T.C.
YEDİTEPE UNIVERSITY

ELECTRICAL & ELECTRONICS ENGINEERING DEPARTMENT

TERM PROJECT

EE242
Microprocessor Systems

Erdinç Özer #20160701102
Mahmud Sami Ünlü #20160701119

DIGITAL ENGINE CONTROL
ECU (Electronic Control Unit)

Purposes:

- **Changing the Ignition Control of Engine Mechanical to Digital**
- **Controlling Gas Throttle of Engine with Servo Motor**
- **Controlling Gas Throttle with UART**
- **Stop Engine with Potantiometer or UART**

Challenges:

- **Servo Motor controler code challenge;**

1) We used Sub Main Clock (SMCLK) because we know Sub Main Clock run with 1MHz frequency.
(TA0CTL = TASSEL_2|ID_3|MC_2|TACLR; // SMCLK, up mode)

2) First We tried and find ADC10MEM and Duty Cycle limits .
ADC10MEM is the value limits of potantiometer between 0 and 1024.
Duty cycle value limits is between 700 and 2500.
However, we were need to 30° angle and we made calculation writing code of 30° angle on duty cycle.
(TA0CCR1 = ((float)ADC10MEM/3.4133)+800; // for maping 1200 between 800 for 30 degree)

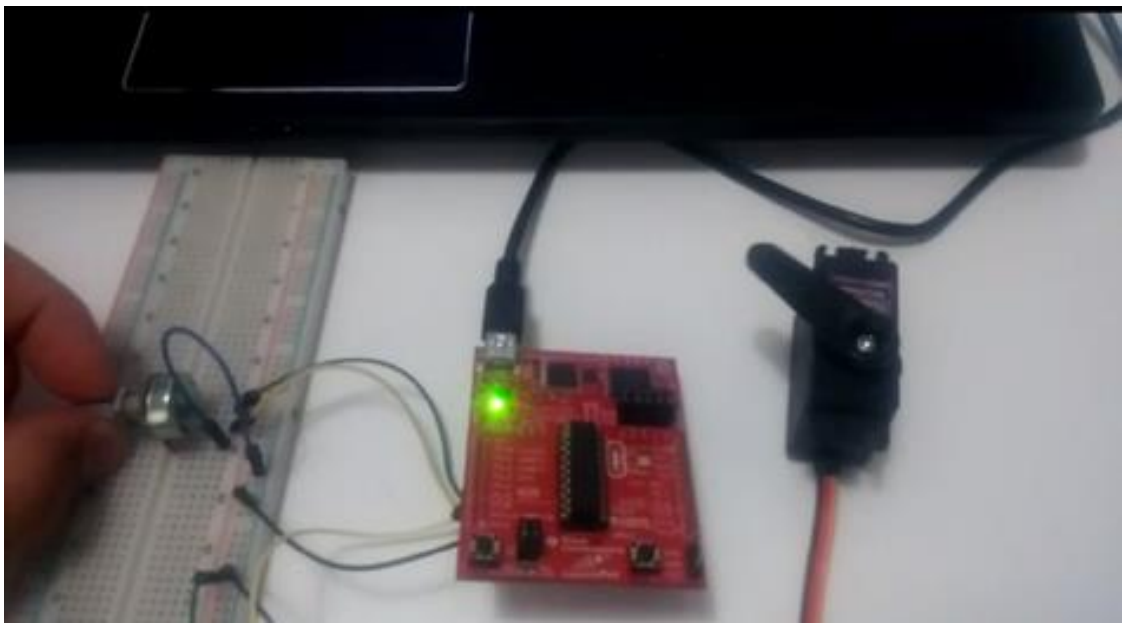

Figure 1: Servo Engine Control Challenge

- **Crank Shaft Positioning Problems**

1) We use 2 stroke gasoline engine and the problem is to determine ignition timing synchronization. To fix this problem we measured ignition time at the oscilospe

