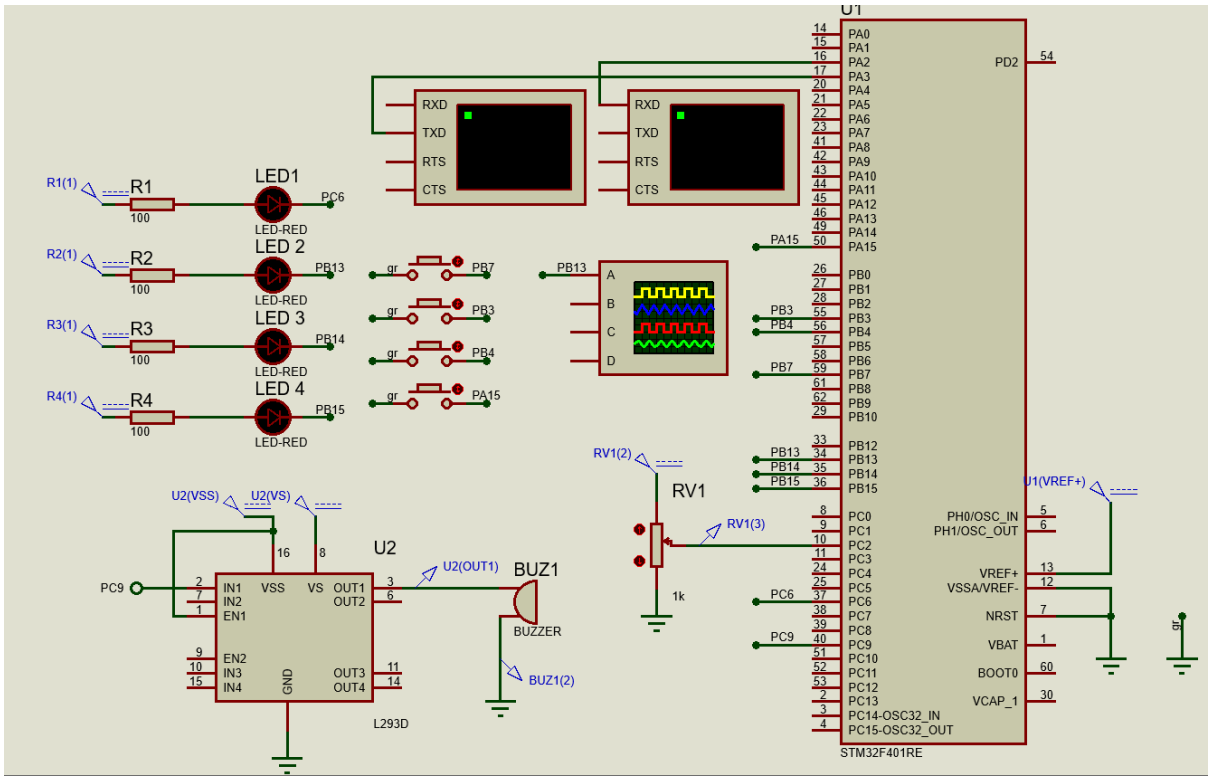
Full name: Nguyen Xuan Chu

Course: VN_ERS_AMJ24_Embedded C_HCMC_1

Student code: 240053

Solution

Circuit On Proteus



Code

8	SRS08	When Ignition Status is High, UART transmission & receive status to be displayed on the Debug Live Expression Window for every 500msec	Functional	ST, UT, IT		

```c
#include<stm32f401xe.h>
#include<stdio.h>
#include<stdint.h>

#define SYS_CLK 16000000
#define PCLK SYS_CLK
#define Baudrate 9600
#define LED3 (1<<14)
#define LED4 (1<<15)

//global variables
uint8_t count = 0;
uint8_t button2 = 0;
uint8_t button3 = 0;
uint8_t button4 = 0;
uint8_t it = 0;
uint8_t ena_print = 0;
```

```c
char buffer[100];
char receive = ' ';
typedef enum
{
	OFF = 0,
	RIGHT_INDICATOR_ON = 1,
	LEFT_INDICATOR_ON = 2,
	PARKING_LIGHT_ON = 3
} Status;
Status current_status = OFF;

void gpio_config() {
	// Enable clocks for ports A, B, and C
	RCC->AHB1ENR = (1<<0)|(1<<1)|(1<<2);

	// Setup GPIO for LED1 PC6 as OUTPUT
	GPIOC->MODER &=~(1<<13);
	GPIOC->MODER |= (1<<12);

	// Setup GPIO for buzzer
	GPIOC->MODER |= (1<<18);
	GPIOC->MODER &=~(1<<19);

	// Setup GPIO for LED2-3-4 PB13 PB14 PB15 as OUTPUT
	GPIOB->MODER &=~((1<<27)|(1<<29)|(1<<31));
	GPIOB->MODER |= (1<<26)|(1<<28)|(1<<30);

	// Setup GPIO for PB7 - PB3 - PB4 as INPUT with Pull-up
	GPIOB->MODER &=~((1<<14)|(1<<15)|(1<<6)|(1<<7)|(1<<8)|(1<<9));
	GPIOB->PUPDR |= (1<<14)|(1<<6)|(1<<8);
	GPIOB->PUPDR &=~((1<<15)|(1<<7)|(1<<9));

	// Setup GPIO for PA15 as INPUT with Pull-up
	GPIOA->MODER &= ~((1<<30)|(1<<31));
	GPIOA->PUPDR |= (1<<30);
	GPIOA->PUPDR &= ~(1<<31);

	//initial turn off all LEDs
	GPIOB->ODR |= (1<<13);
	GPIOB->ODR |= (1<<14);
	GPIOB->ODR |= (1<<15);
	GPIOC->ODR |= (1<<6);
}
void uart_init()
{
	RCC->APB1ENR |= (1<<17);

	GPIOA->MODER |= (1<<5);			//alternate PA2
	GPIOA->MODER &=~(1<<4);

	GPIOA->MODER |= (1<<7);			//alternate PA3
	GPIOA->MODER &=~(1<<6);

	GPIOA->AFR[0] |= (0x7<<8);		//set AF7-TX for PA2 -> 2*4 = 8
	GPIOA->AFR[0] |= (0x7<<12);		//set AF7-RX for PA3 -> 3*4 = 12

//	USART2->CR1 |= (1<<7);			//set interrupt for TX pin PA2
	USART2->CR1 |= (1<<5);			//set interrupt for RX pin PA3
	USART2->CR1 |= (1<<3);			//USART enable transmitter
	USART2->CR1 |= (1<<2);			//USART enable receiver
	USART2->BRR = (PCLK+(Baudrate/2))/Baudrate;
```

```
	USART2->CR1 |= (1<<13);				//enable UART
	NVIC_SetPriority(EXTI1_IRQn,2);
	NVIC_EnableIRQ(USART2_IRQn);
}
void send_char(unsigned char ch)
{
	while(!(USART2->SR&(1<<7)));		//wait till the data register
gets empty
	USART2->DR=ch;
}
void send_string(char *str)
{
	while(*str)
	{
		send_char(*str);
		str++;
	}
}
void TIM1_PWM_Init()
{
	//Configure PC6 for TIM3_CH1
//	RCC->AHB1ENR |= (1<<0);
	GPIOB->MODER &= ~(1<<26);
	GPIOB->MODER |= (1<<27);			//select alternate function mode
	GPIOB->AFR[1] = (1<<20);			// 0 is for AFRL if AFRH is 1
(AF2)//Set alternate function 1 (AF1) for PB13

	//Configure Timer 1 Channel 1
	RCC->APB2ENR |= (1<<0);				// Enable clock for TIMER1
	TIM1->PSC = 0;						// do not divide the input
frequency
	TIM1->ARR = 10000-1;				//	load some default value
	TIM1->CCMR1 |= (1<<3);
	TIM1->CCMR1 |= (1<<5) | (1<<6);		// select PWM1 mode
	TIM1->CCMR1 &= ~(1<<4);				// select PWM1 mode
	// Configure complementary output (CH1N)
	TIM1->CCER &=~(1<<3);				//OC1N active high
    TIM1->CCER |= (1<<2);				// Enable complementary output
(CC1NE)
    TIM1->BDTR |= (1<<15);				// MOE (Main Output Enable) bit
	TIM1->CNT = 0;						//Timer start count from 0
	TIM1->CR1 |= (1<<0);				//enable counter
}
void Set_frq_duty_cycle_TIM1(unsigned long int frequency, unsigned int
duty)
{
	TIM1->ARR = ((16000000/frequency)-1);
	TIM1->CCR1 = (duty*((TIM1->ARR)+1))/100;  //CCR1 because we use
channel 1
}
void TIM3_Init()
{
	__disable_irq();
	RCC->APB1ENR |= (1<<1);
	TIM3->PSC = 16000-1;				//set frequency 1kHz = 1ms
	TIM3->CNT = 0;						//count from 0
