/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FILE : Temprature Control With Relay.c

\* PROGRAM : Engineering Capstone Project

\* PROGRAM CODE : EECE 8040

\* PROJECT : Thermo Electric Cooler for Tesla Model 3

\* PROGRAMMERS : 01)Akash Kachchhi

\* : 02)Sepideh Arabi

\* FIRST VERSION : 2019-07-09

\* DESCRIPTION : This code is used for controlling the temprature of

\* thermoelectric cooler based on the relay. This will turn

\* on the Peltier Module and Fan if the temprature is more

\* than 5.5 and Cooling will be turn on. If the temprature

\* will be less than 3.5 it will turn off the peltier module

\* and cooling fan. That time cooling will be off.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* This notice applies to any and all portions of this file

\* that are not between comment pairs USER CODE BEGIN and

\* USER CODE END. Other portions of this file, whether

\* inserted by the user or by software development tools

\* are owned by their respective copyright owners.

\*

\* COPYRIGHT(c) 2019 STMicroelectronics

\*

\* Redistribution and use in source and binary forms, with or without modification,

\* are permitted provided that the following conditions are met:

\* 1. Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* 2. Redistributions in binary form must reproduce the above copyright notice,

\* this list of conditions and the following disclaimer in the documentation

\* and/or other materials provided with the distribution.

\* 3. Neither the name of STMicroelectronics nor the names of its contributors

\* may be used to endorse or promote products derived from this software

\* without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

\* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

\* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

\* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

\* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

\* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

\* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

\* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "stm32l4xx\_hal.h"

#include <math.h>

#define SERIESRESISTOR 10000.0

#define NUMSAMPLES 5

// resistance at 25 degrees C

#define THERMISTORNOMINAL 7200

// temp. for nominal resistance (almost always 25 C)

#define TEMPERATURENOMINAL 25

// The beta coefficient of the thermistor (usually 3000-4000)

#define BCOEFFICIENT 3950

int samples[NUMSAMPLES];

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* --------------------------Global Variable Declaration----------------------\*/

char temp[10] = { 0 };

\_\_IO float uhADCxConvertedValue = 0.0;

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc1;

TIM\_HandleTypeDef htim1;

UART\_HandleTypeDef huart2;

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_USART2\_UART\_Init(void);

static void MX\_TIM1\_Init(void);

static void MX\_ADC1\_Init(void);

void calculateTemperature(void);

/\*\*\*\*\*\*\*\*\*\*\*LCD Function Declaration\*\*\*\*\*\*\*\*\*\*/

void HD44780\_Init();

void HD44780\_ClrScr(void);

void HD44780\_GotoXY(unsigned char x, unsigned char y);

void HD44780\_PutStr(char \*str);

void calculateTemperature(void) {

float steinhart;

if (HAL\_ADC\_Start(&hadc1) != HAL\_OK) {

printf("HAL\_ADC\_Start Error. \r\n");

}

if (HAL\_ADC\_PollForConversion(&hadc1, 10) != HAL\_OK) {

printf("HAL\_ADC\_PollForConversion Error. \r\n");

} else {

/\* ADC conversion completed \*/

uhADCxConvertedValue = HAL\_ADC\_GetValue(&hadc1);

printf("Analog value: %4.2f\t", uhADCxConvertedValue);

uhADCxConvertedValue = (4095.0 / uhADCxConvertedValue) - 1;

uhADCxConvertedValue = SERIESRESISTOR / uhADCxConvertedValue; // 10K / (1023/ADC - 1)

printf("Thermistor resistance: %4.2f\r\n", uhADCxConvertedValue);

steinhart = uhADCxConvertedValue / THERMISTORNOMINAL; // (R/Ro)

steinhart = log(steinhart); // ln(R/Ro)

steinhart /= BCOEFFICIENT; // 1/B \* ln(R/Ro)

steinhart += 1.0 / (TEMPERATURENOMINAL + 273.15); // + (1/To)

steinhart = 1.0 / steinhart; // Invert

steinhart -= 273.15; // convert to C

printf("Temperature: %4.2f \*C\r\n", steinhart);

