جامعة بيروت العربية
BEIRUT ARAB UNIVERSITY

Controlling a Fan Coil Unit Project

MCHE 416L – Mechatronics System Design Lab

by

Mohammad Omar Shehab, 201801047

Submitted to the
Department of Mechanical Engineering
Faculty of Engineering

Instructor

Eng. Mostafa Najim

Spring 2019-2020

Table of Contents

List of Tables

List of Figures

List of Equations

I. Introduction

The mechanization of buildings in the early 20th century has increased the demand for central heating and cooling making HVAC systems more than a supplementary addition to any building [1]. HVAC, or heating, ventilation, and air conditioning, systems are used in a wide variety of systems from the air conditioner in one's car to large systems of air handling units and fan coil units found in industrial complexes. Furthermore, they combine principles of thermodynamics, fluid mechanics, and heat transfer to provide thermal control and indoor comfort [2]. **Figure 1** shows the basic functioning of an HVAC system.

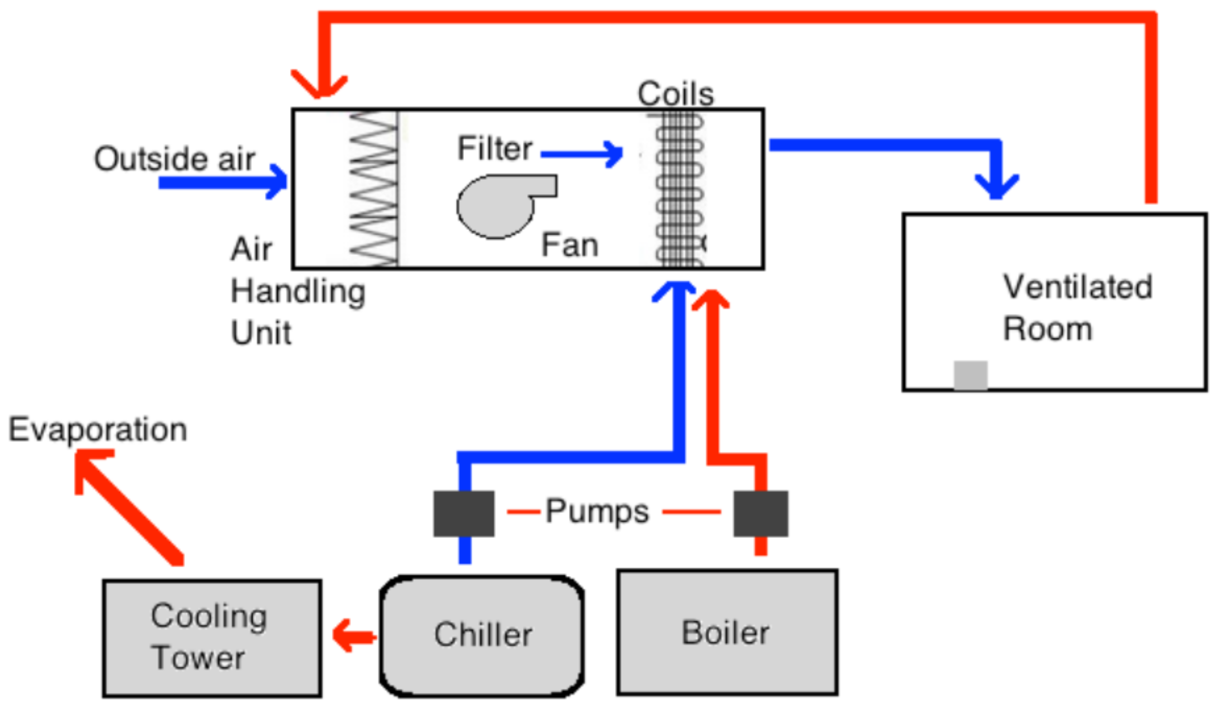


Figure 1 Basic Function of an HVAC System

One of the main components of a HVAC system, especially those found in buildings and commercial centers, is the FCU, or fan coil unit. A FCU consists of a fan for blowing the conditioned air, a heating coil for heating the supply air, a cooling coil for cooling the supply air, and multiple filters for filtering out particulates present in the air [3]. **Figure 2** is basic representation of a FCU.

