A Project Report on

Ambient Temperature Control System

Submitted in partial fulfilment of the Award of

DISTINCTION IN MEE 312: CONTROL SYSTEMS

By

Group 9

MEE 312 CLASS

Dept. of Mechanical Engineering

Obafemi Awolowo University

April 2017

Submitted to

DR. S.A. Adio

(Coordinator, MEE 312)

Group 9 Members

1. AKINOLA Abidemi MEE/2013/016
2. OLADAPO Farouq MEE/2013/043
3. AKINYOSADE Abayomi MEE/2014/053
4. OMOIKHUNU Mark MEE/2013/049
5. IME Anwana MEE/2010/034
6. AFONUGHE Oghenejuvwerhe MEE/2013/014
7. SOLARU Emmanuel MEE/2013/0
8. ADEBIYI Adebayo MEE/2010/002
9. EGWUATU Vincent MEE/2010/025
10. OYEBAMIJI Ayooluwa Paul MEE/2013/054
11. ONI Olayemi MEE/2010/058
12. OSUJI Simon Chukwuma MEE/2013/051
13. OLOWOLAJU Kehinde Adesuyi MEE/2011/055

Abstract

The aim of the project is to control the temperature of the immediate surroundings in a room being cooled by an electric fan. This was achieved by using an Arduino Uno as the controller and DHT-11 temperature sensor as the feedback device to control the rotation of a 28BYJ-48 rotor (demoed fan). The 28BYJ-48 rotor was driven with the aid of a ULN2003 driver board by the controller using a FULL-STEP driving technique.

Table of Contents

Chapter 1: Introduction

1.1

The purpose of the project was the optimisation of the functioning of an electric fan to improve human comfort by thermal conditioning.

The optimisation was achieved by changing the electric fan control system from an open-loop system to a closed loop system by the introduction of a feedback device.

This feedback device is a temperature sensor which detects the ambient temperature of the room and this information (as an electric signal) is supplied to and processed by the controller. The controller takes action based on the processed feedback information whether the fan should be turned on or off.

