Integrated Temperature, Light and Humidity Monitoring System for the Hospital Environment

[Temperature and Light Section]

EE 4BI6 Capstone Project

Student: Christina Tan [0658807]
Faculty Advisor: Dr. T. Doyle
Faculty Supervisor: Dr. J. Deen

4/23/2010

Submitted in partial fulfillment towards a Bachelor of Engineering degree
Faculty of Electrical and Computer Engineering
McMaster University, Hamilton, Ontario, Canada

Table of Contents

Abstract

A smart, integrated temperature, light and humidity monitoring system has been implemented with the use of open standard technology, commercial project and household items which actively monitors the environmental conditions. A main target for this system is to have it designed and implemented as cost efficient as possible. The system allows for a user to input the desired conditions regarding a specific patient's temperature, lighting and humidity requirements. A microcontroller then compares the environmental conditions against the user's input requirements, and actuators change the settings until the desired conditions have been obtained. The designed system is for use in the hospital environment, with emphasis on the neonatal ward, where infants remain in incubators for a period of time. Results indicate the hospital monitoring system is feasible; however, ensuring an enclosed room may be difficult to obtain.

Acknowledgements

Recognition goes to Dr. Thomas Doyle for providing guidance and feedback throughout the course of this project.

Special thanks go to Dr. Jamal Deen allowing his ideas to be explored, as well as for supervising and guiding this project towards completion.

Much appreciation goes to Jonathan Bruer for the guidance given and his assistance with acquiring parts for the prototype design.

Thanks to ZhiYuan Gong for his assistance with troubleshooting the code when problems were encountered.

Lastly, much gratitude goes to Adeel Alam for his contributions toward this project.

List of Figures

Introduction

Definition

The proposed temperature, lighting and humidity monitor is an integrated device that is intended to allow users to input specific requirements for a patient in a hospital environment. The signals for these hospital room settings will be detected and analysed. This data will then be utilized to adjust the current settings as required for each individual patient.

Background

A person who has fallen ill may be require a hospital stay to ensure proper medical attention has been received. While in the hospital, the patient may need specific characteristics of the room to suit his or her condition, such as the temperature, lighting and humidity. These conditions will need to be of a certain value to allow comfort and ideal conditions for the patient on his or her way to recovery. In reality, however, these conditions may change due to external factors, such as the weather, and hospitals lack the resources in order to have these rooms continuously monitored. Because of this, the proposed project is to create a device to monitor these changes in an integrated system. This device would return these values back to the ideal settings, if deviation were to occur.

An example of where this integrated monitoring system would be beneficial is in the case of neonatal jaundice. Bilirubin, which results from the breakdown of red blood cells, is broken down by the liver and removed from the body with urine and stool excretion. A newborn, however, has not completely developed the ability to process the bilirubin, so the excess bilirubin may enter the tissues of the body. When this occurs, yellowing of the skin and eyes appears, leading to jaundice. (How Stuff Works, 2006)

If jaundice were to develop, the baby may need to undergo phototherapy treatment. During this procedure, blue fluorescent lights are used to break down excess bilirubin in the skin (UMHS Newborn Care Committee, 2005). In this process, the temperature of the surrounding area should remain constant, which is where the proposed device would assist in this accomplishment.

Presently, no integrated devices that monitor and control the combination of temperature, lighting and humidity exist, making this project unique. There is a device to monitor all three of these factors, but there are no actuators. Assumptions for this project are that the room is to be air-tight, which allows for better control of the humidity and somewhat, the temperature.

Methodology

Sensors that are used to detect temperature, lighting and humidity will be placed in close proximity to the patient. A computer connected to the microcontroller will allow the user to input the required temperature, light and humidity settings that are ideal for the patient's condition or even for comfort. When the readings from the sensors deviate with respect to the input values from the controller, the microprocessor sends out a signal to the effecters to adjust the settings back to the input values. (Jia, Heiss, Fu, & Gay, 2008)

Scope

The temperature, lighting and humidity monitoring system is to be a cost efficient system and targeted towards patients in the hospital environment (UMHS Newborn Care Committee, 2005). One of the main focuses would be for infants, including infants that are placed in incubators, and for the treatment of neonatal jaundice.

Literature Review

After reviewing many articles, there are presently no papers that mention monitoring the combination of temperature, lighting and humidity in one integrated system and have actuators to modify these settings. There are, however, journals for the automatic detection and control of light. In one such article, a light sensor in the control module detects a change in light intensity and a radio frequency module is used to change the lighting (Bai & Ku, 2008).

In addition to this, there is one research paper that has discussed monitoring these three environmental conditions; however, there has been no mention about having actuators to modify

these surroundings. An advantage of this sensor system is that it is wireless, based on Zigbee technology, which has low power consumption. (Tatsiopoulos & Ktena, 2009)

Another research paper discussed a process to effectively control humidity in an incubator. The humidity was controlled based on a passive and active system. In the passive system, air was passed over a water reservoir, causing the air to have an increase in humidity, before being distributed throughout the room. The level of control for the passive system was minimal, as opposed to the active model, which used a step motor that rotated to allow humidity in. (Costa, et al., 2009)

Table 1: Comparison between the Articles

Paper	Environmental Focus	Notes
1	Light	
2	Light, temperature, Humidity	Research focused on sensing, not controlling
3	Humidity	
This Project	Light, temperature, Humidity	Focuses on sensing and controlling