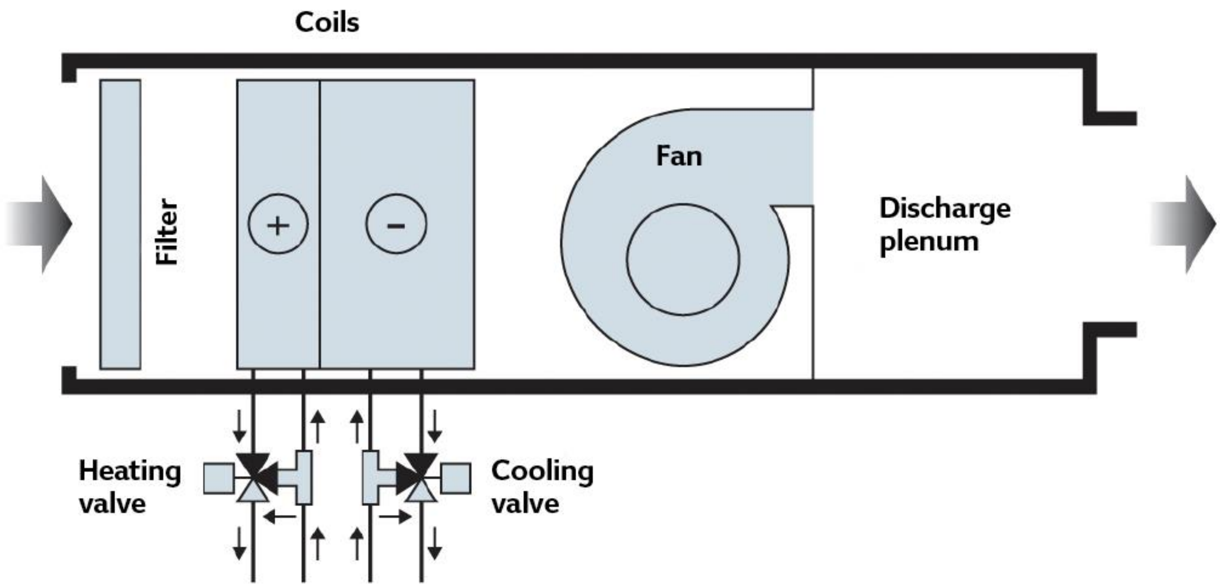


Figure 2 Schematic Representation of an FCU

II. Objective

The aim of this project to control a FCU using an Arduino Uno where a certain sequence of operations is to be respected. Moreover, the action to be taken is determined according to a set of inputs to the Arduino.

Usually, a fan coil unit can be controlled individually or as a part of the Building Management Services. In both cases, the sequence of operations is as follows:

1) The FCU is activated when the residents of the conditioned space toggle the system’s on/off switch (and this simulated by the press of a momentary push button).
2) Then the outside brightness is measured, and a LED turns on/off accordingly (turns on if dark and vice versa).
3) A temperature sensor measures the room temperature, and the Arduino then compares it to the desired temperature set by the occupants via a thermostat (this is simulated using a potentiometer).
4) If the setpoint temperature is greater than the room temperature, heating mode is activated by the opening the heating valve. However, if the former is less than the latter, cooling mode is activated by opening the cooling valve. Both valves are connected to

a relay where the cooling valve is connected to the normally closed contacts, and the heating valve is connected to the normally open contacts.

5) Now, if the absolute difference between the 2 input temperatures, the setpoint and room temperatures, is greater than 2°C, the fan is set on high speed. If it is between 1°C and 2°C, the fan is set on medium speed. Moreover, if it is between 0°C and 1°C, the fan is set on low speed. Lastly, if it equals zero, the fan turns off.
6) The sequence should continue looping until the residents press the momentary button again after which every component turns off.

The following components are used in this project:

1) An Arduino Uno
2) One momentary button
3) An LM35DZ temperature sensor
4) A potentiometer
5) A 10KΩ LDR
6) One blue LED
7) One red LED
8) One yellow LED
9) Three 220Ω resistors
10) One 2KΩ resistor
11) One 10KΩ resistor
12) One relay
13) One Tip120 transistor
14) One 9V DC motor
15) One 9V DC power supply
16) One breadboard
17) Connecting wires

III. Arduino Code

Before writing the code, the input and outputs should be identified.

