FINAL REPORT

TEMPERATURE INDICATOR CIRCUIT FOR BETTER COMFORT AND ENERGY EFFICIENT AIR CONDITIONING SYSTEMS WITH SAFETY FUNCTION

EC6020: EMBEDDED SYSTEMS DESIGN – PROJECT

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

SENEVIRATHNA S.J.

2018/E/110

INTRODUCTION:

According to the International Energy Agency (IEA), electricity consumption for air conditioning will be the main trigger for the increase in world electricity demand in the next 2050. Air conditioning becomes inefficient and uncomfortable Due to the inability to set the most efficient and best comfortable temperature. Air conditioners consume more power when the outside temperature is high. Air Conditioners are depending on the ambient air to remove heat. When the outside temperature is high, the air conditioner can't remove heat effectively. Thus, the air conditioner may struggle to provide sufficient and energy efficient cooling. If the outside temperature is below the air conditioning limit, we may experience the following consequences. The unit's inner coils will freeze. The lubricating fluid will thicken, and the unit will not function properly and may ultimately damage the air conditioner.

An air conditioner not only consumes more power during hot days, but it also may not provide sufficient and efficient cooling for a house. We all know that when the outside temperature is high, the air conditioner needs to work harder to cool the room. Therefore, it consumes more power. But it is more than that.

In order to measure both outside and inside temperature and suggest a best cooling temperature of air conditioning system, an embedded system is created using the ATmega-328 Microcontroller. By an LCD display suggested temperature is shown for the customer. If the outside temperature goes down beyond the inside temperature the relay is turned on and turn off the AC system. This will protect the whole AC system. The system is inexpensive and simple to implement in residential areas for comfortable and energy efficient Air Conditioning System with safety function.

REQUIREMENT ANALYSIS:

1. **FUNCTIONAL REQUIREMENTS**
    - Temperature sensor should detect the temperature in both indoor and outdoor.
    - LCD display should always show the indoor temperature, outdoor temperature and best temperature or A/C is turned off.
    - Relay should be turned on the A/C or Ventilator.

2. **NON-FUNCTIONAL REQUIREMENT**
    - Low-Power Consumption.
    - Smaller in Size
    - Reliable product in day-to-day environment

PROJECT DESIGN AND IMPLEMENTATION:

The Temperature Indicator System is a low-cost and reliable system which is powered by USB 5V DC supply.

Working Principle:

- LM-35 Sensor reading is transmitted to microcontroller.
- Compare the temperature difference and suggest a best temperature value.
- If Indoor > Outdoor temperature AC is turned off and Ventilator will be turned on.

