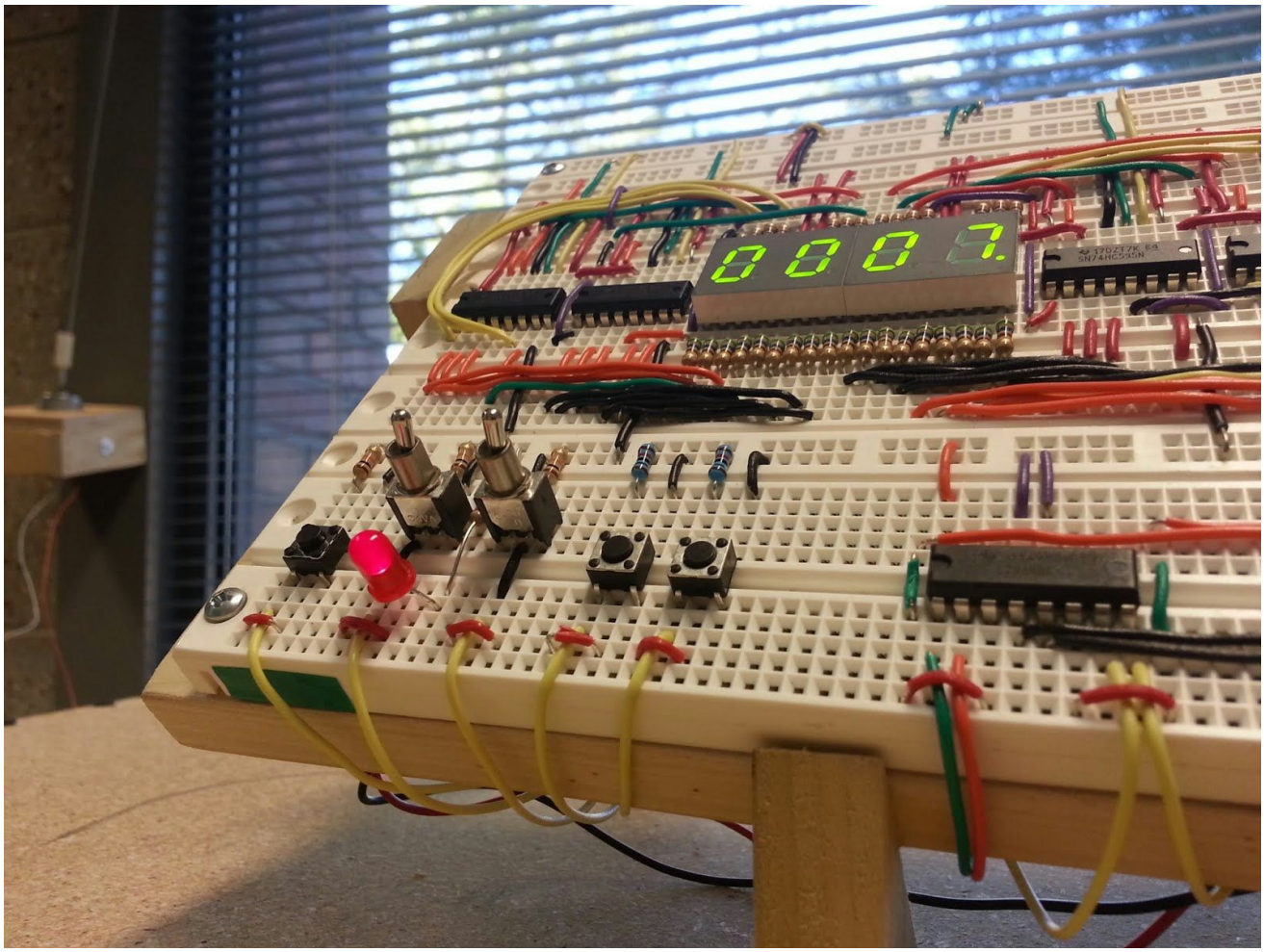
Christopher Marry, James Buckland, Steve Heffernan

Alarm Clock Automation



Christopher Marry, James Buckland, Steve Heffernan

Project Description

For many, a standard alarm clock is ineffective at waking them up. Some people can completely ignore sounds when they sleep. Others wear earplugs so they can sleep despite a snoring roommate. Traditional sound-based alarm clocks won't do anything for either of these people. However, a sudden, bright light is almost always enough to force people out of bed. We decided to build an alarm clock system that wakes people up via both light and sound. The sound is provided by a standard buzzer. The light is provided via a motor attached to the blinds that can open and close the blinds. When the alarm goes off, the blinds open and a buzzer sounds. Sometimes, you need a little extra sleep, which is why there is also a snooze function. When the snooze button is hit, the alarm stops and the blinds close, ready to go off again in a few minutes.

