Final Report

Arduino Snooze-Proof Alarm Clock

by Jethro Leroux

MECH307
Mechatronics
Colorado State University
Dr. Haile Endeshaw
5/2/2021

Table of Contents

Table of Contents 1
Design Summary 2
System Details 3
Design Evaluation 8
  Justifications for Grade Adjustments 9
Partial Parts List 10
Lessons Learned 11
Appendix 13
  Alarm-Clock Program Flowchart 13
  Alarm-Clock Code 15
  Alarm-Clock Wiring Diagram 23

Design Summary

This project was designed to fix my deep-sleeping friend's bad habits. He never gets up after his phone alarm goes off and he just hits 'snooze'. That is why I have designed this perfect alarm clock to make sure he wakes up every morning. As shown in Figure 1, the device will have an LED for light noise, a buzzer for normal noise, and 2 DC motors on the side to cause chaotic motion and noise to force him to get out of bed to stop the alarm. There will be no snooze capabilities, which will hopefully force him to get up the first and only time the clock alarms. Additionally, the clock will have a photoresistor to be able to wake him when the sun comes up so he can have a productive day for once.

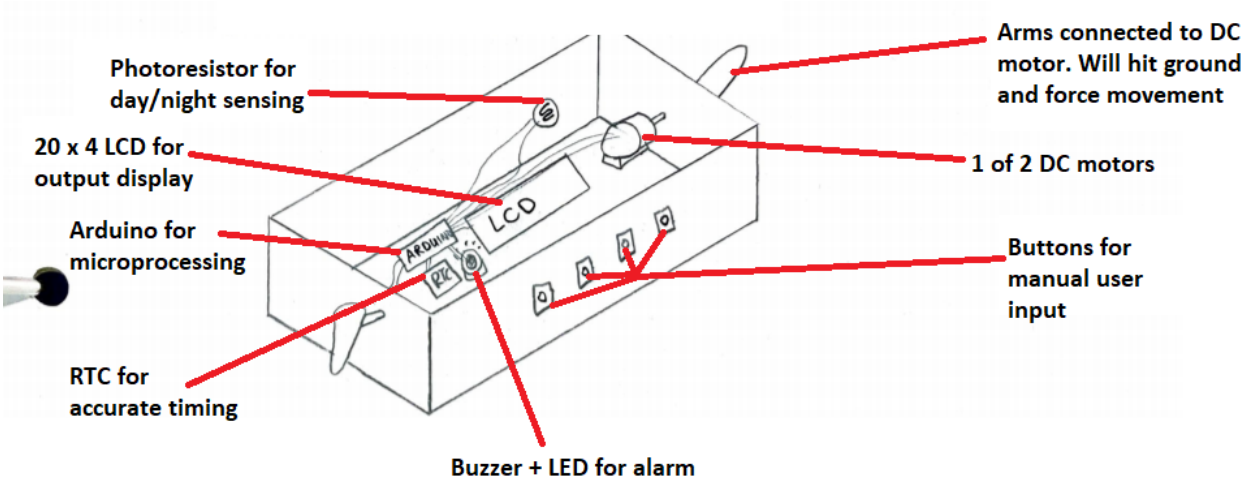


Figure 1: Entire Assembly Sectioned Labeled Isographic View

The motors and connecting arms can be better visualized as shown in Figure 2, below.

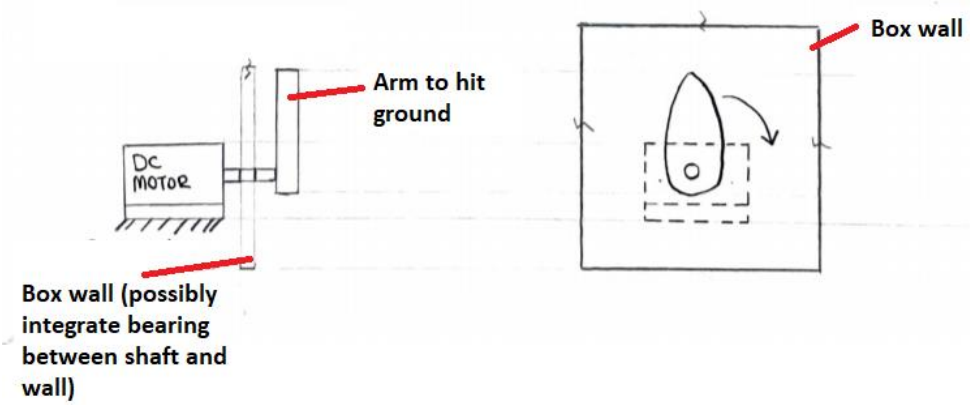


Figure 2: Mechanism Labeled Orthographic View

System Details

There were 3 modes for the alarm clock alarm setting: timed, sunrise, and off. The user can press button 4 (buttons labeled incrementally from left to right) to toggle between the alarm modes. The timed alarm setting allows the user to input an alarm time by pressing buttons 2 and 3 simultaneously to access the alarm time set menu. After a time has been set, and the system has been put in the alarm time mode, the Arduino will check for if the current time (hour and minute) is equal to the set alarm time. The current time is constantly returned to the Arduino via the RTC module, and button 1 could be used to access a menu to set the current time and upload the updated user-input time into the Arduino. If so, it will activate the alarm over that entire minute, or until the system has had its alarm mode set to 'off' via button 4. The photoresistor alarm mode makes use of the photoresistor and Arduino input to check for a voltage value over the photoresistor that is high enough for what is believed to be the light emitted from the sun in the morning.