}

if (steinhart >= 27.5) {

//peltier ON Fan ON

HD44780\_GotoXY(0, 1);

HD44780\_PutStr("COOLING ON ");

HAL\_GPIO\_WritePin(GPIOA, Peltier\_Relay\_Pin, GPIO\_PIN\_SET); //On

HAL\_GPIO\_WritePin(GPIOA, Motor\_Relay\_Pin, GPIO\_PIN\_SET);

} else if (steinhart <= 25.5) {

HD44780\_GotoXY(0, 1);

HD44780\_PutStr("COOLING OFF");

HAL\_GPIO\_WritePin(GPIOA, Motor\_Relay\_Pin, GPIO\_PIN\_RESET); //Off

HAL\_GPIO\_WritePin(GPIOA, Peltier\_Relay\_Pin, GPIO\_PIN\_RESET);

}

HD44780\_GotoXY(0, 0);

sprintf(temp, "Temp: %0.1f 'C", steinhart);

HD44780\_PutStr(temp);

HD44780\_PutStr(" ");

printf("\r\n");

HAL\_Delay(1000);

}

int main(void) {

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* Configure the system clock \*/

SystemClock\_Config();

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_USART2\_UART\_Init();

MX\_TIM1\_Init();

MX\_ADC1\_Init();

HD44780\_Init();

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1) {

calculateTemperature();

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void) {

RCC\_OscInitTypeDef RCC\_OscInitStruct;

RCC\_ClkInitTypeDef RCC\_ClkInitStruct;

RCC\_PeriphCLKInitTypeDef PeriphClkInit;

/\*\*Configure LSE Drive Capability

\*/

HAL\_PWR\_EnableBkUpAccess();

\_\_HAL\_RCC\_LSEDRIVE\_CONFIG(RCC\_LSEDRIVE\_LOW);

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_LSE

| RCC\_OSCILLATORTYPE\_MSI;

RCC\_OscInitStruct.LSEState = RCC\_LSE\_ON;

RCC\_OscInitStruct.MSIState = RCC\_MSI\_ON;

RCC\_OscInitStruct.MSICalibrationValue = 0;

RCC\_OscInitStruct.MSIClockRange = RCC\_MSIRANGE\_6;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_MSI;

RCC\_OscInitStruct.PLL.PLLM = 1;

RCC\_OscInitStruct.PLL.PLLN = 16;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV7;

RCC\_OscInitStruct.PLL.PLLQ = RCC\_PLLQ\_DIV2;

RCC\_OscInitStruct.PLL.PLLR = RCC\_PLLR\_DIV2;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK

| RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_1) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_USART2

| RCC\_PERIPHCLK\_ADC;

PeriphClkInit.Usart2ClockSelection = RCC\_USART2CLKSOURCE\_PCLK1;

PeriphClkInit.AdcClockSelection = RCC\_ADCCLKSOURCE\_PLLSAI1;

PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC\_PLLSOURCE\_MSI;

PeriphClkInit.PLLSAI1.PLLSAI1M = 1;

PeriphClkInit.PLLSAI1.PLLSAI1N = 16;

PeriphClkInit.PLLSAI1.PLLSAI1P = RCC\_PLLP\_DIV7;

PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC\_PLLQ\_DIV2;

PeriphClkInit.PLLSAI1.PLLSAI1R = RCC\_PLLR\_DIV2;

PeriphClkInit.PLLSAI1.PLLSAI1ClockOut = RCC\_PLLSAI1\_ADC1CLK;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure the main internal regulator output voltage

\*/

if (HAL\_PWREx\_ControlVoltageScaling(PWR\_REGULATOR\_VOLTAGE\_SCALE1)

!= HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure the Systick interrupt time

\*/

HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq() / 1000);

/\*\*Configure the Systick

\*/

HAL\_SYSTICK\_CLKSourceConfig(SYSTICK\_CLKSOURCE\_HCLK);

/\*\*Enable MSI Auto calibration

\*/

HAL\_RCCEx\_EnableMSIPLLMode();

