# IMPLEMENTATION OF CAR SPEED CONTROL USING MSP430 FR4133 LAUNCHPAD DEVELOPMENT KIT.

**UNIVERSITY :** SRH HOCHSCHULE HEIDELBERG.

**NAME :** CHETHAN MEDAHALLI NARAYANAMURTHY.

**DEGREE :** MASTER OF ENGINEERING IN INFORMATION TECHNOLOGY.

**MATRICULATION NUMBER :** 11012471.

**UNDER THE GUIDANCE OF :** PROF. VIMALA BAUER.

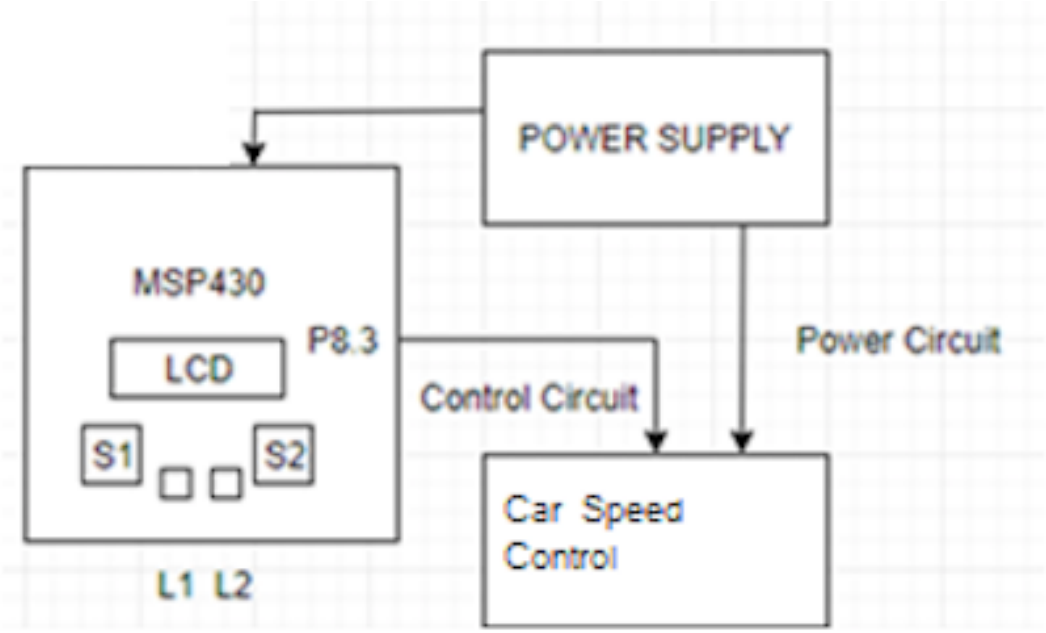
## ABSTRACT

Road safety is an issue of concern to all authorities. Excessive speed, resulting in collisions, injury, death, as well as damage to public property is one of the most important safety issues. Therefore excessive speed control is necessary and critical for preventing and fighting in the municipality's arsenal. In this project we control the car speed using MSP430 FR4133 Launchpad Development Kit.

## IMPLEMENTATION

In this project we are using MSP430 FR4133 to increase and decrease car speed by using its GPIO pins and PWM pin as output pin (P1.7). The implementation can be done manually, by pressing button S1 (SPEED INCREASES) and S2 (SPEED DECREASES). In this process we use debouncing technique, Interrupts and Timers (A0 and A1).

## BLOCK DIAGRAM




## ALGORITHM

1. Initially, Car is OFF. Display on LCD ‘0’.
2. Display on LCD: Switch on Car  Press S1 to increase and S2 to decrease speed.
3. Press S1 to increase the speed from 0 to 3. Once it reaches the speed 3, it displays “MAXIMUM SPEED REACHED” on LCD.
4. Press S2 to decrease the speed from 3 to 0. Once it reaches the speed 0, it displays “STOPPED” on LCD.