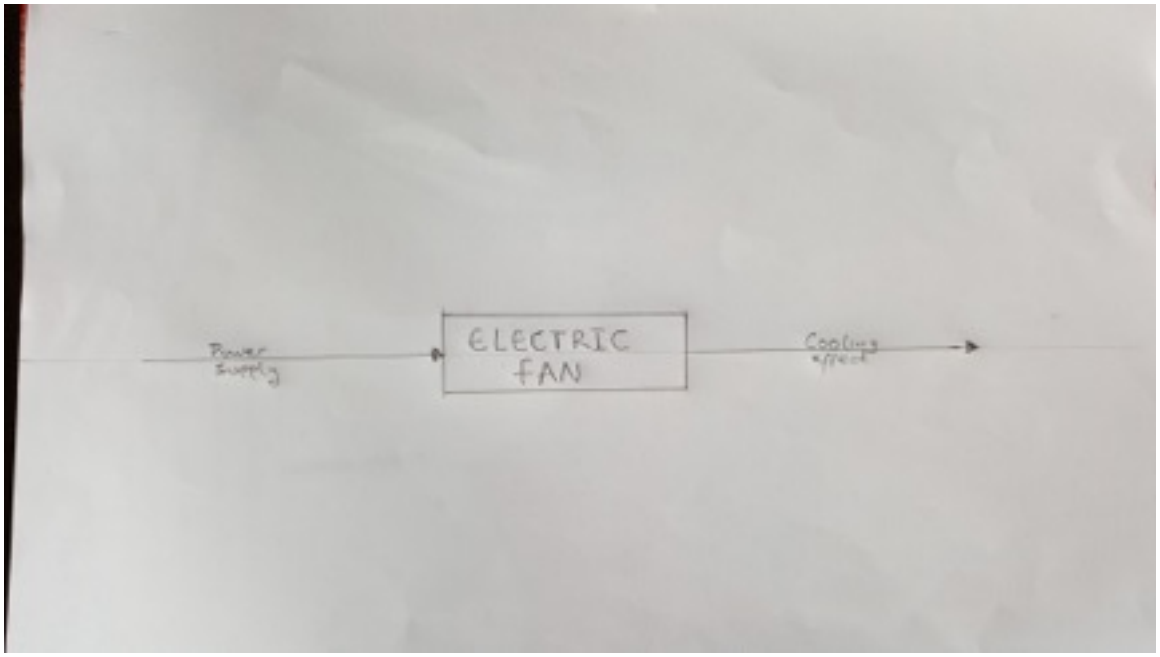

Figure 1: Open-Loop Control System of an Electric Fan

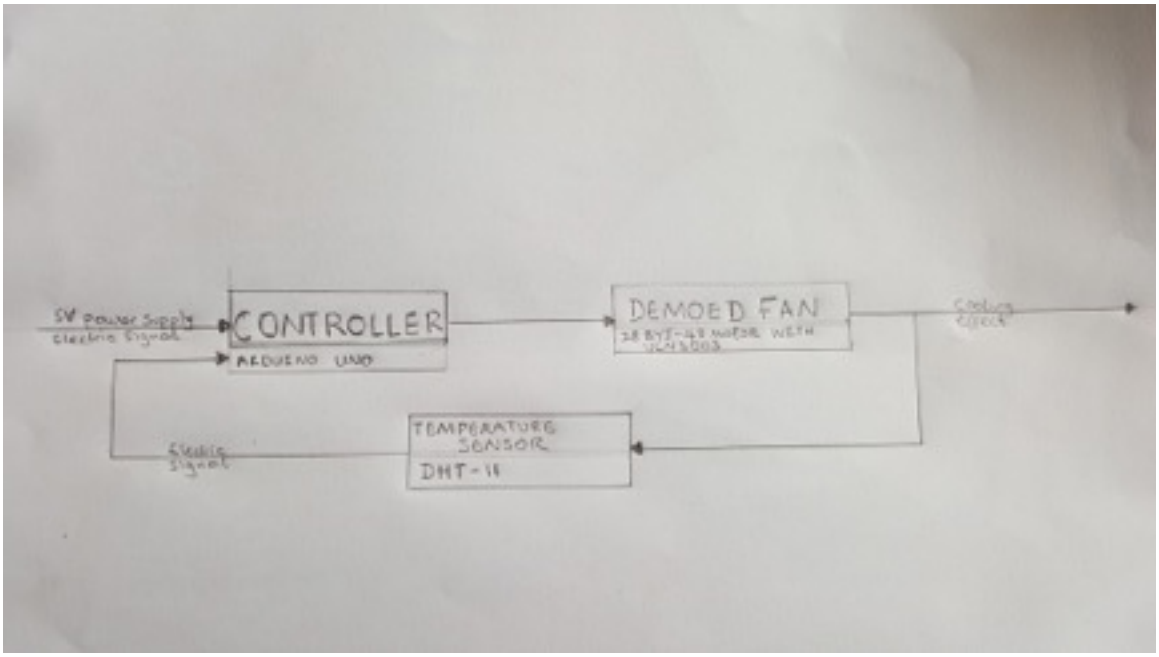

Figure 2: Optimised Closed-Loop Control System

1.2 COMPONENTS USED

1. Micro-controller (Arduino Uno board)
2. Solder-less breadboard
3. Jumper-wires for electrical connections
4. 28BYJ-48 Step Motor with ULN2003 driver
5. DHT-11 sensor
6. External power cable for the Micro-controller
7. External power source (9V batteries)

1.3 QUICK REVIEW OF THE COMPONENTS USED

Arduino Uno is a programmable electronic controller which can receive digital or analog inputs from various types of sensors or transducers and this inputs can be used to affect outputs from electrical devices like LED, electric motor and many other actuator devices.

Electrical Inputs(Voltages) from the sensors can be measured and manipulated with the Arduino programming and computing interface which basically works on Objective-C programming language.

The Arduino Uno basically has three main sets of pins peculiar to this project. They are the power, digital and the analog pins. The power pins contain: ground pins(GND) and 5V pin. There exists 14 digital input/out pins and 5 analog input/output. Other parts of the Arduino uno board are: USB inter face, External power supply port and TX/RX LEDs. A systematic diagram of the Arduino is shown in figure 3 and 4 below.