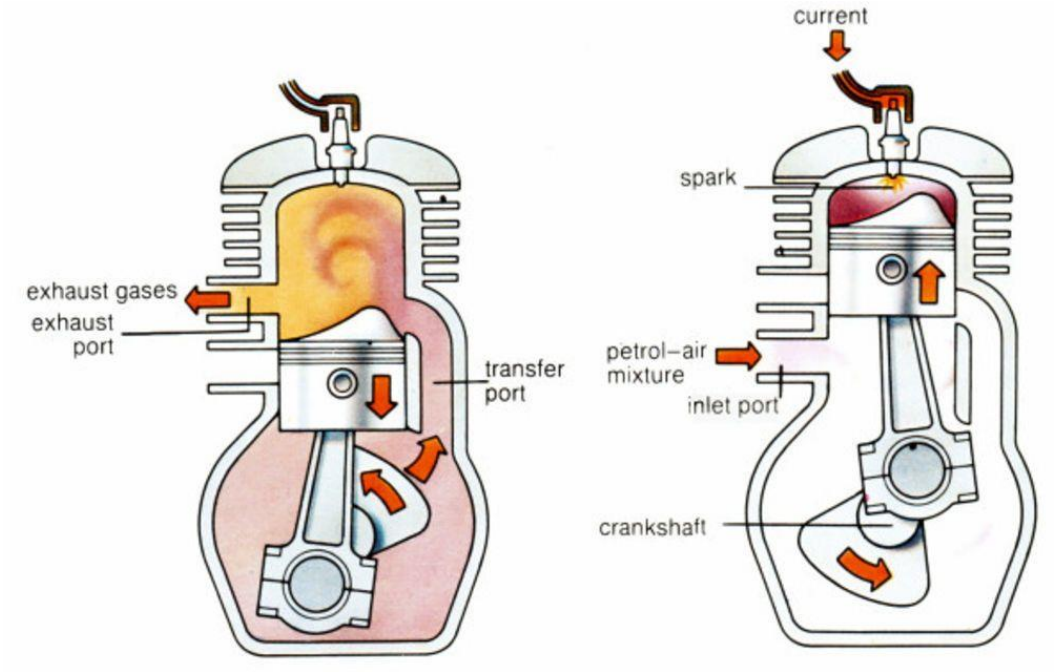

Figure 2 : 2 Stroke Engine

Figure 3 : CH1 Hardware Igniton Timing measurement

We devolop algortihm to ignite at the right time. At steady state situation the engine runs at 50 Hz 20ms each crank cycle.