When the alarm is activated, an LED blinks, a buzzer changes between notes 'A5' and 'A6' simultaneously simulating a siren, and the motors spin in opposite directions, hitting the ground and causing chaotic motion and noise. The LCD backlight will also flash bright and dull at the same frequency as the buzzer and LED when the alarm is activated.

Overall, the device functioned exactly as imagined. It also looked very similar in its final design compared to the initial idealized illustrations. In Figure 3 below, the basic inputs and outputs of the system can be visualized to gain a brief and concise understanding on how the device works altogether.

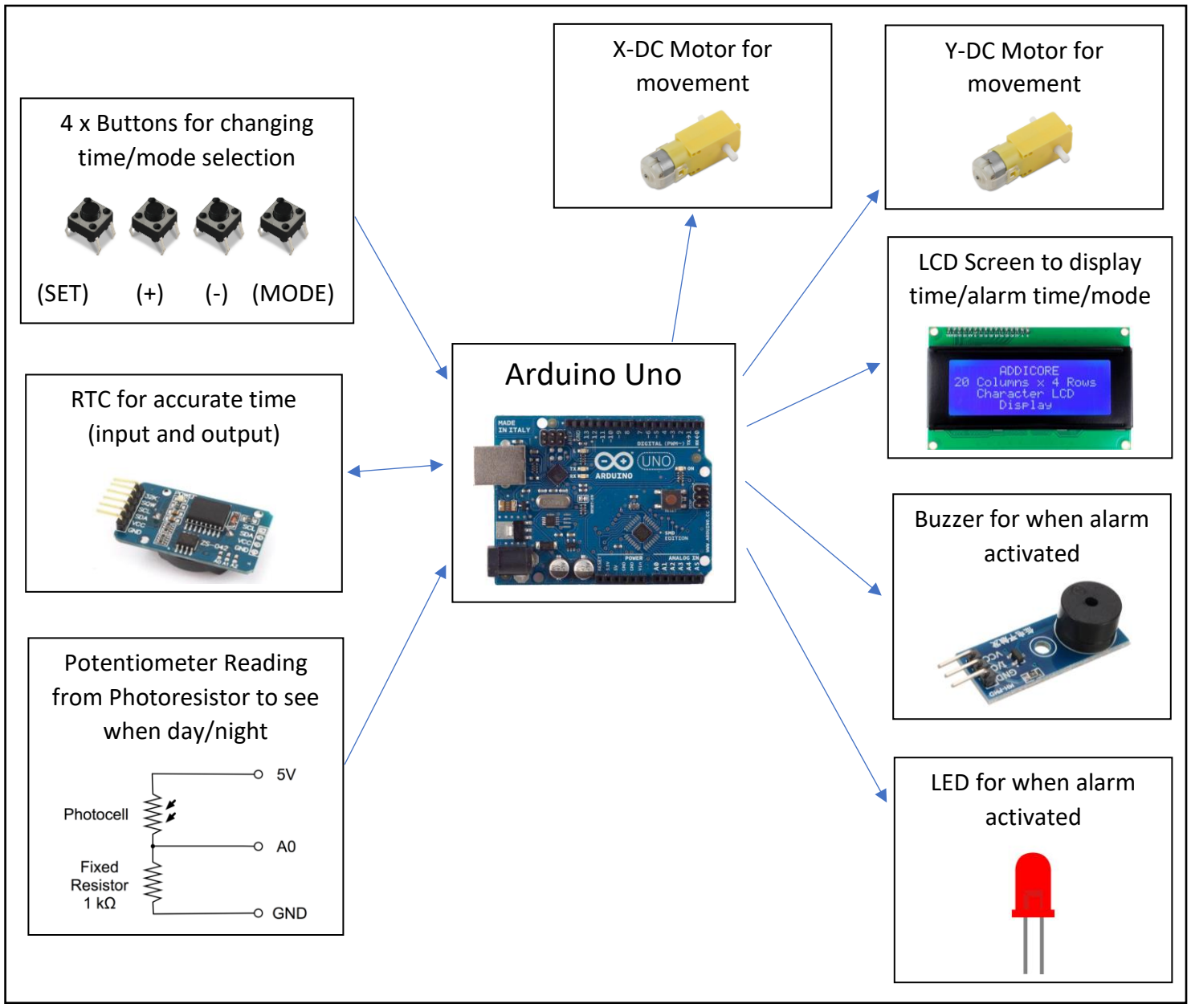


Figure 3: Device Overall Functional Diagram

The system of connections of all the electronic components listed above in Figure 3, evolved into Figure 4 below, which illustrates all the physical wiring and components together before final assembly.