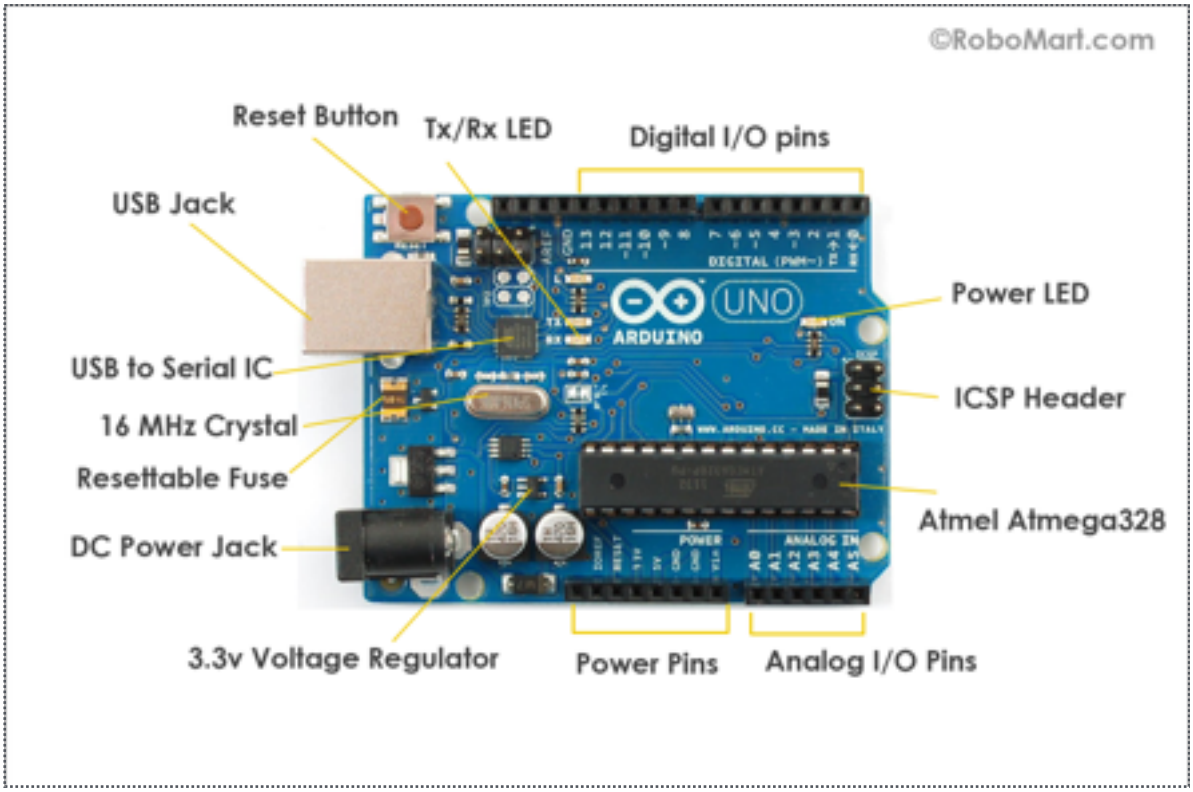

Figure 3: Arduino Uno board Layout 1

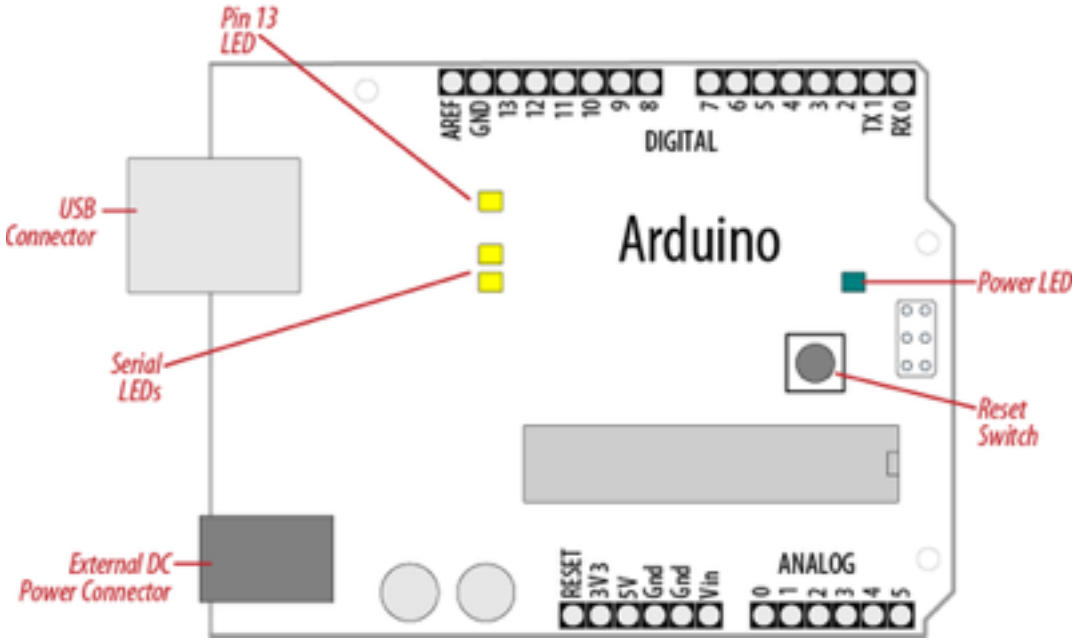

figure 4: Arduino Uno board Layout 2

Solder-less breadboard (or terminal array board)is a construction base for prototyping of electronics. The Breadboard consist of several contact points which are precisely arrayed in rows and columns. Contact points that are inter-connected electrically are called Strips. On a bread board, there exists two classes of strips: Bus strip and Terminal Strip. There are two sets of strips Bus strip and Terminal Strip: a set on the right and the other on the left. The terminal strips are radially connected and a labelled a-j on the board. The Bus strips usually contains two rows: one for ground and one for a supply voltage. Each of the rows are electrically connected to each other.