## APPENDIX:

1. Open the main.c file and include the files to project.

```
#include "lcd.h"
#include "driverlib.h"
```

2. Assign a count value to hold the count and to increment and decrement.

```
volatile unsigned int counter = 0;
unsigned long counter1 = 0;
```

3. Set S1 and S2 buttons debounce flag's:

```
//In running time to vary the battery backup memory variable to track status through LPM3.5
volatile unsigned char *S1buttonDebounce = &BAKMEM2_L; //S1 button debounce flag
volatile unsigned char *S2buttonDebounce = &BAKMEM2_H; //S2 button debounce flag
```

4. Main function:
   - ❖ Stop Watchdog Timer.
   - ❖ Initialize GPIO.
   - ❖ Display count on LCD.
   - ❖ Set debounce for S1 and S2 to '0'.
   - ❖ Setting the PWM period in Timer A1 Capture/Compare 0 register to 1000us.
   - ❖ Using ACLK, Up mode, clear TAR.
   - ❖ Enabling interrupts globally and configure Low Power Mode (LPM3_bits).

```
int main(void)
{
    WDT_A_hold(WDT_A_BASE); //Stop Watchdog timer

    initGPIO(); //Initialize gpio pins
    LCD_init(); //Initialize LCD
    LCD_displayNumber(counter1);

    *S1buttonDebounce = *S2buttonDebounce = 0; // S1 S2 Debounce
    TA1CCR0 = 1000-1; //Set the PWM Period in the Timer A1 Capture/Compare 0 register to 1000us.
    TA1CTL = TASSEL__ACLK | MC__UP | TACLR;//ACLK, up mode, clear TAR
    __bis_SR_register(GIE + LPM3_bits); // Enable interrupts globally
    return 0;
}
```

5. Use interrupts and set the functionality for S1, also we include debouncing.
   - ❖ Port 1 ISR (Interrupt Service Routine).
   - ❖ Using Switch case to toggle button 1 (Pin 2).
   - ❖ When button is pressed, count is displayed.

- Initially the count is “0” → Car is OFF.

```
#pragma vector=PORT1_VECTOR
__interrupt void pushbutton_ISR(void)
{
    switch (__even_in_range(P1IV, P1IV_P1IFG7))
    {
    case P1IV_NONE:
        break;  // None

    case P1IV_P1IFG0:
        __no_operation();
        break;  // Pin 0

    case P1IV_P1IFG1:
        __no_operation();
        break;  // Pin 1

    case P1IV_P1IFG2:   // Pin 2 (button 1)
        if((*S1buttonDebounce) == 0)
        {
            *S1buttonDebounce = 1; //High to low transition
            counter = 0;
        }
        S1();
        break;

    case P1IV_P1IFG3:
            __no_operation();
            break;  // Pin 3

    case P1IV_P1IFG4:
            __no_operation();
            break;  // Pin 4

    case P1IV_P1IFG5:
            __no_operation();
            break;  // Pin 5

    case P1IV_P1IFG6:
            __no_operation();
            break;  // Pin 6

    case P1IV_P1IFG7:
            __no_operation();
           break;   // Pin 7

    default:
       _never_executed();
    }
}
```

6. Use interrupts and set the functionality for S2, also we include debouncing.
   - ❖ Port 2 ISR (Interrupt Service Routine).
   - ❖ Using Switch case to toggle button 2 (Pin 6).
   - ❖ When button is pressed, count is displayed.

```
#pragma vector=PORT2_VECTOR
__interrupt void pushbutton_ISR2(void)
{
    switch (__even_in_range(P2IV, P2IV_P2IFG7))
    {
    case P2IV_NONE:
        break;  // None

    case P2IV_P2IFG0:
        __no_operation();
        break;  // Pin 0

    case P2IV_P2IFG1:
        __no_operation();
        break;  // Pin 1

    case P2IV_P2IFG2:
       __no_operation();
        break;  // Pin 2

    case P2IV_P2IFG3:
        __no_operation();
        break;  // Pin 3

    case P2IV_P2IFG4:
        __no_operation();
        break;  // Pin 4

    case P2IV_P2IFG5:
        __no_operation();
        break;  // Pin 5

    case P2IV_P2IFG6:   // Pin 6 (button 2)
        if((*S2buttonDebounce) == 0)
                {
                    *S2buttonDebounce = 1;   // First high to low transition
                    counter = 0;
                }
                S2();
             break;

    case P2IV_P2IFG7:
             __no_operation();
           break;   // Pin 7

    default:
       _never_executed();
    }
}
```

