HVAC EMBEDDED TWIN FAN CONTROLLER (REDUNDANT BACK-UP SYSTEM)

COMPHREHENSIVE PROJECT REPORT

Submitted in partial fulfilment of the requirement of

BACHELOR OF TECHNOLOGY

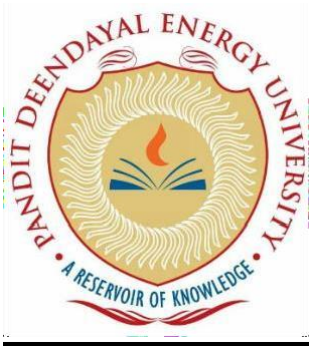
by

DHYEY SHAH
DEPARTMENT: ELECTRICAL ENGINEERING

Roll no. 18BEE036

Under the guidance of

MR. NIRAV KARELIA



SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

May 2022

CHAPTER 7: APPLICATION LAYER DEVELOPMENT

7.1 MAIN CODE

Below is the main code section for the developed Application Layer using PIC 16F1939:

Main File Code / Application Layer (Mode Operation – main.c):

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/Delay.h"
#include "mcc_generated_files/io.h"
#include "mcc_generated_files/Global.h"
#include "mcc_generated_files/digit4segment.h"
#include "mcc_generated_files/EEPROM.h"
#include "mcc_generated_files/Functions.h"

APP_DATA            appData;
TIMER_DATA          timer;
SYSTEM_DATA         system;

/*
                    Main application
 */

void HardwareControls(void);

void main(void)
{
    uint8_t index,ID = false;
    uint16_t LOCALTMR_CNT    = 4;
    uint8_t LOCALSECTMR      = 14;

    uint8_t TestModeStat     = 0;

    // initialize the device
    SYSTEM_Initialize();

    // When using interrupts, you need to set the Global and Peripheral Interrupt Enable bits
    // Use the following macros to:

    // Enable the Global Interrupts
    INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    INTERRUPT_PeripheralInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();
```

```
appData.state = APP_STATE_INIT;



while (1)
{
   // Add your application code
   switch ( appData.state )
   {
      /* Application's initial state. */
      case APP_STATE_INIT:
      {
         LED_AUTO_ON();
         LED_FAN1_ON();
         LED_FAN2_ON();
         LED_AIRSW_ON();
         InitSegments();
         LED_AUTO_OFF();
         LED_FAN1_OFF();
         LED_FAN2_OFF();
         LED_AIRSW_OFF();
         UpdateDisplay(Display_Blank);
         ID = GetData(EEP_SYSDEFAULT_ADD);
         if(ID != SYS_DEFAULT_VAL){
            //printf("\r\nSet System Defaults");
            SetDefault();
         }
         ReadConfig();
         system.setupmode = DispCurrent;
         UpdateDisplay(200);
         DelayMS(2000);
         system.setupmode = false;
         UpdateDisplay(Display_Blank);
         ID = true;
         timer.secTMR1 = 0;
         ADC_SelectChannel(ADCCURRENTCH()); ADC_StartConversion();
         LoadSecTMR1(3);UpdateDisplay(timer.secTMR1); DelayMS(1000);
         while(timer.secTMR1){UpdateDisplay(timer.secTMR1);}
         ID = false;

         if(READ_EMRGSWPIN())
            appData.state = APP_STATE_EMRGSTOP;
         else{
            timer.secTMR3 = 0;
            system.startFunc = true;  appData.state = APP_STATE_RUN;
         }
         break;
      }
      case APP_STATE_RUN:
      {
         if(timer.secTMR3 != LOCALSECTMR){
            if(system.automode == true){
               system.hrrunTMR  = (((system.hrTMR*60)-timer.minTMR)/60);
               system.minrunTMR = (((system.hrTMR*60)-timer.minTMR)%60);

               if(system.flash) system.flash = false; else system.flash = true;
               UpdateDisplay(Display_DECTIMER_MODE);
            }

            LOCALSECTMR = timer.secTMR3;
```

```
            }
            UpdateLoadCurrent();
            if((timer.minTMR != LOCALTMR_CNT)&&(system.automode==true)){
               if(AutoChangeoverLoop()) appData.state = APP_STATE_INIT;
               LOCALTMR_CNT = timer.minTMR;
            }

            UpdateLoadCurrent();
            if((system.runningfan == FAN1)&&(system.fan1Fault == false)){



               if(system.functionmode == AIRMODE){
                  if(READ_AIRSW1PIN()){
                     if(timer.airswTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();
                        system.fan1Fault = true;
                        UpdateStat(EEP_FAN1FAULT_ADD,system.fan1Fault);
                        appData.state = APP_STATE_INIT;
                     }
                  } else timer.airswTMR = AIRSW_RUNTMR;