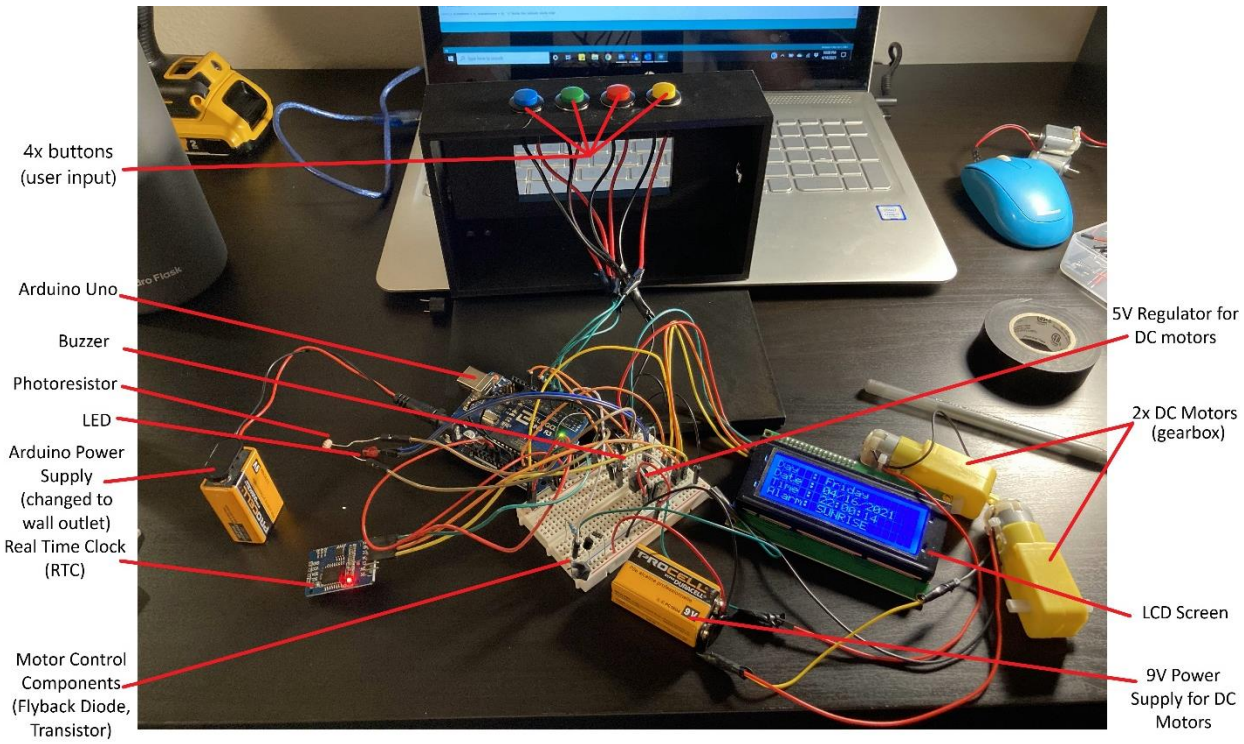


Figure 4: Device Wired Electronic Components Pre-Assembly

Parts where printed using a 3D printer to house the components and form the structure of the alarm clock. 4 parts where printed; the main housing, the lid, and 2 arms (mentioned above), some of which are pictured below in Figures 5 and 6, as well as above in the background of Figure 4.

Figure 5: 3D Printed Arms

Figure 6: 3D Printed Main Housing

When fully put together, and the wiring components were secured to the 3D printed housing components, the system looked as such in Figure 7, below.

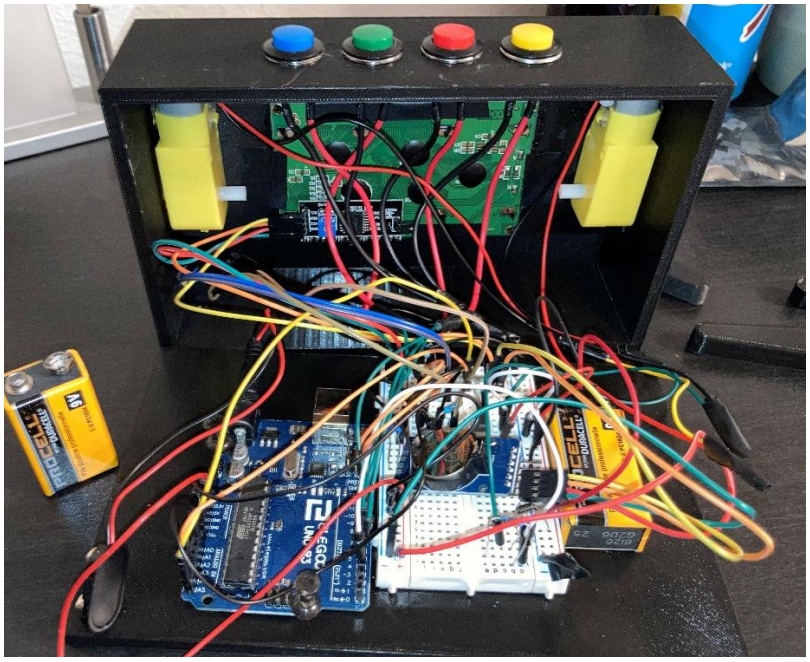


Figure 7: Electrical and Printed Components Assembled

The system ran using instructions via code uploaded to the Arduino Uno. A full software flowchart and well-commented code can be found within the Appendix labeled “Alarm-Clock Program Flowchart” and “Alarm-Clock Code”, respectively. **The code fully describes how the Arduino handles electrical inputs and outputs to and from all the components listed within Figure 3: Device Overall Functional Diagram.** Finally, the last iteration of the system was wired per the schematic in the Appendix, “Alarm-Clock Wiring Diagram.” A final image of the fully assembled system can be seen below in Figure 8.

Figure 8: Final Assembled Alarm Clock

Design Evaluation

Elements present in the final system design within each functional element category are highlighted below.