Methodology of Design

High Level Design

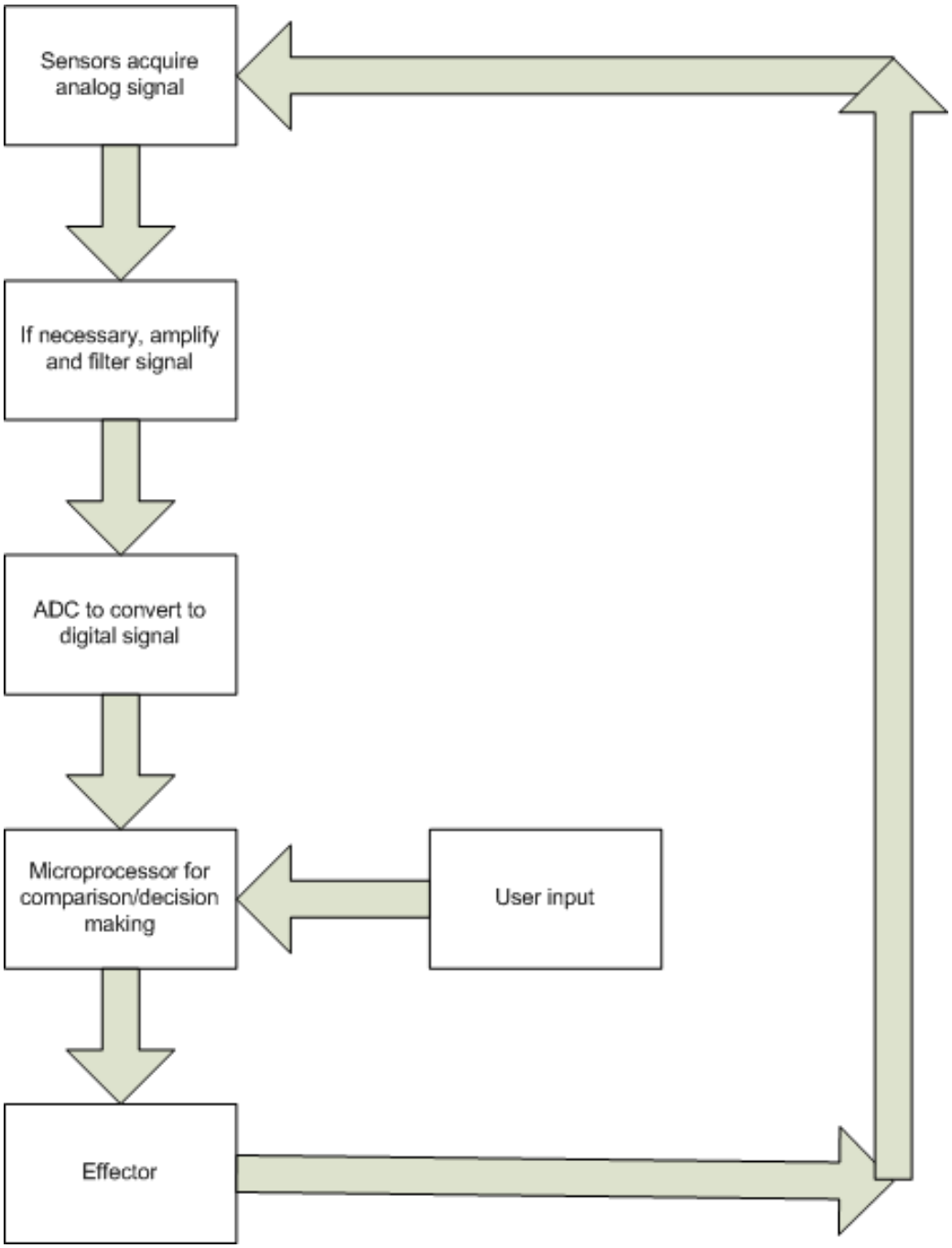

Figure 1: Overall Experimental Design

A very systematic approach was used for the overall design of the project, in which three environmental factors, lighting, temperature and humidity, were to be controlled. An input is required from an external source in order to effectively control these variables. The inputs from the external sources were obtained using a variety of sensors. These sensors detected the surrounding environment, and these inputs were utilized to the control effecters. (Patriciu, 2009)

The control mechanism used was a microcontroller, which read the output from the sensors and used this output to compare if has deviated from its intended value. The user was manually required to input the desired value for each environmental setting.

When the variable being monitored has deviated from its required value, a signal is sent in order for an actuator to start affecting the environment. An actuator is "a mechanism that puts something into automatic action" (Dictionary.com, 2010). In this case, the actuator functioned to either increase or decrease the effects of that environmental variable, so that eventually, the actuator returns the detected value to the desired one, which in turn behaves like a negative feedback loop. A flow diagram of the overall design is given in Figure 1.

Description of Apparatus

This section discusses the components used throughout this project and includes explanations as to why they were selected. For specifications of each component, please refer to Appendix A.

Microcontroller

The Arduino Duemilanove, which uses the ATMega328, was selected as the microcontroller for this project. This was an ideal selection, as the processor is extremely strong and cost efficient. An input voltage ranging from 0 – 5V is required, which corresponds with the humidity sensor. An on-board 10-bit analog to digital converter (ADC), aids in the digitization of the analog signal acquired from the sensor. Additionally, the temperature and light sensors used in the prototype are specifically designed for the Arduino, which allows for simple integration with this microcontroller. The board also features 8-bit PWM pins, which are used to control the light and temperature effecters. The Arduino platform uses a language similar to C, so having been

familiarized with the C programming language, it was convenient to use this microcontroller. Another reason the Arduino was selected is that it is a well known board with plenty of examples and references on the web to aid this project. Lastly, uploading the code from the computer onto the board was simple, since the software was available for at no cost on the Arduino website. The Arduino has a serial port to allow communication with the computer. The USB connection from the computer goes directly onto the Arduino board, where a USB to serial converter then allows communication to occur. This was necessary, since this project requires a user's input. (Robot Shop, 2009)

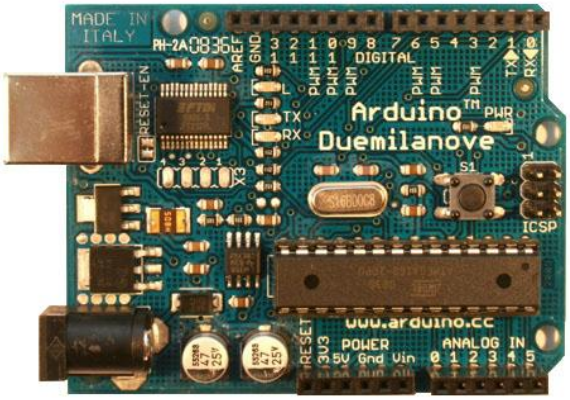

Figure 2: Arduino Duemilanove

Other microcontrollers were considered, including the PIC series of microcontrollers. These PICs also contained 10-bit ADCs; however they had operating voltages ranging from 1.8 – 5.5V. Additionally, to upload a code from the computer, a bootloader program along with its corresponding hardware was required, another factor that made the PIC series unfavourable.

Light

For the light sensor, a major requirement was that the sensor would be able to detect a wide range of light. The sensor would need to be able to detect a brightness ranging from 0 – 20 000 lux. This range was selected because the sensor would need to detect values ranging from a completely dark room, to double what the brightest light setting is in the hospital room. Research indicates that for patients with neonatal jaundice, treatment is done with the use of blue fluorescent lights in order to remove the build-up of bilirubin from the blood (UMHS Newborn

Care Committee, 2005). These treatment lights have a brightness of 10 000 lux (Specialty Lights, 2009). Very few sensors were able to detect this high a range. The only sensor that could be acquired was the LX1971IDU by Microsemi-IPG, and has a current output that is mostly linear. Since the microcontroller reads voltages, a resistor can be added to the circuit to accommodate this. This sensor is also ideal because the response is mostly linear, as shown in Figure 4. This linearity is beneficial because the output of the microcontroller is proportional to the reading from the sensor.