	TIM3->ARR = 500-1;					//interrupt after 500ms
	TIM3->DIER |= (1<<0);				//enable interrupt for timer 2
	TIM3->CR1 |= (1<<0);				//enable counter
	NVIC_SetPriority(TIM3_IRQn, 2);
	NVIC_EnableIRQ(TIM3_IRQn);
```

```
	__enable_irq();
}
void TIM3_IRQHandler()
{
	if ((TIM3->SR & (1<<0)))		//check if the timer reach the
value or not
	{
		if (current_status == RIGHT_INDICATOR_ON || current_status ==
LEFT_INDICATOR_ON)
		{
			it++;
		}
		if (current_status == PARKING_LIGHT_ON)
		{
			it = 1;
		}
		ena_print = 1;
	}
	TIM3->SR &= ~(1<<0);			//reset the flag
}
void adc_init()
{
	RCC->APB2ENR|=(1<<8);			//enable clock for ADC1
	// Setup  PC2 mode adc input
	GPIOC->MODER |= ((1<<4)|(1<<5));
	//configure adc
	ADC1->SQR1&=~(0xF<<20);			// bit [23:20] 1 conversion
	ADC1->SQR3 = (12<<0);			// write the channel of
ADC1_IN12?
	ADC1->CR2|=(1<<1);				//continuous conversion
	ADC1->CR2|=(1<<0);				//enable ADC converter
	ADC1->CR2|=(1<<30);				// start conversion
}
void Ext_init_PB7()
{
    __disable_irq();                // Disable global interrupt
    RCC->APB2ENR |= (1 << 14);      // Enable System configuration control
clock
    SYSCFG->EXTICR[1] |= (1 << 12); // Select PB7 as External Interrupt pin
    EXTI->IMR |= (1 << 7);          // Unmask on line PB7
    EXTI->FTSR |= (1 << 7);         // Falling edge triggered on line PB7
    NVIC_SetPriority(EXTI1_IRQn,1);
    NVIC_EnableIRQ(EXTI9_5_IRQn);   // Enable EXTI7
    __enable_irq();                 // Enable global interrupt
}
/* External interrupt button 2 as PB3 */
void Ext_init_PB3()
{
    __disable_irq();                // Disable global interrupt
    RCC->APB2ENR |= (1 << 14);      // Enable System configuration control
clock
    SYSCFG->EXTICR[0] |= (1 << 12); // Select PB3 as External Interrupt pin
    EXTI->IMR |= (1 << 3);          // Unmask on line PB3
    EXTI->FTSR |= (1 << 3);         // Falling edge triggered on line PB3
    NVIC_SetPriority(EXTI1_IRQn,1);
    NVIC_EnableIRQ(EXTI3_IRQn);     // Enable EXTI3
    __enable_irq();                 // Enable global interrupt
}
/* External interrupt button 3 as PB4 */
void Ext_init_PB4()
{
```

```
    __disable_irq();                 // Disable global interrupt
    RCC->APB2ENR |= (1 << 14);       // Enable System configuration control
clock
    SYSCFG->EXTICR[1] |= (1 << 0);  // Select PB4 as External Interrupt pin
      EXTI->IMR |= (1 << 4);              // Unmask on line PB4
      EXTI->FTSR |= (1 << 4);             // Falling edge triggered on line
PB4
      NVIC_SetPriority(EXTI1_IRQn,1);
      NVIC_EnableIRQ(EXTI4_IRQn);         // Enable EXTI4
      __enable_irq();                     // Enable global interrupt
}
/* External interrupt button 4 as PA15 */
void Ext_init_PA15()
{
      __disable_irq();                 // Disable global interrupt
      RCC->APB2ENR |= (1 << 14);       // Enable System configuration
control clock
      SYSCFG->EXTICR[3] |= (0 << 15); // Select PA15 as External Interrupt
pin
      EXTI->IMR |= (1 << 15);          // Unmask on line PA15
      EXTI->FTSR |= (1 << 15);         // Falling edge triggered on line
PA15
      NVIC_SetPriority(EXTI1_IRQn,1);
      NVIC_EnableIRQ(EXTI15_10_IRQn); // Enable EXTI15
      __enable_irq();                  // Enable global interrupt
}
int is_ignited()
{
      return (((GPIOC->ODR)>>6)&1);
}
void blink(int pin)//(1<<14)|(1<<15)
{
      //toggle after 2000ms
      if (it == 4)
      {
            GPIOB->ODR ^= pin;
            GPIOC->ODR ^= (1<<9);
            it = 0;
      }
}
void blink_off(int pin)
{
      GPIOB->ODR |= pin;
      GPIOC->ODR &=~(1<<9);
}
void parking_light(int pin)
{
      //toggle after 500ms
      if (it)
      {
            GPIOB->ODR ^= pin;
            GPIOC->ODR ^= (1<<9);
            it = 0;
      }
}

int main()
{
      gpio_config();
      uart_init();
      adc_init();
```

