

Project Report “Refrigerator Alert System”

Group 2:
Chad Huard
Prabhjot Kaur
Sean Lechkun
Gary Garcia

BE1200
Instructor: Dr. E. Ayorinde

Article I. Abstract

A common problem in homes is that the refrigerator door is often left open which causes premature food spoilage, food borne illness and increased power consumption. It therefore seems advisable to employ an alert system that ensures that the refrigerator door is closed in a timely fashion. Our team proposes a solution. Using an arduino kit we have created a design that keeps the user alerted to the temperature in his/her refrigerator and alerts him or her when the door has been open too long. The report discusses the main objective and the proposed solution in detail.

Article II. Background Material / Case Studies

Section 2.01 Food Storage Temperatures

According to the Food Safety and Inspection Service, a division of the United States Department of Agriculture (USDA):

“Bacteria grow most rapidly in the range of temperatures between 40 and 140 °F, the “Danger Zone,” some doubling in number in as little as 20 minutes. A refrigerator set at 40 °F or below will protect most foods [1].”

This indicates that the refrigerator should never reach 40°F, but on most refrigerators the user would never know if temperature had risen to dangerous levels. Furthermore, while keeping the inside temperature below 40°F will prevent the growth of harmful bacteria it does not stop the bacteria that cause spoilage [1]. Spoilage bacteria will grow at in any refrigeration temperature, but will grow more slowly as temperature decreases. Optimal food storage temperatures are as follows [2]:

<i>Food</i>	<i>Optimal Temp.</i>
Dairy	34°F - 38°F
Meat	33°F - 36°F
Fruit & Veg.	35°F - 40°F

Section 2.02 Temperature Change Study

In order to fully understand how long the door must be left open before it adversely affects the food storage conditions we studied the temperature change of the refrigerator when the door is left open for different periods of time.

- (a) Trial 1 [Figure 1]: For the first trial we left the refrigerator open for 150s and allowed it to cool for another 300s. The pattern is quite linear. With the door open the temp

increases at about 3°/min, and when closed the refrigerator will cool at about half that rate (1.5°/min).

- (b) Trial 2 [Figure 2]: The second trial only left the door open 60s. The temp increase when the door is open is very similar at about 3°/min, but we can see that as the temp returns to its lower range the cooling slows considerably to less than .5°/min below 35.5°. This very slow cooling means it takes almost 20-30 min for the refrigerator to recover to its original temperature after being left ajar for this time period.
- (c) Trial 3 [Figure 3]: In this trial the door was left open for 30s, which is much closer to an average use than the previous trials. This trial was repeated 3 times. These trials show the standard 3°/min rise with the door open, but show a much slower cooling when the door is closed. The 3 iterations were spaced 5min apart. In the 5min between trials the refrigerator was not able to cool down to the original temp in that time as shown by the stacked plots in the graph.

Article III. Project Objectives / Design Criteria

Section 3.01 Main Objective

The main objective of this project is to create a prototype device that will alert the user when the internal temperature of the refrigerator is not at optimal levels for food storage. Specific cases that should be addressed are when the temperature is raised due to the door being left ajar and when the temperature raises due to refrigerator malfunction.

Section 3.02 Project Specific Design Criteria

- (a) Provide user with refrigerator temperature information.
 - i) Provide text readout of temperature on LCD screen.
 - ii) Change LED color to correspond with internal temp.
 - iii) Provide active warning when temperature is not within optimal storage temperature range.
- (b) Provide multiple levels of alarm to prevent the temperature from reaching unsafe levels due to leaving the door ajar.
 - i) Use flashing light to alert user that internal temp has risen 3°F since door was opened.
 - ii) Use audible alarm when door has been open for longer than 45s.
 - iii) Use servo to close door after being left ajar more than 60s (optional).
- (c) Alert user when temperature has risen to unsafe levels in event of refrigerator malfunction.
 - i) Use flashing red LED when temp is above 39°F even if door is closed.

Article IV. Proposed Solution

Section 4.01 Overview

Our proposed solution consists of a two box solution, with a control unit mounted outside of the refrigerator and a sensor unit on the inside. The decision to use two boxes arose due to the possibility of condensation taking place in the refrigerator. While the arduino board is rated for use in temperatures much lower than a refrigerator, condensation can quickly cause shorts and damage the electronics.

The sensor unit that will go inside of the refrigerator will house a light sensor and temperature sensor. The light sensor will be used to judge if the door is open or closed by sensing the light from the refrigerators internal light or outside light coming in through the open door [3]. Its sensitivity is set fairly high so very little light is needed for the arduino to register the door as open, in case the door is partially closed and the internal light has been turned off. The temperature sensor continuously reads the internal temperature and converts it to an analog voltage that the arduino board can then convert directly into °C and then to °F for display [4].

Output to the user will be provided by an LCD display, a RGB LED that will change color with temperature fluctuations and audio buzzer to alert the user when the door has been left ajar too long. Additionally, when the door has been open for long enough to assume the user has forgotten to close it even with provided warnings we will implement a servo motor to shut the door [Figures 5 & 6].

Due to the design criteria of monitoring and displaying temperature constantly we decided to split the processing between two arduino boards. One board will be responsible for all of the continuous functions, including monitoring the sensor unit and displaying temperature via the LCD and LED. The other arduino will be responsible for playing the audio alarm and closing the door via the servo. The necessity for this arose from the inability of the arduino board to play an alarm tone and read the input devices or run any other output simultaneously.

Section 4.02 Control Loop

The main loop of the control board is used as a central monitoring service. It is responsible for monitoring temperature and door state and passing that information on to the other elements of the design [Figure 7].