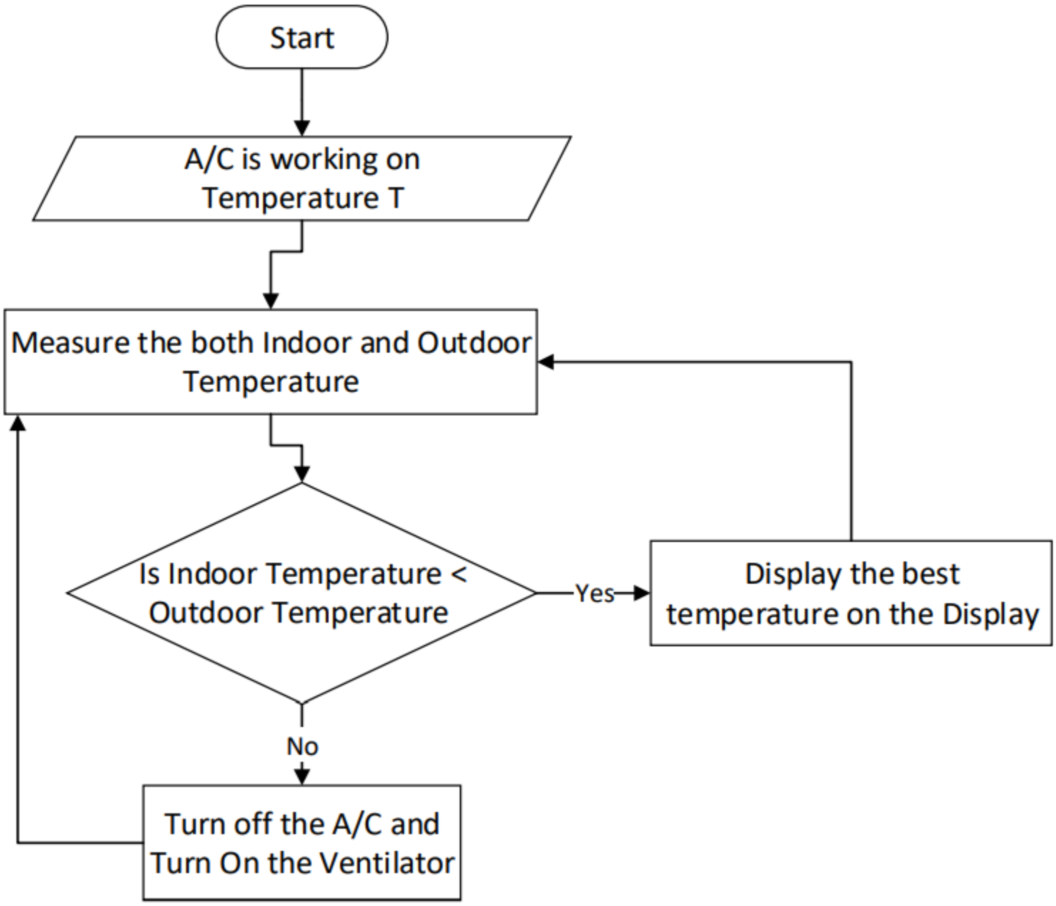

Figure 1: Functional Flow chart of the System

System Devices:

a. LM-35 Temperature sensor.

LM35 is a temperature sensor that outputs an antilog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius. The advantage of LM35 over thermistor is it does not require any external calibration.

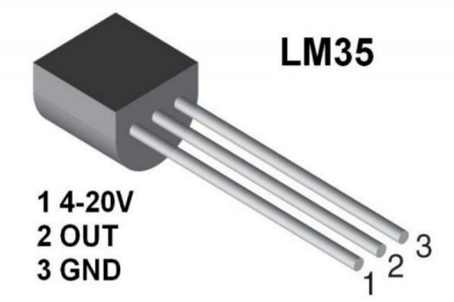

Figure 2 : LM-35 Temperature sensor

b. LCD 16x2 (Liquid Crystal Display)

The most basic and commonly used LCDS are the 16×2 because they are cheap, easy to program and can display wide range of characters. This is a 16-pin device which displays 16 × 2 characters.

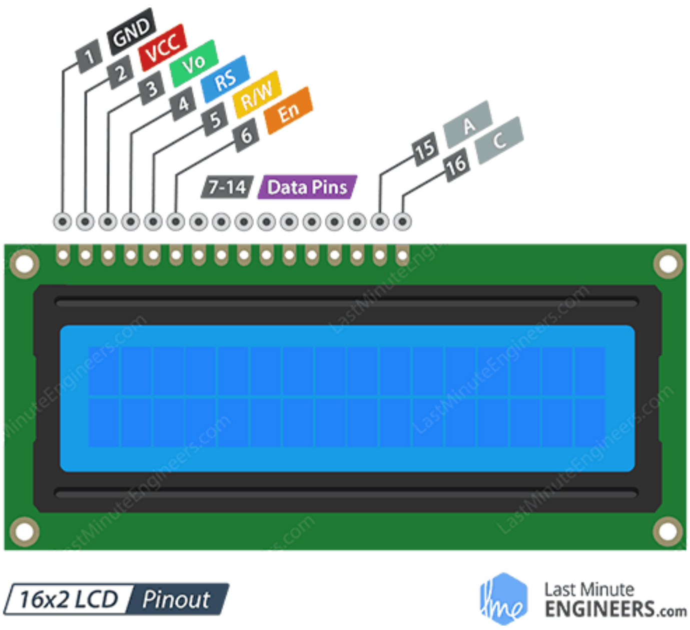

Figure 3 : 16 ×2 Liquid Crystal Display

c. ATmega328

The ATmega328 is a single-chip microcontroller created by Atmel in the mega AVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core.

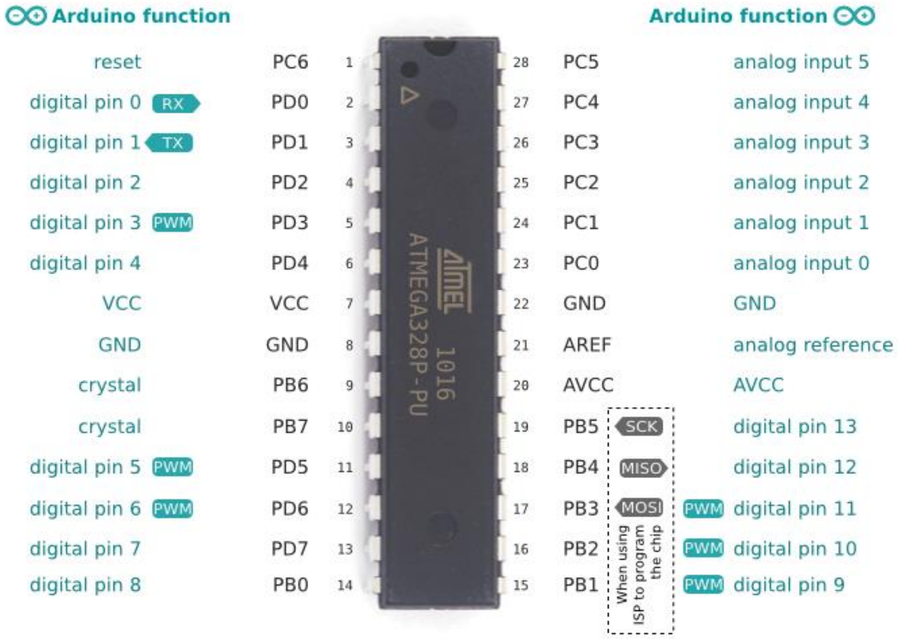

Figure 4 : ATmega328 Controller

d. 5V Relay

5V relay is an automatic switch that is commonly used in an automatic control circuit and to control a high-current using a low-current signal. The input voltage of the relay signal ranges from 0 to 5V.