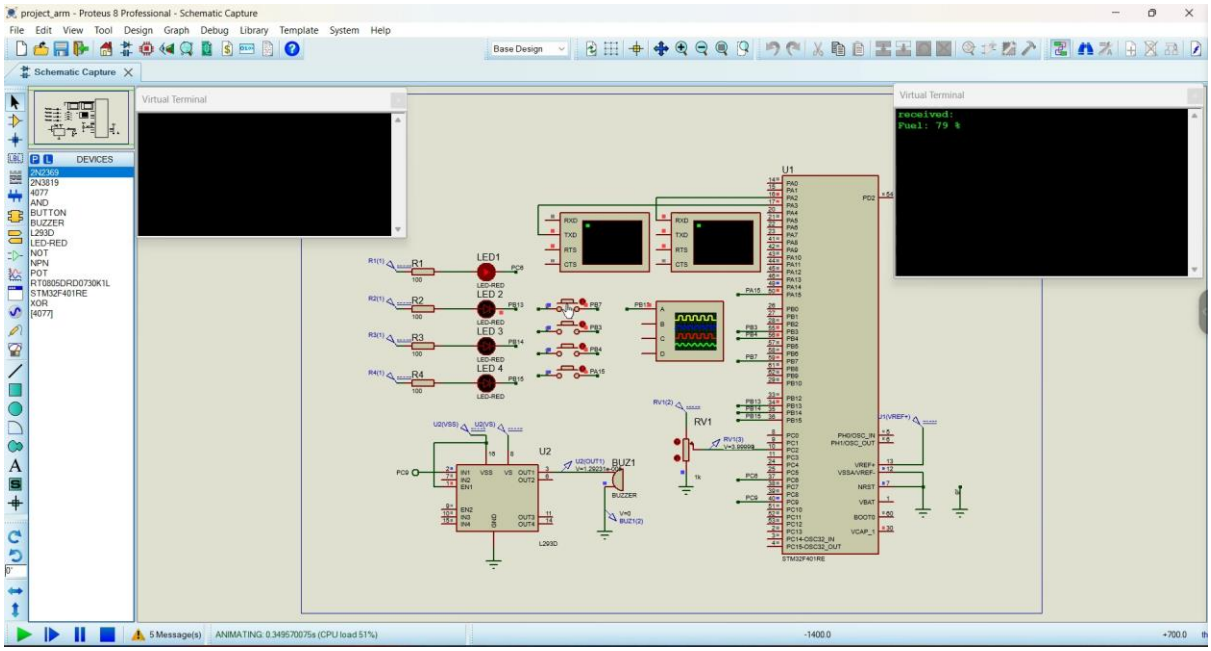
```
	TIM1_PWM_Init();
	TIM3_Init();
	Ext_init_PB7();
	Ext_init_PB3();
	Ext_init_PB4();
	Ext_init_PA15();

	int adc_data = 0;
	uint8_t adc_percentage = 0;

	while(1)
	{
		if (is_ignited() == 0)
	//ignition ON
		{
//			sprintf(buffer, "s:%d b2:%d b3:%d b4:%d it:%d\r",
//					current_status,
//					count,
//					button3,
//					button4,
//					it
//			);
//			send_string(buffer);
			//button 3
			switch (button3) {
				case 1:
					current_status = RIGHT_INDICATOR_ON;
					blink(LED3);
					break;
				case 2:
					current_status = OFF;
					blink_off(LED3);
					button3 = 0;
				default:
					break;
			}
			//button 4
			switch (button4) {
				case 1:
					current_status = LEFT_INDICATOR_ON;
					blink(LED4);
					break;
				case 2:
					current_status = OFF;
					blink_off(LED4);
					button4 = 0;
				default:
					break;
			}
			//button 2
			switch (count)
			{
				case 1:
					Set_frq_duty_cycle_TIM1(10000, 10);
					break;
				case 2:
					Set_frq_duty_cycle_TIM1(10000, 90);
					break;
				case 3:
					if (current_status== OFF || current_status ==
PARKING_LIGHT_ON)
```

```
                                    {
                                          current_status = PARKING_LIGHT_ON;
                                          parking_light(LED3|LED4);
                                    }
                                    break;
                              case 4:
                                    Set_frq_duty_cycle_TIM1(10000, 0);
                                    if (current_status == PARKING_LIGHT_ON)
                                    {
                                          blink_off(LED3|LED4);
                                    }
                                    count = 0;
                                    break;
                              default:
                                    break;
                        }
                        //receive message and transmit fuel level every 500ms
                        if (ena_print == 1)
                        {
                              sprintf(buffer, "received: %c\r", receive);
                              send_string(buffer);
                              while ((ADC1->SR>>1 & 1) != 1);
                              //get adc value 12bit [11:0]
                              adc_data = (ADC1->DR & 0xFFF);
                              adc_percentage = (adc_data*100/4096);
                              sprintf(buffer, "Fuel: %d %c\r", adc_percentage,
'%');
                              send_string(buffer);
                              ena_print = 0;
                        }
                  }
                  else                                          //ignition OFF
                  {
                        Set_frq_duty_cycle_TIM1(10000, 0);
                        blink_off(LED3|LED4);
                  }
            }
}
void EXTI9_5_IRQHandler(void)
{
    if ((((EXTI->PR) >> 7) & 1))
    {
      GPIOC->ODR ^= (1<<6);
    }
    EXTI->PR |= (1 << 7);
}
void EXTI3_IRQHandler(void)
{
    if ((((EXTI->PR) >> 3) & 1))
    {
      if (is_ignited() == 0)
      {
                  count++;
                  if (count == 3 && current_status == OFF)
                  {
                        GPIOB->ODR &=~(LED3|LED4);
                        GPIOC->ODR |= (1<<9);
                  }
                  if (count == 4)
                  {
                        current_status = OFF;
```

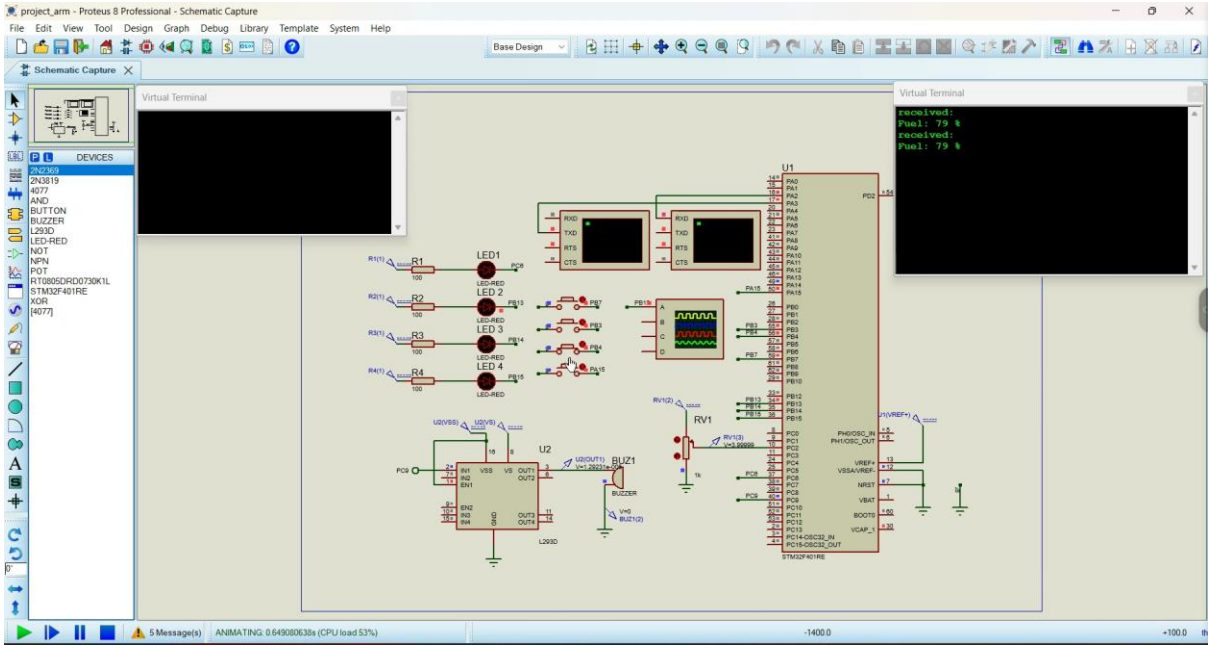
```
                    }
        }
    }
    EXTI->PR |= (1 << 3); // Clear the interrupt flag
}
void EXTI4_IRQHandler(void)
{
      if ((((EXTI->PR) >> 4) & 1))
      {
            if (is_ignited() == 0)
            {
                  if (current_status == OFF || current_status ==
RIGHT_INDICATOR_ON)
                  {
                        GPIOB->ODR &=~LED3;
                        GPIOC->ODR |= (1<<9);
                        button3++;
                  }
            }
      }
      EXTI->PR |= (1 << 4); // Clear the interrupt flag
}
void EXTI15_10_IRQHandler(void)
{
      if ((((EXTI->PR) >> 15) & 1))
      {
            if (is_ignited() == 0)
            {
                  if (current_status == OFF || current_status ==
LEFT_INDICATOR_ON)
                  {
                        GPIOB->ODR &=~LED4;
                        GPIOC->ODR |= (1<<9);
                        button4++;
                  }
            }
      }
      EXTI->PR |= (1 << 15);
}
void USART2_IRQHandler()
{
      if ((USART2->SR&(1<<5)))            //receive char
      {
            receive = USART2->DR;
      }
}
```

Result

1		ST01	Pres the Ignition Key "1st time"	Board Power Up • Program Running • Ignition Status Off	Ignition Status to High & Ignition LED to be ON, ADC Value & UART status to be shown on Debug Live Expression window		Not Tested	