/\* SysTick\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(SysTick\_IRQn, 0, 0);

}

/\* ADC1 init function \*/

static void MX\_ADC1\_Init(void) {

ADC\_ChannelConfTypeDef sConfig;

/\*\*Common config

\*/

hadc1.Instance = ADC1;

hadc1.Init.ClockPrescaler = ADC\_CLOCK\_ASYNC\_DIV1;

hadc1.Init.Resolution = ADC\_RESOLUTION\_12B;

hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;

hadc1.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

hadc1.Init.LowPowerAutoWait = DISABLE;

hadc1.Init.ContinuousConvMode = DISABLE;

hadc1.Init.NbrOfConversion = 1;

hadc1.Init.DiscontinuousConvMode = DISABLE;

hadc1.Init.NbrOfDiscConversion = 1;

hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;

hadc1.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_NONE;

hadc1.Init.DMAContinuousRequests = DISABLE;

hadc1.Init.Overrun = ADC\_OVR\_DATA\_PRESERVED;

hadc1.Init.OversamplingMode = DISABLE;

if (HAL\_ADC\_Init(&hadc1) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

/\*\*Configure Regular Channel

\*/

sConfig.Channel = ADC\_CHANNEL\_12;

sConfig.Rank = ADC\_REGULAR\_RANK\_1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_2CYCLES\_5;

sConfig.SingleDiff = ADC\_SINGLE\_ENDED;

sConfig.OffsetNumber = ADC\_OFFSET\_NONE;

sConfig.Offset = 0;

if (HAL\_ADC\_ConfigChannel(&hadc1, &sConfig) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* TIM1 init function \*/

static void MX\_TIM1\_Init(void) {

TIM\_ClockConfigTypeDef sClockSourceConfig;

TIM\_MasterConfigTypeDef sMasterConfig;

htim1.Instance = TIM1;

htim1.Init.Prescaler = 0;

htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim1.Init.Period = 255;

htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim1.Init.RepetitionCounter = 0;

htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE;

if (HAL\_TIM\_Base\_Init(&htim1) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterOutputTrigger2 = TIM\_TRGO2\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig)

!= HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\* USART2 init function \*/

static void MX\_USART2\_UART\_Init(void) {

huart2.Instance = USART2;

huart2.Init.BaudRate = 9600;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

huart2.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart2.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

if (HAL\_UART\_Init(&huart2) != HAL\_OK) {

\_Error\_Handler(\_\_FILE\_\_, \_\_LINE\_\_);

}

}

/\*\* Configure pins as

\* Analog

\* Input

\* Output

\* EVENT\_OUT

\* EXTI

\*/

static void MX\_GPIO\_Init(void) {

GPIO\_InitTypeDef GPIO\_InitStruct;

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE()

;

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE()

;

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE()

;

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOB,

LCD\_RS\_Pin | LCD\_E\_Pin | LCD\_D4\_Pin | LCD\_D5\_Pin | LCD\_D6\_Pin | LCD\_D7\_Pin,

GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, Motor\_Relay\_Pin | Peltier\_Relay\_Pin,

GPIO\_PIN\_RESET);

/\*Configure GPIO pins : LCD\_RS\_Pin LCD\_E\_Pin LCD\_D4\_Pin LCD\_D5\_Pin

LCD\_D6\_Pin LCD\_D7\_Pin \*/

GPIO\_InitStruct.Pin = LCD\_RS\_Pin | LCD\_E\_Pin | LCD\_D4\_Pin | LCD\_D5\_Pin

| LCD\_D6\_Pin | LCD\_D7\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);

/\*Configure GPIO pins : Motor\_Relay\_Pin Peltier\_Relay\_Pin \*/

GPIO\_InitStruct.Pin = Motor\_Relay\_Pin | Peltier\_Relay\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @param file: The file name as string.

\* @param line: The line in file as a number.

\* @retval None

\*/

void \_Error\_Handler(char \*file, int line) {

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

while (1) {

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t\* file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

/\*\*

\* @}

\*/

/\*\*

\* @}

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/