Description of Work

User Interface

Originally, we had planned to take apart an alarm clock and wire it to an Arduino, but this proved to be far more complicated that we anticipated. It was far easier to build an alarm clock from scratch than to reverse engineer one and add extra wires. As a result, we studied the functionality of a normal alarm clock in order to replicate it. Just like a regular alarm clock, ours is controlled with several buttons and switches. There is an alarm on/off switch, and a switch that allows the user to set the time or the alarm. We also have buttons to increment the hours or minutes of the time or alarm by one, as well as a snooze button. The time is displayed on a set of four seven-segment displays. In addition, there is a red indicator LED that turns on when the alarm is set to go off.

Electronics

We used four 74HC595N shift registers, an SN754410 H-Bridge two QDSP-G545 dual seven-segment displays, three pushbuttons, two switches, an LED, a buzzer, a custom-built motor mount, a motor, and more resistors than you can shake a stick at. The wire was all custom-cut and mounted onto breadboards borrowed from 265 Everitt, which were attached to a wooden mount. The electronics were installed as they were coded and given daily builds and tests with user input. We began with the display, and added time functionality using the open source Arduino Time library, and then buttons/UI features, and finally motor control.

The shift registers allowed us to control all four of the seven segment displays using only just three outputs from the Arduino. This was a major hurdle because the arduino only has 12 inputs/outputs, which we needed to consolidate in order to use all the components we needed.

We made several sample-design first draft breadboard to make sure the earlier drafts of the code worked, and tested the power of the motor and shape of the motor mount, verifying that the project would work far before the entire thing was finished.

Code

For the code itself, see the end of our project. A short summary: our code relies on the user to set the time and alarm, but counts forward in seconds using the internal clock that the Arduino comes equipped with. It initially sets the time to be 00:00 and the alarm to be 00:05, and the user can bump up the times from there. When the alarm goes off, the blinds open and the alarm sounds. Turning off the alarm will turn off the buzzer, but leave the blinds open; hitting snooze bumps the alarm up by five minutes, silences the alarm, and closes the blinds temporarily.

We had some experience with code, but the java coding was a challenge for us; not because the language was unfamiliar, but because the internal logic required for the code was complex, requiring a single processor to approximate parallel processing (counting time *and* opening blinds). Eventually, we figured out how to simulate parallel processing by making the arduino check the states of the buttons, refresh the time, and check to see if it should be sounding an alarm in a never-ending loop. This loop executes dozens of times per second, so from a user's point of view, it does everything simultaneously.

Motor

We started out by using the small motor that came with the Arduino. By experimenting with the code, we found that we could spin the motor in either directions at any speed we wanted. However, the stock motor was not powerful enough to turn the blind rods, so we needed something bigger.

We decided to purchase a DCM-330 motor from the ECE store. This presented new challenges because the motor drew too much current for the Arduino to handle. We needed an external power supply for the motor, which we created by linking two nine volt batteries in series. This necessitated a new control method for the motor. We tried connecting the motor, batteries, and arduino with transistors. This proved too difficult due to our lack of knowledge about electronics.

One of the ELAs suggested an H-bridge, an IC circuit that is designed to allow, block, or reverse current. This was perfect for our needs. The motor, batteries and arduino were all hooked up to the H-bridge. Depending on the states of three inputs, the motor would turn left, turn right, or stay still.

Mounting

In order to control the blinds, we had to find a way to attach a motor to the blinds. We decided early on that the easiest way to do this was to attach the motor to the hexagonal blind rod. By

turning the rod, the angle of the blinds changes, allowing us to open and close the blinds. In order to attach the motor to the rod, we needed to build a connector. This connector was custom built using the 3d printer. It is epoxied to the shaft of the motor, and slides freely on and off the blind rod.