Both the temperature sensor and the light sensor required some processing before the readings were published to the rest of the functions. The temperature sensor is very noisy so a reading is only taken every 200ms. The reading is stored in an array which is used to take a rolling average of the last 6 readings, smoothing the readings and eliminating noise. The light sensor is also prone to some noise, so a de-bounce technique is employed. When a light sensor reading indicates that the door state has changed the board waits 5ms then re-measures. Only if the reading is still different than the previous door-state is the current door-state updated [5].

Section 4.03 Timer Function

The timer function is called with an argument that lets the timer know if it should turn on, turn off or increment. When turned on, it reads the current time and begins keeping track of the time elapsed since the door opened. The timer function is also responsible for keeping track of how much the temperature has changed since the door opened and calling the 2nd arduino board to start the audio alarm after 45s has elapsed since the door opened.

Section 4.04 **LCD Function**

The LCD function uses the LiquidCrystal.h library that is included with the arduino distribution to attach and send data to a 16x2 LCD screen [6] [7]. The function displays different data depending on if the door is open or closed. When the door is closed the display shows the current internal temperature. When the door is open it displays a notification that the door is ajar and the number of degrees the temperature has risen in the refrigerator since the door was opened.

Section 4.05 **BlinkM Function (RGB LED)**

The BlinkM is a RGB LED with built-in microcontroller to simplify programming [8]. The light() function is responsible for communicating with the BlinkM through an I²C serial connection. The function relies on the BlinkM_funcs.h file from BlinkM which in turn uses the Wire.h library from arduino to make the I²C connection [9].

When the door is closed the color of the light is dependent on the temperature, with 34°F being blue, 39°F being red and any temperature between the two being a proportional mixture of the two. Any time the temperature exceeds 39°F the light begins to blink red, warning the user that the refrigerator is no longer at a safe storage temperature. When the door is open the color is dependent on the temperature increase since the door opened, with 3°F being red. Above 3°F the light will blink red. This should allow about 60s before the light begins to blink as seen in our temperature study [Figures 1-3].

The flashing red state of the light is accomplished with the use of a light script that is programmed into the microcontroller on the BlinkM instead of using the arduino to turn the light on or off. This frees up the main arduino board to continue to perform its continuous functions while the light blinks without an extra timer running [Figure 8].

Section 4.06 **2nd Arduino: Alarm & Servo**

The audio alarm and servo will be controlled by a 2nd arduino board. This board is housed in the main control unit alongside the main arduino board and is triggered by a digital out pin on the main board. After the door has been open for 45s the control board sets the alarm pin to HIGH triggering the alarm board. When the alarm pin is HIGH the alarm board immediately starts the alarm and a separate timer. After the alarm has been going off for 15s the alarm board uses the servo to close the door. Once the door has been closed (either by the user or the servo) the control board will stop triggering the alarm board and the alarm board will reset its timer.

The servo is controlled using the Servo.h arduino library [10]. When the door has been open for 60s, it attempts to close the door. If the door is not closed after 5s the alarm will continue to sound for another 15s and then the servo will try again. Closing the door is accomplished by rotating the servo 90° from its at rest position using the write() command. Since a servo reacts to absolute position commands and can only rotate through 180° it must be returned to its rest position using another write() command. In order to accommodate this return rotation without opening the door the servo will be connected to the door with a one directional slip clutch, although that is not implemented in the prototype.

Article V. Design Evaluation / Challenges / Improvements

Section 5.01 **Technical Performance**

The technical performance of this prototype is excellent. We have met all of our design criteria and expanded the original objective in meaningful and productive ways. Our design goes beyond

simply notifying the user that the door has been open too long, offering useful information about temperature that can be directly related to power consumption and food preservation. We also do not rely on a single method of conveying information to the user. Instead we use several ways to notify the user, creating an intuitive, simple and enjoyable user experience.

Challenges / Improvements:

- (a) Alarm: A major challenge was trying to incorporate the audio alarm into the main arduino program. Since the arduino board can only do one thing at a time it is impossible for it to be making a tone (which requires a loop that quickly switches between HIGH and LOW on a pin to simulate a sine wave) and measuring temperature, writing to the LCD or LED or any of the other functions that we want to happen continuously. We were forced to choose between using two arduino boards or alternating between playing the alarm tone and running the other functions. In the end we thought it was worth the extra cost for the temperature to be monitored continuously so we used two arduino boards.
- (b) Temp. Sensor: The temperature sensor uses an analog output. One unforeseen result of this is that when the boards are powered from a battery or from another board the voltage out is not exactly 5v. This makes the accuracy of the temperature sensor lower in an absolute sense, although the relative precision is unaffected. We have tried to account for this lower voltage by measuring the voltage at the sensor with a multi-meter and replacing the 5v in the code with the appropriate value. To properly solve this problem an external voltage reference could be used or the temperature sensor could be replaced with one that uses a digital output.
- (c) Servo: The servo was relatively simple to implement in the arduino code, but the physical design was challenging. In the end we decided that our prototype would show the basic functionality but not be a fully functioning unit. We see this element as an option that could be sold separately and attached to the main control unit. In order for it to be fully functional a much larger servo motor would need to be employed. This would require a separate power supply and a control unit with transistor or optoisolation to allow the arduino board to control the larger voltages needed. There would also need to be a clutch mechanism that connects the servo to the refrigerator door. This clutch would be a one way slip clutch which would allow the servo to disengage when the door closed. It would also be a safety feature, disengaging the servo if it came into contact with a person still using the refrigerator after 60s.

Section 5.02 Ergonomics

The ergonomics of this design is not completely designed at this stage, but the design lends itself to be easily optimized ergonomically. The main control unit can be mounted directly to the refrigerator door, or close by on the wall or counter. The user-interface is simple and easy to understand. Using the color RGB allows the user to see the temperature from across the room. The sensor unit takes up very little space and can be mounted discreetly in the refrigerator. The prototype uses a wire from the control unit to the sensor unit, but the arduino board can easily support wireless communication with a few extra parts. This would increase ergonomics greatly.