A. Inputs

There are 3 analog inputs and 1 digital input. The room temperature, setpoint temperature, and the outside brightness are the analog inputs, and the system button is the digital input. The room temperature is measured by the LM35DZ sensor, the outside brightness is measured by the LDR, and the setpoint temperature is selected by a potentiometer. **Table 1** lists specific details about each input.

Table 1 The Inputs of The System

Input	*Name*	*Pin*	*Input Data Type*	*Data Container*
Potentiometer	temp_setpoint	Analog Pin A0	Float	user_setpoint
LM35DZ	temp_sensor	Analog Pin A1	Float	room_temp

LDR	brightness_sensor	Analog Pin A2	Float	outside_brigthness
System Button	occupancy_toggle	Digital Pin 2	Boolean	occupancy_state

B. Outputs

The indoor lights, simulated by turning on/off a yellow LED, are to be turned first after initiating the system. Next, depending on the setpoint temperature and room temperature, the air is either cooled or heated, and this is achieved through using a relay that turns the determined mode. In addition, a LED indicator (blue LED to indicate cooling and red LED to indicate heating) will turn on accordingly. Lastly, the absolute difference between the two input temperatures determines the speed at which the fan is to operate. Therefore, there are five outputs. **Table 2** lists specific details about each output.

Table 2 The Outputs of The System

Output	*Name*	*Pin*	*Output Data Type*
Yellow LED	room_bulb	Digital Pin 3	Boolean
Valve Control Relay	valve_relay	Digital Pin 4	Boolean
Fan	fan	Digital Pin 5	PWM
Blue LED	cooling_indicator	Digital Pin 6	Boolean
Red LED	heating_indicator	Digital Pin 7	Boolean

C. Transfer Functions

The Arduino measures quantization levels instead of the actual physical property. Hence, the transfer function that relates the quantization level to the temperature should be determined.

1) LM35DZ Transfer Function

This sensor, a datasheet of which is found in the appendix, is linear and can measure a range of [0, 100°C] with a gain of 10mV/°C. Thus, the relationship between the voltage output and temperature is:

Equation 1 Relationship Between Room Temperature and Voltage Output of LM35DZ

$$V = 0.01 \times T \quad (1)$$

The analog pins of the Arduino are 10-bit. Furthermore, using internal or external reference voltages affects the output of the LDR and the potentiometer; thus, it is best to use the default reference voltage that is 5V. Hence, the relationship between the voltage and quantization level is the following:

Equation 2 Relationship Between Voltage Input and Quantization Level of Analog Pins

$$Q_{level} = \frac{V - V_{min}}{V_{max} - V_{min}} \times Q_{level,max} = \frac{V - 0}{5 - 0} \times 1023 = 204.6 \times V \qquad (2)$$

Therefore, the transfer function between the room temperature and the quantization level is:

Equation 3 Transfer Function of LM35DZ

$$T = \frac{1}{2.046} Q_{level} \qquad (3)$$

2) Potentiometer Transfer Function

The setpoint temperature range is [16°C, 30°C], and the potentiometer is connected in a way such that rotating it clockwise increases the setpoint temperature. That is why the following boundary conditions are chosen:

$$\begin{cases} if\ Q_{level} = 0\ \ then\ T_{setpoint} = 16℃ \\ if\ Q_{level} = 1023\ \ then\ T_{setpoint} = 30℃ \end{cases}$$

Consequently, the transfer function between the quantization level and the setpoint temperature is:

$$Q_{level} = \frac{T - T_{min}}{T_{max} - T_{min}} \times 1023 = \frac{T\ -\ 16}{30\ -\ 16} \times 1023$$

Equation 4 Transfer Function of Potentiometer

$$T = \frac{1023}{14} \times Q_{level} + 16 \qquad (4)$$

D. The Code

The code is written using simple C programming language, and it is the following:

The first of the code contains all variables to be used.

```
//Inputs
const int temp_setpoint = A0; //pot for controlling setpoint temp
const int temp_sensor = A1; //LM35DZ for measuring room temp
const int brightness_sensor = A2; //LDR for measuring outside brightness
const int occupancy_toggle = 2; //Push button that controls the whole system

//Outputs
const int room_bulb = 3; //the bulb that turns on based on the outside brightness
const int fan = 5; //fan that pushes the air into the room
const int valve_relay = 4; //relay that controls which mode (heating or cooling) turns on
const int cooling_indicator = 6; //blue LED that indicates when cooling is turned on
const int heating_indicator = 7; //red LED that indicates when heating is turned on

//Input Containers
float room_temp; //temp measured by the temp sensor
float user_setpoint; //temp at which the user wants room to be
float delta_T;
float outside_brightness; //outside brightness measured by the LDR
int occupancy_state; //determines if people are present

//Current Variables
float current_room_temp = 0; //to display room temp once
float current_delta_T = 0;
float current_outside_brightness = 0; //to display outdoor brightness once
float current_user_setpoint = 0; //to display desired room temperature once

//Non-physical Variables
int p = 0; //changes push button input to a pulse
boolean occupancy_state_save = LOW; //saves state of push button
```

```
//Time Delay
int time = 500;

//Fan Speeds
int half_speed = 128; //Half speed is 1/2 of 255 (approximately 128) since the digital pins are 8-bit
int low_speed = 64; //I chose low speed to be 1/4 of full speed i.e. 255/4
```

The second part of the code contains the setup of the inputs and outputs and the initial states of the outputs.

```
void setup(){

  //Inputs
  pinMode(temp_setpoint, INPUT);
  pinMode(temp_sensor, INPUT);
  pinMode(brightness_sensor, INPUT);
  pinMode(occupancy_toggle, INPUT);

  //Outputs
  pinMode(room_bulb, OUTPUT);
  pinMode(fan, OUTPUT);
  pinMode(valve_relay, OUTPUT);
  pinMode(cooling_indicator, OUTPUT);
  pinMode(heating_indicator, OUTPUT);

  //Initial states
  digitalWrite(fan, LOW);
  digitalWrite(room_bulb, LOW);
  digitalWrite(valve_relay, LOW);
  digitalWrite(cooling_indicator, LOW);
  digitalWrite(heating_indicator, LOW);

  //Serial Communication
  Serial.begin(9600);
```

```
  delay(time);
  Serial.println("System is now online.");
  delay(time);
  Serial.println("Jarvis is awaiting your command.");
  Serial.println("---------------------------------------");
}
```

