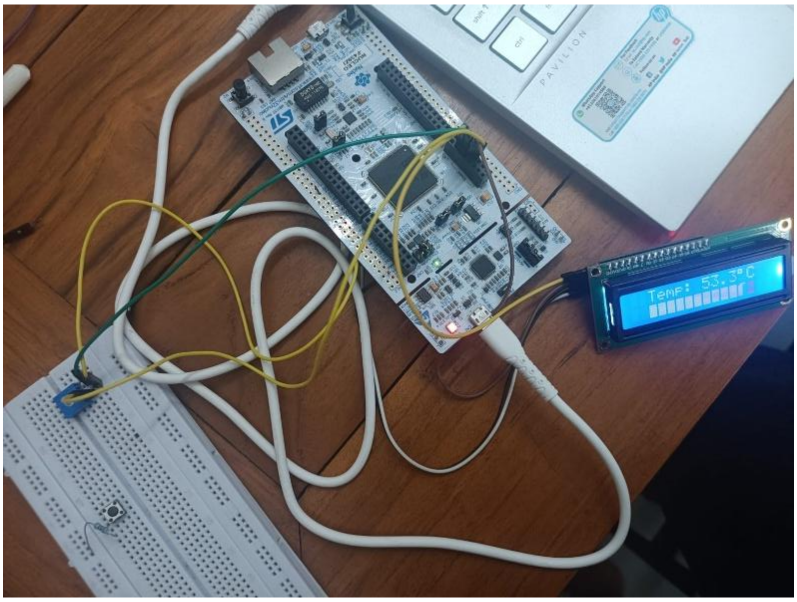
Lab Assignment 5

Mohit Maurya (22110145)

Parag Sarvoday Sahu (22110179)

Q1: ADC based live indicator



An image of the circuit

Initialising the required variables.

```
/* USER CODE BEGIN PV */
int m=0,j=0;
float data;
char DATA_ARRAY[11];
uint32_t adc_value;
uint32_t voltage_integer;
uint32_t integer_part;
uint32_t decimal_part;
int full;
int deci;
int speed[3]= {1000,5000,10000};
int previoustime;
int currenttime;
int debouncetime=200;
/* USER CODE END PV */
```

Function to initialize patterns.

```
void Load_graph(void){
char line_1[] = {0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10};
char line_2[] = {0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18};
char line_3[] = {0x1c,0x1c,0x1c,0x1c,0x1c,0x1c,0x1c,0x1c};
char line_4[] = {0x1E,0x1E,0x1E,0x1E,0x1E,0x1E,0x1E,0x1E};


 lcd_send_cmd(0x40);
 for(int i=0;i<8;i++){
 lcd_send_data(line_1[i]);
 }
 lcd_send_cmd(0x40+8);
 for(int i=0;i<8;i++){
 lcd_send_data(line_2[i]);
 }
 lcd_send_cmd(0x40+16);
 for(int i=0;i<8;i++){
 lcd_send_data(line_3[i]);
 }
 lcd_send_cmd(0x40+24);
 for(int i=0;i<8;i++){
 lcd_send_data(line_4[i]);
 }
}
```

Function to read the value from pin and convert it into desired value.

```
void read_sensor(void){
// PA3 as ADC
HAL_ADC_Start(&hadc1);
 HAL_ADC_PollForConversion(&hadc1, 200);

 adc_value = HAL_ADC_GetValue(&hadc1);
 voltage_integer = (adc_value * 800) / 4095;

 integer_part = voltage_integer / 10;
 decimal_part = voltage_integer % 10;
 sprintf(DATA_ARRAY, "Temp: %02d.%01d", integer_part, decimal_part);

}
```

Function to display information on the LCD display.

```
void display_graphs(void){
 lcd_put_cur(0, 0);
 lcd_send_string(DATA_ARRAY);
 lcd_put_cur(0, 10);
 lcd_send_data(0xDF);
 lcd_put_cur(0, 11);
 lcd_send_string("C");

full=integer_part/5;
for(j=0;j<16;j++){
lcd_put_cur(1, j);
lcd_send_string(" ");
}
for(j=0;j<full;j++){
lcd_put_cur(1, j);
lcd_send_data(255);
}
deci= integer_part%5;
lcd_put_cur(1, full);
if (deci!=0){
lcd_send_data(deci-1);
}
else {
lcd_send_string(" ");
}

}
```

Starting the timer, ADC and initializing the LCD display.

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
lcd_init();
lcd_clear();
Load_graph();

/* USER CODE END 2 */
```

Calling the read_sensor function inside the while loop.

```
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */
 read_sensor();
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

Handling the interrupt generated by the buttons.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
/* Prevent unused argument(s) compilation warning */
UNUSED(GPIO_Pin);
currenttime = HAL_GetTick();
// user button to change speed
if (GPIO_Pin == GPIO_PIN_13 && (currenttime-previoustime>debouncetime) ){
 m=(m+1)%3;
 __HAL_TIM_SET_AUTORELOAD(&htim2, speed[m]-1);
 TIM2->CNT = 0; // Reset counter
 TIM2->EGR |= TIM_EGR_UG; // Force update
 TIM2->CR1 |= TIM_CR1_CEN; // Restart Timer
 previoustime=currenttime;
 }
}
```

Handling the interrupt generated by the timer and using it to update the LCD display.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
/* Prevent unused argument(s) compilation warning */
UNUSED(htim);

/* NOTE : This function should not be modified, when the callback is
needed,
the HAL_TIM_PeriodElapsedCallback could be implemented in the user file
*/
if (htim->Instance==TIM2){
 display_graphs();
 }

}
/* USER CODE END 4 */
```