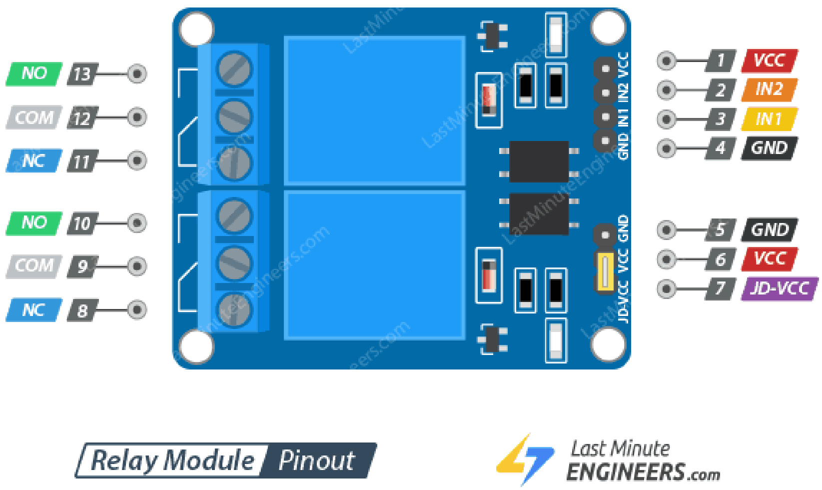

Figure 5: 5V Relay Module

METHOD:

The measured temperature is identified by the microcontroller and temperature readings are sent to the 16x2 LCD display.

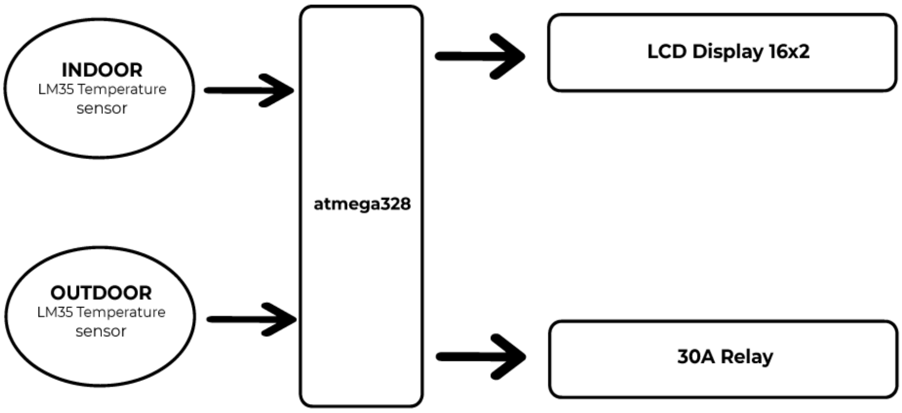

Figure 6 : Block diagram of temperature indication and control system

If;

Outside temperature > Inside temperature - LCD display will show the suggested temperature to the consumer for setting the Air conditioner temperature.

Outside temperature < Inside temperature – Relay will be activated and turn off the Air conditioner for protecting the Air conditioning system components and Ventilator will be turned on