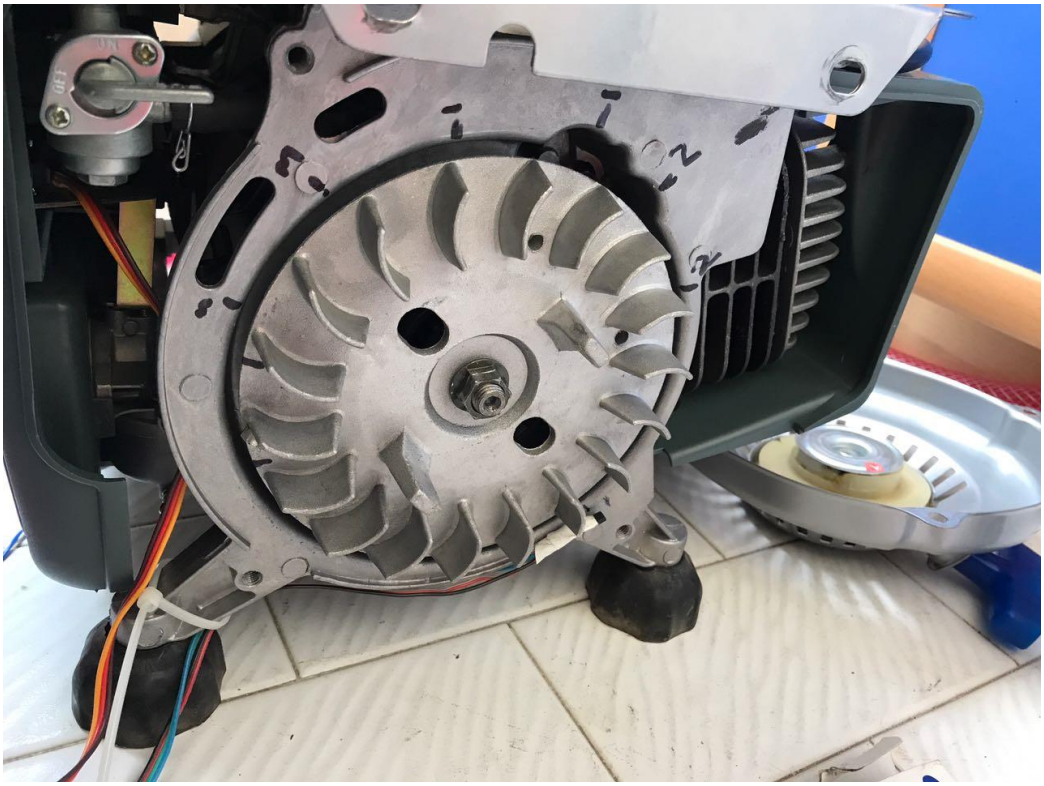

Figure 4 : Cooling Fan connected to Crank Shaft

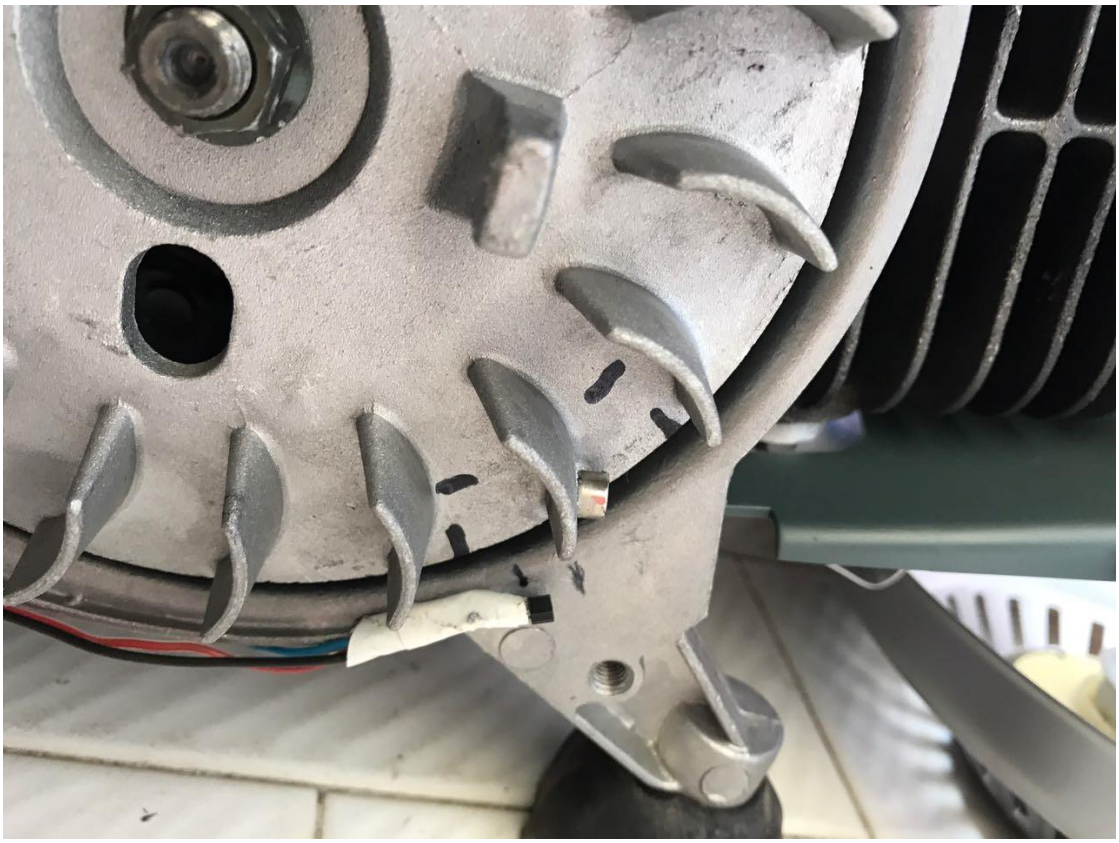

Figure 5 : Hall Effect Sensor, and magnet. Determine the RPM

Figure 6 : Crank Shaft timing tests

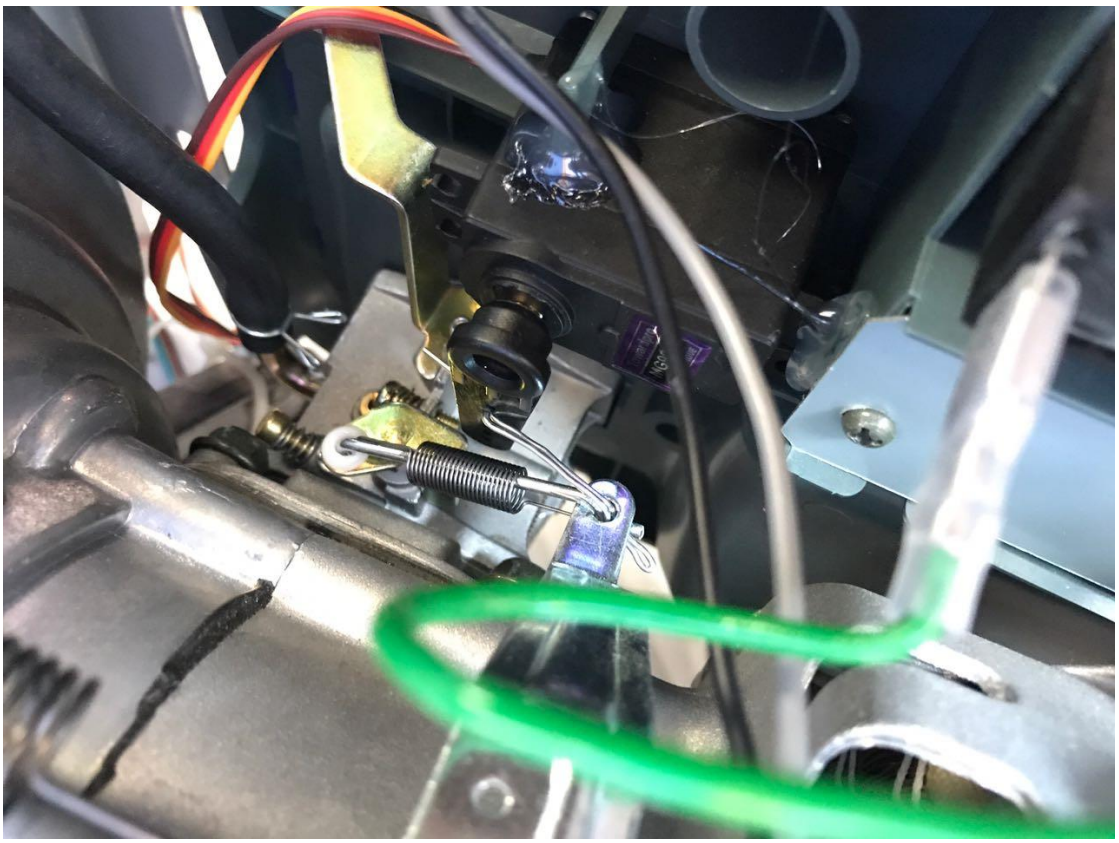

Figure 7 : Throttle Control with Servo Motor