A. **Output Display (Required)**
   - LED (Red)
   - 7-segment digit display
   - LCD (20x4 IIC)
   - display screen

B. **Audio Output Device (Optional)**
   - buzzer
   - speaker with digitally pre-recorded music or voice
   - speaker with software-generated sound effects
   - speaker with software-controlled synthesized music or voice
   - any of the above with higher volume (e.g., through a transistor or amplifier circuit)

C. **Manual User Input** (for interaction with the user) **(Required)**
   - switch or button (x4)
   - potentiometer
   - joystick
   - keypad
   - keyboard
   - touch screen

D. **Automatic Sensor** (for response without user input) **(Required)**
   - limit or proximity switch
   - photo-optic pair
   - potentiometer
   - photocell (photoresistor)
   - temperature sensor
   - accelerometer
   - encoder

E. **Actuators, Mechanisms & Hardware (Required)**
   - actuators:
     - RC servo motor
     - solenoid
     - on-off dc motor
     - reversible dc motor
     - stepper motor (unipolar, bipolar)
     - PWM speed-controlled motor (x2)
   - mechanisms and hardware:
     - solid and reliable mechanical design, manufacturing, and assembly

- interesting and effective use of linkages, cams, screws, levers, gears, etc.
- appropriate and effective use of 3D-printed and machined parts (3D Printed clock housing, lid, and 2x arms)

F. **Logic, Processing, and Control; AND Miscellaneous (functional elements not covered in the categories above) (Optional)**
    - open-loop control
    - programmed logic (button response, multiple modes)
    - menu-driven software (alarm setting/time changing)
    - calculations and data storage/retrieval (from RTC)
    - advanced and/or multiple interfaced microcontrollers
    - closed-loop feedback control
    - components not included in other categories
        - Real Time Clock (RTC)
        - PN2222A transistor
        - 1 x 1N4007 diodes
        - 2 x 220 ohm resistors, 1 x 10k ohm resistor

Justifications for Grade Adjustments

I believe that I went above and beyond in the creation of my well manufactured 3D-printed housing for the clock. Not only do I have a fully functional system, but a well designed and robust housing to encompass my electronics. The housing was well designed to keep all the electronics secured within. The LCD screen is perfectly flush which took some adjustments to the housing; however, it provided a very attractive end result that I am proud of and will keep on my bedside table for years to come. The inside of the main housing had to be slightly complex to account for this, as well as having the motors mounted at the correct height for the arms to work and hit the ground adequately. The housing is small enough to be attractive and not clunky, but just large enough to fit all the components.

I also used a buzzer, which was within an optional and not required category. Great menu-controlled software was also implemented perfectly, with the clock running and acting exactly like an off-the-shelf alarm clock would.

Partial Parts List

Item	Vendor	Cost
DS1307 I2C RTC	Diymore (Amazon)	$4.50
2004 I2C LCD	Sunfounder (Amazon)	$12.99
9V AC/DC Wall Adapter	TB Tbuymax (Amazon)	$9.67
TT DC Gearbox Motor (x2)	S-snail (Amazon)	$5.99
L7805 5V Regulator	MCIGICM (Amazon)	$1.00
Momentary SPST Button (x4)	Weideer (Amazon)	$6.99
TOTAL		**$41.14**

Lessons Learned

Coding

For the coding part of my project, I used years of knowledge of JAVA programming to help me. Obviously, everyone does not have this knowledge, so I suggest finding a great source online to start off your code. For example, I used some code from a project, here: https://create.arduino.cc/projecthub/tittiamo68/alarm-clock-f61bad. This code formed a great foundation for the rest of my code to be added to, immensely reducing the work I had to do on the coding side of things and allowing me time to work on other parts of the mechatronics process, such as the categories below.

Electronic and Wiring Assembly

I cannot stress enough how little I know about these types of things and how they work. The best skill you can learn from the Mechatronics labs is how to troubleshoot when something is not working as it should. Many times, I would add a component and then my system would stop working. For example, when I added my DC motors, my LCD screen would start to show weird characters when the motors turned off after running. Google did not help me too much, I just had to try new things. I added an external power source to make sure that the motors were not drawing current that the LCD needed from the Arduino. That did not work so I added a capacitor to my main + and – lines on the breadboard. That seemed to help a bit, so I bought a 5V voltage regulator – that fixed everything. It turned out that turning the 2 motors off would cause some fluctuations in my system that somehow made its way to my LCD – I am not sure how because I was using a flyback diode, but the regulator over the 9V source for the motors fixed my issue. This is just an example of how I troubleshot my project to get the working final product I had today. Obviously not many people are going to be electrical engineering experts in a Mech course, so troubleshooting and a growth mindset/attitude are key to getting your project to run – persevere!

3D Printing

I learnt a lot about 3D printing from this project. Here are a few tips:

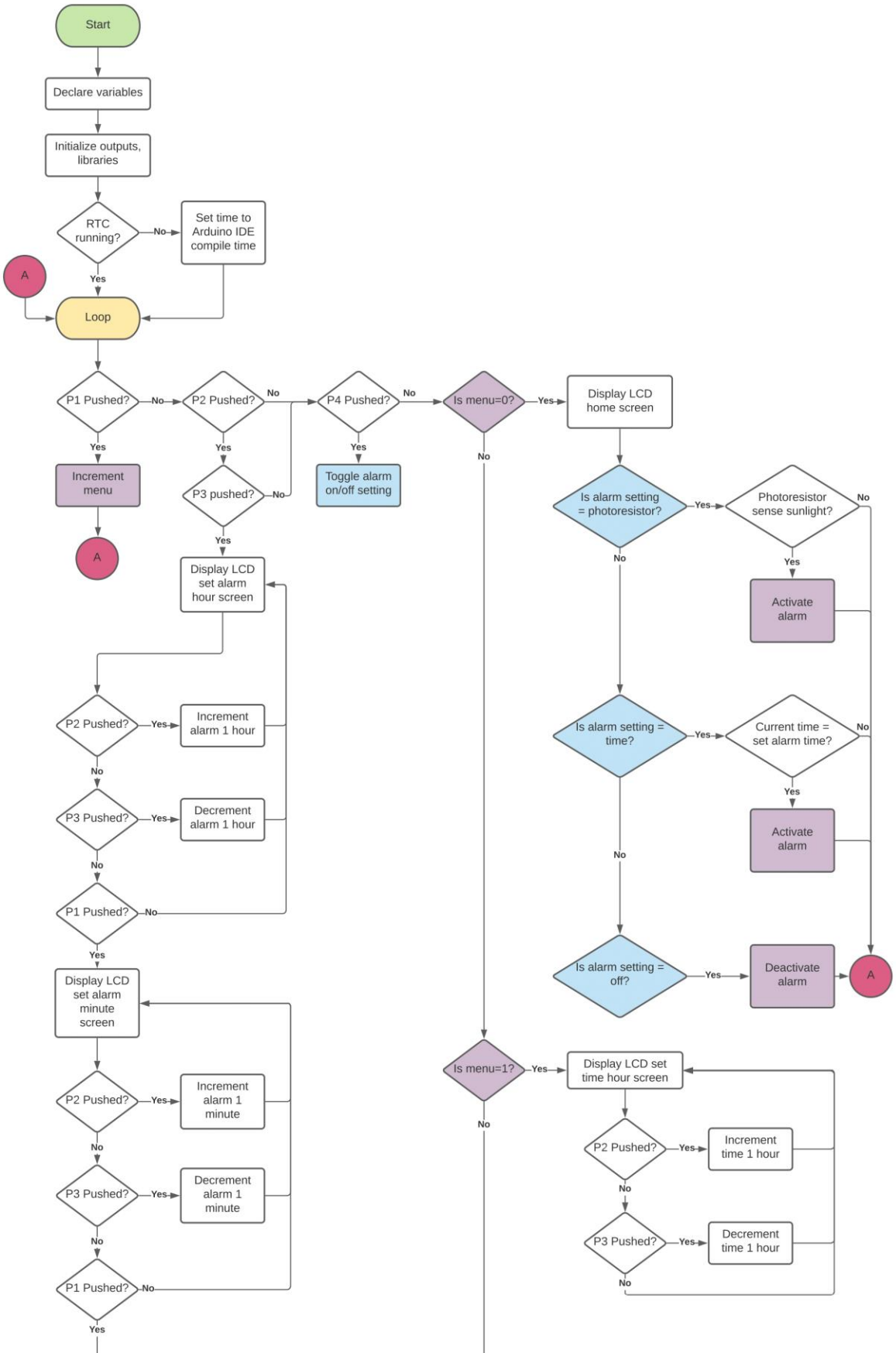
- Keep your 3D printer isolated from drafts, temperature changes and vents. The plastic is so sensitive to changing temperatures that even being next to a closed window on a cold day can cause your plastic to warp and have your part knocked off the bed mid-print.
- Printing is going to take longer than you think – allow yourself lots of time, especially if you are printing at CSU's I2P lab.
- Use decent filament – cheap filament will mean a bad or unfinished print.
- Keep the part on the bed for at least 30 minutes after the print is done. Taking it off immediately can cause it to warp, it needs to cool down slowly and uniformly. Let larger, skinnier parts such as a thin rectangular prism stay on longer.