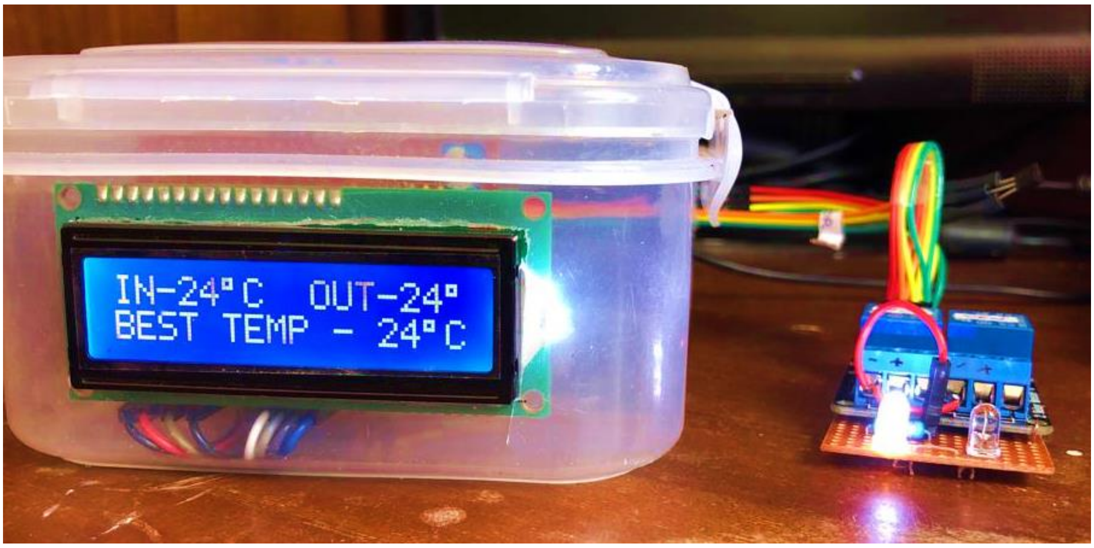

Figure 7: Suggesting a Best Temperature value for the Customer

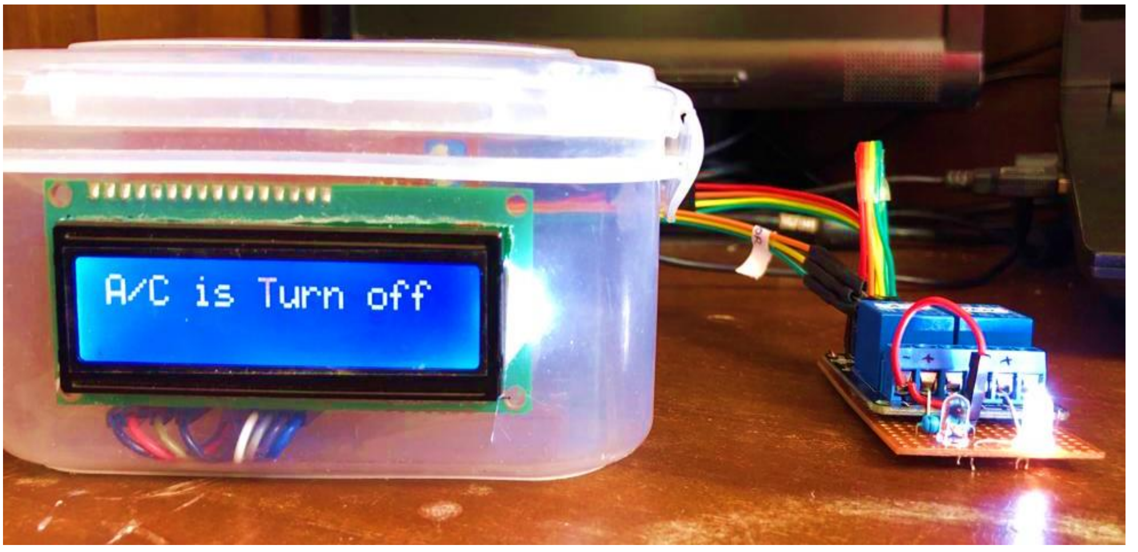

Figure 8 : A/C is Turned-off and Ventilator turned on

CHALLENGES:

Here we mentioned the challenges and problems we faced during the implementing the system.

Sensor selection: We have chosen LM-35 over DHT-22 sensor for this system, because DHT-22 sensor gives the both humidity and Temperature values one by one. So, separating the temperature value is quite difficult. But from LM-35 sensor we can obtain the temperature value easily.

Simulation Issues: For simulation purposes proteus software was used. Some libraries for LCD Display and atmega328p were not included in the proteus.

Purchasing Components: Obtaining the components for the project was difficult because of the economic crisis in the country. As well as, price of the components rise time to time.

Initializing LCD: Initializing LCD display is somewhat difficult using C embedded language without any libraries.

Soldering works: We have done soldering works using Dot boards and Stripboards. So, Soldering the Circuit board is challengeable with available equipment.

Initializing Atmel Studio: Initializing the Atmel studio was somewhat difficult. Because, initially Atmel studio hasn't a tool for uploading via Arduino uno board.

BUDGET:

Table 1 : The estimated budget of the project

Expenses	**Unit**	**# of Units**	**Unit rate (USD)**	**Cost (USD)**
Temperature Sensor (LM35)	per item	2	0.41	0.82
atmega328p	per item	1	1	1
Relay Module	per item	1	1.73	1.73
Jump wire	per item	30	0.1	3
16x2 LCD display	per item	1	2.75	2.75
Stripboard	per item	2	0.25	0.5
Resistors and Capacitor	per item	6	0.1	0.6
Rotary Potentiometer	per item	1	0.8	0.8
Cristal Oscillator	per item	1	0.1	0.1
USB Cable	per item	1	1	1
Total Expenses				**12.3**

Note - All the values are calculated using USD.

- Total expenses in USD - $12.30
- Total expenses in SLR – RS. 4450.00

TIMELINE:

Table 2 : The Timeline of the Project

ALLOCATED WORK	**WEEK**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Topic Selection, Initial Discussions								
Feasibility Study, Literature Review								
Proposal creation, Proposal submission, Implementing								
Software Implementation, Mid Presentation								
Implement circuit using Stripboard. Package design.								
Final Evaluation, presentation & Demonstration.								
Final Report								

SIGNIFICANT OF THE PROJECT:

This project is led to make more energy efficient and more comfortable temperature for living. There are many advantages in this system and we can use this system in different residential, commercial, and industrial applications.