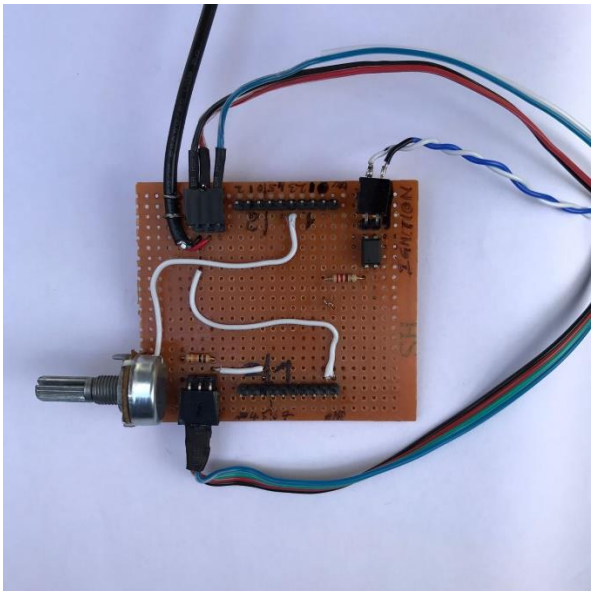

Figure 8 : ECU Shield for MSP430 Launchpad

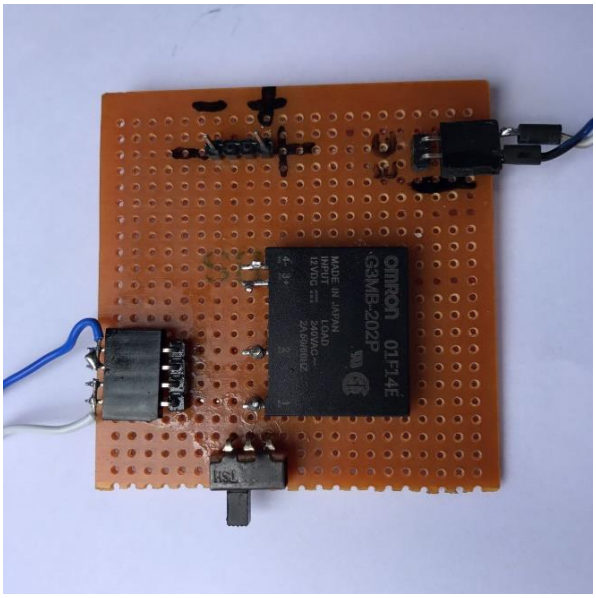

Figure 9 : Ignition Shield

ECU Shield

Contains optocoupler, potentiometer and low pass filters.
This shield have Analog input, Servo PWM output, RPM Sensor input and Ignition output.
Blue White wire connect to ECU Shield to IGNITION Shield.