The breadboard layout is shown in the figures below.

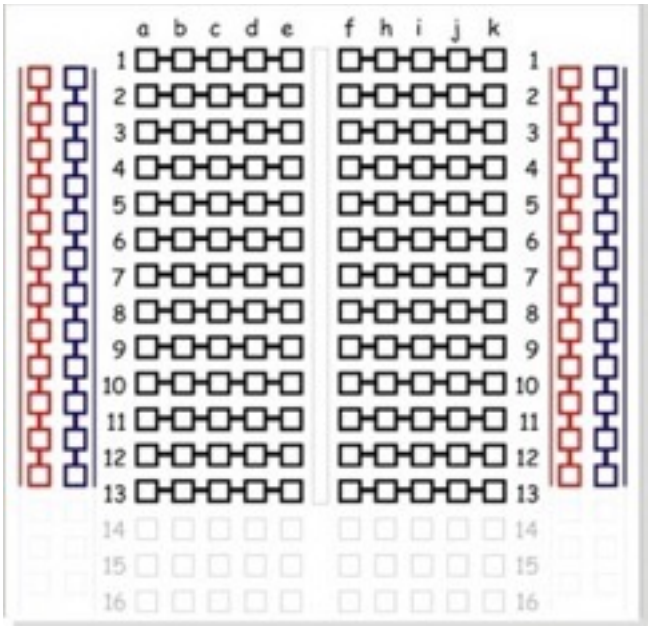

Figure 5: Schematics of the breadboard

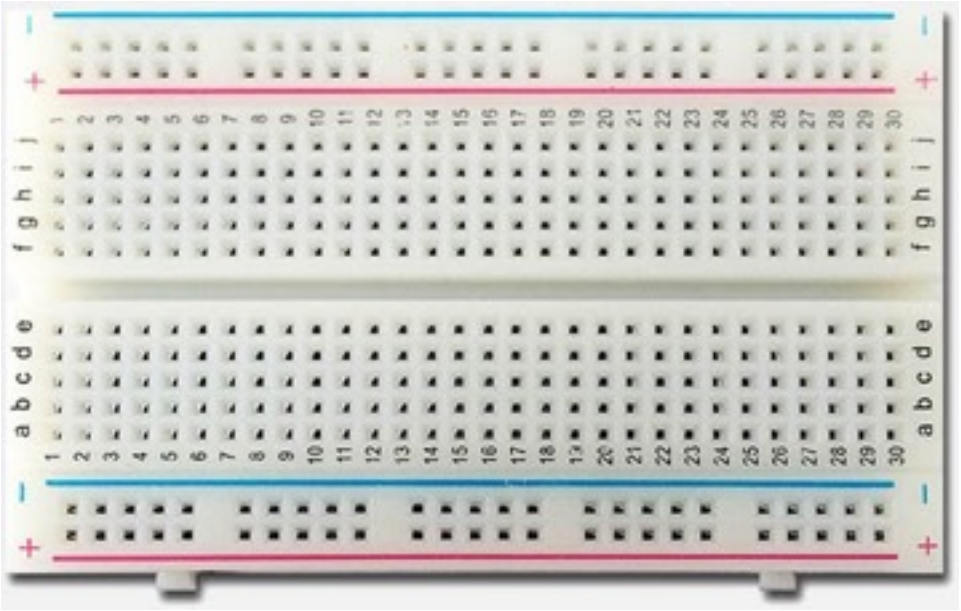

Figure 6: Layout of the solder-less breadboard

Jumper Wires are basically used for electric inter-connections between the contact points on the Bread board.

figure 5: Image of Jumper wires

28BYJ-48 Step Motor is a unipolar stepper motor controlled by a series of electromagnetic coils. It basically works in conjunction with a ULN2003 driver board. 28BYJ-48 stepper motor consists of two electromagnetic coils with each connected to several gears meshed together and in-turn with the output gear. These two coils are independently powered electrically through four different colour-coded wires (blue, pink yellow and orange) with a fifth wire (red) as the ground .

figure 6: Picture of the 28BYJ-48 motor

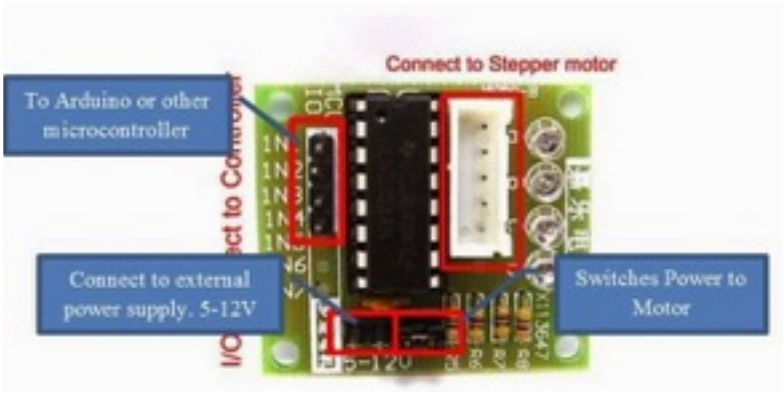

Figure 7: ULN2003 driver board layout

DHT-11 is a sensor for measuring temperature and humidity. it has a temperature measuring range of 0 - 50 degree celsius and humidity measuring range of 20 - 95%. The DHT-11 has three line: GND(-), +5V and a single data line(S). The data line was connected to the digital pin 7 in this experiment.