Section 5.03 Economy

Our design would cost more to manufacture than a simple alarm, but the extra functionality far outweighs the extra cost. With careful electrical design of the final product the design should be able to run on 4 AA batteries making the operation cost low. Since the design uses all lead free components no expensive disposal or recycling is required.

Section 5.04 **Aesthetics**

The aesthetics of the design are not directly addressed by this prototype. The finished product could be much smaller and thinner than the prototype. With its high-end functionality it would be worth while to emphasize aesthetics in this design. Stainless steel is very popular in custom kitchens and with a similar aesthetic this design would have very good high-end appeal. The design has many features that are eye-catching, particularly the RGB LED.

Section 5.05 **Ecological Impact**

While many of the aspects of the designs ecological impact would be decided in the manufacturing engineering phase, one of the design goals is to lower energy consumption of the user. This makes the conceptual design very strong from an ecological viewpoint. We are also using only components that can be purchased lead free, making the design easy to dispose of without introducing toxins into the environment.

Article VI. Conclusion

Overall this prototype is very successful. We have expanded the objectives of the project and met or exceeded all of our design objectives and criteria. Our design is well researched, well executed, useful and user friendly. Due to our ambitious design goals some elements of the design are in more of a conceptual phase, while other elements are closer to completion. Overall, our group worked well together and produced a very well engineered prototype.

Article VII. Resources

Section 7.01 **Code: Arduino 1 – Main Control Unit**

```
#include <LiquidCrystal.h>
#include <Wire.h>
#include "BlinkM_funcs.h"
#define AVE 6 // Number of samples in temp rolling average
#define FREQ 200 // Frequency of temperature readings
#define DOOR 1 // door sensor PIN (analog)
#define TEMP 0 // temp sensor PIN (analog)
#define ALARM_OUT 9
#define ALARM_TIME 10000
#define L_THRESH 500 // light sensor reading that = door open
#define RED_THRESH 3 // degrees refrigerator must get warmer to turn red w/ door open

// Declare PINS & Constants & Objects:
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2); // LiquidCrystal Display: RS=12, R/W=11, EN=10, D4=5, D5=4, D6=3, D7=2
byte addr = 0x00; // BlinkM address

// Declare global variables:

float cTemp = 40; // current temperature
float dTemp = 0; // change in temp (while timer is running)
float startTemp = 0; // temp when door opens
boolean doorRead = false;
boolean doorState = false; // true=open, false=closed
boolean timerState = false; // true=timer on, false=timer off
```

```

long startTime = 0;           // time millis() when door opens
long dTime = 0;             // change in time (actual timer)
int readingTimer = 0;       // time since last temp reading
long prevReading = 0;       // time of previous reading in millis()
int tempReading;           // direct reading of analog temp signal (0-1023)
int tempArray[AVE];        // array to hold the temp readings to be averaged
float aveTempRead;         // average of temp readings in tempArray

float defaultBlueTemp = 68; // temperature below which light is completely blue
float defaultRedTemp = 80;  // temperature above which the light is completely red
float blueTemp;
float redTemp;
int prev_r = 0;

void setup(){
  lcd.begin(2, 16);          // set up the LCD's number of rows and columns
  Serial.begin(9600);
  pinMode(ALARM_OUT, OUTPUT);
  digitalWrite(ALARM_OUT, LOW);
  BlinkM_begin();          // init BlinkM funcs
  BlinkM_stopScript(addr); // stop the BlinkM startup script
}

void loop()
{
  //*****CHECK DOOR STATE*****

  int lightRead = analogRead(DOOR); // read check if door is open or closed

  if (lightRead > L_THRESH)
  {
    doorRead = true;
  }else{
    doorRead = false;
  }

  if (doorRead != doorState) // if doorState has changed
  {
    delay(5); // wait 5ms (for debounce)
    lightRead = analogRead(DOOR); // check the door again, to make sure it wasn't noise or bounce
    if (lightRead > L_THRESH)
    {
      doorRead = true;
    }else{
      doorRead = false;
    }
  }
  if (doorRead != doorState) // if door state still doesn't = old door state
  {
    doorState = !doorState; // change doorstate. If it was open, it is now closed and vice versa
    if (doorState == true) // if it is open start the timer
    {
      timer(2); // START timer
    }else { // if it has just been closed stop the timer
      timer(3); // STOP timer
      digitalWrite(ALARM_OUT, LOW);
    }
  }
}

//*****READ TEMP*****

if (readingTimer > FREQ) // check to see if the last reading was longer than FREQ ago, if it was:
{
  int i;
  for (i = AVE-1; i > 0; i = i-1) // loop through tempArray cells [1] through [AVE] (not [0])
  {
    int prev = i-1; // move all values of tempArray one place to right
  }
}