                  if((system.loadcurrent > (system.cutoffF1Hi*100))){
                     if(timer.currentTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();
                        system.fan1Fault = true;
                        UpdateStat(EEP_FAN1FAULT_ADD,system.fan1Fault);
                        appData.state = APP_STATE_INIT;
                     }
                  } else if(timer.currentTMR < CURRENTSENSE_RUNTMR) timer.currentTMR =
CURRENTSENSE_RUNTMR;
               }
               if(system.functionmode == CURRMODE){
                  if((system.loadcurrent > (system.cutoffF1Hi*100)) || (system.loadcurrent <
(system.cutoffF1Lo*100))){
                     if(timer.currentTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();
                        system.fan1Fault = true;
                        UpdateStat(EEP_FAN1FAULT_ADD,system.fan1Fault);
                        appData.state = APP_STATE_INIT;
                     }
                  } else if(timer.currentTMR < CURRENTSENSE_RUNTMR) timer.currentTMR =
CURRENTSENSE_RUNTMR;
               }
            }
            if((system.runningfan == FAN2)&&(system.fan2Fault == false)){
               if(system.functionmode == AIRMODE){
                  if(READ_AIRSW2PIN()){
                     if(timer.airswTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();
                        system.fan2Fault = true;
                        UpdateStat(EEP_FAN2FAULT_ADD,system.fan2Fault);
                        appData.state = APP_STATE_INIT;
                     }
                  } else timer.airswTMR = AIRSW_RUNTMR;

                  if(system.loadcurrent > (system.cutoffF2Hi*100)){
```

```
                    if(timer.currentTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();
                        system.fan2Fault = true;
                        UpdateStat(EEP_FAN2FAULT_ADD,system.fan2Fault);
                        appData.state = APP_STATE_INIT;
                    }
                } else if(timer.currentTMR < CURRENTSENSE_RUNTMR) timer.currentTMR =
CURRENTSENSE_RUNTMR;
            }
            if(system.functionmode == CURRMODE){
                if((system.loadcurrent > (system.cutoffF2Hi*100)) || (system.loadcurrent <
(system.cutoffF2Lo*100))){
                    if(timer.currentTMR == 0){
                        RLY_FAN1_OFF();
                        RLY_FAN2_OFF();



                        system.fan2Fault = true;
                        UpdateStat(EEP_FAN2FAULT_ADD,system.fan2Fault);
                        appData.state = APP_STATE_INIT;
                    }
                } else if(timer.currentTMR < CURRENTSENSE_RUNTMR) timer.currentTMR =
CURRENTSENSE_RUNTMR;
            }
        }
        if((READ_INCPIN() == true) && (READ_DECPIN() == true) && (READ_SETPIN() == false)){
            LoadSecTMR1(5);
            while((!READ_SETPIN()) && timer.secTMR1);
            if(timer.secTMR1 == 0){
                system.setupmode = true;
                system.startFunc = false;
                ID = false;
                UpdateDisplay(Display_HYFN);
                DATAEE_WriteByte(EEP_PROGRAM_ADD,true);
                system.automode = true;
                LED_AUTO_ON();
                UpdateStat(EEP_AUTOMODE_ADD,system.automode);
                DelayMS(2000);
                UpdateDisplay(Display_AUTO_MODE);
                DelayMS(800);
                appData.state = APP_STATE_SETAUTO_MODE;
            }
        }
        if((READ_INCPIN() == false) && (READ_DECPIN() == true) && (READ_SETPIN() == true)){
            LoadSecTMR1(5);
            while((!READ_INCPIN()) && timer.secTMR1);
            if((timer.secTMR1 == 0) && (system.automode == false)){
                if(system.runningfan == FAN1){
                    if(system.fan2Fault == false) {system.runningfan = FAN2;
UpdateStat(EEP_RUNNINGFAN_ADD,system.runningfan);}
                    system.startFunc = false;
                } else if(system.runningfan == FAN2){
                    if(system.fan1Fault == false) {system.runningfan = FAN1;
UpdateStat(EEP_RUNNINGFAN_ADD,system.runningfan);}
                    system.startFunc = false;
                } else {

                }
                appData.state = APP_STATE_INIT;
```

```
            }
        }
        if((READ_INCPIN() == true) && (READ_DECPIN() == false) && (READ_SETPIN() == true)){
            LoadSecTMR1(5);
            while((!READ_DECPIN()) && timer.secTMR1);
            if(timer.secTMR1 == 0){
                UpdateStat(EEP_FAN1FAULT_ADD,false);
                UpdateStat(EEP_FAN2FAULT_ADD,false);
                system.runningfan = FAN1;
                UpdateStat(EEP_RUNNINGFAN_ADD,system.runningfan);
                UpdateStat(EEP_LOCALTIMER_ADD,0);
                appData.state = APP_STATE_INIT;
            }
        }
        if((READ_INCPIN() == false) && (READ_DECPIN() == false) && (READ_SETPIN() == true)){
            LoadSecTMR1(5);
            while((!READ_INCPIN())&&(!READ_DECPIN())&&timer.secTMR1);
            if(timer.secTMR1 == 0){
                UpdateDisplay(Display_HYFN);
                DelayMS(1000); system.startFunc = false;
                DATAEE_WriteByte(EEP_SYSDEFAULT_ADD,0);
                DelayMS(10);
                appData.state = APP_STATE_INIT;
            }