Advantages:

- ✓ Cost effective.
- ✓ Energy Efficient and convenient
- ✓ User friendly

Application:

- ✓ HVAC almost consumed half of the energy in buildings and 20% of the overall national energy consumption. Therefore, this small system will help to decrease the energy consumption of ACs in residential and commercial buildings
- ✓ This system is very useful in greenhouse to ensure the necessary temperature inside the greenhouse for better crops.

REFLECTION ON APPLIED KNOWLEDGE FROM THINGS LEARNED IN THE COURSE:

The basic knowledge used for the design and implementation of the device;

- Using ATMega328P microcontroller for making the prototype.
- How to Interfacing the LCD 16x2 Display.
- Characteristics of embedded systems like real-time operation, low manufacturing cost, lower power consumption, etc. were considered before manufacturing the prototype.
- For designing the prototype, bottom-up design approach was selected. That means we have worked from the small components to big system.
- Using the datasheet of the microcontroller for identifying the pin out for ATMEGA328p.
- Design methodology

CONCLUSION:

Air-Conditioning (HVAC) system are to help maintain indoor temperature to provide thermal comfort. Mostly the consumer does not care about the outdoor temperature for controlling the air conditioning temperature. That will be led to more energy consumption, bad thermal comfort and safety of the air conditioning systems. From this small, cost effective circuit we hope to minimize above shortcomings.

For the future implementations we can develop the circuit which works automatically and remote controllable for maximize the user’s easiness.

APPENDIX:

01. C Embedded Code.

```
 #define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define degree_sysmbol 0xdf
//LCD PORT Define
#define lcd_D7_port     PORTD                   // LCD D7 connection
#define lcd_D7_bit      PORTD7
#define lcd_D7_ddr      DDRD

#define lcd_D6_port     PORTD                   // LCD D6 connection
#define lcd_D6_bit      PORTD6
#define lcd_D6_ddr      DDRD

#define lcd_D5_port     PORTD                   // LCD D5 connection
#define lcd_D5_bit      PORTD5
#define lcd_D5_ddr      DDRD

#define lcd_D4_port     PORTD                   // LCD D4 connection
#define lcd_D4_bit      PORTD4
#define lcd_D4_ddr      DDRD

#define lcd_E_port      PORTD                   // LCD Enable pin
#define lcd_E_bit       PORTD3
#define lcd_E_ddr       DDRD

#define lcd_RS_port     PORTD                   // LCD Register Select pin
#define lcd_RS_bit      PORTD2
#define lcd_RS_ddr      DDRD


// LCD module information
#define lcd_LineOne     0x00                    // start of line 1  0000 0000
#define lcd_LineTwo     0x40                    // start of line 2  0100 0000

// LCD instructions
#define lcd_Clear           0b00000001          // replace all characters with ASCII 'space'
#define lcd_Home            0b00000010          // return cursor to first position on first line
#define lcd_EntryMode       0b00000110          // shift cursor from left to right on read/write
#define lcd_DisplayOff      0b00001000          // turn display off
#define lcd_DisplayOn       0b00001100          // display on, cursor off, don't blink character
#define lcd_FunctionReset   0b00110000          // reset the LCD
#define lcd_FunctionSet4bit 0b00101000          // 4-bit data, 2-line display, 5 x 7 font
#define lcd_SetCursor       0b10000000          // set cursor position

// Function Prototypes
void lcd_write_4(uint8_t);
void lcd_write_instruction_4d(uint8_t);
void lcd_write_character_4d(uint8_t);
void lcd_write_string_4d(uint8_t *);
void lcd_init_4d(void);
void ADC_Init();
int ADC_Read(char channel);
void ADC_Init2();
int ADC_Read2(char channel);

/******************************* Main Program Code ************************/
int main(void)
{
    DDRB|=(1<<0); //AC port
    DDRB|=(1<<1); //venti port
    PORTB|=(1<<1); // venti high

// configure the microprocessor pins for the data lines
    lcd_D7_ddr |= (1<<lcd_D7_bit);                  // 4 data lines - output
    lcd_D6_ddr |= (1<<lcd_D6_bit);
    lcd_D5_ddr |= (1<<lcd_D5_bit);
    lcd_D4_ddr |= (1<<lcd_D4_bit);

// configure the microprocessor pins for the control lines
```

```c
    lcd_E_ddr |= (1<<lcd_E_bit);                    // E line - output
    lcd_RS_ddr |= (1<<lcd_RS_bit);                  // RS line - output

// initialize the LCD controller as determined by the defines (LCD instructions)
    lcd_init_4d();                                  // initialize the LCD display for a 4-bit interface

    char Temperature[10];
    char Temperature2[10];
    char DIF[10];
    char BEST[10];
    float celsius;
    float celsius2;

    ADC_Init();                 /* initialize ADC*/
    ADC_Init2();                 /* initialize ADC2*


    while(1){
        celsius = (ADC_Read(0)*4.03);
        celsius = (celsius/10.00);
        sprintf(Temperature,"%d%cC  ",(int)celsius, degree_sysmbol);/* convert integer to ASCII string*/

        celsius2 = (ADC_Read2(1)*4.03);
        celsius2 = (celsius2/10.00);
        sprintf(Temperature2,"%d%cC  ", (int)celsius2, degree_sysmbol);/* convert integer to ASCII string */

        lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);    // set cursor to start of first line
        _delay_us(80);

        lcd_write_string_4d("IN-");   // display the first line of information
        lcd_write_string_4d(Temperature);
        _delay_ms(2000);

        lcd_write_string_4d("OUT-");
        lcd_write_string_4d(Temperature2);
        _delay_ms(2000);


        if (celsius2+1> celsius)
        {

                lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);  // set cursor to start of second line
                _delay_us(80);

                float diff = celsius - celsius2;   // Getting the Difference between 2 temperature
                sprintf(DIF,"%d%cC  ", (int)diff, degree_sysmbol);/* convert integer value to ASCII string */
                _delay_us(80);

                float best_t = celsius2 + diff/2;  //Best temperature Calculation
                sprintf(BEST,"%d%cC  ", (int)best_t, degree_sysmbol);/* convert integer to ASCII string */

                lcd_write_string_4d("BEST TEMP - ");  //best temp suggesion
                lcd_write_string_4d(BEST);
                _delay_ms(1000);

                PORTB&=~(1<<0);  // ac low
                PORTB|=(1<<1); //venti high

        }

        else {

            do
            {
                lcd_write_instruction_4d(lcd_Clear);             // clear display RAM
                _delay_ms(4);                                   // 1.64 mS delay (min)

                lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);  // set cursor to start of first line
                _delay_us(80);

                PORTB&=~(1<<1);  // venti low
                PORTB|=(1<<0); // ac high

                lcd_write_string_4d("A/C is Turn off");
                _delay_ms(10000);

                lcd_write_instruction_4d(lcd_Clear);             // clear display RAM
                _delay_ms(40);                                   // 1.64 mS delay (min)

            } while (celsius2 > celsius);   // indoor <outdoor
```

```c
        }
        memset(Temperature2,0,10);
        memset(Temperature,0,10);
    }