The last part contains the continuous loop.

```
void loop(){

 //System Initialization
  occupancy_state = digitalRead(occupancy_toggle);
  if (occupancy_state == HIGH && p == 0){
    p = 1;
    Serial.println("This room shows signs of life");
    Serial.println("---------------------------------------");
  }

  while (p == 1){
    //Measurements
    delay(0.5*time);
    user_setpoint = 14*analogRead(temp_setpoint)/1023 + 16;
    room_temp = analogRead(temp_sensor)/2.046;
    outside_brightness = analogRead(brightness_sensor);
    delta_T = room_temp - user_setpoint;

    if(outside_brightness != current_outside_brightness ||
room_temp != current_room_temp || user_setpoint !=
current_user_setpoint){

      //Evaluating Outside Temperature
      if (outside_brightness >= 600){
        digitalWrite(room_bulb, LOW);
        Serial.print("It is daytime of brightness value: ");
        Serial.println(outside_brightness);
```

```
   Serial.println("------------------------------------");
  }

  else{
    digitalWrite(room_bulb, HIGH);
    Serial.print("It is nighttime of brightness value: ");
    Serial.println(outside_brightness);
    Serial.println("------------------------------------");
  }
  delay(4*time);

  //Displaying Room Temperature
  Serial.print("The room's temperature is: ");
  Serial.print(room_temp);
  Serial.println("°C");

  //Displaying Setpoint Temperature
  Serial.print("The desired room temperature is: ");
  Serial.print(user_setpoint);
  Serial.println("°C");

  //Controlling Heating and Cooling Valves
  if (delta_T >= 0) { //This means room is hot.
    digitalWrite(valve_relay, LOW);
    digitalWrite(cooling_indicator, HIGH);
    digitalWrite(heating_indicator, LOW);
    delay(time);
    Serial.println("Cooling valve has been opened.");
  }
  else { //This means room is cold.
    digitalWrite(valve_relay, HIGH);
    digitalWrite(heating_indicator, HIGH);
    digitalWrite(cooling_indicator, LOW);
    delay(time);
    Serial.println("Heating valve has been opened.");
  }
```

```
    delay(6*time);

    //Fan Control
    if (abs(delta_T) >= 2){
      digitalWrite(fan, HIGH);
      delay(time);
      Serial.println("Fan is on high speed");
      Serial.println("------------------------------------");
    }

    else if (abs(delta_T) >= 1){
      analogWrite(fan, half_speed);
      delay(time);
      Serial.println("Fan is on medium speed.");
      Serial.println("------------------------------------");
    }

    else if (abs(delta_T) > 0){
      analogWrite(fan, low_speed);
      delay(time);
      Serial.println("Fan is on low speed.");
      Serial.println("------------------------------------");
    }
    else {
      digitalWrite(fan, LOW);
      delay(time);
      Serial.println("Fan is off.");
      Serial.println("------------------------------------");
    }
  }

  //Exitng while(p == 1)
  occupancy_state = digitalRead(occupancy_toggle);
  if (occupancy_state == HIGH && p == 1){
    occupancy_state_save = HIGH;
    p = 2;}
}
```

```
//System Standby
if (occupancy_state_save == HIGH && p == 2){
  p = 0;
  occupancy_state_save = 0;
  current_room_temp = 0;
  current_user_setpoint = 0;
  current_delta_T = 0;
  current_outside_brightness = 0;
  Serial.println("Humans are evacuating the room.");
  delay(time);
  Serial.println("Initiating standby mode in...");
  delay(2*time);
  Serial.println("3...");
  delay(2*time);
  Serial.println("2...");
  delay(2*time);
  Serial.println("1...");
  delay(2*time);

  //Turning off HVAC System (Fan and Valves)
  Serial.println("Turning off HVAC system...");
  delay(time);
  digitalWrite(fan, LOW);
  delay(time);
  Serial.println("Fan is turned off.");
  delay(time);
  digitalWrite(valve_relay, LOW);
  digitalWrite(cooling_indicator, LOW);
  digitalWrite(cooling_indicator, LOW);
  digitalWrite(heating_indicator, LOW);
  Serial.println("Valves are closed.");
  delay(time);
  Serial.println("HVAC system is now off.");
  delay(2*time);

  //Turning off Indoor Lighting
```

```
    Serial.println("Turning off indoor lighting...");
    delay(time);
    digitalWrite(room_bulb, LOW);
    delay(time);
    Serial.println("Indoor lighting is off.");
    delay(time);
    Serial.println("Jarvis is on standby mode.");
    delay(time);
    Serial.println("Press the system button to wake me up.");
    Serial.println("---------------------------------------");
  }
}
```

IV. Circuit Photo

Figure 3 is picture of the actual circuit.