        }
        break;
    }
    case APP_STATE_SETAUTO_MODE:
    {
        if(!READ_INCPIN()){
            while(!READ_INCPIN());
            UpdateDisplay(Display_AUTO_MODE);
            DelayMS(1); UpdateStat(EEP_LOCALTIMER_ADD,0);
            system.automode = true;
            LED_AUTO_ON();
            UpdateStat(EEP_AUTOMODE_ADD,system.automode);
        }
        if(!READ_DECPIN()){
            while(!READ_DECPIN());
            UpdateDisplay(Display_MN_MODE);
            DelayMS(1); UpdateStat(EEP_LOCALTIMER_ADD,0);
            system.automode = false;
            LED_AUTO_OFF();
            UpdateStat(EEP_AUTOMODE_ADD,system.automode);
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            LED_AUTO_OFF();
            UpdateDisplay(Display_CURRENTSENSE_MODE);
            system.functionmode = CURRMODE;
            DelayMS(800);
            appData.state = APP_STATE_CURRENT_AIR;
            ID = false;
        }
        //appData.state = APP_STATE_RUN;
        break;
    }
    case APP_STATE_CURRENT_AIR:
```

```
{
    if(!READ_INCPIN()){
        while(!READ_INCPIN());
        UpdateDisplay(Display_CURRENTSENSE_MODE);
        system.functionmode = CURRMODE;
        DelayMS(1);
        UpdateStat(EEP_FUNCTIONMODE_ADD,system.functionmode);
    }
    if(!READ_DECPIN()){
        while(!READ_DECPIN());
        UpdateDisplay(Display_AIRSENSE_MODE);
        system.functionmode = AIRMODE;
        DelayMS(1);
        UpdateStat(EEP_FUNCTIONMODE_ADD,system.functionmode);
    }
    if(!READ_SETPIN()){
        while(!READ_SETPIN());
        system.setupmode = true;
        LED_FAN1_ON();
        LED_FAN2_OFF();
        DelayMS(800);
        ID = false;
        appData.state = APP_STATE_CALIBRATION_FAN1;
    }
    break;
}
case APP_STATE_CALIBRATION_FAN1:
{
    system.setupmode = DispCurrent;
    UpdateLoadCurrent();



    if(timer.secTMR3 != LOCALSECTMR){
        UpdateDisplay(system.loadcurrent/100);
        //PeriodicLogs();
        LOCALSECTMR = timer.secTMR3;
    }
    LED_FAN1_ON();
    LED_FAN2_OFF();
    RLY_FAN1_ON();
    RLY_FAN2_OFF();
    if(!READ_SETPIN()){
        while(!READ_SETPIN());
        system.setupmode = true;
        RLY_FAN1_OFF();
        RLY_FAN2_OFF();
        UpdateDisplay(Display_F1Hi);
        ID = 0;
        DelayMS(800);
        appData.state = APP_STATE_SETF1Hi;
    }
    break;
}
case APP_STATE_CALIBRATION_FAN2:
{
    system.setupmode = DispCurrent;
    RLY_FAN1_OFF();
    RLY_FAN2_ON();
    UpdateLoadCurrent();
    if(timer.secTMR3 != LOCALSECTMR){
```

```
            UpdateDisplay(system.loadcurrent/100);
            //PeriodicLogs();
            LOCALSECTMR = timer.secTMR3;
        }
        LED_FAN1_OFF();
        LED_FAN2_ON();
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            RLY_FAN1_OFF();
            RLY_FAN2_OFF();
            UpdateDisplay(Display_F2Hi);
            ID = 0;
            DelayMS(800);
            appData.state = APP_STATE_SETF2Hi;
        }
        break;
    }
    case APP_STATE_SETF1Hi:
    {
        if(!READ_INCPIN()){
            timer.msTMR1 = 200;
            while((!READ_INCPIN())&&timer.msTMR1);
            system.setupmode = DispCurrent;
            if(system.cutoffF1Hi != 200) system.cutoffF1Hi++;
            UpdateStat(EEP_CUTOFF_F1Hi_ADD,system.cutoffF1Hi);
            DelayMS(5);
            system.cutoffF1Hi = GetData(EEP_CUTOFF_F1Hi_ADD);
            UpdateDisplay(system.cutoffF1Hi);
        }
        if(!READ_DECPIN()){
            timer.msTMR1 = 200;
            while((!READ_DECPIN())&&timer.msTMR1);
            system.setupmode = DispCurrent;
            if(system.cutoffF1Hi > system.cutoffF1Lo) system.cutoffF1Hi--;
            UpdateStat(EEP_CUTOFF_F1Hi_ADD,system.cutoffF1Hi);
            DelayMS(5);



            system.cutoffF1Hi = GetData(EEP_CUTOFF_F1Hi_ADD);
            UpdateDisplay(system.cutoffF1Hi);
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            if(system.functionmode == CURRMODE){
                UpdateDisplay(Display_F1Lo);
                LED_FAN1_ON();
                LED_FAN2_OFF();
                DelayMS(800);
                appData.state = APP_STATE_SETF1Lo;
            } else {
                DelayMS(800);
                UpdateStat(EEP_CUTOFF_F1Lo_ADD,0);
                DelayMS(5);
                system.cutoffF1Lo = GetData(EEP_CUTOFF_F1Lo_ADD);
                appData.state = APP_STATE_CALIBRATION_FAN2;
            }
        }
        break;
```