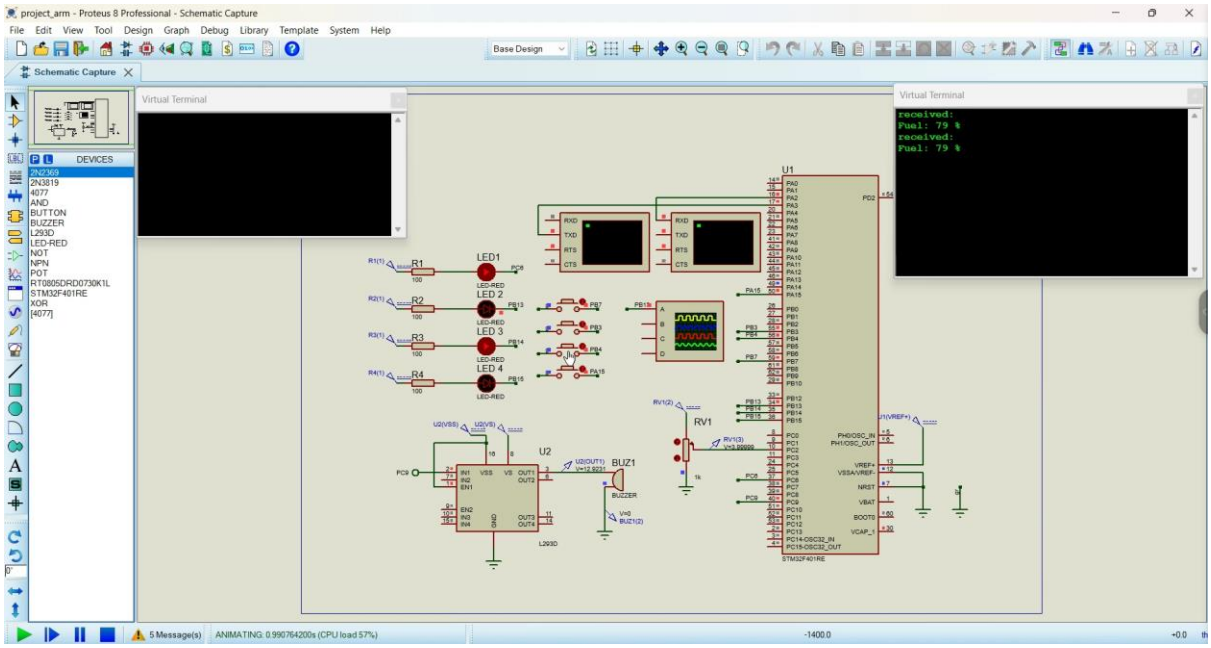


2		ST02	Press the Ignition key "2nd time"	Board Power Up • Program Running • Ignition LED ON • ADC value on the Live expression window. • UART value on live expression window	Ignition Status Off All other operations off	Not Tested

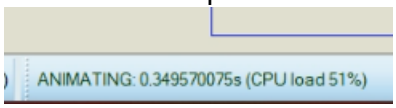


Press switch 3 but it is not operated its function.

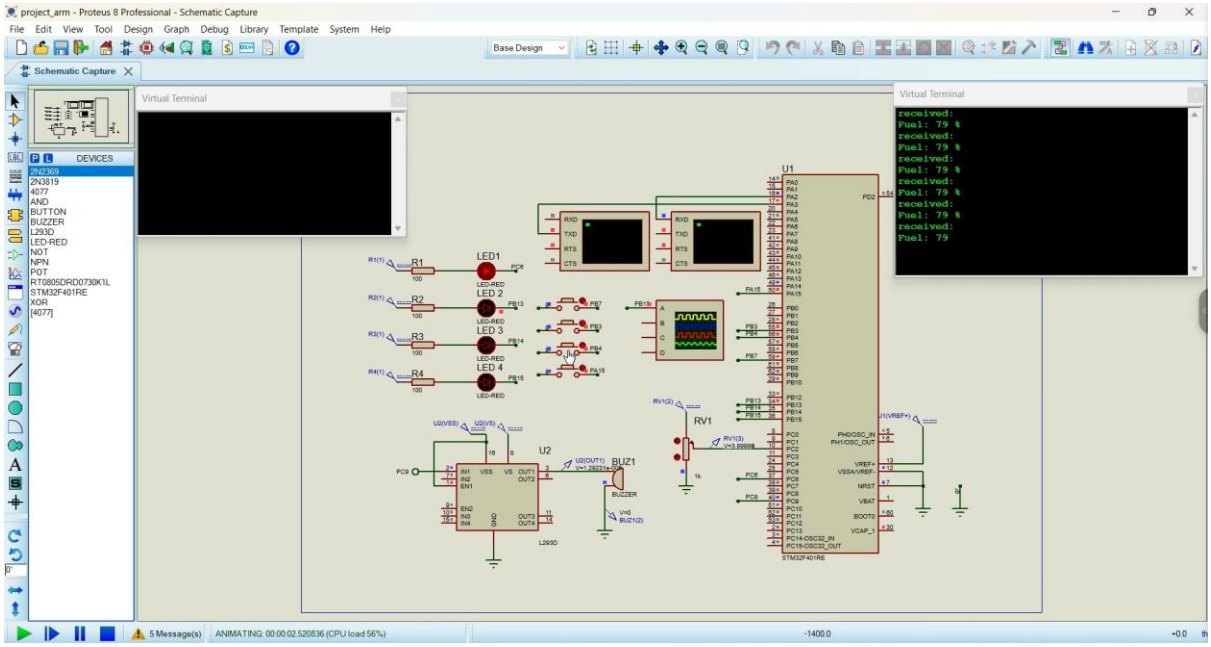
3		ST03	Press the Right Indicator key "1st time	Board Power Up • Program Running • Ignition LED ON • ADC value and UART shown on debug live expression window	Right indicator LED & Buzzer to blink with 0.5Hz frequency	Not Tested



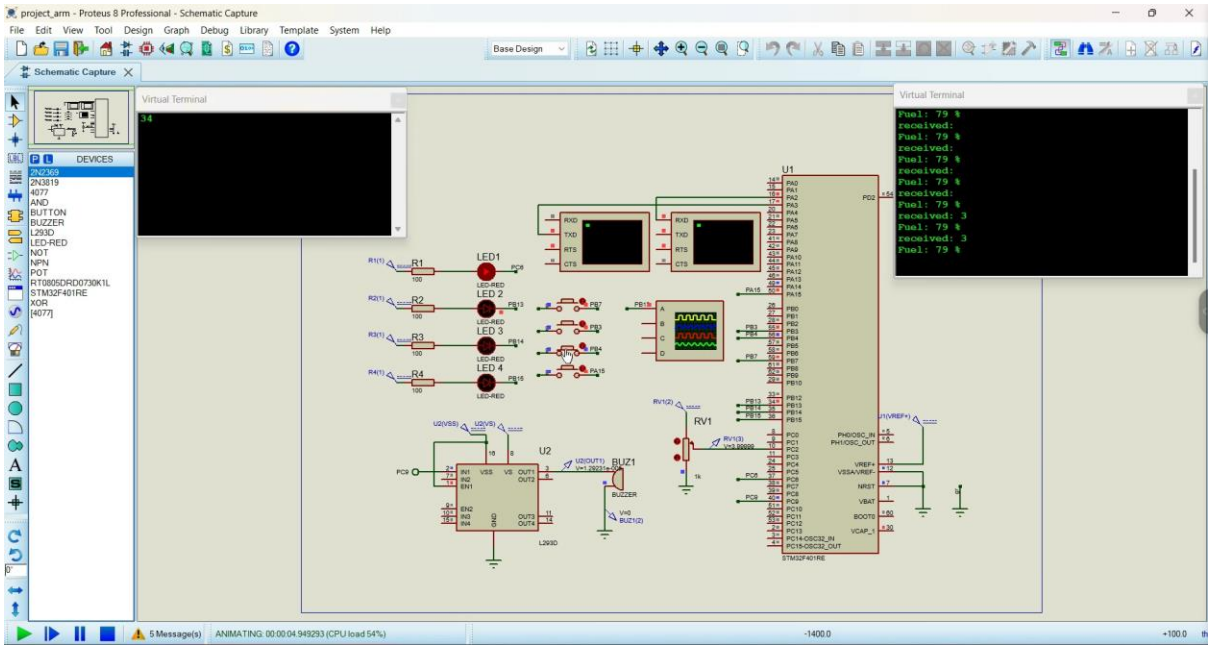
The switch 3 is pressed at the 1st time at 0.34 sec (time simulation)