Figure 8: DHT-11 overview

External power cable was used to connect the 9V battery to the Arduino via the power port on the Arduino.

Digital Multi-meter was used to test for inter-connectivity or continuity in the electrical connections and also for raw output measurement.

Chapter 2: Methods and processes

2.3.1 SETTING UP INDEPENDENT COMPONENTS OF THE PROJECT

2.3.1.1 SETTING UP THE ARDUINO UNO

A end of two different jumper wires (preferably red and white in colour respectively) were connected to the 5V port and any one of the ground(GND) port on the micro-controller board. The USB cable was connected to the USB port of the Arduino board and the other end was not connected immediately to the computer until all electrical connections to and on the bread board are completed and verified.

The Arduino interface was downloaded from the Arduino website and installed on the computer to complete the Arduino set-up.

2.3.1.2 SETTING UP THE 28BYJ-48 STEPPER MOTOR

All these wires from the 28BYJ-48 were connected to the ULN2003 driver board. From this board, the four input pins(IN1, IN2, IN3, IN4) were connected with the aid of jumper wires to digital pins 2,3, 4 and 5 respectively. For convenience, the input channels to the Arduino were colour coded with jumper wires (blue,pink,yellow and orange respectively) in other to match the colour coding for power inputs to the motor to the driver board.

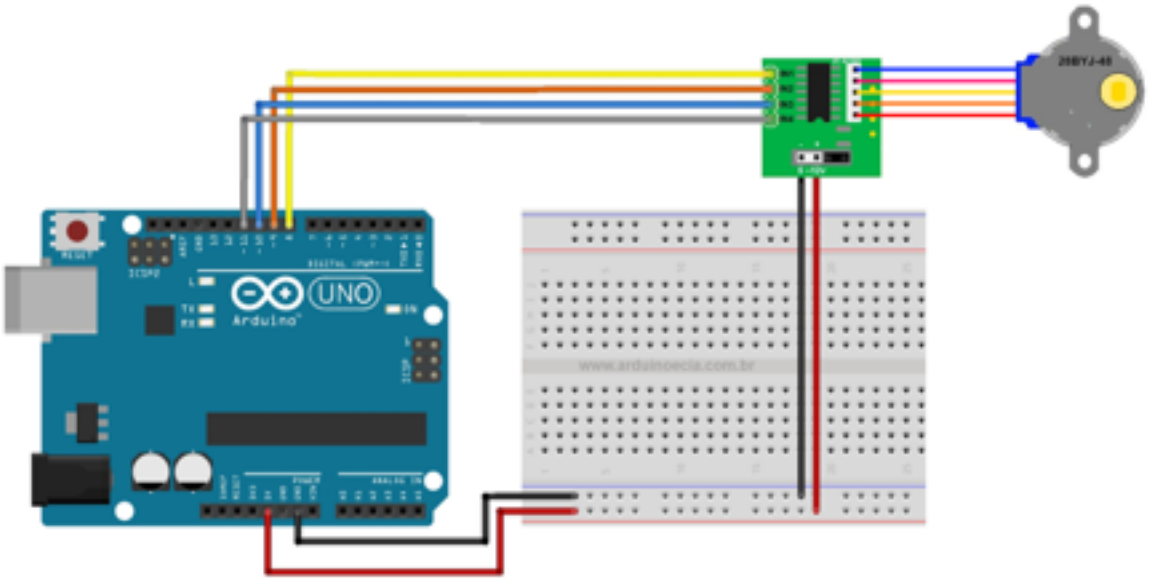

Figure 9: Wiring the Rotor to the Arduino

Of the different driving methods for the stepper motor, the FULL-STEP drive was used in order to produce the highest possible torque to appropriately demo or simulate an electric fan, nevertheless the motor is a relatively low speed motor. The power pins for the driver board were also connected to the used bus terminal on the breadboard.

There are eight different steps per cycle of the motor with the first four steps identical to the other four. The peculiarity of the FULL-STEP drive is that, only two different combinations of the input power **to the motor** are on at each step of a half cycle.