```

```

tempArray[i] = tempArray[prev]; // this will dump oldest measurement and make room in tempArray[0] for new
temp
}

tempReading = analogRead(TEMP); // take actual reading
tempArray[0] = tempReading; // put reading in first cell of tempArray

aveTempRead = 0; // clear the previous value in aveTempRead
for (i = 0; i < AVE; i++) // loop through all tempArray cells
{
aveTempRead = aveTempRead + tempArray[i]; // sum of all tempArray cells
}
aveTempRead = aveTempRead / AVE; // convert sum to average of tempArray cells

float tempVoltage = aveTempRead * 3.8 / 1024; // convert the average reading to a voltage
float cTemp_celcius = (tempVoltage - .5) * 100; // convert voltage to celcius
cTemp = (cTemp_celcius * 9 / 5) + 32; // convert celcius to fahrenheit

prevReading = millis(); // update previous reading value to current time
readingTimer = 0; // clear time

}else{
readingTimer = millis() - prevReading; // if it isn't time to take a new measurement, update the timer.
}

}

//*****CALL REPEATING FUNCTIONS
lcdDisplay(); // update data on LCD
timer(1); // increment timer
light();

}
//*****END LOOP / START FUNCTIONS*****
//*****

// The timer receives the state argument and does one of 3 things:
// 1 = increment: called each time through the loop
// 2 = start timer: called when door is opened
// 3 = stop timer: called when door is closed
void timer(int state)
{
if (state == 2) // ****START timer****
{
startTime = millis(); // set time when door was opened
startTemp = cTemp; // set temp when door was opened
timerState = true; // start timer running
}else if (state == 3) // ****STOP timer****
{
timerState = false; // stop timer running
dTime = 0; // zero timer variables
dTemp = 0;
}else if (state == 1) // ****INCREMENT timer****
{
if (timerState == true) // if timer is running update dTime and dTemp
{
dTime = millis() - startTime;
dTemp = cTemp - startTemp;
if (dTime > ALARM_TIME)
{
digitalWrite(ALARM_OUT, HIGH);
}
}
}else{ // if timer is not running do nothing
return;
}
}
}
}

```

```

// lcdDisplay() runs all of the LCD functions each time through the main loop
// it can display a different screen of info if the door is open or closed
void lcdDisplay() // Prints data to LCD each time through loop
{
  if (doorState == true)
  {
    lcd.home();
    lcd.print("Door is OPEN ");
    lcd.setCursor(0,1);
    lcd.print("Deg lost: ");
    lcd.print(dTemp);
    lcd.print(223, BYTE); // degree symbol
    lcd.print("F ");
  }else{
    lcd.home();
    lcd.print("Temp: ");
    lcd.print(cTemp);
    lcd.print(223, BYTE);
    lcd.print("F ");
    lcd.setCursor(0,1); // move cursor to bottom row
    lcd.print(" "); // print 16 spaces (clear line)
  }
}

}

// the light() function runs the BlinkM unit. It is connected via I2C
// and most of its functions are built into the BlinkM_funcs.h file.
void light()
{
  // the scale from blue to red will be different for when the door is open or closed
  // when door open: blue = the temperature when the door was first opened :: red = blue plus RED_THRESH (around 3
  // degrees?)
  // when door closed: blue = default blue :: red = default red

  if (doorState == true)
  {
    blueTemp = startTemp;
    redTemp = startTemp + RED_THRESH;
  }else{
    blueTemp = defaultBlueTemp;
    redTemp = defaultRedTemp;
  }

  float mapTemp = cTemp-blueTemp; // current temp minus the low temp (blue)
  float tempDiff = redTemp - blueTemp; // difference in blue and red temp
  float mult = 255 / tempDiff;
  mapTemp = mapTemp * mult; // this maps the temp to values of 0 for blueTemp and 255 for redTemp
  mapTemp = constrain(mapTemp, 0, 255); // this constrains the values so below blueTemp it is always blue and above
  // redTemp it is always red
  int r = mapTemp; // red value
  int b = 255-mapTemp; // blue value redValue + blueValue = 255 to maintain constant brightness

  if (r == 255 && prev_r < 255) // if the light has just turned red... start blinking
  {
    BlinkM_playScript(addr, 3, 0, 0);
    BlinkM_setFadeSpeed(addr, 65);
    BlinkM_setTimeAdj(addr, -10);
  }else if (r == 255 && prev_r == 255){ // if the light is already blinking and is still red... keep blinking
    return;
  }else{ // if light is not completely red... set color appropriately
    BlinkM_stopScript(addr);
    BlinkM_fadeToRGB(addr, r, 0, b); // send command to BlinkM to change colors of light
  }
  prev_r = r;
}
}

```

Section 7.02 Code: Arduino 2 – Alarm & Servo

```
#include <Servo.h>
#define SPEAKER 9 // the pin for the LED
#define IN 7 // the input pin where the pushbutton is connected
#define SERVO 10 // servo output pin

Servo servo; // declare servo object

int inRead = LOW; // will be used to store the state of the input pin
int alarmState = LOW; // is alarm running or not?
int nextNote = HIGH; // keeps track of which note is going to be played next
long startTime = 0; // time when timer started
long timer = 0; // actual running timer
float freq1 = 16744; // 2 frequencies make the alarm sound like a siren
float freq2 = 16644;

void setup()
{
  pinMode(SPEAKER, OUTPUT); // tell Arduino LED is an output
  pinMode(IN, INPUT); // and BUTTON is an input
  Serial.begin(9600);
  servo.attach(SERVO);
}
void loop()
{
  inRead = digitalRead(IN); // read the in pin to see if alarm should sound

  if (inRead != alarmState) // if the pin is different than the current alarm state
  {
    alarmState = inRead; // change the alarm state accordingly
    if (alarmState == HIGH) // if the alarm just turned on
    {
      startTime = millis(); // set the time that the current cycle of timer started
      nextNote = HIGH; // reset the next note, so each time the alarm starts, it starts with the same note
    }else{ // if timer just stopped
      timer = 0; // reset the timer to 0
    }
  }

  if (alarmState == HIGH) // if the alarm is on
  {
    if (nextNote == HIGH) // and the next note is HIGH
    {
      for (int i=0; i<= 500; i++) // play the HIGH note
      {
        digitalWrite(SPEAKER, HIGH);
        delayMicroseconds(freq1);
        digitalWrite(SPEAKER, LOW);
        delayMicroseconds(freq1 - 1); // - 1 to make up for digitalWrite overhead
      }
      nextNote = LOW; // and update the next note
    }else{
      for (int i=0; i<= 500; i++) // if next note was LOW, play the low note
      {
        digitalWrite(SPEAKER, HIGH);
        delayMicroseconds(freq2);
        digitalWrite(SPEAKER, LOW);
        delayMicroseconds(freq2 - 1); // - 1 to make up for digitalWrite overhead
      }
      nextNote = HIGH; // and update the next note
    }
  }
  if (timer > 15000) // if the alarm has been running for 15s close door using servo
  {
    servo.write(5); // rotate door to closed position
    delay(500); // wait .5s for door to finish closing
    servo.write(90); // return servo to default (open) location
  }
}
```

```
delay(5000);           // wait 5s for control board to recognize that door is now closed and turn off alarm
alarmState = LOW;     // turn off the alarm
timer = 0;            // reset the timer
return;              // skip the last line
}

timer = millis()-startTime; // update the timer
}
}
```