Figure 3: Light Sensor for Room

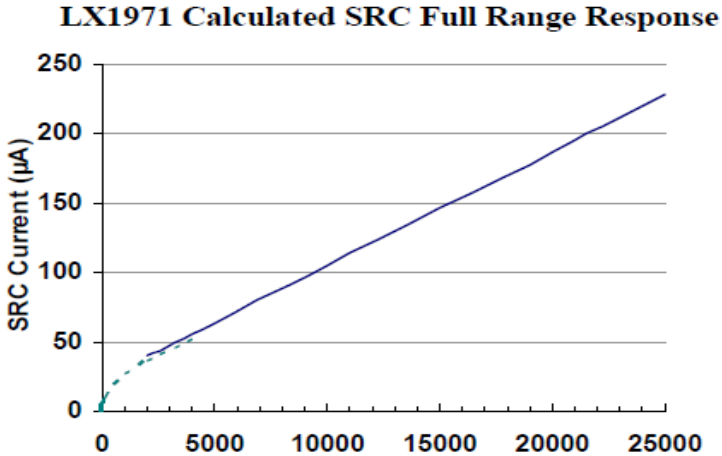

Figure 4: Response of Light Sensor

The problem with this sensor, though, is that it is too tiny to manually solder to a circuit board, and the actual circuit board that the sensor uses is quite expensive, priced around $70. For the prototype, a different sensor will be used, the DFRobot Ambient Light Sensor, which has been designed specifically for the Arduino microcontroller. This sensor detects the intensity of ambient light with the use of a photo resistor. The resistance value decreases with a higher light

intensity, and a voltage reading is outputted. Since the sensor already produces a voltage, it will be more accurate than using a resistor along with the current output, like with the LX1971, since the resistor will have its own tolerance. The DFRobot Ambient Light Sensor is also very cost-efficient, at under $4, meeting one of the major criteria for this project. (Robot Shop, 2009)

Figure 5: Light Sensor for Prototype

To control the lighting of the prototype, two ideas were considered: LEDs and servo motors. If servo motors were to be used, the motor would be used to manually control a dimmer switch operated with a variable resistor. The other option, LEDs, was selected due to low power, cost-efficiency and ease of use, since the LEDs can be directly controlled by the Arduino microcontroller. In order to control the lighting, pulse width modulation (PWM), a theory learned in a previous McMaster University course, COE 2DP4, will be used. A brief explanation would be that the longer the pulse is a “1” (turned on) in one period, the average voltage would be higher giving the impression that the LEDs are brighter. The Arduino microcontroller already has PWM pins for an 8-bit PWM output, allowing for ease of use. Blue LEDs were selected for the prototype to represent the blue fluorescent lighting used for the treatment of neonatal jaundice. (Specialty Lights, 2009)

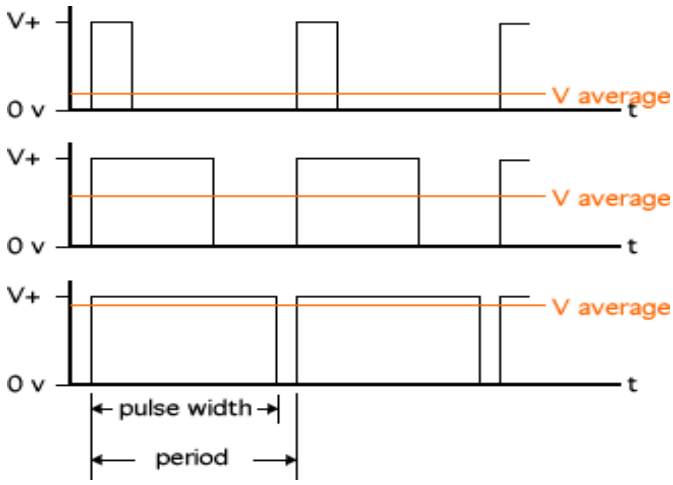

Figure 6: Concept of PWM

Figure 7: LED Used for Dimming the Brightness

Temperature

For the temperature sensor, the DFRobot LM35 was selected, which is based on the LM35 by National Semiconductor. This sensor was chosen because it was specifically designed for the Arduino microcontroller, outputs a voltage, and operates from -55 – 150°C, covering the range that is necessary for sustaining human life. This sensor is also fairly accurate, with ±¼°C at 25°C and ±¾°C over the entire range. Since the LM35 sensor is already mounted on a circuit board, it allows for ease of use, in addition to being cost-efficient, with the price being less than $4. (National Semiconductor, 2000)

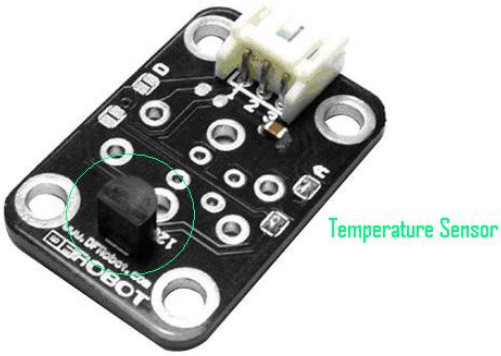

Figure 8: Temperature Sensor Used

Other sensors were researched, and although some were very inexpensive, they were not mounted on a circuit board, adding inconvenience as well as increased cost in order to mount it.

For temperature management, different ideas were reviewed for the cooling and heating of the room. To control cooling, the use of a computer case fan and a CPU fan were both considered. Both fans are cost-efficient, each costing less than $15, although the CPU fan was slightly more expensive. To control the heating, using a mini-heater, a blow dryer and a power resistor were all considered. The power resistor was chosen for the prototype, a cubicle for an infant, because a mini-heater would be much too large to fit into the cubicle, and the blow dryer would generate too much noise for an infant. Although the CPU fan was a bit more costly than the computer case fan, the CPU fan was selected because it contains a heat sink, which is where the power resistor is mounted, to ensure that the resistor does not overheat.

Figure 9: CPU Fan Used

Figure 10: Power Resistor Used

Experimental Design

The following section describes how each component coordinated with other components throughout the project to complete the objective that was required of this design.