// endless loop
    while(2);
    return 0;
}
/****************************** End of Main Program Code *****************/

// ADC conversation

void ADC_Init(){
    DDRC = 0x00;            /* Make ADC port as input */
    ADCSRA = 0x87;          /* Enable ADC, with freq/128  */
    ADMUX = 0x40;           /* Vref: Avcc, ADC channel: 0 */
}

int ADC_Read(char channel)
{
    ADMUX = 0x40 | (channel & 0x07);   /* set input channel to read */
    ADCSRA |= (1<<ADSC);               /* Start ADC conversion */
    while (!(ADCSRA & (1<<ADIF)));     /* Wait until end of conversion by polling ADC interrupt flag */
    ADCSRA |= (1<<ADIF);               /* Clear interrupt flag */
    _delay_ms(1);                      /* Wait a little bit */
    return ADCW;                       /* Return ADC word */
}

void ADC_Init2(){
    DDRC = 0x00;            /* Make ADC port as input */
    ADCSRA = 0x87;          /* Enable ADC, with freq/128  */
    ADMUX = 0x41;           /* Vref: Avcc, ADC channel: 0 */
}

int ADC_Read2(char channel)
{
    ADMUX = 0x41 | (channel & 0x07);   /* set input channel to read */
    ADCSRA |= (1<<ADSC);               /* Start ADC conversion */
    while (!(ADCSRA & (1<<ADIF)));     /* Wait until end of conversion by polling ADC interrupt flag */
    ADCSRA |= (1<<ADIF);               /* Clear interrupt flag */
    _delay_ms(1);                      /* Wait a little bit */
    return ADCW;                       /* Return ADC word */
}


/*============================== 4-bit LCD Functions ======================*/

void lcd_init_4d(void)
{
// Power-up delay
    _delay_ms(100);                                 // initial 40 mSec delay

// Set up the RS and E lines for the 'lcd_write_4' subroutine.
    lcd_RS_port &= ~(1<<lcd_RS_bit);                // select the Instruction Register (RS low)
    lcd_E_port &= ~(1<<lcd_E_bit);                  // make sure E is initially low

// Reset the LCD controller
    lcd_write_4(lcd_FunctionReset);                 // first part of reset sequence
    _delay_ms(10);                                  // 4.1 mS delay (min)

    lcd_write_4(lcd_FunctionReset);                 // second part of reset sequence
    _delay_us(200);                                 // 100uS delay (min)

    lcd_write_4(lcd_FunctionReset);                 // third part of reset sequence
    _delay_us(200);                                 // this delay is omitted in the data sheet

    lcd_write_4(lcd_FunctionSet4bit);               // set 4-bit mode
    _delay_us(80);                                  // 40uS delay (min)

// Function Set instruction
    lcd_write_instruction_4d(lcd_FunctionSet4bit);   // set mode, lines, and font
    _delay_us(80);                                  // 40uS delay (min)

// Display On/Off Control instruction
    lcd_write_instruction_4d(lcd_DisplayOff);        // turn display OFF
    _delay_us(80);                                  // 40uS delay (min)

// Clear Display instruction
```

```
    lcd_write_instruction_4d(lcd_Clear);              // clear display RAM
    _delay_ms(4);                                    // 1.64 mS delay (min)

// ; Entry Mode Set instruction
    lcd_write_instruction_4d(lcd_EntryMode);          // set desired shift characteristics
    _delay_us(80);                                   // 40uS delay (min)

// Display On/Off Control instruction
    lcd_write_instruction_4d(lcd_DisplayOn);          // turn the display ON
    _delay_us(80);                                   // 40uS delay (min)
}

void lcd_write_string_4d(uint8_t theString[])
{
    volatile int i = 0;                              // character counter*/
    while (theString[i] != 0)
    {
        lcd_write_character_4d(theString[i]);
        i++;
        _delay_us(80);                               // 40 uS delay (min)
    }
}

void lcd_write_character_4d(uint8_t theData)
{
    lcd_RS_port |= (1<<lcd_RS_bit);                  // select the Data Register (RS high)
    lcd_E_port &= ~(1<<lcd_E_bit);                   // make sure E is initially low
    lcd_write_4(theData);                            // write the upper 4-bits of the data
    lcd_write_4(theData << 4);                       // write the lower 4-bits of the data
}

void lcd_write_instruction_4d(uint8_t theInstruction)
{
    lcd_RS_port &= ~(1<<lcd_RS_bit);                 // select the Instruction Register (RS low)
    lcd_E_port &= ~(1<<lcd_E_bit);                   // make sure E is initially low
    lcd_write_4(theInstruction);                     // write the upper 4-bits of the data
    lcd_write_4(theInstruction << 4);                // write the lower 4-bits of the data
}

void lcd_write_4(uint8_t theByte)
{
    lcd_D7_port &= ~(1<<lcd_D7_bit);                         // assume that data is '0'
    if (theByte & 1<<7) lcd_D7_port |= (1<<lcd_D7_bit);      // make data = '1' if necessary

    lcd_D6_port &= ~(1<<lcd_D6_bit);                         // repeat for each data bit
    if (theByte & 1<<6) lcd_D6_port |= (1<<lcd_D6_bit);

    lcd_D5_port &= ~(1<<lcd_D5_bit);
    if (theByte & 1<<5) lcd_D5_port |= (1<<lcd_D5_bit);

    lcd_D4_port &= ~(1<<lcd_D4_bit);
    if (theByte & 1<<4) lcd_D4_port |= (1<<lcd_D4_bit);

// write the data
                                         // 'Address set-up time' (40 nS)
    lcd_E_port |= (1<<lcd_E_bit);        // Enable pin high
    _delay_us(1);                    // implement 'Data set-up time' (80 nS) and 'Enable pulse width' (230 nS)
    lcd_E_port &= ~(1<<lcd_E_bit);     // Enable pin low
    _delay_us(1);                      // implement 'Data hold time' (10 nS) an'Enable cycle time' (500 nS)
}
```