The end of the blind rod is several feet off the ground, so we also needed a way to hold the motor in place on the wall. To accomplish this, we built a wooden box with a hole drilled in it for the motor. The motor is secured with two screws, but it has a little room to move back and forth, making it adjustable to fit any window. The box is attached to the wall via Velcro, making it easy to remove the alarm clock if desired.

The mounting system also allows the user to manually open and close the blinds by twisting the rod. It does not impede normal operation of the blinds.

Statement of Budget

Purchases:

$35	Arduino Kit
$5	Speaker
$3	Three seven-segment displays
$12	Switches
$12	Breadboard
$9	Shift Registers
$10	Wood and Hardware
$86	**Total Spent**

Other Materials: (free)

Large Breadboard
Wire
H-Bridge
Extra Switches

After spending quite a bit of money at the ECE store, we found out about the Electronics Service Shop in 265 Everitt. The ESS offered many small electrical components for free. We also rented a large breadboard that allowed us to consolidate all of our wiring onto one massive breadboard.

Time:

We had 10 weeks, with four hours of class per week. With three group members, this gave us 120 in-class man hours. We also met outside of class time on most Fridays and on some Saturdays, pushing the total number of man-hours spent on the project to around 180.

Reflection/Discussion

Overall, the project went very well. Our team was cohesive and we all got along together very well. There were no cross words or heated arguments; we all stayed calm and rational whenever things got difficult. We were very lucky to get a group of personalities that worked well together.

When it came to the panel reviews, they were great towards helping us on the project. The panel reviews forced us to have a deeper understanding of our project so that we would be able to answer difficult questions. This caused us to realize many flaws in our ideas before we started wasting time and money on building something that would eventually not work. The other students and the ELAs asked questions that made us reconsider what we were doing and alerted us to when we were doing something in an overly complicated way. The panel reviews were also great to just see what other interesting projects other groups were doing and being able to give them ideas on how to improve their project.

We learned quite a bit about practical coding throughout the project. None of us had ever coded any hardware before; it was all software. This was completely new ground for us. We learned how to code for Arduino by looking at other examples, as well as through trial and error. One thing that we never quite figured out was parallel processing. It should be something that arduino could do, but we eventually just cheated it by putting all of our code in a very quickly-executing loop.

None of us had any experience with electronics before this project, either. Everything that we learned about electronics was picked up on the fly. All of this spur of the moment research helped us to learn how to research better. We had to look beyond just Wikipedia in order to figure out everything that we needed to know.

Another great thing that we learned in this class was all of the resources we have available to us on campus. Without this class, we would have never found out about the ECE store, the Fab Lab, or the Electronics Service Shop. These places will surely be useful for projects in future classes.

This class also prepared us for our senior design class. In the senior design class, we will

be under much more stress as the project will be indicative of all that we have learned throughout our college career. This class allowed us to tackle a free-form problem in a similar way to a senior design class, but with much lower academic risk should our project fail.