Light

The LEDs were placed at the top of the cubicle, while the light sensor was placed at the bottom of the cubicle towards the centre. The sensor detects the brightness level of the cubicle, resulting in a voltage value from 0 – 5V. This voltage value is converted with the on-board 10-bit analog to digital converter (ADC), resulting in a value between zero and 1023 (Robot Shop, 2009). These values are scaled down to a range from zero to seven. When the Arduino microcontroller

is connected to a computer, the user selects a brightness level from zero to seven, with zero being completely dark, and seven being the brightest setting. The LEDs turn on to the specific brightness level, controlled by PWM. If the lighting of the room were to change, for example, an increase in sunlight, then the sensor will detect this deviation and dim the LEDs in order to maintain the brightness level of the room. The LEDs will brighten if the opposite were to occur.

Temperature

The CPU fan and power resistor were placed towards the edge of the cubicle, with the temperature sensor located closer to the patient. The sensor works similar to the light sensor, reading a voltage between 0 – 5V. Again, the voltages are converted with the 10-bit ADC, resulting in a value between zero and 1023. Code has been implemented to convert these values into degrees Celsius. When the Arduino is connected to a computer, the user is able to select a temperature in Celsius values. These values are then compared against the readings given from the temperature sensor. If the user inputted value is higher than the sensor reading, then the power resistor will turn on until the desired temperature is reached. If the input temperature is lower than the one being read on the sensor, the CPU fan will turn on until the target temperature has been obtained.

Design for Prototype

The objective of the project was to build a cost-efficient prototype to simulate a hospital room. To assemble this, a small, enclosed storage cube was purchased from Wal-Mart. To cover up the front opening, Plexiglas was placed and screwed on.

Figure 11: Storage Cube

There was a block of wood attached to the ceiling of the cubicle. Protruding from this piece of wood, a breadboard was fastened. The breadboard contained four LED lights for light control. There were wires from the breadboard traveling to a second breadboard on the floor in order to supply the LEDs with power.

Figure 12: The Wood Glued from the Top

The second breadboard was placed near the back of the cube. On this breadboard, the connection wires from the two relays that control the power resistor and fan, for temperature control, were connected. Furthermore, the wires coming from the temperature and light sensors were connected to this breadboard, which were powered by the Arduino's power supply.

The power resistor was placed on the left hand side of the cube along with the CPU fan. Aluminum foil was wrapped around the power resistor for easier portability and a wider area for heat dispersion. The power resistor was attached to the heat sink of the fan in order to prevent overheating. The power resistor and heat sink were powered by a 12V, 2.5A laptop power supply, which used a terminal block to allow for multiple connections to the power supply. The temperature sensor was placed in close proximity to the power resistor and the light sensor was placed on the bottom of the cubicle directly underneath the LED lights. See Figures 22 and 23.

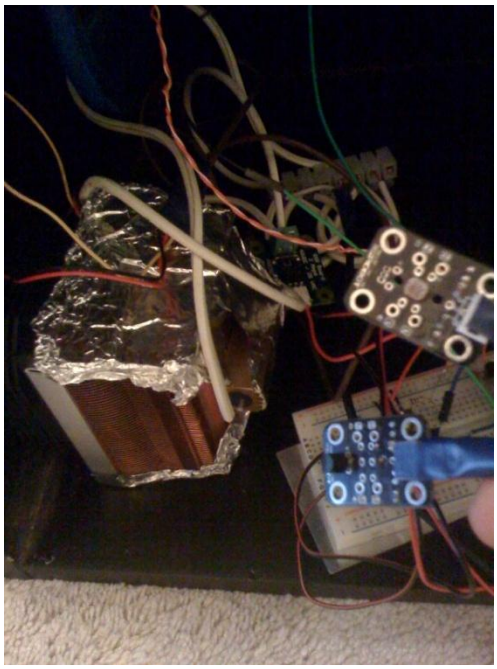

Figure 13: Power Resistor, Temperature and Light Sensor

Code

The code used to program the Arduino essentially had a similar type of code for each of the environmental factors. The code was based on “if” statements. If the user input variable was greater than the value from the sensor, an actuator would be turned on to increase the effect of that specific variable. In contrast, if the user input variable was less than the value from the sensor, an actuator would be turned on to decrease the effect of the specific variable.

Cases were used to allow the user to select which environmental control to input a value for. When controlling the brightness level of the room, the user is to input a brightness value ranging

from zero to seven, with zero resulting in the LEDs being turned off, and seven being the brightest. The 10-bit ADC reads in the voltages from the sensor and the resulting 1024 bits are scaled from zero to seven to match the range given for the users to select. For the LEDs, the effecter range is a scaled version of the PWM output from the Arduino. The Arduino has 8-bit PWM pins, which results in a range from zero to 255 bits. Code was also written to scale to this range from zero to seven in order to correlate with the above mentioned items.

To control the temperature of the room, PWM is also used to control the effecter, which is either the fan or the power resistor. After the sensor reads in the temperature as a voltage, the ADC is used to convert this into a digital signal. Code has been written to convert this value into a temperature reading in degrees Celsius (int room_temp = (125*analogRead(temp_pin)) >> 8). A range from zero to 49°C has been selected for the sensor to read in, and a range from 10 to 39°C has been selected for users to choose from. Any value outside this range will result in an error and requires the user to re-input a value. This range was selected since room temperature is about 25°C and hence, a similar range above and below this value is ideal.

Hysteresis is controlled using a series of "if" statements. For cooling, if the difference between the room temperature and the desired temperature is greater than 5°C, then the fan will turn on at maximum speed, which is a reading of 255 for PWM. When the temperature difference is less than 5°C, code was written to scale the output (analogWrite(9, (room_temp-custom_temp)*6);) . The temperature difference is multiplied by six, since the maximum temperature difference is 39, and the product of these two integers is closest to 255. A similar algorithm was used to control hysteresis in heating. If the temperature difference is greater than 5°C, the PWM output will be 240; 255 may cause the power resistor to overheat, so it was not used for heating. When the difference is less than 5°C and greater than 3°C, the PWM output will be 235. After the temperature difference is less than 3°C, a scaled calculation was given (analogWrite(10, (custom_temp-room_temp)*40);).

Please refer to Appendix B for the code.

Results

Each individual environmental variable was tested in the cube to observe how effective the prototype setting is.

For temperature, Figure 14 shows temperature values read over a period of eight seconds.