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
IN1 (blue)	ON	OFF	OFF	ON	ON	OFF	OFF	ON
IN2 (pink)	ON	ON	OFF	OFF	ON	ON	OFF	OFF
IN3 (yellow)	OFF	ON	ON	OFF	OFF	ON	ON	OFF
IN4 (orange)	OFF	OFF	ON	ON	OFF	OFF	ON	ON

Figure 2: ON-OFF configuration for each step of the FULL-STEP driving method

The step mode of the motor was pre-programmed on the Arduino programming interface. The program used was documented in Appendix D.

After all electrical connections were made and the code written, the USB from the Arduino was connected to the computer and the code was uploaded to the Arduino. Then the motor starts to rotate continuously.

2.3.1.3 SETTING UP THE DHT-11 TEMPERATURE AND HUMIDITY SENSOR

The DHT-11 sensor was longitudinally placed in the breadboard terminal strip such that each of its terminals were in three different terminal strip that are not electrically continuous. With the aid of three different jumper wires, each of these terminal strips connected to the (-), middle and (s) terminals were connected to the ground strip (on the bus terminal of the breadboard), power strip (on the bus terminal of the breadboard), and digital pin 7 (on the Arduino board). Thereafter a computer program is written to get the temperature and humidity values per set time interval (delay). Usually there exist the DHT-11 library which does the digital to analog value manipulations. The computer program written was just to enable appropriate outputting of the sensor values per time. The DHT-11 computer program including where the library used can be found, was documented in Appendix E.

The compiled code was then uploaded to the Arduino and the Arduino serial monitor then produced a visual Output of the temperature and humidity readings.

2.3.2 SETTING UP THE ENTIRE PROJECT AS A WHOLE

After each of these major components had been completely set-up and individually tested, the Arduino program for each of the functioning of the individual components is then appropriately collated and the code is then updated. The main update was to set our threshold temperature and

ensure that the modelled fan(rotor) was allowed to rotate only when the temperature is above a threshold temperature. The combined Arduino program is shown in Appendix F.

Chapter 3: Results and Discussion

The readings obtained from the DHT-11 sensors were displayed and viewed on the serial monitor of the Arduino Programming Interface on the computer. It was observed that at temperatures greater than or equal to the set-point (threshold), the electric fan model (28BYJ-48 rotor) was observed to rotate while at temperatures less than the set-point, no rotation was observed.

From the above observation, we were able, to some degree of efficiency, model an electric fan which operates as a closed-loop temperature control system.

Setting an optimum delay time (in the Arduino program) for the temperature measurement processes performed by the controller (through the sensor) proved to be a very important factor to the proper functioning of the closed loop system.

The “if “ statement method was used in the controller program to compare the measured value of temperature to the set-point and initiate the rotor action as required. It was noted that using a very large delay time for the sensor resulted in very accurate temperature measurements will little or no error. The effect of large sensor delay time resulted in fractional rotation of the rotor for temperatures greater or equal to the set-point as the time between each cycle(8 steps) of rotation of the rotor increases.

As the sensor delay time was reduced, the sensor error was increased but the rotation of the fan was found to be smoother.

Using a “while” statement in the program annulled the effect of large sensor delay time but had an adverse effect that could not be ignored. Once the temperature measured by the sensor is equal to the set-point, it starts up the rotor but the controller ceased to detect any further change in temperature.

A proposed remedy to the effect of the large delay time on the rotor is to use a relay switch for the cutting off power from the rotor when temperature is below the set-point and for switching on the rotor when the sensed temperature is equal to or greater than the threshold temperature.

Appendices

APPENDIX A: ARDUINO UNO

The Arduino Uno is a micro-controller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the micro-controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega16U2 (ATmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.

Stronger RESET circuit.

ATmega 16U2 replace the 8U2.

"Uno" means *one* in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform.

ARDUINO MICROCONTROLLER

Microcontroller	
Architecture	AVR
Operating Voltage	5V
Flash memory	32 KB of which 0.5KB used by bootloader
SRAM	2KB
Clock Speed	16MHZ
Analog I/O Pins	6
EEPROM	1KB
DC Current per I/O	40 mA on I/O Pins; 50mA on 3.3V Pin

GENERAL

Input Voltage	7-12V
Digital I/O Pins	20 (of which 6 provide PWM output)
PWM Output	6
PCB Size	53.4 x 68.6
Weight	25g
Product Code	A000066 (TH); A000073 (SMD)

APPENDIX B: SOLDERLESS BREADBOARD

Typical specifications

A modern solder-less breadboard consists of a perforated block of plastic with numerous tin plated phosphor bronze or nickel silver alloy spring clips under the perforations. The clips are often called tie points or contact points. The number of tie points is often given in the specification of the breadboard.