Ignition Shield

We control the igniton with solid state relay.
We first try mechanic relay but the mechanical relay cannot response behind 15ms so we try with solid state and succed.

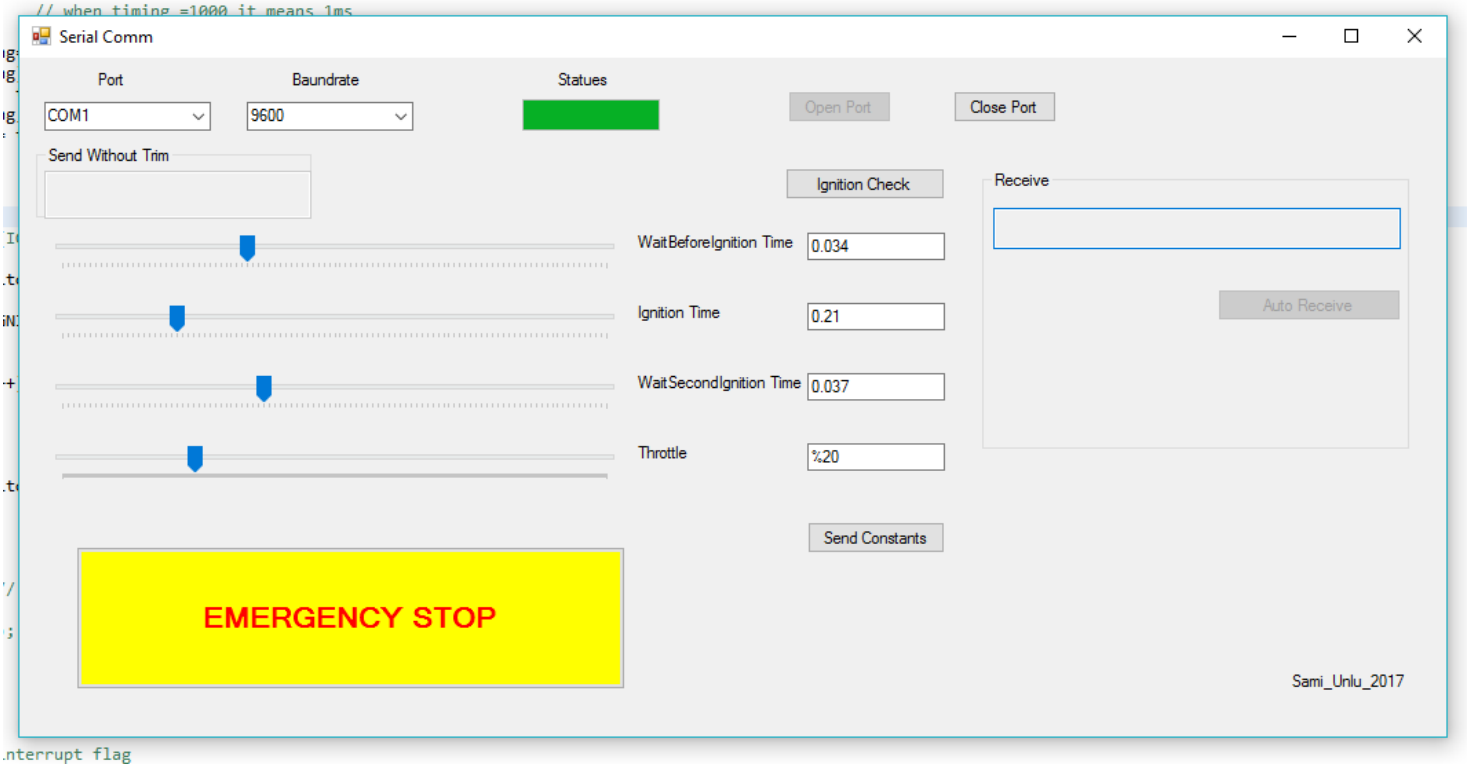

Figure 10 : C# UART ECU Cycle Timing optimization

Distribution Of Duties

Erdinç Özer:

- Servo Control

- Report Writing

- Product Selection and Supply

Mahmud Sami Ünlü:

- Engine Analysis

- Crank Shaft Analysis (Oscilloscope)

- Algorithm and perfom

    - Timer
    - ADC
    - UART
    - PWM

- Circuit design and setup

- Simulation Test

- Serial C# Program

Conclusion

In this project we leanrt how to use ADC, Timers, Port Interrupt, Interrupts and UART. The whole codes wrote ourself. Project videos at Youtube

Code

```
#include <msp430.h>
/*
 *  Toplam 1 tur 60ms suruyor
 * Sensoru tam pıston en usttteyken koyduk
 * for loop 1000 e kadar sayınca 11.7 ms suruyor dalga suresı
 * for loop 500  e kadar sayınca 6.00ms  suruyor dalga suresı
 * for loop 100  e kadar sayınca 1.41ms  suruyor dalga suresı
 * Bu ayarlar elektronik röle ve kendi ateşlemesi ile çalışıyor
 * SENSOR
 * rolanti de 1 tur 18ms toplam
 * rolantide pulse 360us
 * BU AYARLARDA ROLE
 * rolanti de 1 tur 55ms toplam
 * rolantide pulse 17.28ms
 * parazit gelene kadar gecen sure 5.6ms
 *
 * ROLATI ICIN DOGRU AYAR
 * V13
 * Degiskenleri duzenledik
 * Aciklamalar konuldu
 * V14
 * Timerlar Timer0_A ve Timer1_A seklinde duzenlendi
 * !!!! #define IGNITION BIT1 //P1.1 CHANGED WITH P1.4 !!!
 * V15
 * Timer Interrupt Ayarlamaları ve UART icin pin acma islemleri
 * Servo ve potansiyomemetre eklendi
 * Uart icin pin acildi
 * V16
 * !!!! #define IGNITION BIT4 //P1.4 CHANGED WITH P1.5 !!!
 * V17
 * SensorRPM BIT2 den P2.5 tasindi
 * V19
 * SampleCycle eklendi ve bunu otomatik yapıyor ama arıza var
 * V20
 * Uart eklendi.
 * sayiyi 4 digit halinde girip entera basiyoruz ve yeni gaz ayari oluyor
 *  ayrica 3 digit 2 digit veya 1 digiti yazıp entera basinca otomatik decimal bir degere
cevirebiliyor
 * bu yaptigi islemi bir fonksiyona cevirsek iyi olur
 * uarti acmak icin UART_THROTTLE_CONTROL 1 yapmak gerekki adcyi kapatsın
 *
 * ###SIRADAKI ISLEMLER ####
 * Hıza göre gaza basabilme ayarlanıcak
 * Exponansiyel gaz deneneicek
 */

#define DEBUG 1

#define IgnitionLED BIT6       //P1.6
#define STOP_ENGINE BIT0       //P1.6
#define IGNITION BIT5          //P1.5
#define SensorRPM BIT5         //P2.5
#define Throttle_Analog BIT4    //P1.4 ADC option select
#define ServoPWM BIT1          //P2.1
```

```
#define toogleLED (P1OUT^=IgnitionLED)
#define ErrorLED BIT6          //P1.2
#define debugServoValue 400
#define UART_THROTTLE_CONTROL 1 // 1 When we want to use uart bro
// Olaki negatif gelirse diye (Daha once olmustu)
unsigned int IgnitonTime=600; // 65 125 250 de calisiyor // best 125 //2 ms //try 1000
unsigned int WaitSecondIgniton=600; // try on motor
unsigned int WaitBeforeIgniton=600;  //300 de en iyisi
unsigned int MaxTiming =1500; //For Pull start osiloscope data  //24 ms
unsigned int IGNITION_STOP=0;
unsigned int STOP =950;
unsigned int CycleSample=25; //Igniton timing cycle sample
unsigned int UART_THROTTLE=0;
int UART_ENTERED_DIGIT=0;
int UART_ARR[5];
int Decimal_Constant_Array[4]={1,10,100,1000};

int i=0;
int pulse=0;
int result = 0;
int msec = 0; //milisecond
unsigned int timing=0;
/*
 * main.c
 */
int main(void) {
   WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    BCSCTL1 = CALBC1_1MHZ; //Adjust the clock
    DCOCTL = CALDCO_1MHZ; // Uarttan gelen ayarlar

        //////////##### IGNITION TIMERS SETTING #####//////////

     TA0CTL = TASSEL_2|ID_3|MC_2|TACLR; //SMCLK kullandık Clk/8 Timer1_A  Ignition
     //TA1CCR0 = 0xFFFF; //Upper limit for count Counterın sayacagı son sayıyı yazdık

     //////////##### SERVO TIMERS SETTING #####//////////

     TA1CCR0  = 20000-1; // PWM Period
      TA1CCTL1  = OUTMOD_7; // CCR1 reset/set
      TA1CCR1 = 2017; // CCR1 PWM duty cycle
      TA1CTL = TASSEL_2 + MC_1; // SMCLK, up mode

       //////////##### PORT SETTING #####//////////

   P1SEL =  0x00; //
   P1DIR = IgnitionLED|IGNITION|ErrorLED|STOP_ENGINE;
   P1OUT = STOP_ENGINE;
//   P1SEL |= ServoPWM; // P1.2 and P1.3 TA1/2 options
   //P1OUT = IgnitionLED|IGNITION|ErrorLED;
   //P1REN=0xBC;
   P2DIR =  ServoPWM; // P1.2 and P1.3 output
   P2SEL =  ServoPWM; // P1.2 and P1.3 TA1/2 options
   P2REN =  SensorRPM;
   P2IE  =  SensorRPM;//Enable interrupt from port1
   P2IES =  SensorRPM;//SensorRPM;//Interrupt edge select from high to low
   P2IFG =  0x00;//Clear the interrupt flag
```

```
  P2OUT =  SensorRPM;

  /////////#### ADC SETTING ####/////////
 if(UART_THROTTLE_CONTROL==0){
  ADC10CTL0 = ADC10SHT_2 + ADC10ON + ADC10IE; // ADC10ON, interrupt enabled
  ADC10CTL1 = INCH_3+CONSEQ_0+ADC10SSEL_3+ADC10DIV_4; // input A3
  ADC10AE0 |= Throttle_Analog; // PA.1 ADC option select

  while (ADC10CTL1 & ADC10BUSY);   // Wait ADC until its ready to conversion
  ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
  }
 else {

    P1DIR |= 0x41; //Adjust pins
    P1SEL |= BIT1|BIT2;
    P1SEL2 = BIT1|BIT2;

    UCA0CTL1 |= UCSSEL_2; //Setup the UART mode
   //Use SMCLK
    UCA0BR0 = 104;
   //Low bit of UCBRx is 104
    UCA0BR1 = 0;
   //High bit of UCBRx is 0
    UCA0MCTL = UCBRS_1;
   //Second modulation stage select is 1
   //Baud Rate = 9600
    UCA0CTL1 &= ~UCSWRST;
   //Clear SW reset, resume operation
    IE2 |= UCA0RXIE;
   //Enable USCI_A0 RX TX interrupt

  }


  _enable_interrupts();//Enable all interrupts


//if (DEBUG)
 //  TA0CCR1 = (debugServoValue/4)+800;

   while(1){
      //  P1OUT = 0x00;
        // P2OUT |=
      SensorRPM;
        // i=0;
       //  for(i; i<IgnitonTime;i++);
   }

}


//USCI A receiver interrupt
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIA0RX_ISR(void){
  int temp_decimal=0;
  switch(UCA0RXBUF){
  case 0x0D :
```

```
      UART_ENTERED_DIGIT -= 1; //arraydeki 0. elemana ulasamiyor diye
      UART_THROTTLE=0;
      int i=0;
      int  Decimal_Constant_Array_ptr=0;
        for(i=UART_ENTERED_DIGIT; i>-1; i--){
     //when UART_ENTERED_DIGIT =5 i =0 when UART_ENTERED_DIGIT=4 i=1   tersten
saydirmak icin
           Decimal_Constant_Array_ptr=i-UART_ENTERED_DIGIT;
           if((Decimal_Constant_Array_ptr)<0)
             Decimal_Constant_Array_ptr*=-1;
    // buraya kadar hep tersten saydirmak icin 54321 ise 12345 olsun diye
            UART_THROTTLE+=
UART_ARR[i]*Decimal_Constant_Array[Decimal_Constant_Array_ptr];
        }