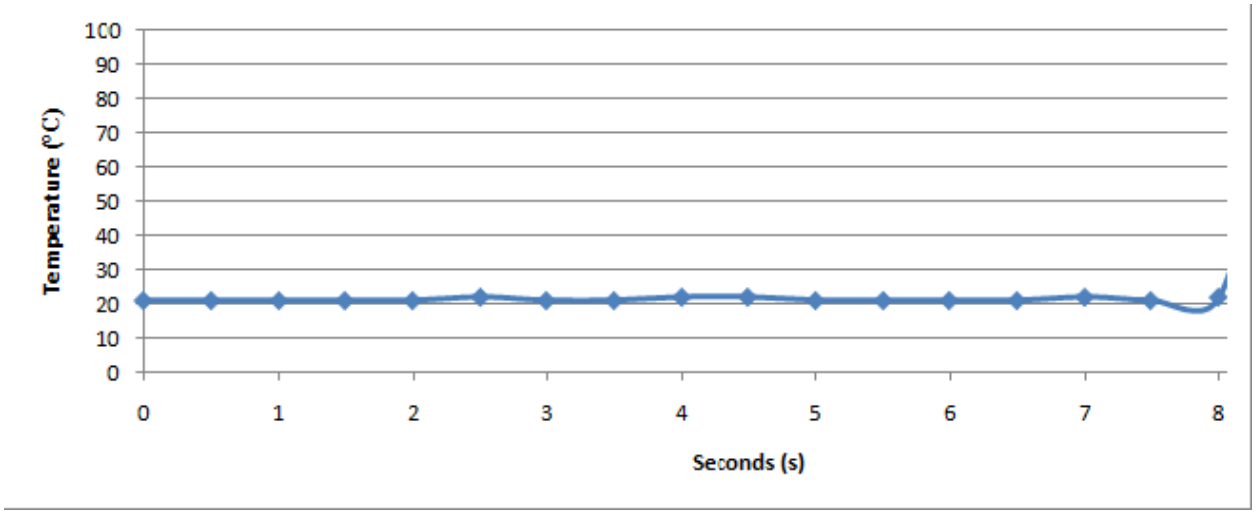

Figure 14: Temperature Values

The temperature sensor gives values in the range of 21 – 22°C. These results were consistent and gave an indication that the sensor is able to provide accurate measurements.

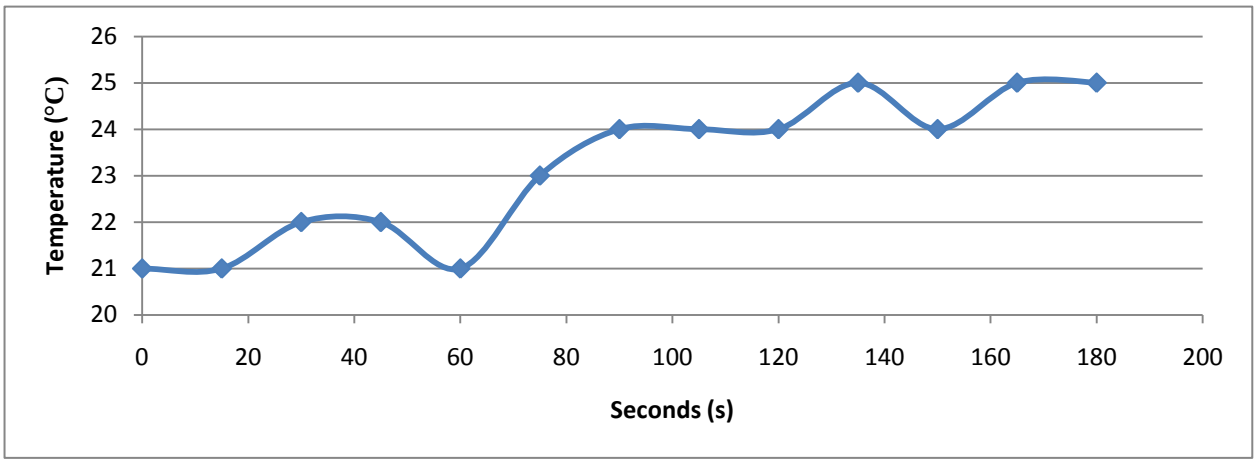

Figure 15: Temperature When the User Inputs 25°C

As shown in Figure 15, the temperature reading reached its desired value after approximately three minutes. The factor affecting this was mainly the placement of the temperature sensor. The closer it was to the power resistor, the quicker it approached the desired temperature reading. The downside to this was that the sensor's reading was only indicative of the temperature near the power resistor, rather than the whole room. Figure 16 displays the results of the temperature sensor being placed at the other end of the room.

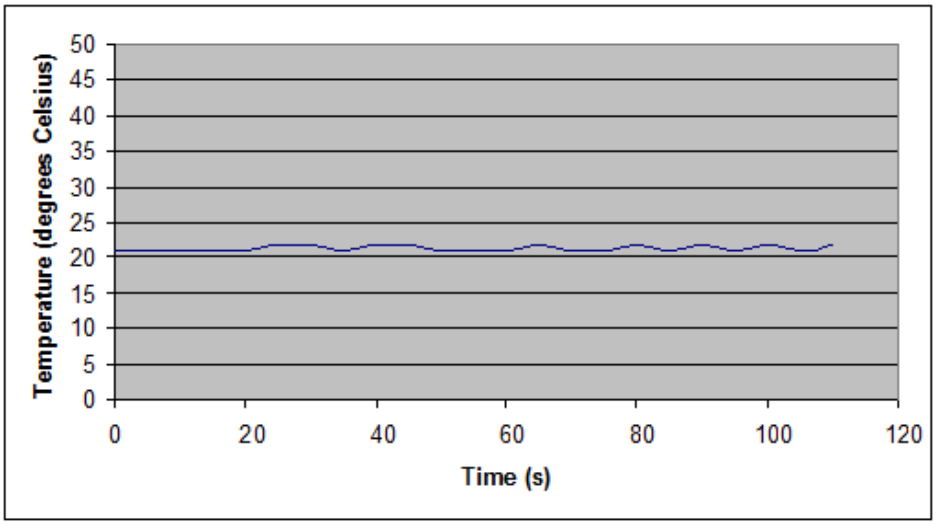

Figure 16: Temperature Sensor Placed at the Other End of the Room when the User input is 25°C

Figure 17 shows when the temperature sensor is placed at approximately the centre of the room.