9. Initializing GPIO pins.

```
void initGPIO(void)
{
    // Set pin P1.0 to output direction and turn LED off
    GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0); // Red LED (LED1)
    GPIO_setOutputLowOnPin(GPIO_PORT_P1, GPIO_PIN0);

    GPIO_setAsOutputPin(GPIO_PORT_P4, GPIO_PIN0); // Green LED (LED2)
    GPIO_setOutputLowOnPin(GPIO_PORT_P4, GPIO_PIN0);

    // Unlock pins (required for FRAM devices)
    PMM_unlockLPM5();  //copy all the pin values when it go to LPM

    // Set P1.2 as input with pull-up resistor (for push button S1)
    // Configure interrupt on low to high transition and then clear flag
    // and enable the interrupt
    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P1, GPIO_PIN2);
    GPIO_selectInterruptEdge(GPIO_PORT_P1, GPIO_PIN2, GPIO_LOW_TO_HIGH_TRANSITION);
    GPIO_clearInterrupt(GPIO_PORT_P1, GPIO_PIN2);
    GPIO_enableInterrupt(GPIO_PORT_P1, GPIO_PIN2);

    // Set P2.6 as input with pull-up resistor (for push button S2)
    // Configure interrupt on low to high transition and then clear flag
    // and enable the interrupt
    GPIO_setAsInputPinWithPullUpResistor(GPIO_PORT_P2, GPIO_PIN6);
    GPIO_selectInterruptEdge(GPIO_PORT_P2, GPIO_PIN6, GPIO_LOW_TO_HIGH_TRANSITION);
    GPIO_clearInterrupt(GPIO_PORT_P2, GPIO_PIN6);
    GPIO_enableInterrupt(GPIO_PORT_P2, GPIO_PIN6);
    displayScrollText("PRESS S1 TO INCREASE AND S2 TO DECREASE THE CAR SPEED");
}
```

10.The logic behind S1

- Displays count value on LCD.
- If the count is '3' then it displays **"MAXIMUM LIMIT"** else it will increment the count by 1.
- We set P1.7 as our PWM output.
- Make CCR1 set/reset and adjust the PWM duty cycle as its half the time, which translates 50% duty cycle.
- As it Increments, LCD will display **"SPEED INCREASING".**
- In the end, we clear the interrupt flag.

```
void S1(void)
{
    LCD_displayNumber(counter1);
    GPIO_toggleOutputOnPin(GPIO_PORT_P1, GPIO_PIN0);

    if(counter1 == 3)
    {
        displayScrollText("LIMIT REACHED");
        return;
    }
    counter1 = counter1 + 1;

        TA1CCR1 = counter1 * 10;// CCR1 PWM duty cycle.
                                //It is half the time, which translates to 50% duty cycle
        TA1CTL = TASSEL_2 | MC_1;
        TA1CCTL1 = OUTMOD_7; // CCR1 reset/set

    P1DIR |= BIT7;  // Setting P1.7 as PWM output
    P1SEL0 |= BIT7; // P1.7 output PWM

    displayScrollText("INCREASING SPEED");
    LCD_displayNumber(TA1CCR1/10);

    GPIO_clearInterrupt(GPIO_PORT_P1, GPIO_PIN0);
}
```