```
}
case APP_STATE_SETF1Lo:
{
   if(!READ_INCPIN()){
      timer.msTMR1 = 200;
      while((!READ_INCPIN())&&timer.msTMR1);
      system.setupmode = DispCurrent;
      if(system.cutoffF1Lo < system.cutoffF1Hi-1) system.cutoffF1Lo++;
      UpdateStat(EEP_CUTOFF_F1Lo_ADD,system.cutoffF1Lo);
      DelayMS(5);
      system.cutoffF1Lo = GetData(EEP_CUTOFF_F1Lo_ADD);
      UpdateDisplay(system.cutoffF1Lo);
   }
   if(!READ_DECPIN()){
      timer.msTMR1 = 200;
      while((!READ_DECPIN())&&timer.msTMR1);
      system.setupmode = DispCurrent;
      if(system.cutoffF1Lo != 0) system.cutoffF1Lo--;
      UpdateStat(EEP_CUTOFF_F1Lo_ADD,system.cutoffF1Lo);
      DelayMS(5);
      system.cutoffF1Lo = GetData(EEP_CUTOFF_F1Lo_ADD);
      UpdateDisplay(system.cutoffF1Lo);
   }
   if(!READ_SETPIN()){
      while(!READ_SETPIN());
      system.setupmode = true;
      LED_FAN1_OFF();
      LED_FAN2_ON();
      ID = 3;
      DelayMS(800);
      appData.state = APP_STATE_CALIBRATION_FAN2;
   }
   break;
}
case APP_STATE_SETF2Hi:
{
   if(!READ_INCPIN()){
      timer.msTMR1 = 200;
      while((!READ_INCPIN())&&timer.msTMR1);
      system.setupmode = DispCurrent;
      if(system.cutoffF2Hi != 200) system.cutoffF2Hi++;
      UpdateStat(EEP_CUTOFF_F2Hi_ADD,system.cutoffF2Hi);
      DelayMS(5);
      system.cutoffF2Hi = GetData(EEP_CUTOFF_F2Hi_ADD);
      UpdateDisplay(system.cutoffF2Hi);


   }
   if(!READ_DECPIN()){
      timer.msTMR1 = 200;
      while((!READ_DECPIN())&&timer.msTMR1);
      system.setupmode = DispCurrent;
      if(system.cutoffF2Hi > system.cutoffF2Lo) system.cutoffF2Hi--;
      UpdateStat(EEP_CUTOFF_F2Hi_ADD,system.cutoffF2Hi);
      DelayMS(5);
      system.cutoffF2Hi = GetData(EEP_CUTOFF_F2Hi_ADD);
      UpdateDisplay(system.cutoffF2Hi);
   }
   if(!READ_SETPIN()){
      while(!READ_SETPIN());
      system.setupmode = true;
```

```
            if(system.functionmode == CURRMODE){
                UpdateDisplay(Display_F2Lo);
                LED_FAN1_OFF();
                LED_FAN2_ON();
                DelayMS(800);
                appData.state = APP_STATE_SETF2Lo;
            } else {
                UpdateStat(EEP_CUTOFF_F2Lo_ADD,0);
                DelayMS(5);
                system.cutoffF2Lo = GetData(EEP_CUTOFF_F2Lo_ADD);
                if(system.automode == true){
                    system.setupmode = DispHr;
                    system.hrTMR = GetData(EEP_HRTIMER_ADD);
                    UpdateDisplay(system.hrTMR);
                    DelayMS(800);
                    appData.state = APP_STATE_SETHRTIMER;
                } else {
                    DelayMS(800);
                    appData.state = APP_STATE_INIT;
                }
            }
        }
        break;
    }
    case APP_STATE_SETF2Lo:
    {
        if(!READ_INCPIN()){
            timer.msTMR1 = 200;
            while((!READ_INCPIN())&&timer.msTMR1);
            system.setupmode = DispCurrent;
            if(system.cutoffF2Lo < system.cutoffF2Hi-1) system.cutoffF2Lo++;
            UpdateStat(EEP_CUTOFF_F2Lo_ADD,system.cutoffF2Lo);
            DelayMS(5);
            system.cutoffF2Lo = GetData(EEP_CUTOFF_F2Lo_ADD);
            UpdateDisplay(system.cutoffF2Lo);
        }
        if(!READ_DECPIN()){
            timer.msTMR1 = 200;
            while((!READ_DECPIN())&&timer.msTMR1);
            system.setupmode = DispCurrent;
            if(system.cutoffF2Lo != 0) system.cutoffF2Lo--;
            UpdateStat(EEP_CUTOFF_F2Lo_ADD,system.cutoffF2Lo);
            DelayMS(5);
            system.cutoffF2Lo = GetData(EEP_CUTOFF_F2Lo_ADD);
            UpdateDisplay(system.cutoffF2Lo);
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            //UpdateDisplay(Display_SETCURRENT_MODE);
            ID = 0;
            DelayMS(800);


            system.cutoffF1Hi = GetData(EEP_CUTOFF_F1Hi_ADD);
            system.cutoffF1Lo = GetData(EEP_CUTOFF_F1Lo_ADD);
            system.cutoffF2Hi = GetData(EEP_CUTOFF_F2Hi_ADD);
            system.cutoffF2Lo = GetData(EEP_CUTOFF_F2Lo_ADD);