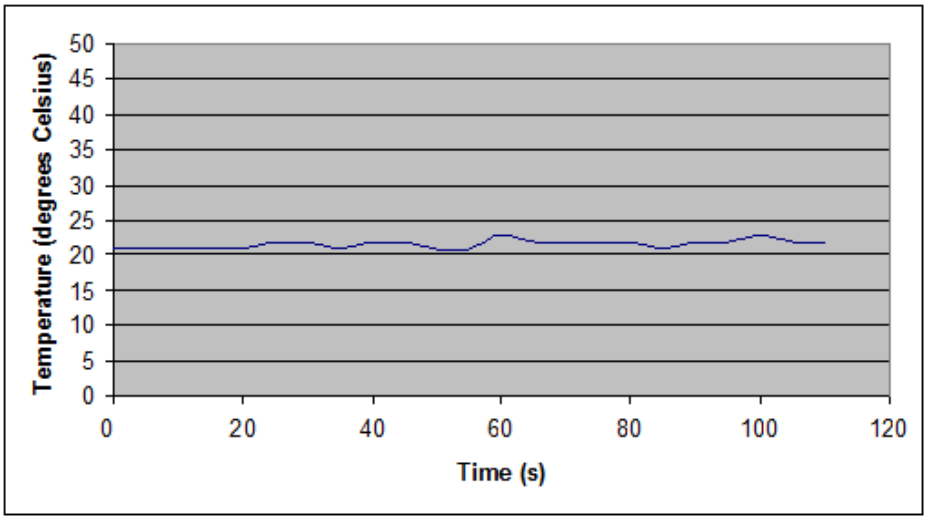

Figure 17: Sensor Placed at the Centre of the Room with the User Input of 25°C

The dependency of the temperature sensor being placed near the power resistor is apparent from both Figure 16 and Figure 17. Both figures show that the heat generated from the power resistor was not sufficient to reach the opposite side of the room. In Figure 17, the temperature starts to increase very slightly after about two minutes; however, it is still not sufficient to produce a fully functional temperature control system. The power resistor can produce 25W of power. However, the air-metal interface from the power resistor to the surrounding air is very inefficient. Therefore, air cannot heat up fast enough for the sensor to detect it at a distance from the power resistor. This is a limitation to this project.

Figure 18 shows the time it takes after the user inputs a value of 15°C.

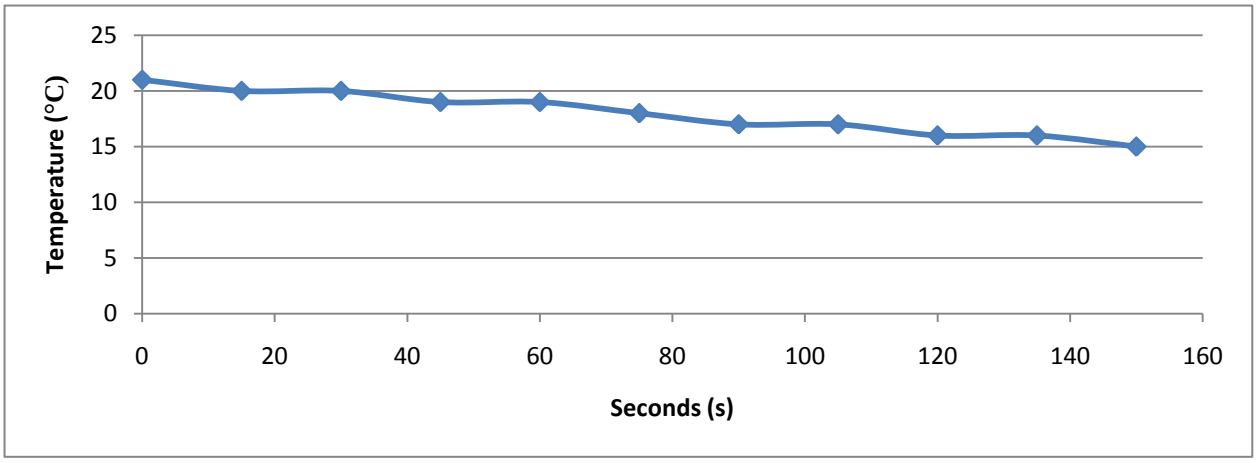

Figure 18: Temperature when the User Inputs 15°C

In approximately 2.5 minutes, the temperature cooled to its desired value. It was a fairly linear relationship. The cooling effect was quicker because of the heatsink along with the fan that was blowing. This allowed the power resistor to cool at a quicker rate as opposed to the power resistor being heated.

Lighting was set up in such a way that the user had to input a value between zero and seven, where zero is off and seven is the brightest. The user inputted the value three. After dimming the lights of the room (not the cubicle LEDs), the LEDs increased in brightness. This proves that the LEDs strived to maintain the brightness level of three by becoming brighter. As the user's hand moved over the light sensor to emulate total darkness, the LEDs became very bright. In contrast,

as the lights of the room were brightened, the LEDs appropriately dimmed to maintain the overall brightness of the room at level three.

Problems

One of the problems encountered was the control of the light. The project was under the assumption that the LEDs were the main power source. When an external power source is present, it sometimes results in a change in the maintenance of the overall lighting. An example of this was when the brightness level was set to four. If the brightness of the overall room was extremely bright, then the LEDs would be unable to dim enough to maintain a balance of brightness level four, even if the LEDs were completely off. A potential solution would be to envelop the room in complete darkness. This would give the LEDs complete control. Otherwise, the external light source used must be weaker than the LEDs within the room itself.

Another problem was the accuracy of the temperature sensor in relation to the entire cubicle. The sensor was purposely placed near the power resistor to effectively read the temperature. If it was placed anywhere else, the temperature readings would not be as affected by the power resistor. The resistor had a power rating of approximately 25W. This would have been a very slow process to heat the entire cubicle. The cubicle could have been smaller to allow heat to build up in the room. This would allow the temperature sensor to read the value of the room, rather than only the power resistor. Otherwise, a blow dryer or a heater could have been used to increase the temperature. Drawbacks to these proposed methods were that the heater is too large and the blow dryer is not quiet enough. For cooling, the CPU fan was not powerful enough to cool the room down to the coldest temperature values that the user is allowed to input. A solution to this would be to use a more powerful fan in order to allow for more cooling.

Future Work

Currently, hysteresis is being controlled by a series of “if” statements. A more effective way to control this would be to use a proportional-integral-derivative (PID) controller.

The future work for this project would be to integrate this into a room. The room can use the sensors for monitoring purposes. The actuators can be the thermostat, lighting and humidifier in the room. This integration would make this a very unique project as it would have total control of these three factors in a room.

Lastly, projects like these can be integrated in a grocery shop. Food items can disintegrate when they are not in an ideal environment with respect to humidity and temperature. This project can monitor the environment that the fruits and vegetables are placed in so that if the ideal conditions change, these sensors can detect this and restore the conditions. This would reduce the amount of food discarded from grocery stores.