Section 7.03 Figures & Tables

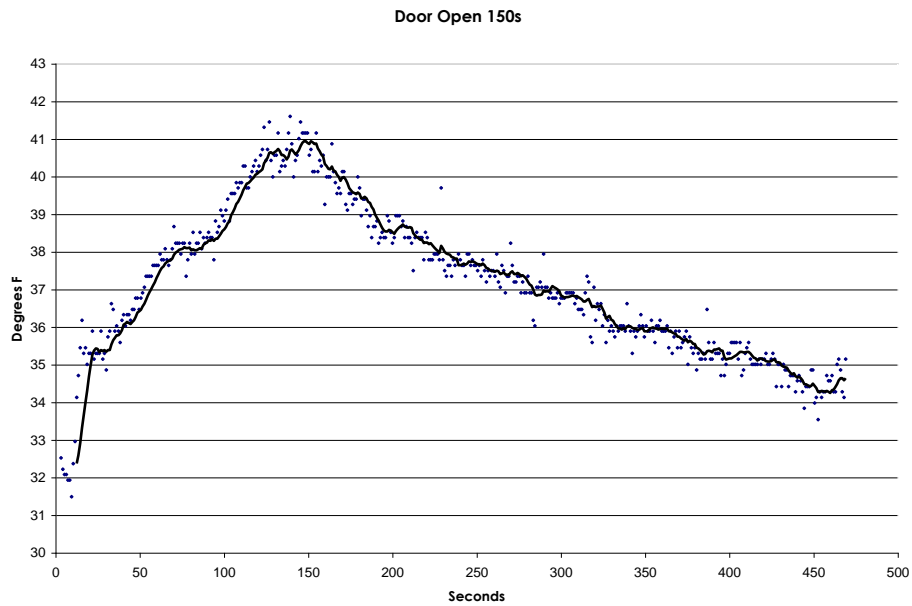


Figure 1

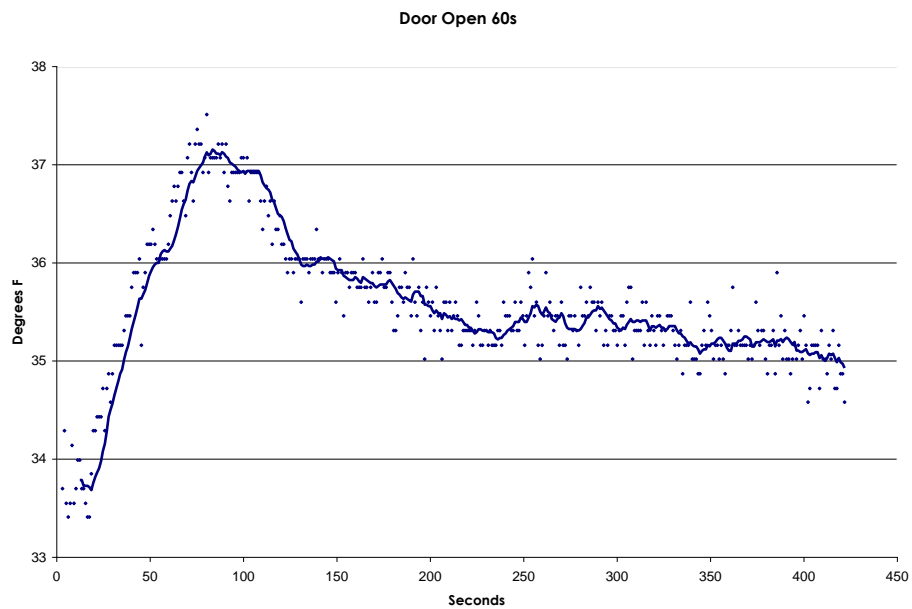


Figure 2

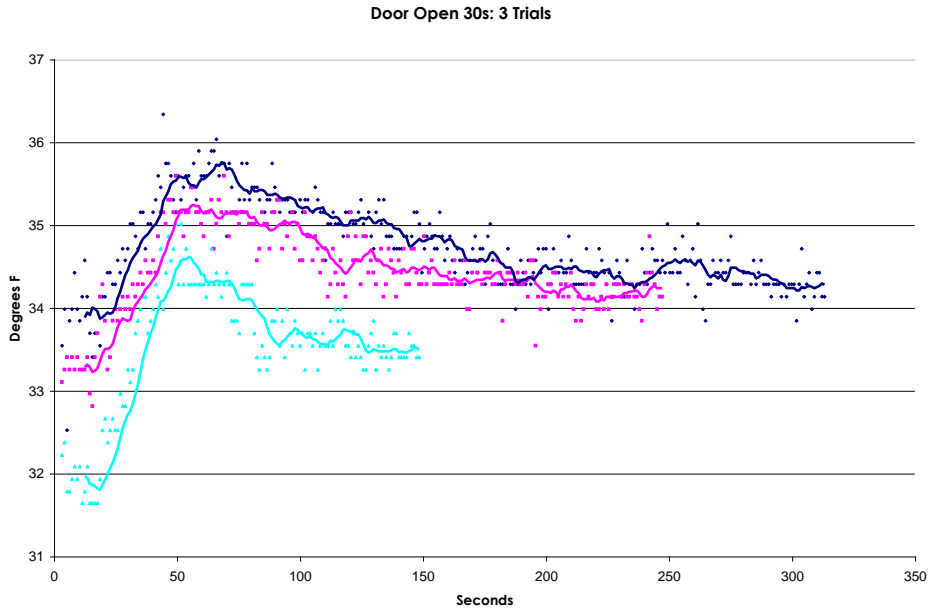


Figure 3

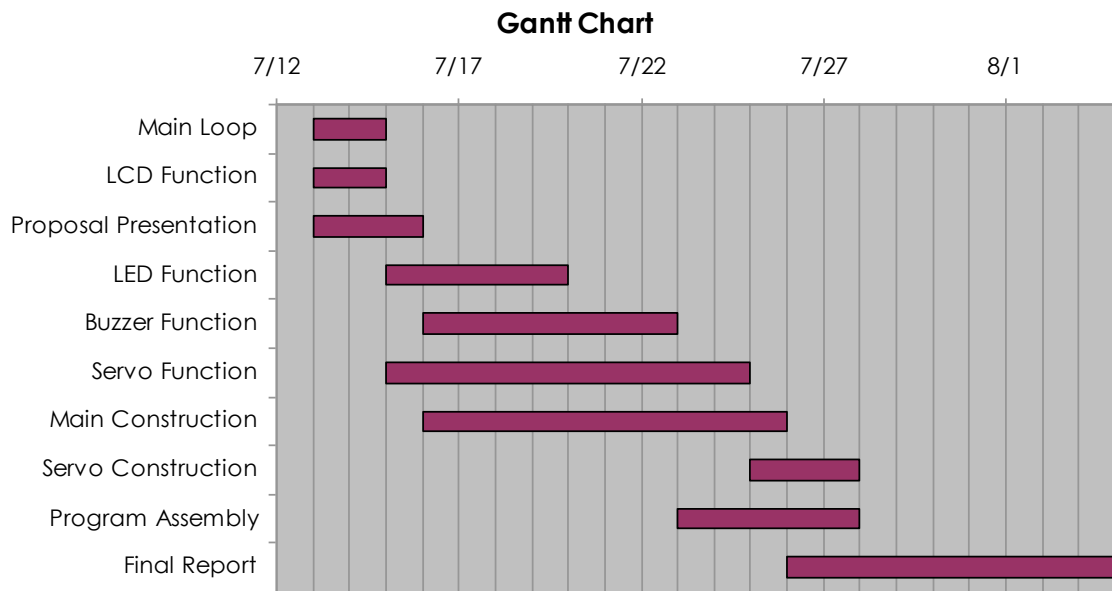


Figure 4

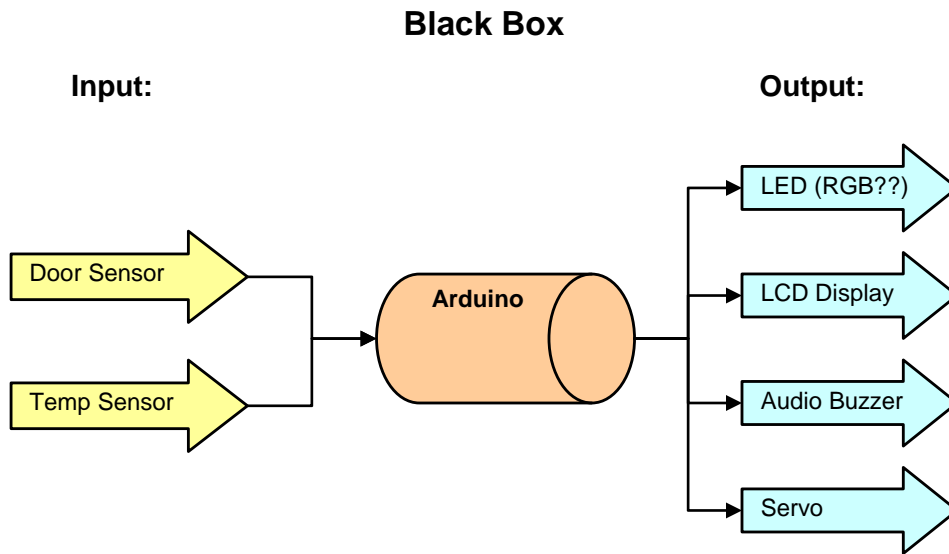


Figure 5

Order of Events Flow Chart

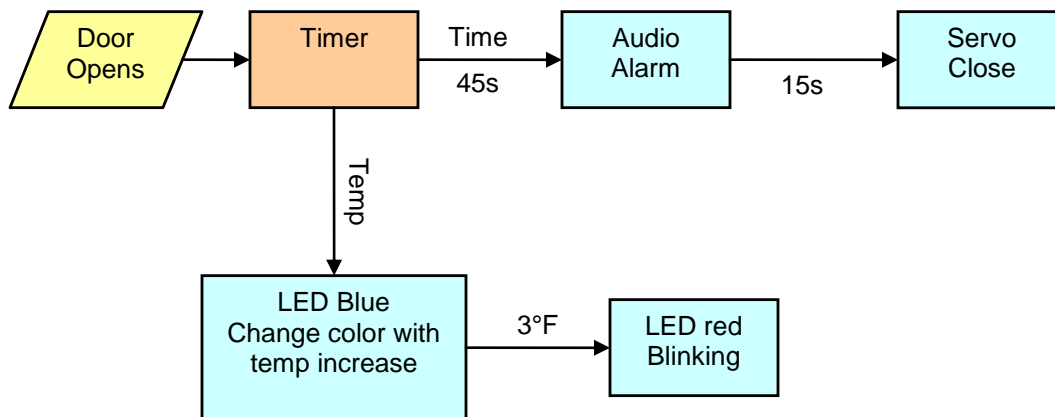


Figure 6

Control Unit Main Loop

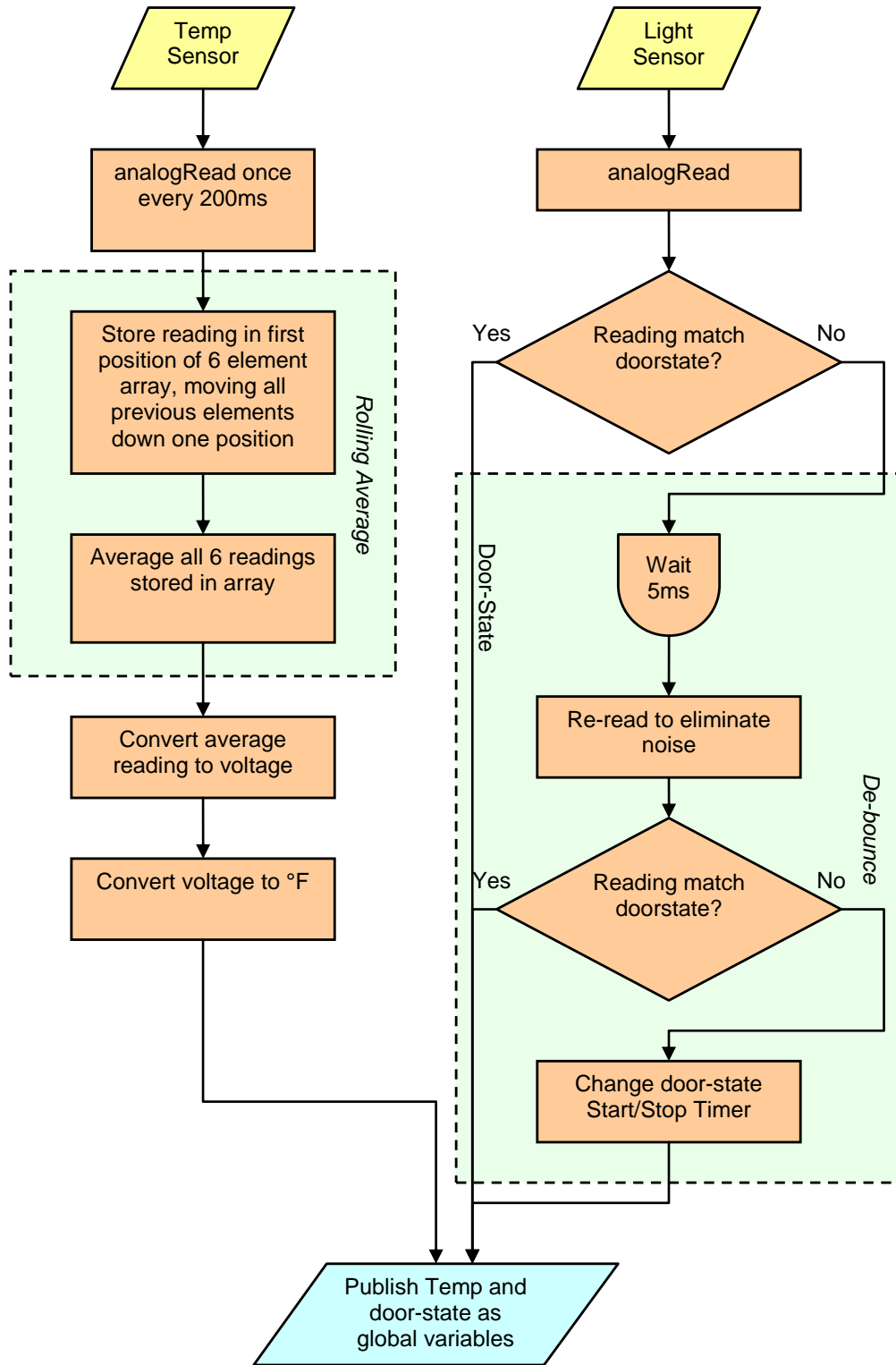


Figure 7

BlinkM Flow Chart

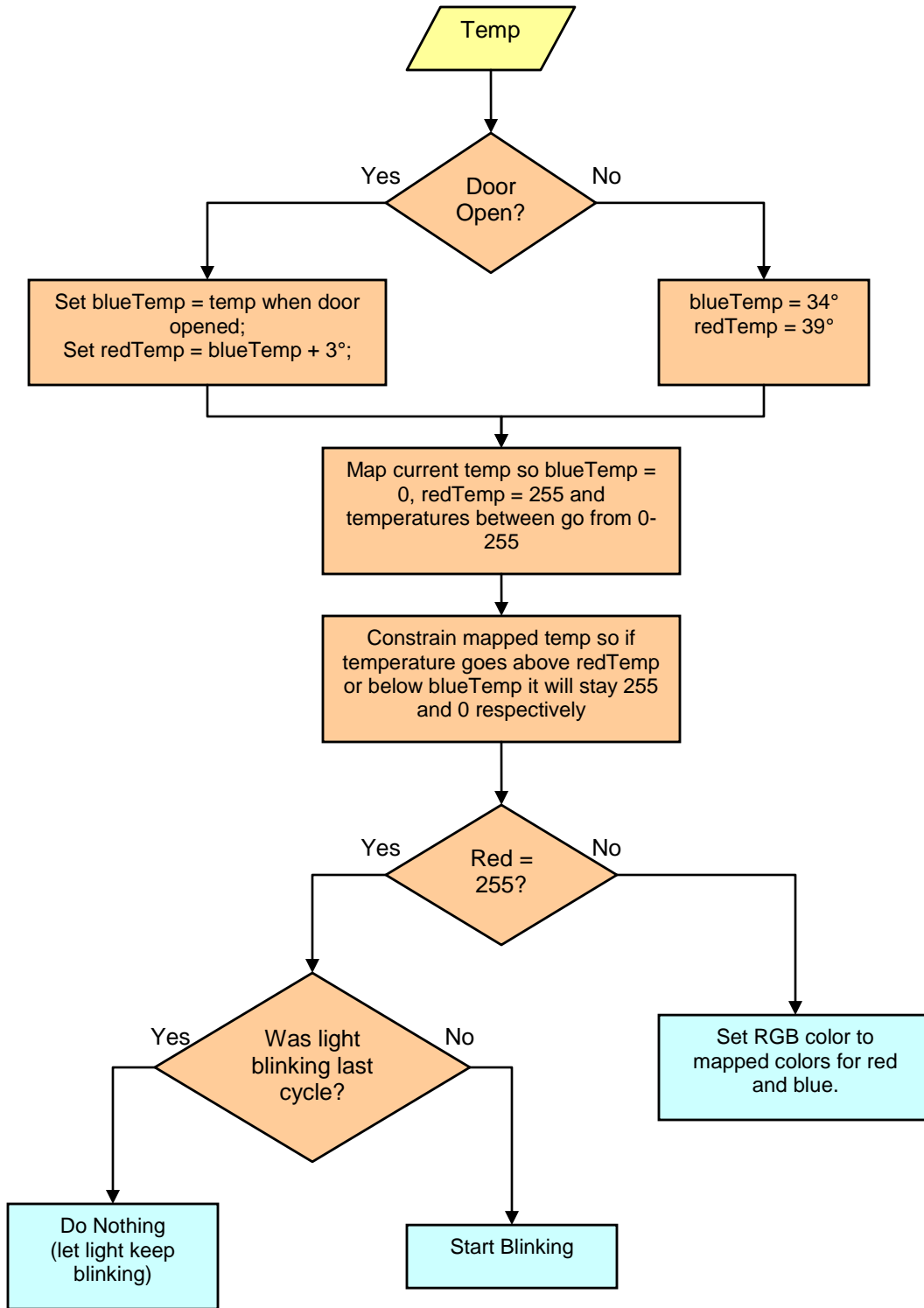


Figure 8

Section 7.04 Schedule & Tasks

At the very beginning of the project we divided the project into single person tasks (Table 1) and assigned each persons responsibilities so as to not overburden any one person or make it so much was due at the same time [Figure 4].