02. Captures during implementation

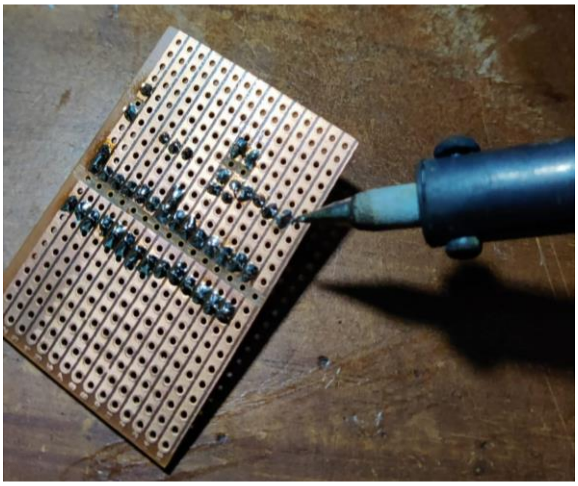

Figure 9 : Soldering the Stripboard

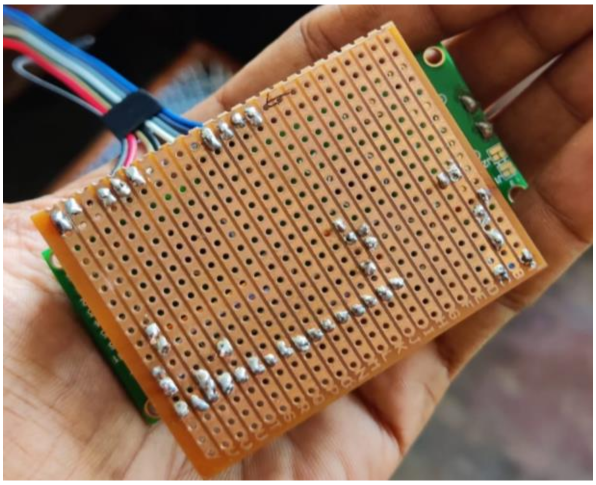

Figure 10 : After Soldering

Figure 11: After Attaching the LCD Display to the board

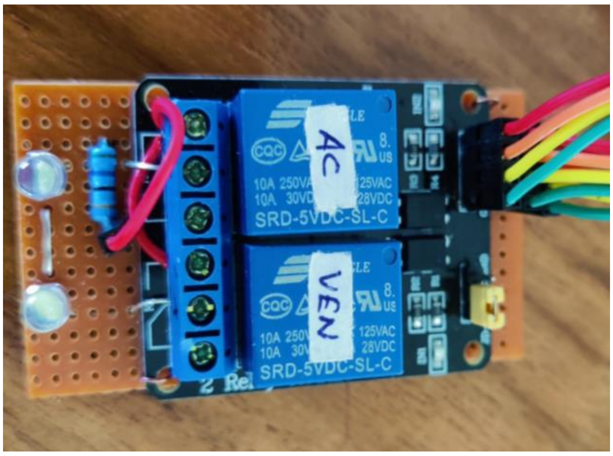

Figure 12: Attach relay module to the Circuit Board

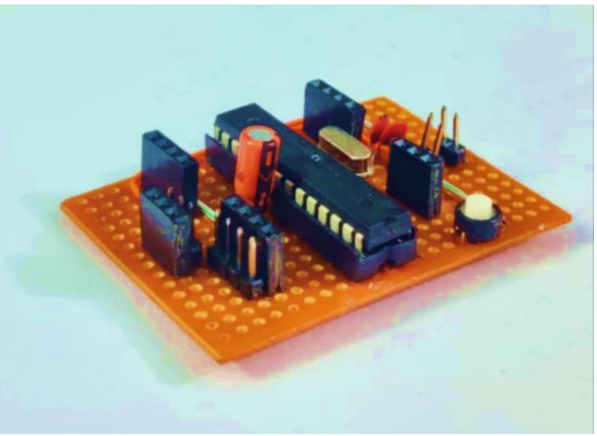

Figure 13: After Design the Main Circuit Board

REFERENCES:

[1] X. Xu, Z. Zhong, S. Deng, and X. Zhang, "A review on temperature and humidity control methods focusing on air-conditioning equipment and control algorithms applied in small-to-medium-sized buildings," *Energy and Buildings*, vol. 162. Elsevier Ltd, pp. 163–176, Mar. 01, 2018, doi: 10.1016/j.enbuild.2017.12.038.

[2] J. Wang, Q. Zhang, and Y. Yu, "An advanced control of hybrid cooling technology for telecommunication base stations," *Energy Build.*, vol. 133, pp. 172–184, Dec. 2016, doi: 10.1016/j.enbuild.2016.08.090.

[3] M. R. Levine, Automatic temperature adjusting system for air conditionerroom. China Patent CN103335385 A, 6 June 2013.

[4] Circuit design app for makers- circuito.io. Circuit Design App for Makers- circuito.io. (n.d.). Retrieved June 25, 2022, from https://www.circuito.io/

[5] M. A. Afandi, S. Nurandi, and I. K. A. Enriko, "Automated Air Conditioner Controler and Monitoring Based on Internet of Things," *IJEIS (Indonesian J. Electron. Instrum. Syst.*, vol. 11, no. 1, p. 83, 2021, doi: 10.22146/ijeis.64563.

[6] Tech, P. B. W. (2012, June 27). The Arduino weather station / thermostat using ATMEGA328 Microcontroller. ATMega32 AVR. Retrieved June 25, 2022, from https://atmega32-avr.com/the-arduino-weather-station-thermostat-using-atmega328-microcontroller/

[7] Zhen, Y. C. (2021, August 17). Does outside temperature affect air conditioner? aircondlounge. Retrieved June 25, 2022, from https://aircondlounge.com/does-outside-temperature-affect-air-conditioner

[8] Http://Www.Avr-Asm-Tutorial.Net/Avr_en/Index.Html. Accessed 17 Aug. 2022.

[9] "ADC in AVR ATmega16/ATmega32 | AVR ATmega Controllers." ADC in AVR ATmega16/ATmega32 | AVR ATmega Controllers, www.electronicwings.com, https://www.electronicwings.com/avr-atmega/atmega1632-adc. Accessed 17 Aug. 2022.

[10] "Fritzing." Fritzing, fritzing.org, https://fritzing.org/. Accessed 17 Aug. 2022.

[11] Lecture notes and Laboratories