The spacing between the clips (lead pitch) is typically 0.1 in (2.54 mm). Integrated circuits (ICs) in dual in-line packages (DIPs) can be inserted to straddle the centreline of the block. Interconnecting wires and the leads of discrete components (such as capacitors, resistors, and inductors) can be inserted into the remaining free holes to complete the circuit. Where ICs are not used, discrete components and connecting wires may use any of the holes. Typically the spring clips are rated for 1 ampere at 5 volts and 0.333 amperes at 15 volts (5 watts). The edge of the board has male and female notches so boards can be clipped together to form a large breadboard.

Bus and terminal strips

Solder-less breadboards are available from several different manufacturers, but most share a similar layout. The layout of a typical solder-less breadboard is made up from two types of areas, called strips. Strips consist of interconnected electrical terminals.

Breadboard consisting of only terminal strips but no bus strips

Terminal strips

The main areas, to hold most of the electronic components.

In the middle of a terminal strip of a breadboard, one typically finds a notch running in parallel to the long side. The notch is to mark the centre line of the terminal strip and provides limited airflow (cooling) to DIP ICs straddling the centreline. The clips on the right and left of the notch are each connected in a radial way; typically five clips (i.e., beneath five holes) in a row on each side of the notch are electrically connected. The five rows on the left of the notch are often marked as A, B, C, D, and E, while the ones on the right are marked F, G, H, I and J. When a "skinny" dual in-line pin package (DIP) integrated circuit (such as a typical DIP-14 or DIP-16, which have a 0.3-inch (7.6 mm) separation between the pin rows) is plugged into a breadboard, the pins of one side of the chip are supposed to go into row E while the pins of the other side go into row F on the other side of the notch. The columns are numbered 1 - 50 or whatever number of columns there are.

Bus strips

To provide power to the electronic components.

A bus strip usually contains two rows: one for ground and one for a supply voltage. However, some breadboards only provide a single-row power distributions bus strip on each long side. Typically the row intended for a supply voltage is marked in red, while the row for ground is marked in blue or black. Some manufacturers connect all terminals in a column. Others just connect groups of, for example, 25 consecutive terminals in a column. The latter design provides a circuit designer with some more control over crosstalk (inductively coupled noise) on the power supply bus. Often the groups in a bus strip are indicated by gaps in the colour marking.

Bus strips typically run down one or both sides of a terminal strip or between terminal strips. On large breadboards additional bus strips can often be found on the top and bottom of terminal strips. Note there are two different common alignments for the power bus strips. On small boards, with about 30 rows, the holes for the power bus are often aligned between the signal holes. On larger boards, about 63 rows, the power bus strip holes are often in alignment with the signal holes. This makes some accessories designed for one board type incompatible with the other. For example, some Raspberry Pi GPIO to breadboard adapters use offset aligned power pins, making them not fit breadboards with aligned power bus rows. There are no official standards, so the users need to pay extra attention to the compatibility between a specific model of breadboard and a specific accessory. Vendors of accessories and breadboards are not always clear in their specifications of which alignment they use. Seeing a close up photograph of the pin/hole arrangement can help determine compatibility.

Some manufacturers provide separate bus and terminal strips. Others just provide breadboard blocks which contain both in one block. Often breadboard strips or blocks of one brand can be clipped together to make a larger breadboard.

In a more robust variant, one or more breadboard strips are mounted on a sheet of metal. Typically, that backing sheet also holds a number of binding posts. These posts provide a clean way to connect an external power supply. This type of breadboard may be slightly easier to handle. Several images in this article show such solder-less breadboards.

APPENDIX C: 28BYJ-48 STEPPER MOTOR

A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied to it in the proper sequence. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied.One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system. Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.

Features

1. The rotation angle of the motor is proportional to the input pulse.
2. The motor has full torque at standstill(if the windings are energised)
3. Precise positioning and repeatability of movement since good stepper motors have an accuracy of – 5% of a step and this error is non cumulative from one step to the next.
4. Excellent response to starting/stopping/reversing.
5. Very reliable since there are no contact brushes in the motor. Therefore the life of the motor is simply dependant on the life of the bearing.
6. The motors response to digital input pulses provides open-loop control, making the motor simpler and less costly to control.
7. It is possible to achieve very low speed synchronous rotation with a load that is directly coupled to the shaft.
8. A wide range of rotational speeds can be realised as the speed is proportional to the frequency of the input pulses.