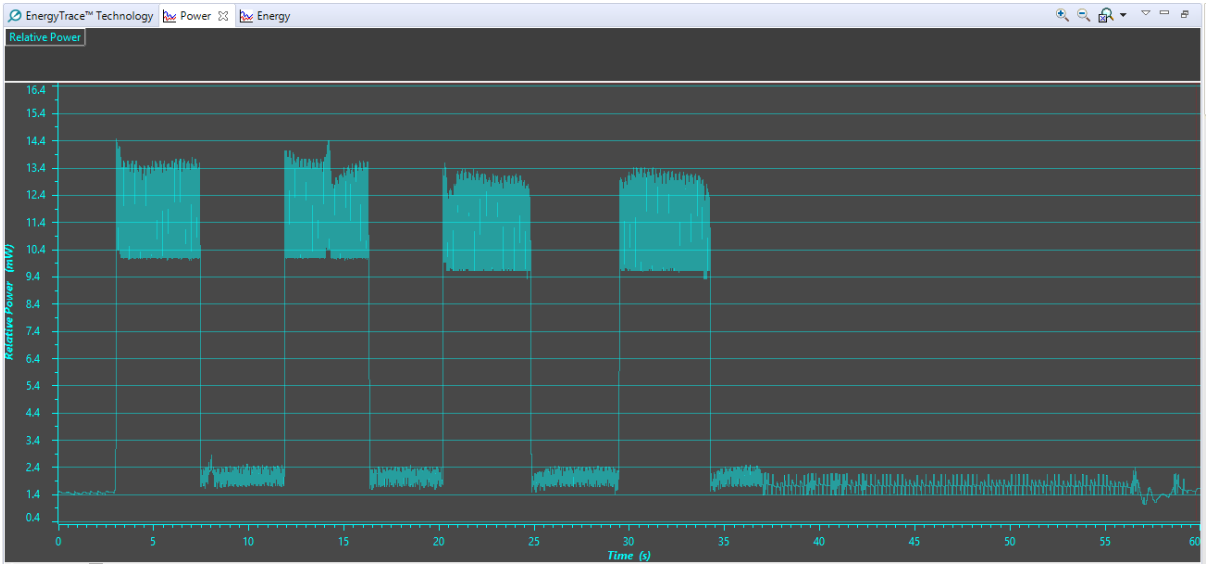
11. The logic behind S2

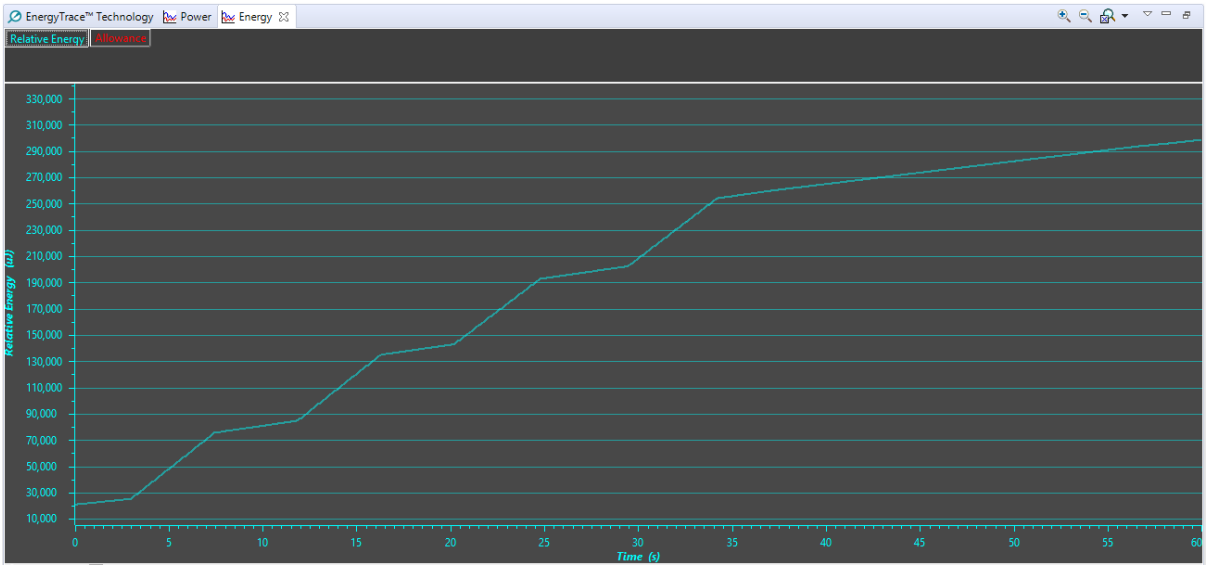
- Displays count on LCD.
- If the count is '0' then it displays **"STOPPED"** else it will decrement the count.
- We set P1.7 as our PWM output.
- Make CCR1 set/reset and adjust the PWM duty cycle as its half the time, which translates 50% duty cycle.
- As it decrements, LCD will display **"SPEED DECREASING".**
- In the end, we clear the interrupt flag.

```
void S2(void)
{
    LCD_displayNumber(counter1);
    GPIO_toggleOutputOnPin(GPIO_PORT_P4, GPIO_PIN0);

    if(counter1 == 0)
    {
        displayScrollText("STOPPED");
        return;
    }
    counter1 = counter1 - 1;

        TA1CCR1 = counter1 * 10;//PWM DUITY CYCLE 50%
        TA1CTL = TASSEL_2 | MC_1;
        TA1CCTL1 = OUTMOD_7;

    P1DIR |= BIT7; // Setting P1.7 as PWM output
    P1SEL0 |= BIT7;

    displayScrollText("SPEED DECREASING");
    LCD_displayNumber(TA1CCR1/10);

    GPIO_clearInterrupt(GPIO_PORT_P2, GPIO_PIN6);
}
```

10. The main.c file is built and then debugged in CCS.

11. Energy and Power Trace:




## CONCLUSION

I conclude that this project is working as per the design and logic implemented. I have used debouncing, Interrupts, Timers A0 & A1 and PWM to build the project.

## REFERENCES


1. Slides provided by Prof. Vimala Bauer (GPIO, Interrupts, Timers, PWM, Clocks).
2. Texas Instruments, "MSP430FR4133 LaunchPad™ Development Kit User's Guide," January 2018.
3. MSP430FR4133 - Technical Document.
4. MSP430FR2xx_4xx_ – DriverLib User Guide.