Conclusions

The designed temperature, lighting and humidity monitoring system has proven to successfully acquire accurate measurements for the above mentioned environmental factors. The actuators have been able to modify these factors by changing these values to what the user has requested. Although the actuators are not completely effective, necessary actions are being taken with respect to the hardware and code in order to become more effective.

Appendix A

Specs for the Microprocessor

Microcontroller	ATmega168
Operating Voltage	5V
Input Voltage (recommended)	7 – 12V
Input Voltage (limits)	6 – 20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	16KB (ATmega168) or 32KB (ATmega328) of which 2KB used by bootloader
SRAM	1KB (ATmega168) or 2KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1KB (ATmega328)
Clock Speed	16MHz

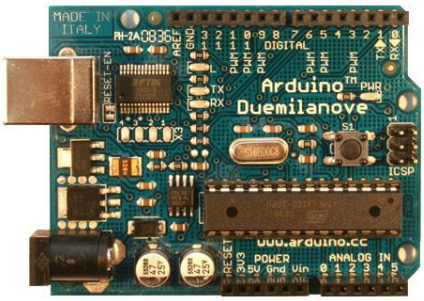

Figure 19: Arduino Duemilanove

Specs for the Fans (12 volts)

Fan Dimension 120 x 120 x 25 mm

Fan Speed 1800 ± 10%

Bearing Type Dual Ball Bearing

Noise Level 29.5dB (Max)

Max. Air Flow 64.286 CFM

Power Connector 3-Pin

Rated Voltage 12 V

Started Voltage ≦ 9 V

Rated Current 0.38A

Figure 20: 12V Fan used for Temperature Control

Specs for LED

Features:

465-470nm Blue

3.0 – 3.4Vdc Forward voltage

80mA Forward current

30 degree viewing angle

Figure 21: LED used for Light Control

Specs for the Relay

Output Contact on Resistance: 0.05 ohms

Maximum DC Switching Voltage: 40Vdc

Maximum AC Switching Voltage; 28Vac

Maximum Switching Current: (AC or DC) 2.5A

Maximum Continuous Current: (AC or DC) 2.5A

Maximum Switching Speed: 20 cycles/second

Figure 22: Solid State Relay used to Control the Transducer, 12 Volt Fan and Power Resistor

Appendix B

<u>Combined Code (Humidity will be italicized)</u>

```
//Data type declaration//
int light_pin = 0; //light sensor pin
int temp_pin = 1; //temperature sensor pin
int hum_pin = 2; //humidity sensor pin
//temporary data type declaration//
int custom_light = 0; //store the expected illuminance
double custom_temp = 0;  //store the expected temperature
int custom_hum = 0; //store the expected humidity
int custom_data = 0;  //store the expected data update
int temp = 0;
int temp1 = 0;
int byte_i;
int serial_read;
char reading_Array[5];

void setup()
{
  //Initialize pin mode//
  pinMode(11, OUTPUT); //LEDs
  pinMode(10, OUTPUT); //heater (temp)
  pinMode(9, OUTPUT); //fan (temp)
  pinMode(8, OUTPUT); //fan for increase humidity
  pinMode(7, OUTPUT); //fan for decreasing humidity
```

```
  pinMode(6, OUTPUT); //transducer to increase humidity
  pinMode(temp_pin, INPUT);
  pinMode(light_pin, INPUT);
  pinMode(hum_pin, INPUT);
  //Initialize Serial port//
  Serial.flush();
  Serial.begin(300);

  Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for humidity):");
}
// the loop() method
void loop()
{
  int light_read = analogRead(light_pin);
  int room_temp = (125*analogRead(temp_pin)) >> 8; //gives temperature in degrees Celsius
  double hum_reading = analogRead(hum_pin);
  double voltageValue = (5*hum_reading)/1023;
  double rh = (voltageValue - 0.788)/0.0314;

  update_data();
  //output light strength value
  temp = map(custom_light,0,7,0,1023);
  if ((temp-(1023-light_read)) <= 0){ //1023-light_read b/c light_read values are inverted
    analogWrite(11, 0);
  }
```

```
  else analogWrite(11, map((temp-(1023-light_read)),0,1023,0,255));// PWM for LED
brightness; range of 0 to 255

  //output heater and fan strength value

  if (custom_temp != 0){ //we are going choose a room temp range from 0 to 49..these are used to
do the below calculations

    if (custom_temp < room_temp) {

      if (room_temp-custom_temp > 5)

        analogWrite(9, 255);

      else

        //analogWrite(9, 255);

        analogWrite(9, (room_temp-custom_temp)*6); //closest to get 255 (39*6 = 234)

      analogWrite(10, 0);

    }

    else if (custom_temp > room_temp){

      if (custom_temp-room_temp > 5)

        analogWrite(10, 240); //not set to 255 in case of overheating

      else if (custom_temp - room_temp > 3)

        analogWrite(10, 225);

      else

        analogWrite(10, (custom_temp-room_temp)*40);

        //analogWrite(10, (custom_temp-room_temp)*5); //multiplied by 5 so max for resistor is
not reached (in case it overheats)

      analogWrite(9, 0);

    }

    else{

      analogWrite(9, 0);
```

```
    analogWrite(10, 0);
  }
}
else
{
  analogWrite(9, 0);
  analogWrite(10, 0);
}
//output transducer and fans
if (custom_hum == 0)
{
  digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, LOW);
}
else
{
 if (custom_hum > (rh + rh*0.02)) //transducer needs to turn on
  {
    if (custom_hum - rh > 8)
    {
      digitalWrite(6, HIGH);
      digitalWrite(7, LOW);   // set the LED on
      digitalWrite(8, HIGH);
    }
```

```
    else
    {
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);
      digitalWrite(8, LOW);
    }
  }
  else if (custom_hum < (rh - rh*0.02)) //fans needs to be on
  {
    digitalWrite(6, LOW);  // set the LED on
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
  }
  else
  {
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
    digitalWrite(8, LOW);
  }
}
//delay(10000);

Serial.print(room_temp);
Serial.print(" ");
Serial.println(rh);
```

```
}


void update_data(){
  byte_i = Serial.available();
  if (byte_i > 0){
    switch (custom_data){
      //case 0 chooses which value will be updated
      case 0: serial_read = Serial.read();
            if (Serial.available() > 0){
              Serial.println("There is no such thing. Input again.");
              Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for
humidity):");
            }
            else{
              if ((serial_read > 48) && (serial_read < 52)){
                custom_data = serial_read - 48;                    //update custom_data
                Serial.print("Please input a value for ");
                if (custom_data == 1)
                  Serial.println("brightness (between 0 to 7):");
                else if (custom_data == 2)
                  Serial.println("temperature (between 10 to 39 degC):");
                else Serial.println("humidity (between 0-100 % RH");
              }
              else{ Serial.println("There is no such thing. Input again.");
                    Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for
humidity):");
```

```
        }
      }
      Serial.flush();
      break;
  //case 1 updates the light value
  case 1: serial_read = Serial.read();
      if (Serial.available() > 0){
        Serial.println("There is no such thing. Input again.");
        Serial.println("Please input a value for brightness:");
      }
      else{
        if ((serial_read > 47) && (serial_read < 56)){
          custom_light = serial_read - 48;
          custom_data = 0;
          Serial.println("Brightness value updated.");
          Serial.println(custom_light);
          Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for
humidity):");
        }
        else{ Serial.println("There is no such thing. Input again.");
            Serial.println("Please input a value for brightness:");
        }
      }
      Serial.flush();
      break;
```

```
//case 2 updates the temperature value
case 2:
{
    float temp_temp, super_temp;
    int flag = 0;
    int i;
    for(i = 1; (i < 5) && (Serial.available() > 0) && (!flag); i++){
      serial_read = Serial.read();
      Serial.println(serial_read);
      if ((i == 1) && (serial_read > 48) && (serial_read < 52))
        temp_temp = (serial_read - 48) * 10; //first digit of temp (1 to 3)
      else if ((i == 2) && (serial_read > 47) && (serial_read < 58))
        temp_temp = temp_temp + serial_read - 48; //second digit of temp (0 to 9) this and
line above give range of 10 to 39 degC
      else if ((i == 3) && (serial_read == 46))
      {} //decimal
      else if ((i == 4) && (serial_read > 47) && (serial_read < 58)){ //point after decimal
        super_temp = serial_read;
        super_temp = (super_temp - 48) / 10;
        temp_temp = temp_temp + super_temp;
      }
      else flag = 1;
    }
    if (((Serial.available() == 0) && (i == 3) && (!flag)) || ((Serial.available() == 0) && (i
== 5) && (!flag))) {
      custom_temp = temp_temp;
```

```
        custom_data = 0;
        Serial.println("Temperature value updated.");
        Serial.println(custom_temp);
        Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for humidity):");
      }
      else{
        Serial.println("There is no such thing. Input again.");
        Serial.println("Please input a value for temperature:");
      }
      Serial.flush();
      break;
  }
  //case 3 updates the humidity value
  case 3: if (Serial.available()){
        delay(5); //make sure we have all the data
        int i=0;
        while(i<5){
          reading_Array[i] = Serial.read();
          i++;
      }

      Serial.flush();

      serial_read = atoi(reading_Array);
```

```
            Serial.println(serial_read);
            if (Serial.available() > 0){
              Serial.println("There is no such thing. Input again.");
              Serial.println("Please input a value for humidity:");
            }
            else{
              if ((serial_read > 0) && (serial_read < 100)){
                custom_hum = serial_read;


                custom_data = 0;
                Serial.println("Humidity value updated.");
                Serial.println(custom_hum);
                Serial.println("Choose a value to update (1 for light, 2 for temperature, 3 for
humidity):");
              }
              else{ Serial.println("There is no such thing. Input again.");
                    Serial.println("Please input a value for humidity:");
              }
            }
            Serial.flush();
            break;
      }
    }
  }
}
```