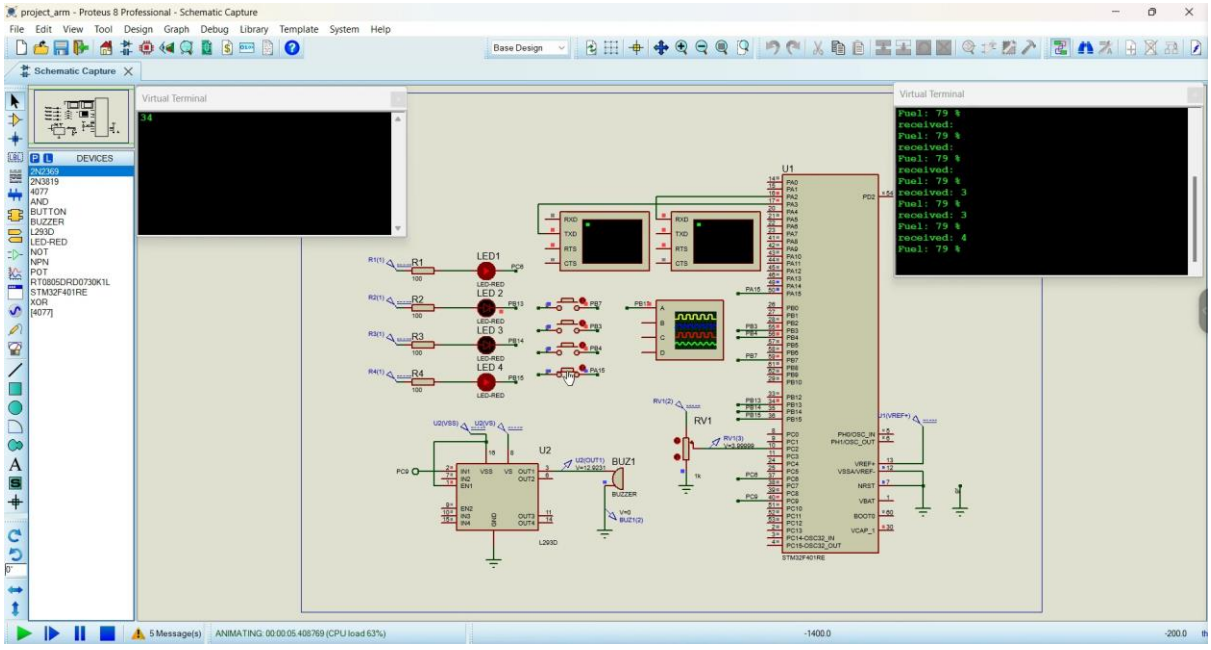
At 2.52 sec (time simulation), the LED 3 is turned off. 



4		ST04	Press the Right Indicator key 2nd time	Board Power Up • Program Running • ADC Value & UART shown on Debug Live Expression window • Right indicator LED & Buzzer to blinking with 0.5Hz frequency	Right Indicator & Buzzer to be Off		Not Tested

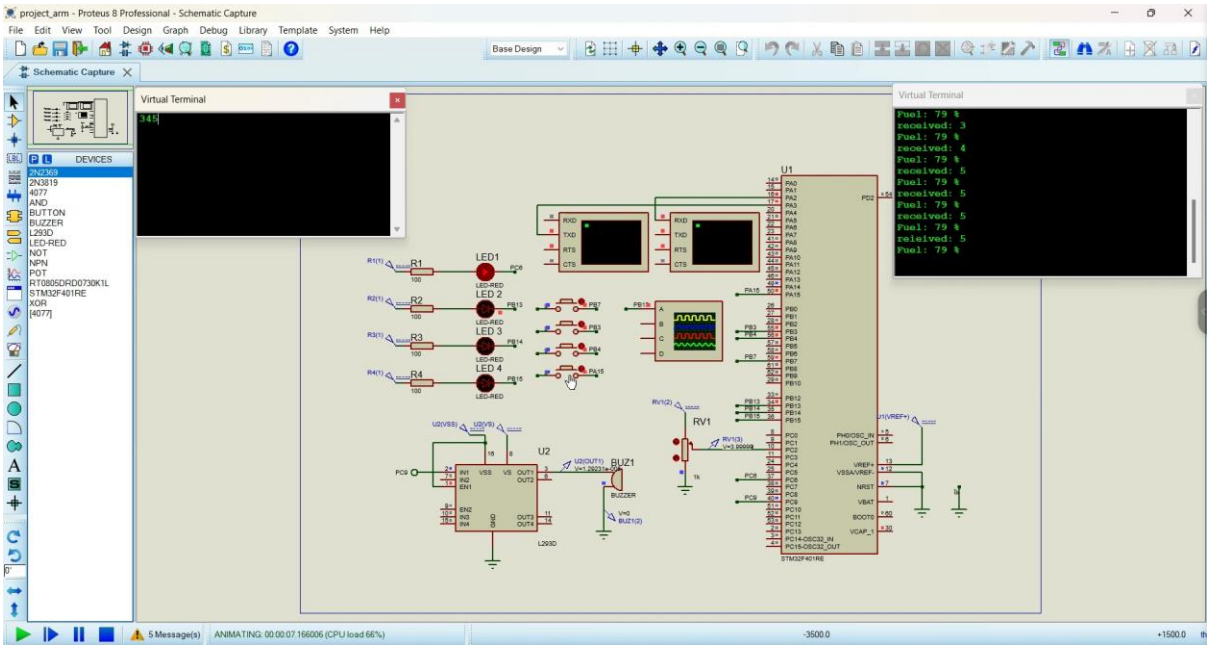


5		ST05	Press the Left Indicator key "1st" time	Board Power Up • Program Running • Ignition LED ON • ADC Value & UART shown on Debug Live Expression window	Left indicator LED & Buzzer to blink with 0.5Hz frequency		Not Tested

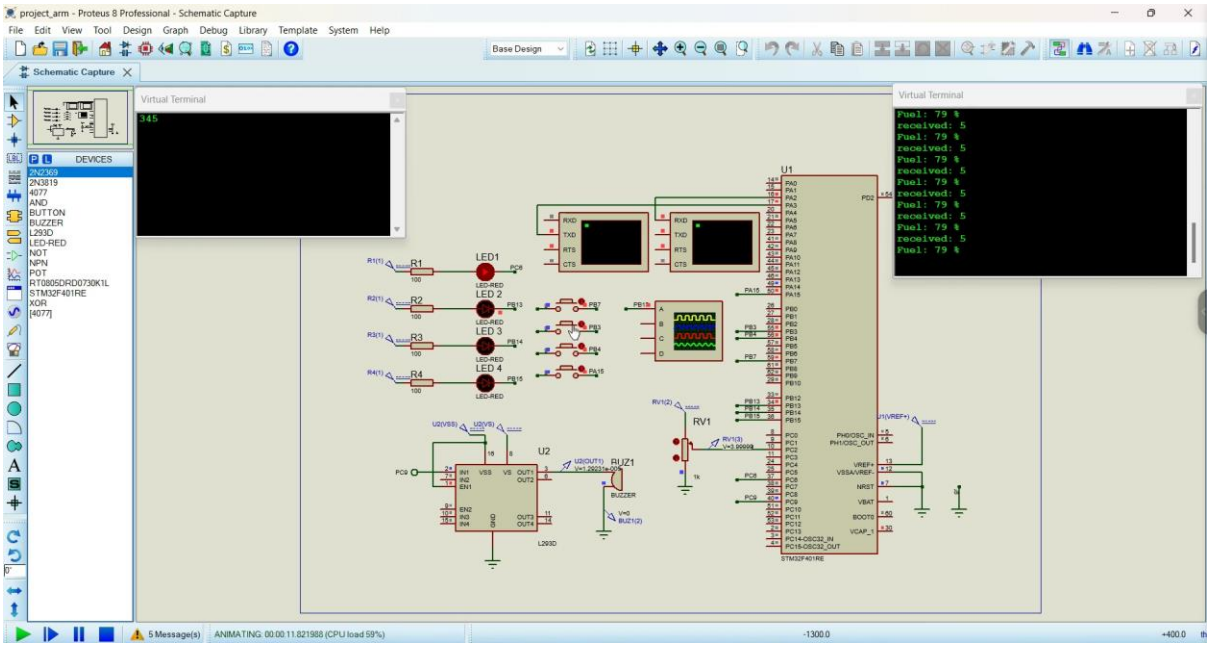


Switch 4 is pressed at ANIMATING: 00:00:05.408769 (CPU load 63%)