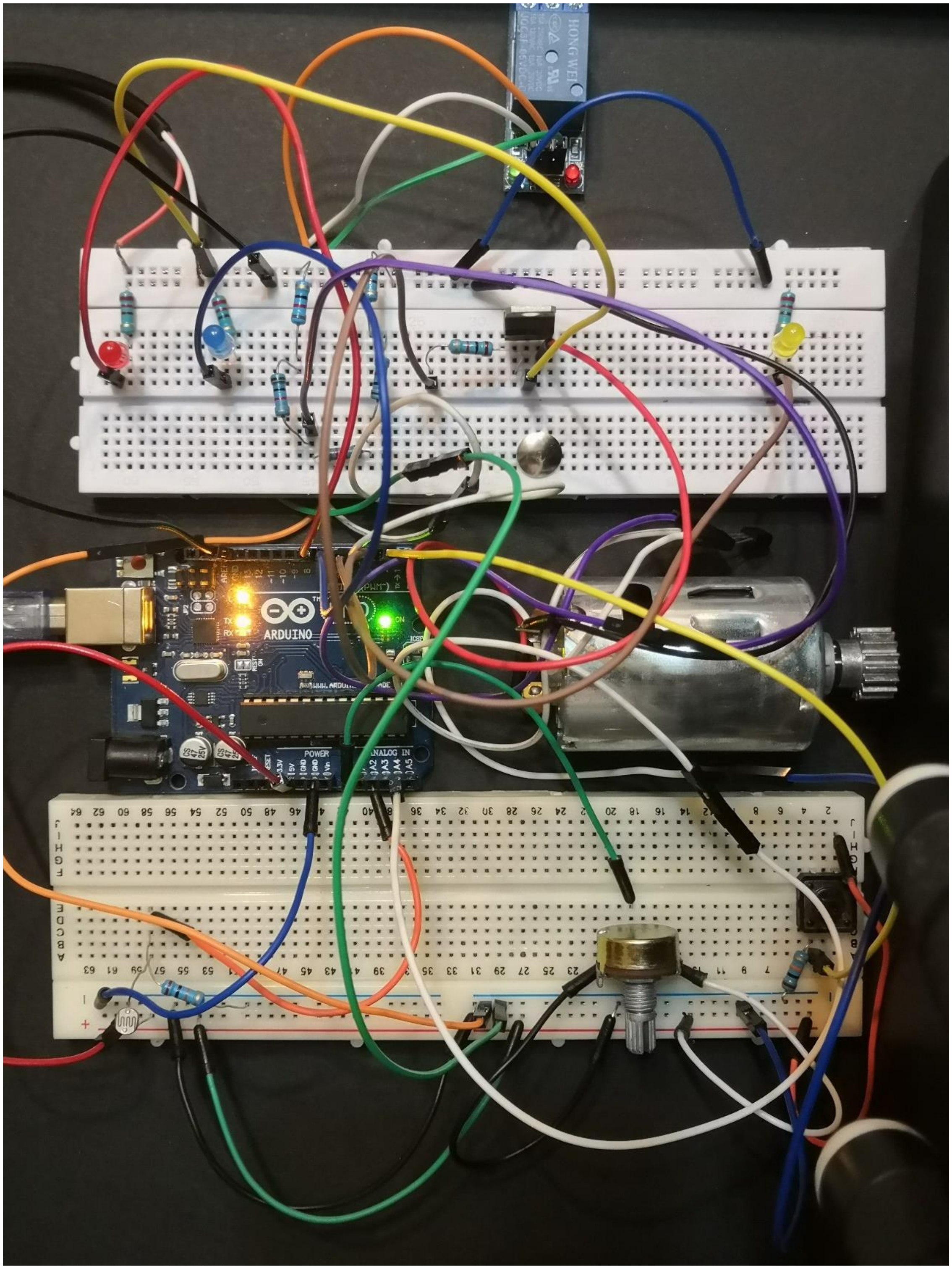


Figure 3 Photograph of The Circuit

V. Circuit Schematic Representation

Using Tinkercad, a schematic diagram (**Figure 4**) of the circuit is drawn. Its is worthy to note that the TMP36 temperature sensor is acts as a placeholder for the LM35DZ.