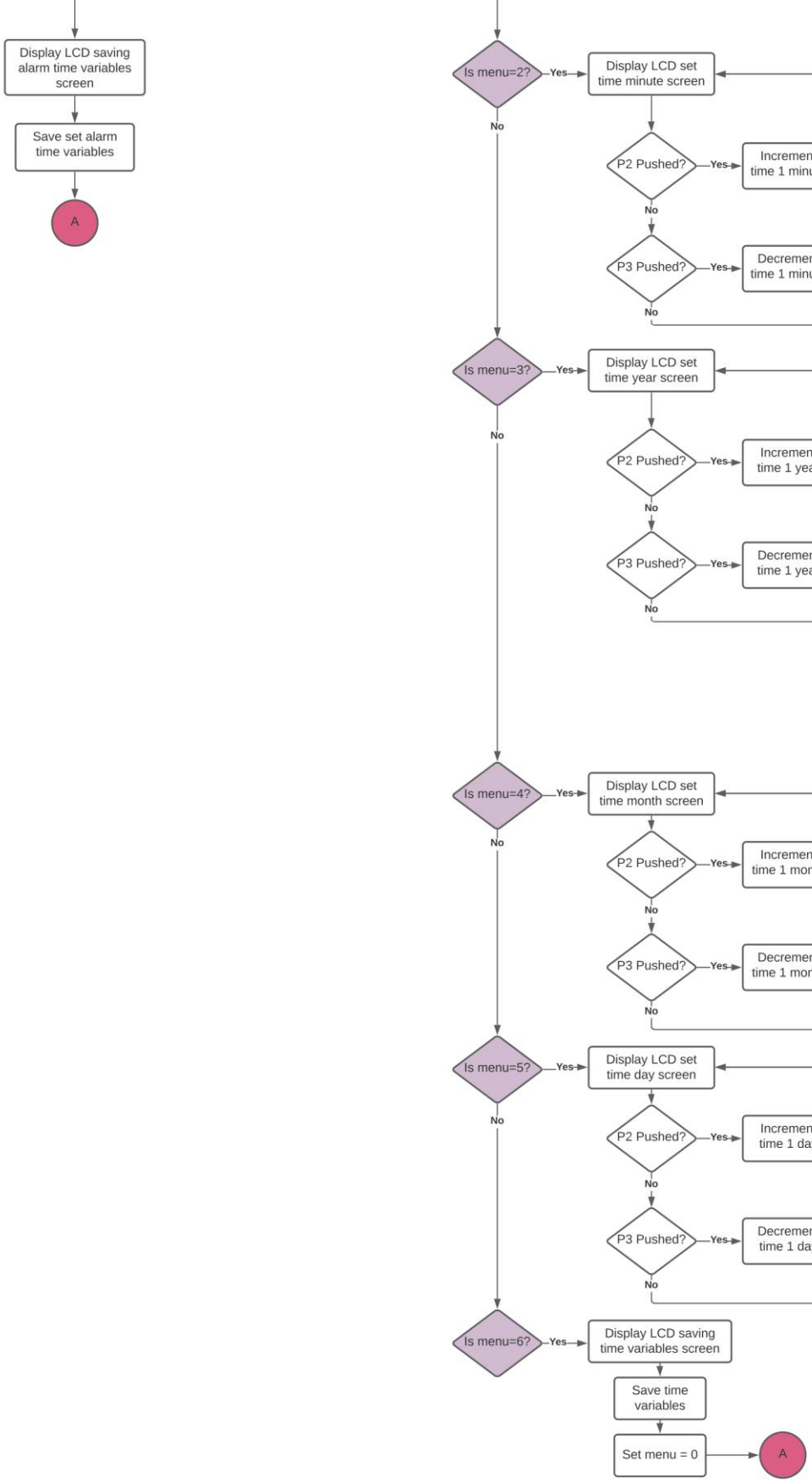
Deadlines (General Projects)

Start early. I started my project in March. I was done a few weeks before the due date. Not only does troubleshooting take a lot of time, but this will help some of the stress. You will also be able to find more 3D printing spots available at the I2P lab – towards the end of the semester, there are none. Nothing ever goes according to plan when you do a project on new topics for the first time such as this. That is why you need to start early, figure out what works and what does not and even reorder new or different parts if the ones you bought first do not work. Orders can take a long time to be delivered, depending on where you get them from, so do not let that effect your project finish date.

Appendix

Alarm-Clock Program Flowchart




Display LCD saving alarm time variables screen
Save set alarm time variables
A
Is menu=2?
Yes
Display LCD set time minute screen
No
P2 Pushed?
Yes
Increment time 1 minute
No
P3 Pushed?
Yes
Decrement time 1 minute
No
Is menu=3?
Yes
Display LCD set time year screen
No
P2 Pushed?
Yes
Increment time 1 year
No
P3 Pushed?
Yes
Decrement time 1 year
No
Is menu=4?
Yes
Display LCD set time month screen
No
P2 Pushed?
Yes
Increment time 1 month
No
P3 Pushed?
Yes
Decrement time 1 month
No
Is menu=5?
Yes
Display LCD set time day screen
No
P2 Pushed?
Yes
Increment time 1 day
No
P3 Pushed?
Yes
Decrement time 1 day
No
Is menu=6?
Yes
Display LCD saving time variables screen
Save time variables
Set menu = 0
A