            LED_FAN1_OFF(); LED_FAN2_OFF();
            if(system.automode == true){
```

```
            system.setupmode = DispHr;
            system.hrTMR = GetData(EEP_HRTIMER_ADD);
            UpdateDisplay(system.hrTMR);
            DelayMS(800);
            appData.state = APP_STATE_SETHRTIMER;
        } else {
            DelayMS(800);
            appData.state = APP_STATE_INIT;
        }
    }
    break;
}
case APP_STATE_SETAIRSW_MODE:
{
    if(ID == 2){
        if(!READ_INCPIN()){
            while(!READ_INCPIN());
            UpdateDisplay(Display_ON);
            UpdateStat(EEP_AIRSWMODFAN2_ADD,true);
            DelayMS(5);
            system.airswFan2 = GetData(EEP_AIRSWMODFAN2_ADD);
        }
        if(!READ_DECPIN()){
            while(!READ_DECPIN());
            UpdateDisplay(Display_OFF);
            UpdateStat(EEP_AIRSWMODFAN2_ADD,false);
            DelayMS(5);
            system.airswFan2 = GetData(EEP_AIRSWMODFAN2_ADD);
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            UpdateDisplay(Display_SETAIRSW_MODE);
            ID = 0;
            DelayMS(800);
        }
    } else if(ID == 1){
        if(!READ_INCPIN()){
            while(!READ_INCPIN());
            UpdateDisplay(Display_ON);
            UpdateStat(EEP_AIRSWMODFAN1_ADD,true);
            DelayMS(5);
            system.airswFan1 = GetData(EEP_AIRSWMODFAN1_ADD);
        }
        if(!READ_DECPIN()){
            while(!READ_DECPIN());
            UpdateDisplay(Display_OFF);
            UpdateStat(EEP_AIRSWMODFAN1_ADD,false);
            DelayMS(5);
            system.airswFan1 = GetData(EEP_AIRSWMODFAN1_ADD);
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            system.setupmode = true;
            UpdateDisplay(Display_FAN2);
            ID = 2;
            DelayMS(800);
        }
    }
```

```
    break;
}
case APP_STATE_SETHRTIMER:
{
    if(!READ_INCPIN()){
        while(!READ_INCPIN());
        if(system.hrTMR != 12) system.hrTMR++;
        UpdateStat(EEP_HRTIMER_ADD,system.hrTMR);
        DelayMS(1); UpdateStat(EEP_LOCALTIMER_ADD,0);
        DelayMS(5);
        system.hrTMR = GetData(EEP_HRTIMER_ADD);
        UpdateDisplay(system.hrTMR);

    }
    if(!READ_DECPIN()){
        while(!READ_DECPIN());
        if(system.hrTMR != 1) system.hrTMR--;
        UpdateStat(EEP_HRTIMER_ADD,system.hrTMR);
        DelayMS(1); UpdateStat(EEP_LOCALTIMER_ADD,0);
        DelayMS(5);
        system.hrTMR = GetData(EEP_HRTIMER_ADD);
        UpdateDisplay(system.hrTMR);
    }
    if(!READ_SETPIN()){
        while(!READ_SETPIN());
        //system.setupmode = DispCurrent;
        LED_FAN1_OFF();
        LED_FAN2_OFF();
        timer.minTMR = 0; UpdateStat(EEP_LOCALTIMER_ADD,timer.minTMR);
        //UpdateDisplay(Display_SETCURRENT_MODE);
        ID = false;
        DelayMS(800);
        //CURRENTCUTOFF = GetData(EEP_CURRENTCUTOFF_ADD);
        appData.state = APP_STATE_INIT;
    }
    break;
}
case APP_STATE_EMRGSTOP:
{
    system.runningfan = 0;
    RLY_FAN1_OFF(); RLY_FAN2_OFF();
    UpdateDisplay(Display_Emrg_Stop);
    if(READ_EMRGSWPIN() == 0){
        timer.swdbncTMR = 1001; while((READ_EMRGSWPIN() == 0) && (timer.swdbncTMR));
        if(!(timer.swdbncTMR)) appData.state = APP_STATE_INIT;
    }
    break;
}
case APP_STATE_SYSTEM_TEST:
{
    switch ( TestModeStat )
    {
        case TM_START:
        {
            UpdateDisplay(Display_Start);
            break;
        }

        case TM_FAN1:
        {
            system.setupmode = DispCurrent; UpdateLoadCurrent();
```