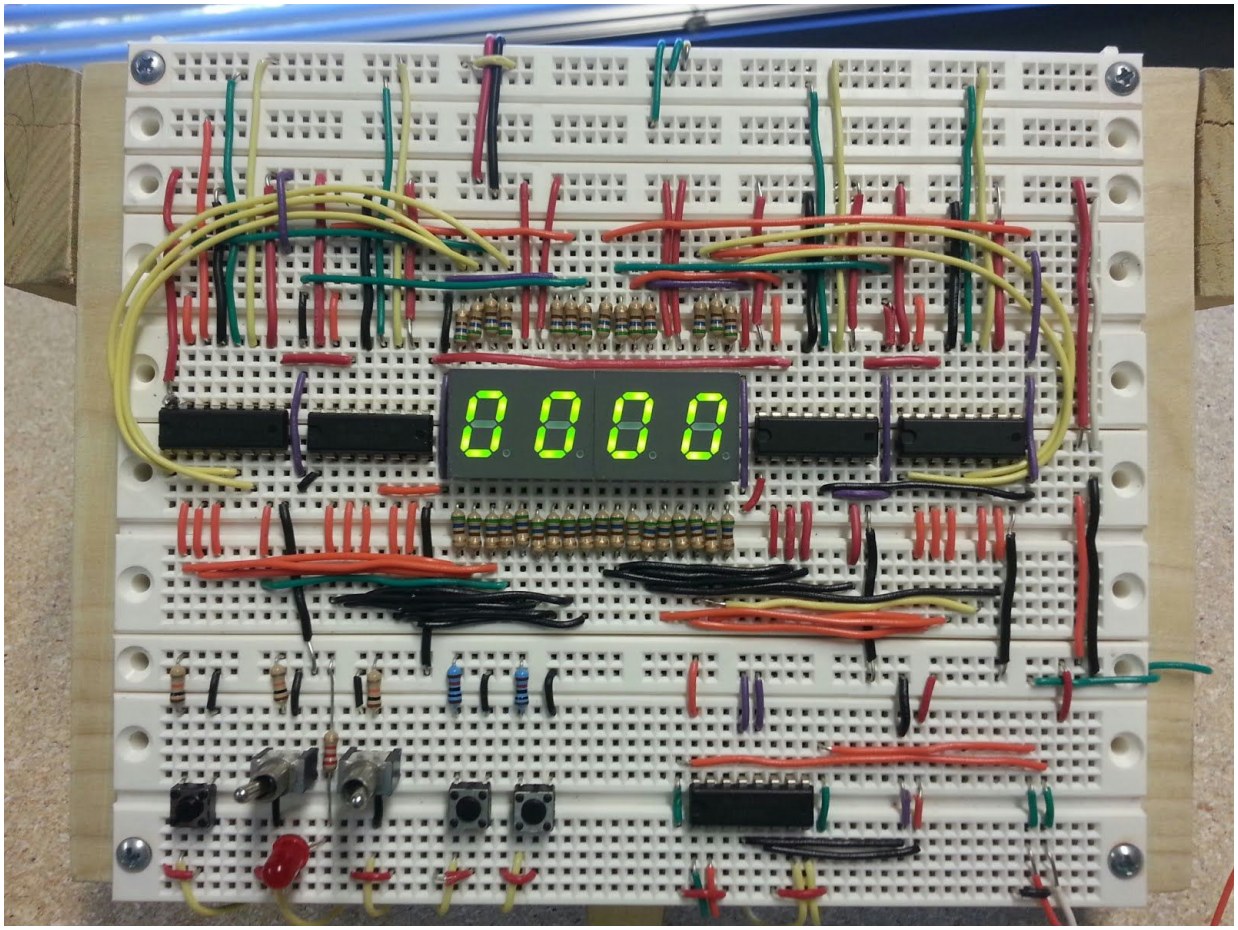
Future Work

There are several improvements that could be made to the project to make it suitable for mass production and sale. Chief among these would a printed circuit board. Our breadboarding was very rough and there were almost a hundred wires running all over it. It was also very fragile, which would be unsuitable for the abuse that alarm clocks are generally put through. Along with the printed circuit board, we would design a plastic body to contain all the electronics and hide them from view. This would allow us to use better buttons and switches than the ones that are compatible with breadboards.