Alarm-Clock Code

The code for my project was adapted from: https://create.arduino.cc/projecthub/tittiamo68/alarm-clock-f61bad

```
/*
:Project:Allarm_Clock
:Author: Tiziano Bianchettin
:Date: 10/02/2017
:Revision: 2
:License: Public Domain
thanks to:
  http://arduinoenonsolo.blogspot.it/2012/12/orologio-con-arduino-e-il-ds1307.html
  http://www.mauroalfieri.it/
  http://www.danielealberti.it/
  http://www.maffucci.it/
  My electronics laboratory professor "Perito Carli"
*/
//************libraries**************//
#include <Wire.h>
#include <RTClib.h> //import RTC library
#include <LiquidCrystal_I2C.h>  //import LCD I2C protocol library

//************************************//
LiquidCrystal_I2C lcd(0x27,20,4); // Display  I2C 20 x 4
RTC_DS3231 RTC; //create object for RTC

//************Button*****************//
int P1=6; // Button SET MENU
int P2=7; // Button +
int P3=8; // Button -
int P4=9; // SWITCH Alarm

//**************Alarm***************//
#define LED 13  //output to LED
#define buzzer 10
int motors = 3;

//************Variables**************//
int hourupg;
int minupg;
int yearupg;
int monthupg;
int dayupg;
int menu =0;  //menu variable seen in program flowchart
int setAll =0;  //alarm toggle variable seen in flowchart

uint8_t alarmHours = 0, alarmMinutes = 0;  // Holds the current alarm time


void setup()
{

  lcd.begin();
  lcd.backlight();
  lcd.clear();

  pinMode(P1,INPUT_PULLUP); // https://www.arduino.cc/en/Tutorial/InputPullupSerial
  pinMode(P2,INPUT_PULLUP); // used pullup resistors of Arduino instead of physical pulldown
resitors in breadboard
  pinMode(P3,INPUT_PULLUP);
  pinMode(P4,INPUT_PULLUP);
  pinMode(LED,OUTPUT);
  pinMode(buzzer, OUTPUT); // Set buzzer as an output
  pinMode(motors, OUTPUT);// set dc motors as output
  printAllOff();
  Wire.begin();
  Serial.begin(9600);
  RTC.begin();
```

```
  if (! RTC.begin()) {
    Serial.println("RTC is NOT running!");
    // Set the date and time at compile time
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }
  int menu=0;
}

void loop() //start of 'A' in program flowchart
{

// check if you press the SET button and increase the menu index
  if(digitalRead(P1)== LOW) //P1 pushed
  {
   menu=menu+1;
  }

  if((digitalRead(P2)== LOW)&&(digitalRead(P3)== LOW))      //P2 and P3
pushed

  {

    DisplaySetHourAll();
    DisplaySetMinuteAll();
    lcd.clear();
    lcd.setCursor(5,0);
    lcd.print("SETTING ALARM");
    lcd.setCursor(5,1);
    lcd.print(alarmHours, DEC);
    lcd.print(":");
    lcd.print(alarmMinutes, DEC);
    delay(1000);
    lcd.clear();
  }
// in which subroutine should we go?
  if (menu==0)  //all menu options based on menu variable shown in program flowchart
    {
     DisplayDateTime(); // void DisplayDateTime
     Alarm(); // Alarm control
          }
  if (menu==1)
    {
    DisplaySetHour();
    }
  if (menu==2)
    {
    DisplaySetMinute();
    }
  if (menu==3)
    {
    DisplaySetYear();
    }
  if (menu==4)
    {
    DisplaySetMonth();
    }
  if (menu==5)
    {
    DisplaySetDay();
    }
  if (menu==6)
    {
    StoreAgg();   //save alarm time variables
    delay(500);
    menu=0;     //reset the menu
    }
    delay(100);
}

void DisplayDateTime () //bunch of LCD control methods to display main home/time screen
{
```

```
// We show the current date and time
  DateTime now = RTC.now();

  lcd.setCursor(0, 2);
  lcd.print("Time : ");

  if (now.hour()<=9)
  {
    lcd.print("0");
  }
  lcd.print(now.hour(), DEC);
  hourupg=now.hour();
  lcd.print(":");
  if (now.minute()<=9)
  {
    lcd.print("0");
  }
  lcd.print(now.minute(), DEC);
  minupg=now.minute();
  lcd.print(":");
  if (now.second()<=9)
  {
    lcd.print("0");
  }
  lcd.print(now.second(), DEC);


  lcd.setCursor(0, 1);
  lcd.print("Date : ");
  if (now.month()<=9)
  {
    lcd.print("0");
  }
  lcd.print(now.month(), DEC);
  monthupg=now.month();
  lcd.print("/");

  if (now.day()<=9)
  {
    lcd.print("0");
  }
  lcd.print(now.day(), DEC);
  dayupg=now.day();
  lcd.print("/");

  lcd.print(now.year(), DEC);
  yearupg=now.year();

  char DOW[][10]={"Sunday   ","Monday   ","Tuesday  ","Wednesday","Thursday
","Friday   ","Saturday "};
  lcd.setCursor(0, 0);
  lcd.print("Day  : ");
  lcd.print(DOW[now.dayOfTheWeek()]); // if it appears error in the code, enter the code given
below
  //lcd.print(DOW[now.dayOfWeek()]);
}

void DisplaySetHour() //if menu variable ==1
{
// time setting
  lcd.clear();
  DateTime now = RTC.now();
  if(digitalRead(P2)==LOW)
  {
    if(hourupg==23)
    {
      hourupg=0;
    }
    else
    {
      hourupg=hourupg+1;
```

```
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if(hourupg==0)
    {
      hourupg=23;
    }
    else
    {
      hourupg=hourupg-1;
    }
  }
  lcd.setCursor(0,0);
  lcd.print("Set Hour:");
  lcd.setCursor(0,1);
  lcd.print(hourupg,DEC);
  delay(200);
}

void DisplaySetMinute()//if menu variable ==2
{
// Setting the minutes
  lcd.clear();
  if(digitalRead(P2)==LOW)
  {
    if (minupg==59)
    {
      minupg=0;
    }
    else
    {
      minupg=minupg+1;