References

Bai, Y., & Ku, Y. (2008). Automatic Room Light Intensity Detection and Control Using a Microprocessor and Light Sensors. *IEEE Transactions on Consumer Electronics* , 1173.

Costa, E. J., Freire, R. C., Silva, J. B., Cursino, C. M., Oliveira, C. R., Pereira, B. A., et al. (2009). Humidity Control System in Newborn Incubator. *XIX IMEKO World Congress Fundamental and Applied Metrology* , 1760-1764.

Dictionary.com. (2010). *Actuator*. Retrieved February 27, 2010, from Dictionary.com: http://dictionary.reference.com/browse/actuator

How Stuff Works. (2006, June 13). *Bili Lights*. Retrieved January 3, 2010, from How Stuff Works: http://www.howstuffworks.com/bili-lights-dictionary.htm

Jia, Y., Heiss, M., Fu, Q., & Gay, N. A. (2008). A Prototype RFID Humidity Sensor for Built Environment Monitoring. *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing* , 496-499.

National Semiconductor. (2000, November). *Precision Centigrade Temperature Sensors*. Retrieved October 25, 2009, from National Semiconductor: http://cache.national.com/ds/LM/LM35.pdf

Patriciu, A. (2009). *Electrical Engineering 4BD4 Lecture Notes.*

Robot Shop. (2009). *Arduino Duemilanove USB Microcontroller Module*. Retrieved December 4, 2009, from Robot Shop: http://www.robotshop.ca/arduino-usb-microcontroller-board-1-1.html

Robot Shop. (2009). *DFRobot Ambient Light Sensor*. Retrieved January 02, 2010, from Robot Shop: http://www.robotshop.ca/dfrobot-ambient-light-sensor.html

Sencera Co. (n.d.). *808H5V5 Humidity Transmitter*. Retrieved December 21, 2009, from Sensor Element: http://www.sensorelement.com/humidity/808H5V5.pdf

Specialty Lights. (2009, December 31). *Specialty-Lights.com*. Retrieved January 02, 2010, from Light Therapy Boxes: http://www.specialty-lights.com/c12hl.html

Steiner & Martins. (1991). *Piezoelectric Ceramic Solutions*. Retrieved October 25, 2009, from Steminc: http://www.steminc.com

Tatsiopoulos, C., & Ktena, A. (2009). A Smart ZIGBEE Based Wireless Sensor Meter System.

UMHS Newborn Care Committee. (2005, May). *Your Baby, Jaundice, and Phototherapy*. Retrieved October 9, 2009, from http://www.med.umich.edu/1libr/pa/umphototherapy.htm

Vitae

NAME:	Christina Tan
PLACE OF BIRTH:	Hamilton, Ontario
YEAR OF BIRTH:	1988
SECONDARY EDUCATION:	Glendale Secondary School (2002 – 2006)
HONOURS and AWARDS:	Honour Roll 2003, 2004, 2006
	Ontario Scholar 2006
	Chad Collins Community Service Award 2006
	McMaster University Entrance Scholarship 2006