Stepper motor 28BYJ-48 Parameters

Model： 28BYJ-48

Rated voltage： 5VDC

Number of Phase： 4

Speed Variation Ratio： 1/64

Stride Angle： 5.625° /64

Frequency : 100Hz

DC resistance： 50Ω±7%(25°C)

Idle In-traction Frequency : > 600Hz

Idle Out-traction Frequency : > 1000Hz

In-traction Torque >34.3mN.m(120Hz)

Self-positioning Torque >34.3mN.m

Friction torque : 600-1200 gf.cm

Pull in torque : 300 gf.cm

Insulated resistance >10MΩ(500V)

Insulated electricity power： 600VAC/1mA/1s

Insulation grade： A

Rise in Temperature <40K(120Hz)

Noise <35dB(120Hz,No load,10cm)

APPENDIX D: CODE FOR THE DRIVING OF THE STEPPER MOTOR

```
void setup() {
 // put your setup code here, to run once:
 // initialise all the pins used for the Output signal to the driver board
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
}
// create an overall class to perform a complete driving cycle
void clockwiserotate(){
 step1();
 step2();
 step3();
 step4();
 step5();
 step6();
 step7();
 step8();
}
// create classes for each step mode configuration of the driving cycle
```

```
void step1() {
 digitalWrite(2,HIGH);
 digitalWrite(3,HIGH);
 digitalWrite(4,LOW);
 digitalWrite(5,LOW);
 delay(3);
}
void step2() {
 digitalWrite(2,LOW);
 digitalWrite(3,HIGH);
 digitalWrite(4,HIGH);
 digitalWrite(5,LOW);
 delay(3);
}
void step3() {
 digitalWrite(2,LOW);
 digitalWrite(3,LOW);
 digitalWrite(4,HIGH);
 digitalWrite(5,HIGH);
 delay(3);
}
void step4() {
 digitalWrite(2,HIGH);
 digitalWrite(3,LOW);
 digitalWrite(4,LOW);
 digitalWrite(5,HIGH);
 delay(3);
}
void step5() {
 digitalWrite(2,HIGH);
 digitalWrite(3,HIGH);
 digitalWrite(4,LOW);
 digitalWrite(5,LOW);
 delay(3);
}
void step6() {
 digitalWrite(2,LOW);
 digitalWrite(3,HIGH);
 digitalWrite(4,HIGH);
 digitalWrite(5,LOW);
 delay(3);
}
void step7() {
 digitalWrite(2,LOW);
 digitalWrite(3,LOW);
 digitalWrite(4,HIGH);
 digitalWrite(5,HIGH);
 delay(10);
}
void step8() {
 digitalWrite(2,HIGH);
 digitalWrite(3,LOW);
 digitalWrite(4,LOW);
 digitalWrite(5,HIGH);
 delay(3);
}
void sleep() {
 digitalWrite(2,LOW);
 digitalWrite(3,LOW);
 digitalWrite(4,LOW);
```

```
 digitalWrite(5,LOW);
 delay(3);
}


void loop() {
  // Repeat each step iteratively
   clockwiserotate();

}
```

APPENDIX E: ARDUINO CODE FOR THE DRIVING OF THE STEPPER MOTOR

```
#include <dht.h>

dht DHT;

#define DHT11_PIN 7

void setup(){
 Serial.begin(9600);
}

void loop()
{
 int chk = DHT.read11(DHT11_PIN);
 Serial.print("Temperature = ");
 Serial.println(DHT.temperature);
 Serial.print("Humidity = ");
 Serial.println(DHT.humidity);
 delay(1000);
}
```

The DHT-11 libraries can be got from this link:

APPENDIX F: COMBINED ARDUINO CODE FOR THE ENTIRE PROJECT

```
#include <dht.h>

dht DHT;

#define DHT11_PIN 7

void setup() {
 // put your setup code here, to run once:
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 Serial.begin(9600);
}

void clockwiserotate(){
 step1();
 step2();
 step3();
 step4();
 step5();
 step6();
 step7();
 step8();
}

void step1() {
 digitalWrite(2,HIGH);
 digitalWrite(3,HIGH);
 digitalWrite(4,LOW);
 digitalWrite(5,LOW);
 delay(3);
}
void step2() {
 digitalWrite(2,LOW);
 digitalWrite(3,HIGH);
 digitalWrite(4,HIGH);
 digitalWrite(5,LOW);
 delay(3);
}
void step3() {
 digitalWrite(2,LOW);
 digitalWrite(3,LOW);
 digitalWrite(4,HIGH);
 digitalWrite(5,HIGH);
 delay(3);
}
void step4() {
 digitalWrite(2,HIGH);
 digitalWrite(3,LOW);
 digitalWrite(4,LOW);
 digitalWrite(5,HIGH);
 delay(3);
```

```
}
void step5() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  delay(3);
}
void step6() {
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  delay(3);
}
void step7() {
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  delay(3);
}
void step8() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  delay(3);
}
void sleep() {
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  delay(3);
}
void loop() {
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(80);
  int i;
  // Threshold temperature Checker
  if (DHT.temperature>=28) {
     // Responsible for the turning on and off of the rotor
     for (i=0;i<50;i++) {
       clockwiserotate();
    }

}
}
```