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if (minupg==0)
    {
      minupg=59;
    }
    else
    {
      minupg=minupg-1;
    }
  }
  lcd.setCursor(0,0);
  lcd.print("Set Minutes:");
  lcd.setCursor(0,1);
  lcd.print(minupg,DEC);
  delay(200);
}

void DisplaySetYear()//if menu variable ==3
{
// setting the year
  lcd.clear();
  if(digitalRead(P2)==LOW)
  {
    yearupg=yearupg+1;
  }
   if(digitalRead(P3)==LOW)
  {
    yearupg=yearupg-1;
  }
  lcd.setCursor(0,0);
  lcd.print("Set Year:");
  lcd.setCursor(0,1);
  lcd.print(yearupg,DEC);
  delay(200);
}
```

```
void DisplaySetMonth()//if menu variable ==4
{
// Setting the month
  lcd.clear();
  if(digitalRead(P2)==LOW)
  {
    if (monthupg==12)
    {
      monthupg=1;
    }
    else
    {
      monthupg=monthupg+1;
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if (monthupg==1)
    {
      monthupg=12;
    }
    else
    {
      monthupg=monthupg-1;
    }
  }
  lcd.setCursor(0,0);
  lcd.print("Set Month:");
  lcd.setCursor(0,1);
  lcd.print(monthupg,DEC);
  delay(200);
}

void DisplaySetDay()//if menu variable ==5
{
// Setting the day
  lcd.clear();
  if(digitalRead(P2)==LOW)
  {
    if (dayupg==31)
    {
      dayupg=1;
    }
    else
    {
      dayupg=dayupg+1;
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if (dayupg==1)
    {
      dayupg=31;
    }
    else
    {
      dayupg=dayupg-1;
    }
  }
  lcd.setCursor(0,0);
  lcd.print("Set Day:");
  lcd.setCursor(0,1);
  lcd.print(dayupg,DEC);
  delay(200);
}

void StoreAgg()
{
// Variable saving
  lcd.clear();
```

```
  lcd.setCursor(0,0);
  lcd.print("SAVING IN");
  lcd.setCursor(0,1);
  lcd.print("PROGRESS");
  RTC.adjust(DateTime(yearupg,monthupg,dayupg,hourupg,minupg,0));
  delay(200);
}
void DisplaySetHourAll()// Setting the alarm minutes
{
  while(digitalRead(P1)==HIGH){

  lcd.clear();

  if(digitalRead(P2)==LOW)
  {
    if(alarmHours==23)
    {
      alarmHours=0;
    }
    else
    {
      alarmHours=alarmHours+1;
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if(alarmHours==0)
    {
      alarmHours=23;
    }
    else
    {
      alarmHours=alarmHours-1;
    }
  }
  lcd.setCursor(0,0);
  lcd.print("Set HOUR Alarm:");
  lcd.setCursor(0,1);
  lcd.print(alarmHours,DEC);
  delay(200);
 }
 delay(200);
}

void DisplaySetMinuteAll()// Setting the alarm minutes
 {
  while(digitalRead(P1)==HIGH){

  lcd.clear();
  if(digitalRead(P2)==LOW)
  {
    if (alarmMinutes==59)
    {
      alarmMinutes=0;
    }
    else
    {
      alarmMinutes=alarmMinutes+1;
    }
  }
   if(digitalRead(P3)==LOW)
  {
    if (alarmMinutes==0)
    {
      alarmMinutes=59;
    }
    else
    {
      alarmMinutes=alarmMinutes-1;
    }
  }
```

```
  lcd.setCursor(0,0);
  lcd.print("Set MIN. Alarm:");
  lcd.setCursor(0,1);
  lcd.print(alarmMinutes,DEC);
  delay(200);
 }
 delay(200);
}
void printAllOn(){  //method to print what the current alarm setting is
   lcd.setCursor(0,3);
  lcd.print("Alarm: ");



  if (alarmHours <= 9)
  {
    lcd.print("0");
  }
  lcd.print(alarmHours, DEC);

  lcd.print(":");
  if (alarmMinutes <= 9)
  {
    lcd.print("0");
  }
  lcd.print(alarmMinutes, DEC);

}
void printAllOnPhoto(){//method to print what the current alarm setting is
   lcd.setCursor(0,3);
  lcd.print("Alarm: SUNRISE");
}
void printAllOff() {//method to print what the current alarm setting is
  lcd.setCursor(0, 3);
  lcd.print("Alarm: Off    ");
}
void Alarm(){ //alarm control
   if(digitalRead(P4)== LOW)
  {
   setAll=setAll+1;
  }
  if (setAll==0)
    {
     printAllOff();
     noTone (buzzer);
     digitalWrite(LED,LOW);
     digitalWrite(motors,LOW);
     }
  if (setAll==1) //time alarm setting
    {

     printAllOn();

     DateTime now = RTC.now();
     if ( now.hour() == alarmHours && now.minute() == alarmMinutes ) //current time = alarm set
time?
        {
         lcd.noBacklight();
         DateTime now = RTC.now();
         digitalWrite(LED,HIGH);
         tone(buzzer,880); //play the note "A5" (LA5)
         digitalWrite(motors,HIGH);
         delay (300);
         tone(buzzer,698); //play the note "F6" (FA5)
         lcd.backlight();
        }
    else{ //turn alarm off
         noTone (buzzer);
         digitalWrite(LED,LOW);
         digitalWrite(motors,LOW);
        }
```

```
    }
     if (setAll==2) //photoresistor setting
    {

     printAllOnPhoto();

     DateTime now = RTC.now();
     int lightVal = analogRead(A0); //read potentiometer value
     if ( lightVal > 300 )// tested value for morning sunlight through bedroom window
        {
         lcd.noBacklight();
         DateTime now = RTC.now();
         digitalWrite(LED,HIGH);
         tone(buzzer,880); //play the note "A5" (LA5)
         digitalWrite(motors,HIGH);
         delay (300);
         tone(buzzer,698); //play the note "F6" (FA5)
         lcd.backlight();
        }
    else{
         noTone (buzzer);
         digitalWrite(LED,LOW);
         digitalWrite(motors,LOW);
        }

    }
  if (setAll==3) //reset alarm menu
    {
     setAll=0;
    }
    delay(200);
}
```

Alarm-Clock Wiring Diagram