   // adcden gelen degerle ayni islemler bu arada UART araligi 0 ile 1024
        IGNITION_STOP= UART_THROTTLE;
          TA1CCR1 = (UART_THROTTLE/4)+1100;//800 // for maping 1200 between 800 //
For the avoid floating process

              if(pulse==0){
                CycleSample=(-1*(IGNITION_STOP-1024))/32;}
              if(IGNITION_STOP>STOP){
                  P1OUT = STOP_ENGINE;
                  }
              else
                P1OUT = P1OUT & ~STOP_ENGINE;

      UART_ENTERED_DIGIT=0;  //Carrige Return tusu yani enter
      break;
   default :

     temp_decimal= UCA0RXBUF-0x30;
     UART_ARR[UART_ENTERED_DIGIT]=temp_decimal;
     UART_ENTERED_DIGIT++;
     break;
   }

}

#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void){

   //TA0CCR1 = ((float)ADC10MEM/3.4133)+800; // for maping 1200 between 800 //Orginal
   IGNITION_STOP= ADC10MEM;
   TA1CCR1 = (ADC10MEM/4)+1100;//800 // for maping 1200 between 800 // For the avoid
floating process
        ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start

        if(pulse==0){
          CycleSample=(-1*(IGNITION_STOP-1024))/32;}
        if(IGNITION_STOP>STOP){
            P1OUT = STOP_ENGINE;
            }
        else
          P1OUT = P1OUT & ~STOP_ENGINE;
}
```

```
#pragma vector=PORT2_VECTOR
//define the interrupt vector
__interrupt void PORT2_ISR(void){
 /*  if (!(ADC10CTL1 & ADC10BUSY)){   // Wait ADC until its ready to conversion
   TA1CCR1 = (ADC10MEM/4)+800; // for maping 1200 between 800 // For the avoid floating
process
     ADC10CTL0 |= ENC + ADC10SC; // Sampling and conversion start
  } */
 i=0;
  // for(i; i<(DELAY);i++);
 if(IGNITION_STOP>STOP){
    P1OUT = STOP_ENGINE;
    }

 else{
  if(( P2IFG & SensorRPM ) == SensorRPM){
     pulse++;
     if (pulse==CycleSample){
         //TACTL |= MC_3;
        //TACTL |= TACLR;
        TA0R = 0x0000;   //TimerA0
        i=0;
        for(i; i<400;i++); // wait 4 ms for the SensorRPM oscillation

     }
     if (pulse==(CycleSample+1)){
        result = TA0R;  //TimerA0
        timing = result/4;      // simplfy calculation

         //msec = result/128;   // orginal calculation
        //timing= 32*msec;    // when timing =1000 it means 1ms
        pulse=0;
        if (timing<1) timing=1;
        if (timing>MaxTiming) timing=MaxTiming;
        WaitBeforeIgniton=  timing/2; // Biraz İlerde kalıyordu gerilettik
        IgnitonTime = timing;
        WaitSecondIgniton = timing*2; // Faz kaydırdık.
     }


     if(pulse<CycleSample){
       // CycleSample=(-1*(IGNITION_STOP-1024))/32;
     i=0;
     for(i; i<WaitBeforeIgniton;i++);  // WaitBeforeIgniton

     P1OUT = (IgnitionLED|IGNITION);

     i=0;
     for(i; i<IgnitonTime;i++);     //IgnitonTime

     P1OUT = 0x00;

     i=0;
     for(i; i<WaitSecondIgniton;i++);   //  WaitSecondIgniton

     }
  }
```

```
            else{
            P1OUT =   P1OUT|ErrorLED; // Test for other pin interrupt
             i=0;
            // for(i; i<IgnitonTime;i++);
             }
           }

            P2OUT |= SensorRPM;
            //P1OUT = 0xBC;
            P2IFG &= 0x00;//Clear the interrupt flag

}
```