```
    if(timer.secTMR3 != LOCALSECTMR){
        UpdateDisplay(system.loadcurrent/100); LOCALSECTMR = timer.secTMR3;
    }
    LED_FAN1_ON(); LED_FAN2_OFF(); RLY_FAN2_OFF(); DelayMS(100); RLY_FAN1_ON();



    break;
}

case TM_FAN2:
{
    system.setupmode = DispCurrent; UpdateLoadCurrent();
    if(timer.secTMR3 != LOCALSECTMR){
        UpdateDisplay(system.loadcurrent/100); LOCALSECTMR = timer.secTMR3;
    }
    LED_FAN1_OFF(); LED_FAN2_ON(); RLY_FAN1_OFF(); DelayMS(100); RLY_FAN2_ON();

    break;
}

case TM_SWITCH:
{
    if(!READ_INCPIN()){
        timer.msTMR1 = 200;
        while((!READ_INCPIN())&&timer.msTMR1);
        system.setupmode = DispCurrent;
        if(system.cutoffF1Hi != 200) system.cutoffF1Hi++;
        UpdateStat(EEP_CUTOFF_F1Hi_ADD,system.cutoffF1Hi);
        DelayMS(5);
        system.cutoffF1Hi = GetData(EEP_CUTOFF_F1Hi_ADD);
        UpdateDisplay(system.cutoffF1Hi);
    }
    if(!READ_DECPIN()){
        timer.msTMR1 = 200;
        while((!READ_DECPIN())&&timer.msTMR1);
        system.setupmode = DispCurrent;
        if(system.cutoffF1Hi > system.cutoffF1Lo) system.cutoffF1Hi--;
        UpdateStat(EEP_CUTOFF_F1Hi_ADD,system.cutoffF1Hi);
        DelayMS(5);
        system.cutoffF1Hi = GetData(EEP_CUTOFF_F1Hi_ADD);
        UpdateDisplay(system.cutoffF1Hi);
    }
    LED_FAN1_OFF(); LED_FAN2_OFF(); RLY_FAN1_OFF(); RLY_FAN2_OFF();

    break;
}

case TM_BMS:
{
    system.setupmode = false; UpdateDisplay(Display_BMS);
    if(timer.secTMR3 != LOCALSECTMR){

        if(index == 0){
            RLY_MODE_TOGGLE();
            index++;
        } else if(index == 1){
            RLY_MODE_TOGGLE(); DelayMS(100); RLY_BMSF1_TOGGLE();
            index++;
        } else if(index == 2){
            RLY_BMSF1_TOGGLE(); DelayMS(100); RLY_BMSF2_TOGGLE();
```

```
            index++;
        } else if(index == 3){
            RLY_BMSF2_TOGGLE(); DelayMS(100); RLY_FAN1_FAULT_TOGGLE();
            index++;
        } else if(index == 4){
            RLY_FAN1_FAULT_TOGGLE(); DelayMS(100); RLY_FAN2_FAULT_TOGGLE();
            index++;
        } else {
            index = 0;
            LED_AIRSW_OFF(); LED_AUTO_OFF(); LED_FAN1_OFF(); LED_FAN2_OFF();
            RLY_MANUAL_MODE(); RLY_FAN1_READY(); RLY_FAN2_READY();



            RLY_FAN1_OFF(); RLY_FAN2_OFF();
        }
        /*
        RLY_BMSF1_TOGGLE(); RLY_BMSF2_TOGGLE(); RLY_MODE_TOGGLE();
        RLY_FAN1_FAULT_TOGGLE(); RLY_FAN2_FAULT_TOGGLE();
        */
        LOCALSECTMR = timer.secTMR3;
    }
    break;
}

case TM_LED:
{
    system.setupmode = false; UpdateDisplay(Display_LED);
    if(timer.secTMR3 != LOCALSECTMR){

        if(index == 0){
            LED_AUTO_TOGGLE();
            index++;
        } else if(index == 1){
            LED_AUTO_TOGGLE(); DelayMS(100); LED_FAN1_TOGGLE();
            index++;
        } else if(index == 2){
            LED_FAN1_TOGGLE(); DelayMS(100); LED_FAN2_TOGGLE();
            index++;
        } else if(index == 3){
            LED_FAN2_TOGGLE(); DelayMS(100); LED_AIRSW_TOGGLE();
            index++;
        } else {
            index = 0;
            LED_AIRSW_OFF(); LED_AUTO_OFF(); LED_FAN1_OFF(); LED_FAN2_OFF();
            RLY_MANUAL_MODE(); RLY_FAN1_READY(); RLY_FAN2_READY();
            RLY_FAN1_OFF(); RLY_FAN2_OFF();
        }
        LOCALSECTMR = timer.secTMR3;
    }
    //    DelayMS(1000);
    break;
}

case TM_INPUTSW:
{
    system.setupmode = false; UpdateDisplay(Display_AIRSENSE_MODE);
    LED_AUTO_ON();
    if(READ_EMRGSWPIN()) LED_FAN1_ON(); else LED_FAN1_OFF();
    if(READ_AIRSW1PIN()) LED_FAN2_ON(); else LED_FAN2_OFF();
    if(READ_AIRSW2PIN()) LED_AIRSW_ON(); else LED_AIRSW_OFF();
```

```
                //    DelayMS(1000);
                break;
            }

            default:
            {
                /* TODO: Handle error in application's state machine. */
                UpdateDisplay(Display_HYFN);
                break;
            }
        }
        if(!READ_SETPIN()){
            while(!READ_SETPIN());
            TestModeStat++; DelayMS(800);
            if(TestModeStat == TM_FAN2){ RLY_FAN1_OFF(); RLY_FAN2_OFF(); DelayMS(1000); }
            if(TestModeStat == TM_SWITCH){ UpdateDisplay(Display_Switch); DelayMS(1000); }
            if(TestModeStat == TM_END){
                UpdateDisplay(Display_Stop); SetDefault(); DelayMS(2000);
                system.test = false; appData.state = APP_STATE_INIT;