Tasks	Start Date	Est. Duration	End Date	Assigned
Main Loop	7/13/2009	2	7/15/2009	Chad, Prabhjot
LCD Function	7/13/2009	2	7/15/2009	Chad
Proposal Presentation	7/13/2009	3	7/16/2009	Sean, Chad
LED Function	7/15/2009	5	7/20/2009	Chad, Sean
Buzzer Function	7/16/2009	7	7/23/2009	Prabhjot
Servo Function	7/15/2009	10	7/25/2009	Sean
Main Construction	7/16/2009	10	7/26/2009	Gary
Servo Construction	7/25/2009	3	7/28/2009	Sean, Gary
Program Assembly	7/23/2009	5	7/28/2009	Chad, Prabhjot
Final Report	7/26/2009	9	8/4/2009	Whole Team

Section 7.05 References

1. USDA. *Safe Food Handling: Refrigeration and Food Safety*. 2005 [cited 2009 July 31]; Available from: http://www.fsis.usda.gov/factsheets/Refrigeration_&_Food_Safety/index.asp.
2. Tim Roberts, P.G. *Food Storage Guidelines For Consumers*. [cited 2009 July 31]; Available from: http://www.healthgoods.com/Education/nutrition_information/Food_Safety_and_Storage/ood_storage_guidelines.htm.
3. ladyada.net. *Photocells a.k.a CdS cells, photoresistors, LDR (light dependent resistor)...* 2008 [cited 2009 July 31]; Available from: <http://www.ladyada.net/learn/sensors/cds.html>.
4. ladyada.net. *Temperature Hot or not? Now you will know to 0.1 degree precision*. 2008 [cited 2009 July 31]; Available from: <http://www.ladyada.net/learn/sensors/tmp36.html>.
5. Banzi, M., *Getting Started with Arduino*. 2009, Sebastopol, CA: O'Reilly Media, Inc.
6. arduino.cc. *LiquidCrystal Library*. [cited 2009 July 31]; Available from: <http://arduino.cc/en/Reference/LiquidCrystal>.
7. Hitachi, L. *HD44780U (LCD-II) Spec Sheet*. 1998; Available from: <http://www.sparkfun.com/datasheets/LCD/HD44780.pdf>.
8. thingm.com. *BLINKM, THE SMART LED*. 2007 [cited 2009 July 31]; Available from: <http://thingm.com/products/blinkm>.
9. arduino.cc. *Wire Library*. [cited 2009 July 31]; Available from: <http://arduino.cc/en/Reference/Wire>.
10. arduino.cc. *Servo library*. [cited 2009 July 31]; Available from: <http://arduino.cc/en/Reference/Servo>.