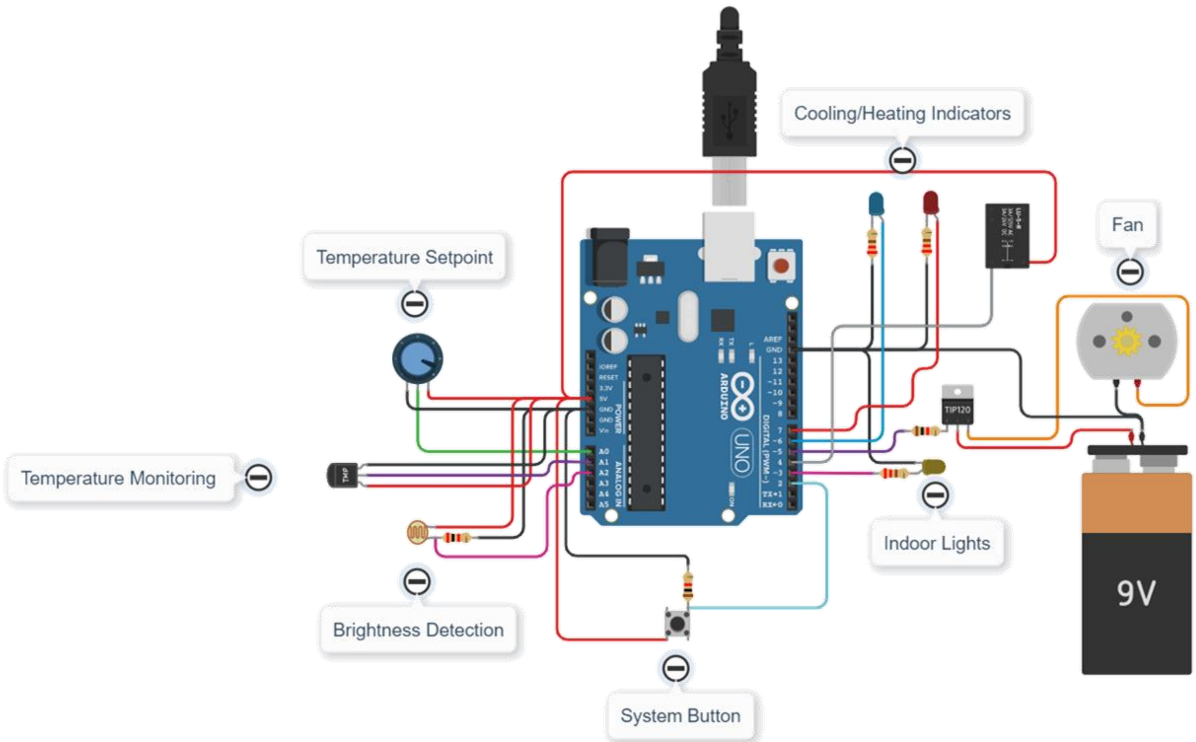


Figure 4 Shematic Diagram of the Circuit

VI. Appendix

Product Folder | Order Now | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

LM35

SNIS159H – AUGUST 1999 – REVISED DECEMBER 2017

LM35 Precision Centigrade Temperature Sensors

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

3 Description

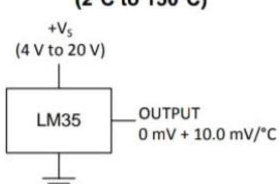
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

Device Information(1)

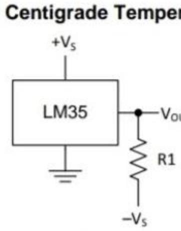
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM35	TO-CAN (3)	4.699 mm × 4.699 mm
	TO-92 (3)	4.30 mm × 4.30 mm
	SOIC (8)	4.90 mm × 3.91 mm
	TO-220 (3)	14.986 mm × 10.16 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Basic Centigrade Temperature Sensor (2°C to 150°C)



Full-Range Centigrade Temperature Sensor



Choose $R_1 = -V_S / 50\ \mu A$
V_{OUT} = 1500 mV at 150°C
V_{OUT} = 250 mV at 25°C
V_{OUT} = −550 mV at −55°C

An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)(2)

		MIN	MAX	UNIT
Supply voltage		–0.2	35	V
Output voltage		–1	6	V
Output current			10	mA
Maximum Junction Temperature, $T_J$max			150	°C
Storage Temperature, T_{stg}	TO-CAN, TO-92 Package	–60	150	°C
	TO-220, SOIC Package	–65	150	

(1) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2500	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
Specified operating temperature: T_{MIN} to T_{MAX}	LM35, LM35A	–55	150	°C
	LM35C, LM35CA	–40	110	
	LM35D	0	100	
Supply Voltage (+V_S)		4	30	V

6.4 Thermal Information

THERMAL METRIC(1)(2)		LM35				UNIT
		NDV	LP	D	NEB	
		3 PINS		8 PINS	3 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	400	180	220	90	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	24	—	—	—	

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) For additional thermal resistance information, see *Typical Application*.

6.7 Electrical Characteristics: LM35, LM35C, LM35D Limits

Unless otherwise noted, these specifications apply: $-55°C \le T_J \le 150°C$ for the LM35 and LM35A; $-40°C \le T_J \le 110°C$ for the LM35C and LM35CA; and $0°C \le T_J \le 100°C$ for the LM35D. $V_S = 5$ Vdc and $I_{LOAD} = 50$ µA, in the circuit of Full-Range Centigrade Temperature Sensor. These specifications also apply from 2°C to T_{MAX} in the circuit of Figure 14.