Q2: Multifunctional up-down counter

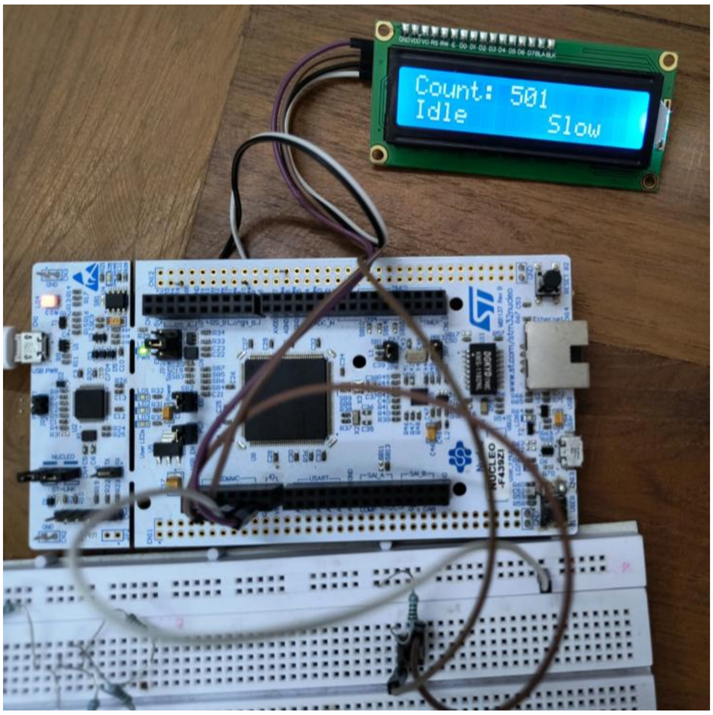


Image of the working circuit

Initialising the required variables.

```
/* USER CODE BEGIN PV */
int counter = -1;
int mode_state = 0; // 0 - up-counter; 1 - down-counter; 2 - idle; PB15
int speed_button=0; // On-board button PC13
int speed[4] ={10000,5000,2000,1000};
char speed_name[4][7] = {"Slow  ", "Medium", "Fast  ", "Turbo "};
int i = 0;
int previous_time = 0;
int current_time;
int debounce_time = 200;
char time_array[11];
/* USER CODE END PV */
```

Function to display information on the LCD display.

```
/* USER CODE BEGIN 0 */
void display_counter(void){

lcd_put_cur(0, 0);
sprintf(time_array, "Count: %03d" , counter);
lcd_send_string(time_array);

lcd_put_cur(1, 0);
if(mode_state == 0){
        lcd_send_string("Up  ");
}

lcd_put_cur(1, 0);
if(mode_state == 1){
        lcd_send_string("Down");
}

lcd_put_cur(1, 0);
if(mode_state == 2){
        lcd_send_string("Idle");
}

lcd_put_cur(1, 10);
lcd_send_string(speed_name[i]);

}
/* USER CODE END 0 */
```

Starting the timer and initializing the LCD display.

```
/* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim2);
  lcd_init();
  lcd_clear();
  /* USER CODE END 2 */
```

Calling the display_counter function inside the while loop.

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
          display_counter();
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Handling the interrupt generated by the buttons.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
/* Prevent unused argument(s) compilation warning */
UNUSED(GPIO_Pin);

// PIN 15 for mode
// PIN 13 (user_button) for speed selection
current_time = HAL_GetTick();
// user button to change speed
if (GPIO_Pin == GPIO_PIN_13 && (current_time - previous_time >
debounce_time)
){
        i=(i+1)%4;
        __HAL_TIM_SET_AUTORELOAD(&htim2, speed[i]-1);
        TIM2->CNT = 0; // Reset counter
        TIM2->EGR |= TIM_EGR_UG; // Force update
        TIM2->CR1 |= TIM_CR1_CEN; // Restart Timerprevioustime=currenttime;
}
// external button to change the mode
if (GPIO_Pin == GPIO_PIN_15 && (current_time - previous_time >
debounce_time) ){

        mode_state = (mode_state+1)%3;
}
        previous_time = current_time;
}
```

Handling the interrupt generated by the internal timer.

The Prescalar (PSC) and AutoReload Register (ARR) were chosen in the following way:

$$Prescalar = 8400 - 1$$

$$ARR = 10000 - 1$$

$$Natural\ clock\ frequency\ of\ TIM2 = 84\ MHz$$

The timer interrupt function counts to 10000 with each clock pulse and then provides an interrupt. Hence, the interrupt occurs every:

$$\frac{8400}{84\ MHz} * 10^4 = 1\ second$$

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
/* Prevent unused argument(s) compilation warning */
UNUSED(htim);
if (htim->Instance==TIM2){
        if(mode_state == 0){
                counter = (counter+1)%1000;
        }
        if(mode_state == 1){
                if(counter == 0){
                        counter = 1000;
                }
                counter = (counter-1)%1000;
        }
        if(mode_state == 2){
                counter = counter;
        }

}
}
```

References:

https://maxpromer.github.io/LCD-Character-Creator/

https://www.youtube.com/watch?v=diwjZPmFUKo