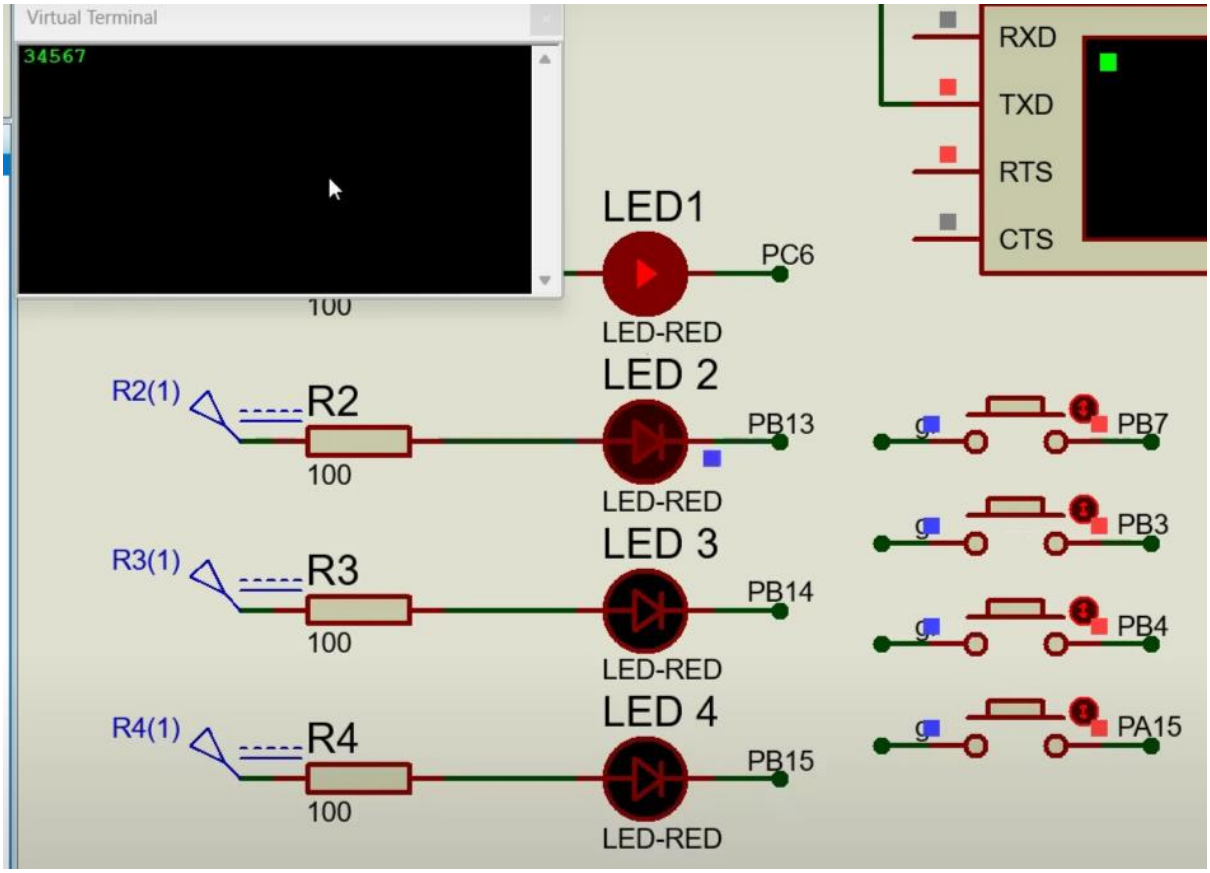
At ANIMATING: 00:00:07.166006 (CPU load 66%), LED 4 is turned off



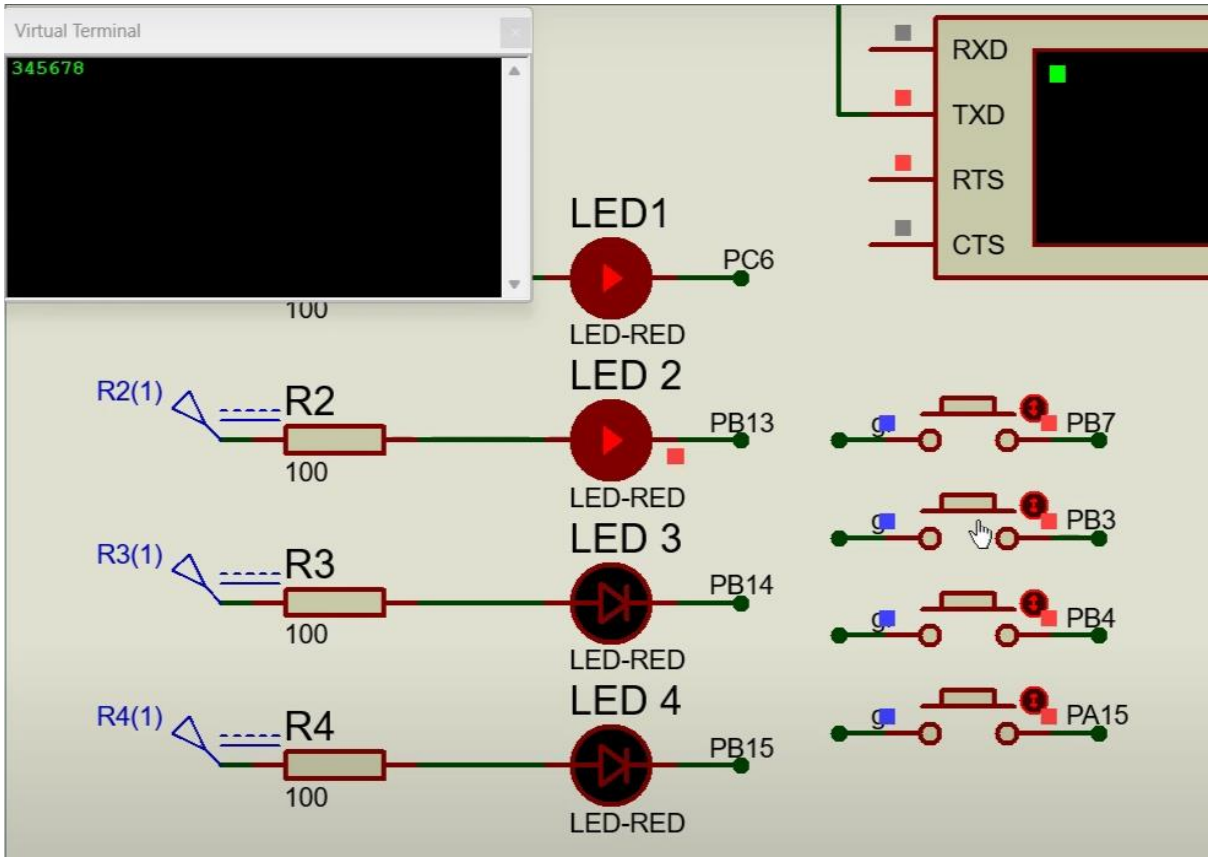
6		ST06	Press the Left Indicator key "2nd" time	Board Power Up • Program Running • ADC Value & UART shown on Debug Live Expression window Left indicator LED & Buzzer to blinking with 0.5Hz frequency	Left Indicator & Buzzer to be Off		Not Tested



7		ST07	Press the Light Switch "1st" time	Board Power Up • Program Running Ignition LED ON • ADC Value & UART shown on Debug Live Expression window	The Low Beam Light to be activated in LED with 10% Duty Cycle		Not Tested