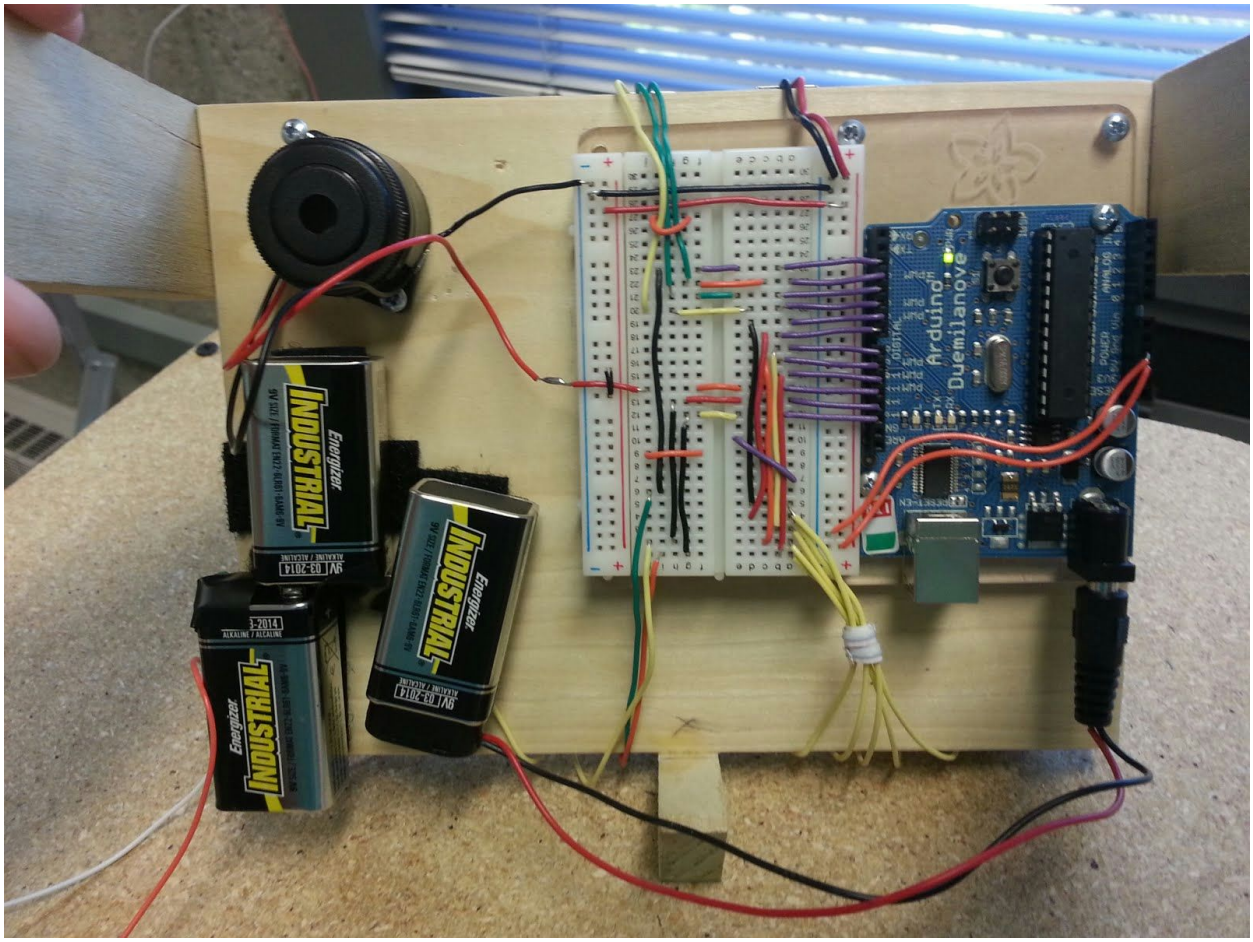
If this were to be mass-produced, we would need to replace the arduino with several ICs. We have done some research into how real clocks work and they use binary to seven-segment converters, clock chips, among others. In addition, the Arduino is very expensive and cannot be attached to a printed circuit board. We would also make the alarm clock powered by a wall outlet. The current design requires 3 nine volt batteries, which is expensive and impractical. Since this alarm clock is attached to the wall, it does not need to be portable.

Some larger-scale improvements could be done to improve the functionality of the clock. Currently, the clock is connected to the motor by long wires that might get in the way in the bedroom. If we could control the motor wirelessly via Bluetooth or some other means, it would eliminate the unsightly wires and a potential tripping hazard.

We could also improve the way the clock brings light into the room. The blinds are limited by the amount of light outside when the alarm goes off. If the user needs to wake up before the sun is up, then the blinds won't do much. We can fix this by also controlling the ceiling lights in the room. This would be similar to how we control the blinds: we would mount a servo next to the light switch that would flick the lights on and off.



The Front of the clock. In the center are the four seven segment displays, along with the massive array of resistors and wires. The four chips on the sides of the displays are the shift registers. On the bottom row, starting from the left are the snooze buttons, alarm on/off switch, time set/alarm set switch, hour button, and minute button. The fifth chip is the H-bridge, which has wires leading to the motor and is connected to a separate power source on the back of the mount.



The back of the mount. It has the three batteries, one for the arduino and two for the motor. A smaller breadboard is used to organize the wires leading to the main board. The buzzer and the arduino board are screwed into the wood.