            }
            LED_AIRSW_OFF(); LED_AUTO_OFF(); LED_FAN1_OFF(); LED_FAN2_OFF();
            RLY_MANUAL_MODE(); RLY_FAN1_READY(); RLY_FAN2_READY();
            RLY_FAN1_OFF(); RLY_FAN2_OFF();
        }
        break;
      }
      default:
      {
        /* TODO: Handle error in application's state machine. */
        break;
      }
    }
    if((READ_EMRGSWPIN())&&(system.test != true)){
      timer.swdbncTMR = 1001; while((READ_EMRGSWPIN()) && (timer.swdbncTMR));
      if(!(timer.swdbncTMR)) appData.state = APP_STATE_EMRGSTOP;
    }
    HardwareControls();

    if((READ_INCPIN() == false) && (READ_DECPIN() == false) && (READ_SETPIN() == false)){
      LoadSecTMR1(5); while((!READ_SETPIN()) && timer.secTMR1);
      if(timer.secTMR1 == 0){ RLY_MANUAL_MODE(); RLY_FAN1_READY(); RLY_FAN2_READY();
RLY_FAN1_OFF(); RLY_FAN2_OFF();
        UpdateDisplay(Display_TEST); SetDefault(); DelayMS(1000); ReadConfig();
        system.test = true; system.startFunc = false; TestModeStat = 0;
        appData.state = APP_STATE_SYSTEM_TEST;
      }
    }
  }
}

void HardwareControls(void){
  if(system.setupmode == false){
    if(system.startFunc){
      if(system.automode) RLY_AUTO_MODE(); else RLY_MANUAL_MODE();

      if((system.runningfan == FAN1) && (READ_AIRSW1PIN()==1) && (system.functionmode ==
AIRMODE)) LED_AIRSW_OFF();
      if((system.runningfan == FAN1) && (READ_AIRSW1PIN()==0) && (system.functionmode ==
AIRMODE)) LED_AIRSW_ON();
```

```
        if((system.runningfan == FAN2) && (READ_AIRSW2PIN()==1) &&  (system.functionmode ==
AIRMODE)) LED_AIRSW_OFF();
        if((system.runningfan == FAN2) && (READ_AIRSW2PIN()==0) &&  (system.functionmode ==
AIRMODE)) LED_AIRSW_ON();

        if(system.automode) LED_AUTO_ON(); else LED_AUTO_OFF();

        if(system.fan1Fault){
           RLY_FAN1_OFF(); RLY_FAN1_FAULT();
        } else RLY_FAN1_READY();

        if(system.fan2Fault){
           RLY_FAN2_OFF(); RLY_FAN2_FAULT();
        } else RLY_FAN2_READY();

        if((system.runningfan == FAN1) && (system.fan1Fault == false)){
           LED_FAN1_ON(); RLY_FAN2_OFF(); RLY_FAN1_ON();}

        if(system.runningfan == ALLOFF){
           RLY_FAN1_OFF(); RLY_FAN2_OFF();}

        if((system.runningfan == FAN2) && (system.fan2Fault == false)){
           LED_FAN2_ON(); RLY_FAN1_OFF(); RLY_FAN2_ON(); }
     } else if(system.test){



     } else {
           RLY_MANUAL_MODE();
           RLY_FAN1_READY();
           RLY_FAN2_READY();
           RLY_FAN1_OFF();
           RLY_FAN2_OFF();
     }
  } else if(system.setupmode == DispCurrent){

     RLY_MANUAL_MODE();
     RLY_FAN1_READY();
     RLY_FAN2_READY();

  } else {
     system.startFunc = false;
     RLY_MANUAL_MODE();
     RLY_FAN1_READY();
     RLY_FAN2_READY();
     RLY_FAN1_OFF();
     RLY_FAN2_OFF();
  }
}