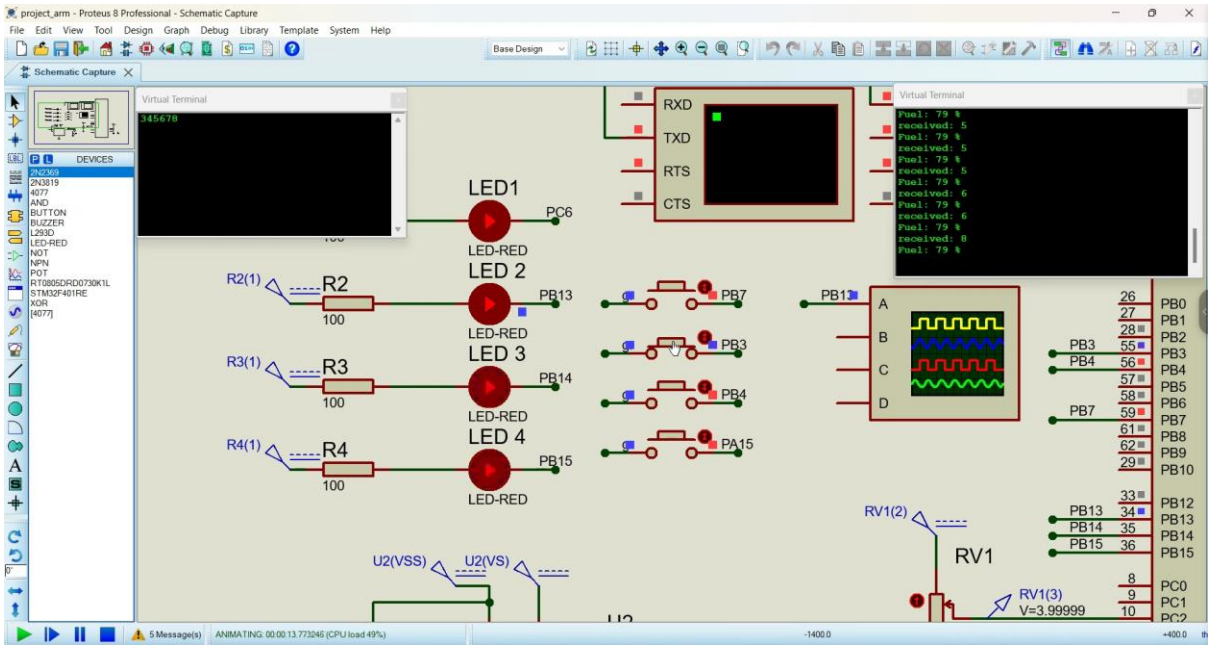


8		ST08	Press the Light Switch "2nd" time	Board Power Up • Program Running Ignition LED ON • ADC Value & UART shown on Debug Live Expression window • Low beam light ON	The High Beam Light to be activated in LED with 90% Duty Cycle		Not Tested	

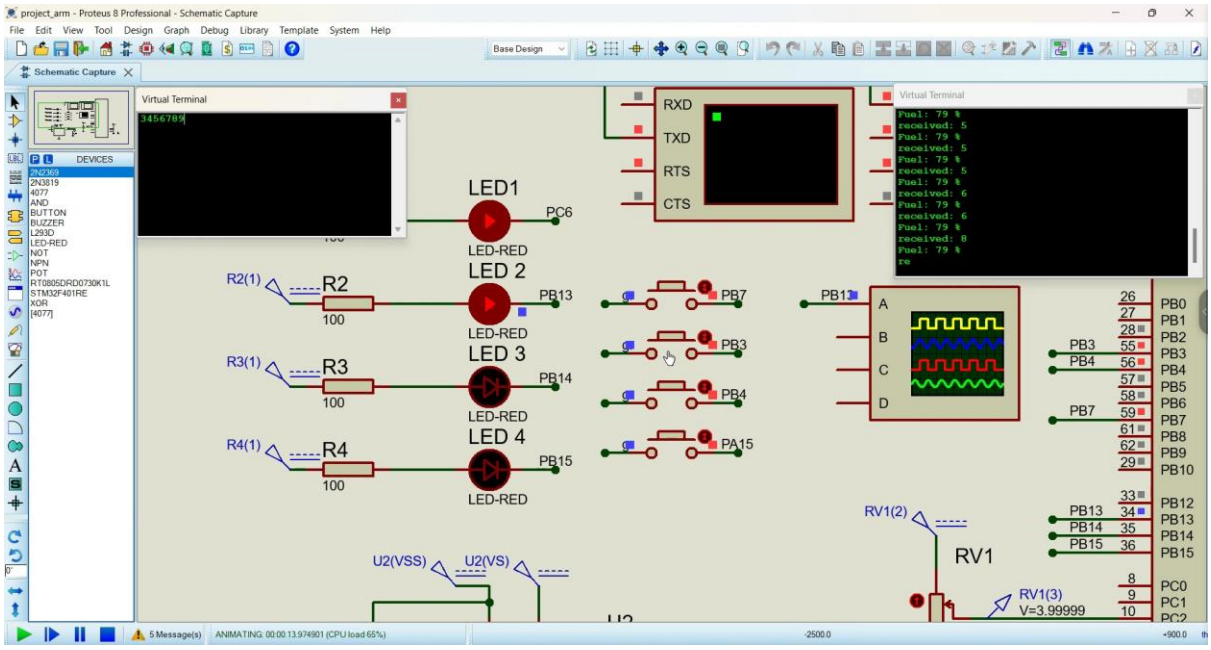


9		ST09	Press the Light Switch "3rd" time	Board Power up • Program running • Ignition LED ON • ADC Value & UART shown on Debug Live Expression window • High Beam Light ON	The Parking Light to be activated i.e. Left & right Indicator LED & Buzzer to blink with 2Hz frequency		Not Tested

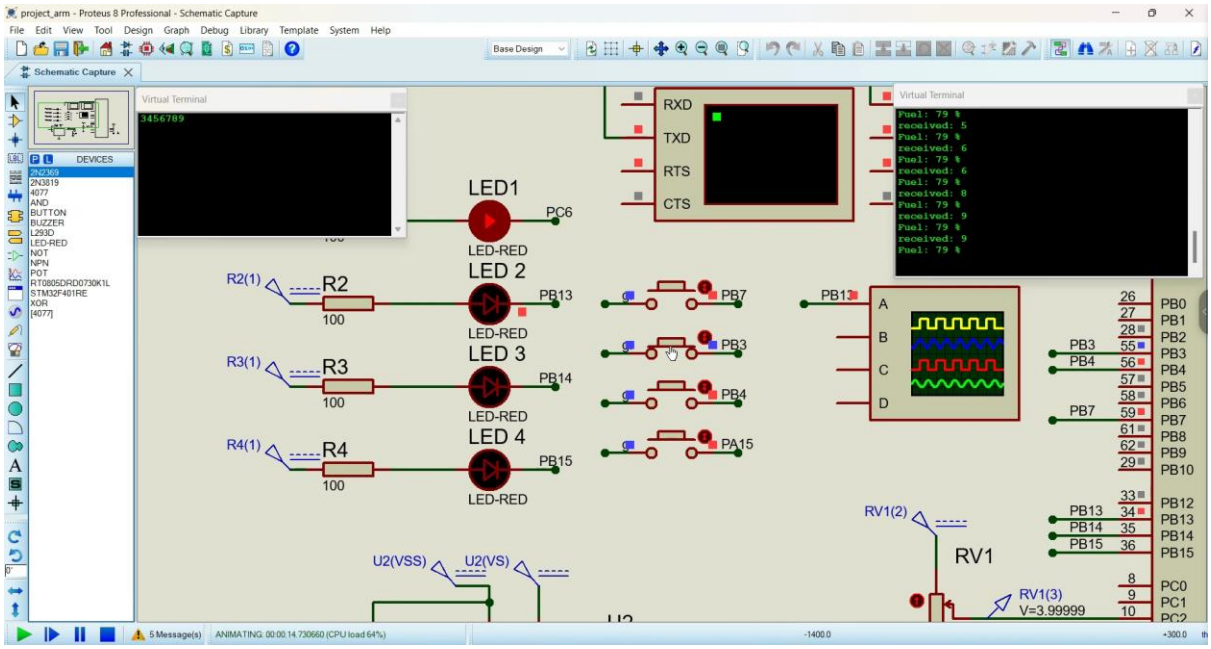
Light switch is pressed 3rd time at ANIMATING: 00:00:13.773246 (CPU load 49%)



After 500ms, at ANIMATING: 00:00:13.974901 (CPU load 65%), parking light is turned off.