The motor and mount. You can see one of the screws that holds the motor in place on the left side. The motor is loosely screwed in so that it can bend to match the angle of the blind rod. The white plastic mount is epoxied to the motor, but can freely slip on and off the rod. The whole wooden assembly is attached to the wall via velcro.

```
#include <Time.h>               //include time module

void setup() {                  //define pins
#define dataPin 3
#define latchPin 4
#define clockPin 5
#define minPin 6
#define hourPin 7
#define setPin 8
#define alarmPin 9
#define snoozePin 10
#define soundPin 11

#define enPin 2
#define leftPin 12
#define rightPin 13

  pinMode(dataPin,   OUTPUT); //define which pins are inputs and outputs
  pinMode(latchPin,  OUTPUT);
  pinMode(clockPin,  OUTPUT);
  pinMode(minPin,    INPUT);
  pinMode(hourPin,   INPUT);
  pinMode(setPin,   INPUT);
  pinMode(alarmPin,  INPUT);
  pinMode(snoozePin, INPUT);
  pinMode(soundPin, OUTPUT);

  pinMode(enPin, OUTPUT);
  pinMode(leftPin, OUTPUT);
  pinMode(rightPin, OUTPUT);

  Serial.begin(9600);
  Serial.println("started");
}

int minState          = 0;       // define a minute and hour state; this indicates whether the button
is being pressed
boolean minPressed    = false;  // define whether the button is being pressed
boolean minTrigger    = false;  // define whether the button has *just* been in the pressed state
int hourState         = 0;
boolean hourPressed   = false;
boolean hourTrigger   = false;
int snoozeState       = 0;      // the snooze if off
boolean snoozePressed = false;
boolean snoozeTrigger = false;
//initial parameters
int aHour             = 0;          // define an initial hour for the alarm
int aMin              = 5;          // define an initial minute for the alarm
boolean displayAlarm  = false;  // is the alarm being displayed when the clock is turned on? no.
boolean isAlarm       = false;
boolean firstTime = true;
boolean blindsClosed = true;


int length = 1; // the number of notes
char notes[] = "c"; //char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = {1}; //int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

```
int tempo = 300; //tempo is 300, come on, this stuff is basic

void loop() {
  isFirstTime(); // check to see if the alarm clock has just been turned on
  refreshSSD();  // refresh the display
  minCheck();  // check to see if the minute button is being pushed
  hourCheck(); // check to see if the hour button is being pushed
  snoozeCheck(); // check to see if the snooze button is being pushed
  if(digitalRead(alarmPin) == LOW){
    matchCheck(); // check to see if the alarm ought to be going off
  } else {isAlarm = false;}
}

void isFirstTime(){     // if it's the first time, turn off the motor
 if (firstTime == true){digitalWrite(enPin, LOW);}
 firstTime = 0;
}

void matchCheck(){  // check to see if the alarm ought to be going off
    if ( ( aHour == hour() ) && ( aMin == minute() ) ){
      isAlarm = true;
      if (blindsClosed == true){openBlinds(); blindsClosed = false; soundAlarm();}
      if (blindsClosed == false){soundAlarm();}
    } else {isAlarm = false;}
}

void openBlinds(){ // how to open the blinds in a curve pattern
  for (int i = 0; i < 75; i++){
    digitalWrite(enPin, HIGH);
    digitalWrite(leftPin, LOW);
    digitalWrite(rightPin, HIGH);
    delay(10);
    digitalWrite(enPin, LOW);
    delay(10);
  }
}

void closeBlinds(){ //how to close the blinds smoothly
  for (int i = 0; i < 75; i++){
    digitalWrite(enPin, HIGH);
    digitalWrite(leftPin, HIGH);
    digitalWrite(rightPin, LOW);
    delay(10);
    digitalWrite(enPin, LOW);
    delay(10);
  }
}

void soundAlarm(){ //defines the ringtone that the buzzer uses to go off
  digitalWrite(soundPin, HIGH);
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') { delay(beats[i] * tempo); } else { playNote(notes[i], beats[i] * tempo); }
    delay(tempo / 2);
  }
}

void playTone(int tone, int duration) { //how to send a tone to the speaker
```

```
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(soundPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(soundPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) { // define the tuning tones of the scale
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {playTone(tones[i], duration);}
  }
}

void refreshSSD(){ // check to see which way the time/alarm pin is, and display the according time
value
  if(digitalRead(setPin) ==  LOW){
    if(displayAlarm == true){tset(); displayAlarm = false;}
    else{refreshTime(); displayAlarm = false;}
  }
  if(digitalRead(setPin) == HIGH){
    if(displayAlarm == false){aset(); displayAlarm = true;}
    else{refreshAlarm(); displayAlarm = true;}
  }
}