PARAMETER	TEST CONDITIONS	LM35			LM35C, LM35D			UNIT
		TYP	TESTED LIMIT(1)	DESIGN LIMIT(2)	TYP	TESTED LIMIT(1)	DESIGN LIMIT(2)	
Accuracy, LM35, LM35C(3)	$T_A = 25°C$	±0.4	±1		±0.4	±1		°C
	$T_A = -10°C$	±0.5			±0.5		±1.5	
	$T_A = T_{MAX}$	±0.8	±1.5		±0.8		±1.5	
	$T_A = T_{MIN}$	±0.8		±1.5	±0.8		±2	
Accuracy, LM35D(3)	$T_A = 25°C$				±0.6	±1.5		°C
	$T_A = T_{MAX}$				±0.9		±2	
	$T_A = T_{MIN}$				±0.9		±2	
Nonlinearity(4)	$T_{MIN} \le T_A \le T_{MAX}$, $-40°C \le T_J \le 125°C$	±0.3		±0.5	±0.2		±0.5	°C
Sensor gain (average slope)	$T_{MIN} \le T_A \le T_{MAX}$, $-40°C \le T_J \le 125°C$	10	9.8		10		9.8	mV/°C
		10	10.2		10		10.2	
Load regulation(5) $0 \le I_L \le 1$ mA	$T_A = 25°C$	±0.4	±2		±0.4	±2		mV/mA
	$T_{MIN} \le T_A \le T_{MAX}$, $-40°C \le T_J \le 125°C$	±0.5		±5	±0.5		±5	
Line regulation(5)	$T_A = 25°C$	±0.01	±0.1		±0.01	±0.1		mV/V
	$4 V \le V_S \le 30 V$, $-40°C \le T_J \le 125°C$	±0.02		±0.2	±0.02		±0.2	
Quiescent current(6)	$V_S = 5$ V, 25°C	56	80		56	80		µA
	$V_S = 5$ V, $-40°C \le T_J \le 125°C$	105		158	91		138	
	$V_S = 30$ V, 25°C	56.2	82		56.2	82		
	$V_S = 30$ V, $-40°C \le T_J \le 125°C$	105.5		161	91.5		141	
Change of quiescent current(5)	$4 V \le V_S \le 30 V$, 25°C	0.2	2		0.2	2		µA
	$4 V \le V_S \le 30 V$, $-40°C \le T_J \le 125°C$	0.5		3	0.5		**3**	
Temperature coefficient of quiescent current	$-40°C \le T_J \le 125°C$	0.39		0.7	0.39		0.7	µA/°C
Minimum temperature for rate accuracy	In circuit of Figure 14, $I_L = 0$	1.5		2	1.5		2	°C
Long term stability	$T_J = T_{MAX}$, for 1000 hours	±0.08			±0.08			°C

(1) Tested Limits are ensured and 100% tested in production.
(2) Design Limits are ensured (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.
(3) Accuracy is defined as the error between the output voltage and 10 mv/°C times the case temperature of the device, at specified conditions of voltage, current, and temperature (expressed in °C).
(4) Non-linearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the rated temperature range of the device.
(5) Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.
(6) Quiescent current is defined in the circuit of Figure 14.

References

[1] The Editors of Encyclopaedia Britannica, "Mechanical System," Encyclopædia Britannica, 3 August 2011. [Online]. Available: https://www.britannica.com/technology/mechanical-system#ref1035138. [Accessed 10 May 2020].

[2] 21Celsius, "Everything You Need To Know About HVAC Systems," 21Celsius, [Online]. Available: http://twentyonecelsius.com.au/blog/everything-you-need-to-know-about-hvac-systems/. [Accessed 10 May 2020].

[3] G. Hundy, A. Trott and T. Welch, "Air Conditioning Methods and Applications," in *Refrigeration, Air Conditioning and Heat Pumps, Fifth Edition*, Butterworth-Heinemann, 2016, p. 510.