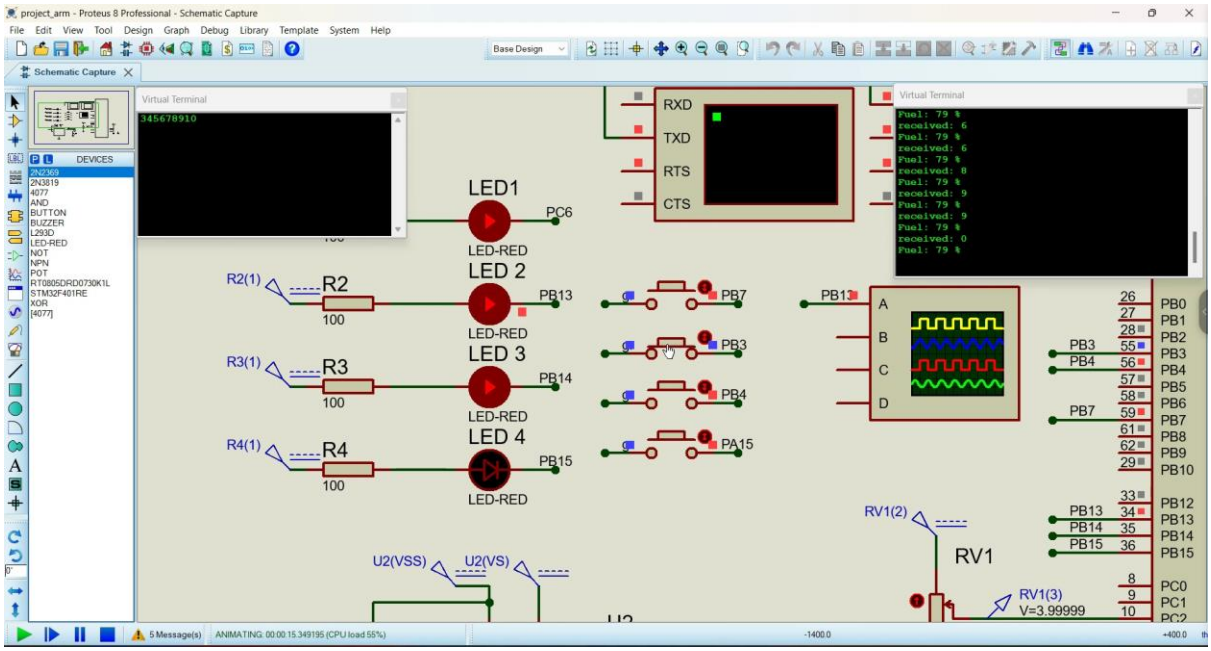


10		ST10	Press the Light Switch 4th time	Board Power up • Program Running • Ignition LED ON • ADC Value & UART shown on Debug Live Expression window • Parking LEDS & Buzzer ON	The Parking Light to be deactivated i.e. Left & right Indicator LED & Buzzer to be OFF		Not Tested



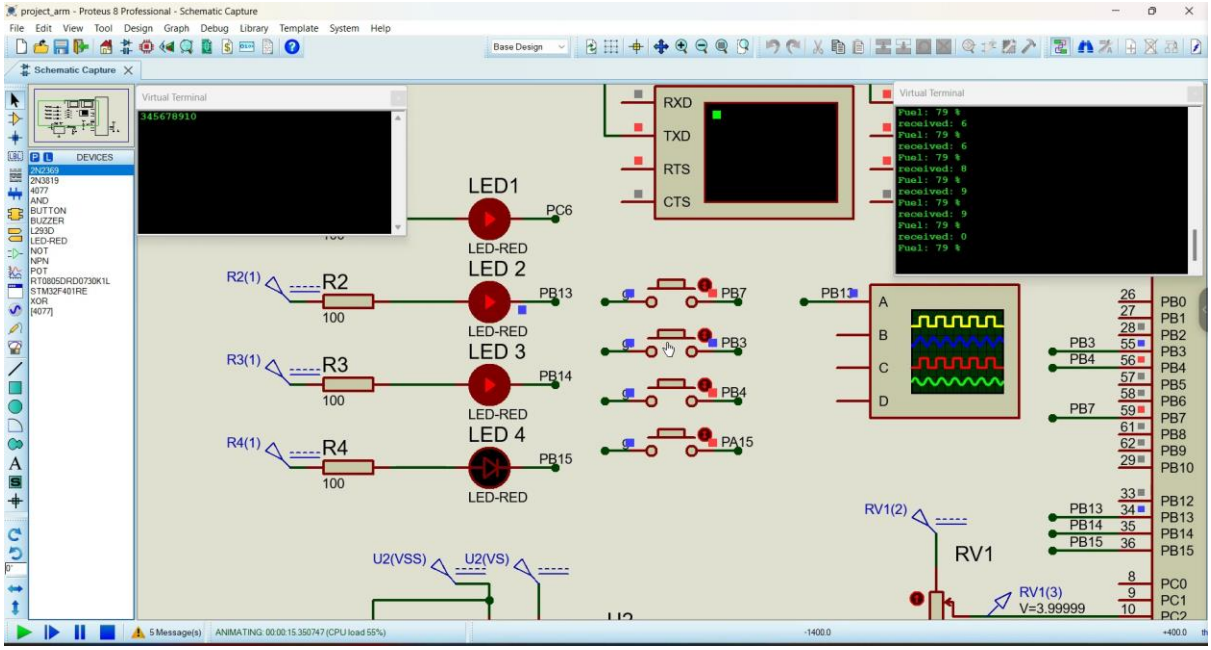
11		ST11	If the Left/Right Indicator is ON: Light Switch is pressed 3rd Time	Board Power up • Program Running • ADC Value & UART shown on Debug Live Expression window • Left/Right indicator LED & Buzzer to blinking with 0.5Hz frequency	The Indicator status LED & Buzzer continues to work & Parking doesn't get activated		

At ANIMATING: 00:00:15.349195 (CPU load 55%) switch 2 is pressed 2nd time.



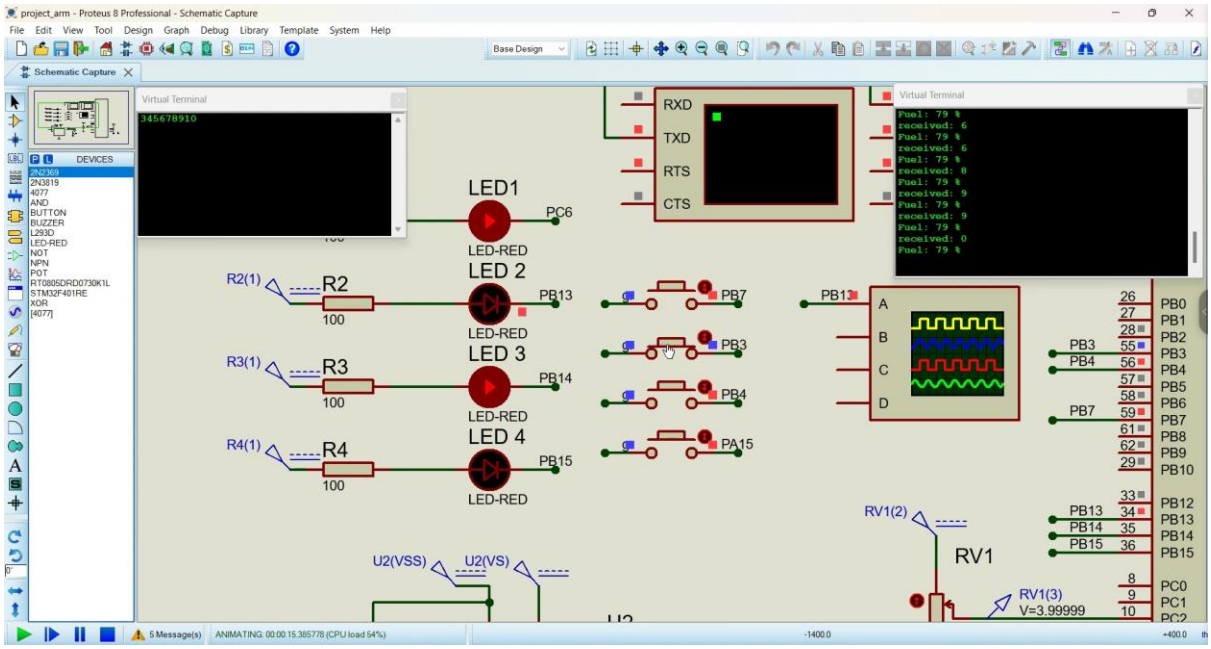
At 

switch 2 is pressed 3rd time.



At 

, switch 2 is pressed 4th time.



When indicator is turned on, the parking light cannot affect to the indicator function.