void aset(){ // display the words ‘aset’ when the alarm is turned on
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, B10000111);
  shiftOut(dataPin, clockPin, MSBFIRST, B10000110);
  shiftOut(dataPin, clockPin, MSBFIRST, B10010010);
  shiftOut(dataPin, clockPin, MSBFIRST, B10001000);
  digitalWrite(latchPin, HIGH);
  delay(1000);
}

void tset(){ // display the words ‘tset’ when the time is turned on
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, B10000111);
  shiftOut(dataPin, clockPin, MSBFIRST, B10000110);
  shiftOut(dataPin, clockPin, MSBFIRST, B10010010);
  shiftOut(dataPin, clockPin, MSBFIRST, B10000111);
  digitalWrite(latchPin, HIGH);
  delay(1000);
}


void refreshTime(){ //how to display the time live
  byte a = assign(minute()%10) + seconds();
  byte b = assign(minute()/10) + blink2();
  byte c = assign(  hour()%10) + blink();
  byte d = assign(  hour()/10) + blink2();
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, a);
```

```
  shiftOut(dataPin, clockPin, MSBFIRST, b);
  shiftOut(dataPin, clockPin, MSBFIRST, c);
  shiftOut(dataPin, clockPin, MSBFIRST, d);
  digitalWrite(latchPin, HIGH);
}

void refreshAlarm(){ //how to display the alarm
  byte a = assign( aMin%10) + B10000000;
  byte b = assign( aMin/10) + B10000000;
  byte c = assign(aHour%10) + B10000000;
  byte d = assign(aHour/10) + B10000000;
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, a);
  shiftOut(dataPin, clockPin, MSBFIRST, b);
  shiftOut(dataPin, clockPin, MSBFIRST, c);
  shiftOut(dataPin, clockPin, MSBFIRST, d);
  digitalWrite(latchPin, HIGH);
}


void minCheck(){ //check to see if the minute button is being pressed; if it is, bump the minute value
up one
  minState = digitalRead(minPin);
  if (minState == LOW){minTrigger = true;}
  if (minState == HIGH){
    if (minTrigger == true){
      if(digitalRead(setPin) == HIGH){aMin = (aMin + 1)%60;}
      if(digitalRead(setPin) ==  LOW){setTime( hour(), minute()+1, second(), day(), month(), year() );}
      minTrigger = false;
    }
  }
}

void hourCheck(){ //same for hour
  hourState = digitalRead(hourPin);
  if (hourState == LOW){hourTrigger = true;}
  if (hourState == HIGH){
    if (hourTrigger == true){
      if(digitalRead(setPin) == HIGH){aHour = (aHour + 1)%24;}
      if(digitalRead(setPin) ==  LOW){setTime( hour()+1, minute(), second(), day(), month(), year() );}
      hourTrigger = false;
    }
  }
}

void snoozeCheck(){ //check to see if the snooze button is being pressed; if it is, increment the alarm
by five minutes
  snoozeState = digitalRead(snoozePin);
  if (snoozeState == LOW){snoozeTrigger = true;}
  if (snoozeState == HIGH){
    if (snoozeTrigger == true){
      if (aMin < 55){aMin = (aMin + 5)%60;}
      else {aMin = (aMin + 5)%60; aHour = (aHour + 1)%24;}
      if (isAlarm == true){closeBlinds(); blindsClosed = true;}
      snoozeTrigger = false;
    }
  }
```

```
}

byte seconds(){ //define the seconds byte
  byte j = B00000000;
  if (second()%2 == 0){j = B10000000;}
  return j;
}

byte blink(){ // define what it means for a dot to blink on...
  byte j = B00000000;
  if (isAlarm){j = seconds();}
  return j;
}

byte blink2(){ //and off
  byte j = B00000000;
  if (isAlarm){j = seconds() + B10000000;}
  return j;
}

byte alarmNotifier(){ //define a visual alarm sequence
 byte j = B00000000;
  if (isAlarm){j = B10000000;}
  return j;
}

byte assign(int i){ // the key array of the project; turns integers into seven-segment displays
  byte array[] = {B11000000, B11111001, B10100100, B10110000, B10011001, B10010010, B10000010,
B11111000, B10000000, B10011000};
  byte j = array[i];
  return j;
}
```