/**
 End of File
*/
```

CHAPTER 8: ASSEMBLY AND TESTING

Testing Setup :

Figure 32 HVAC Test Setup under Load condition

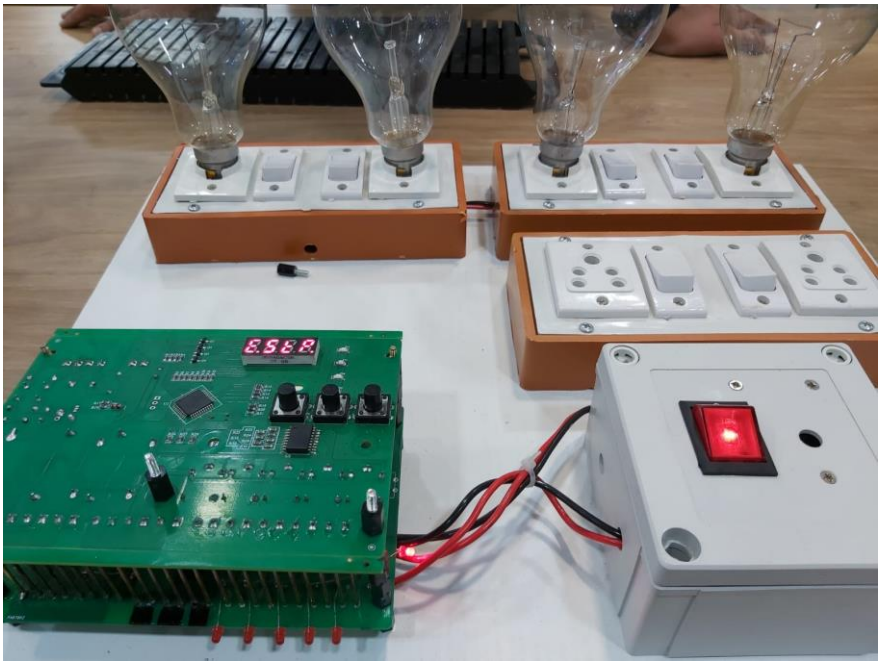


Figure 33 Test Setup for PCB

The testing setup included a vigorous followed sequence of testing procedures to ensure the proper and safe working of the whole developed board.

The HVAC board was developed with a Test/Debugging Mode which automatically guides the used through a number of guided steps to auto check and proceed through its performance.
This test mode consists of testing of various peripherals, working under no-load and different quantity of loads, switching and indication actions, input value actions etc.

Hence the boards were setup on the configured set and tested for correct hardware and software configurations and setups before finally packing them to their product version.

After the checking of terminals, correct code uploading and connections the board was dispatched, if any board is found to misbehave or not function properly in the test run-through, the master reset or other modes of operation are tried before checking it for issues and debugging manually before proceeding.

CHAPTER 9: SUMMARY AND CONCLUSION

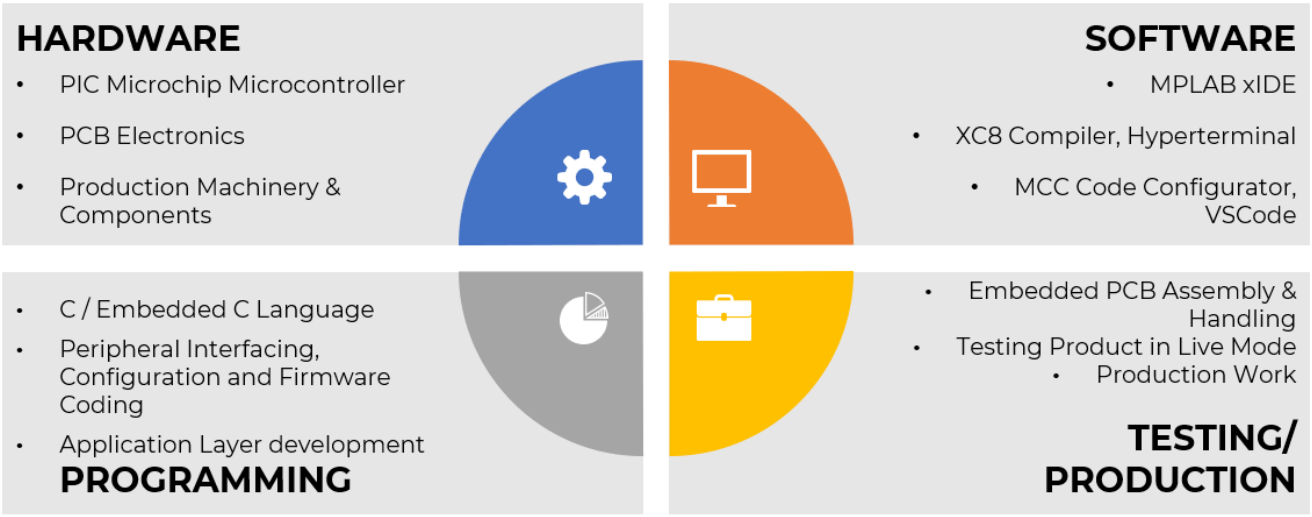
To summarize the Comprehensive Project, a ton of learnings and experiences were gained and the final product was established, changed for design optimizations, tested and mass assembled in the Industry.

The HVAC Twin Fan Controller is a device of vital use in commercial and industrial area in the recent times of Automation and are majorly used to ensure the System safety to the best of the abilities. The device provides the end user with a large variety of options to control the fan flow and ensure safe and sound operation of the AC cooling system and hance leads to an increase of operating life of the system in general, saving long term costs for the industries/users.

Hence a device of high value and long term endurance was developed and assembled, through application of various interdisciplinary methods.

The various Hardware components were used, tested and assembled on the main embedded PCB along with the PIC 16F1939 Microcontroller, while the software used included MPLAB xIDE with MCC for the development of the Application layer at the end.

My Learnings in brief include:



CHAPTER 10: REFERENCES & BIBLIOGRAPHY

A. *www.microchip.com; https://www.microchip.com/en-us/product/PIC16F1939*

B. *Microchip Technical learning center (Peripheral Interfacing)*

C. *https://microchipdeveloper.com/*

D. *https://microcontrollerslab.com/*

E. *https://openlabpro.com/*

F. *https://electrosome.com/*

G. Embisol technologies: End User Requirements Sheet