KOCAELİ UNIVERSITY ENGINEERING FACULTY

DATA TRANSFER
BETWEEN 3
MSP430G2553 VİA
BLUETOOTH MODULE
HC-05

FURKAN YARDIMCI
210207089
2021-2022

DEPARTMENT: Electronic and Communacition Engineering
TEACHER: Dr.Öğr.Üyesi AYHAN KÜÇÜKMANİSA

KOCAELİ, 2021

TABLE OF CONTENTS

ABSTRACT ....
BACKGROUND...
- MSP430G2553 ....
- HC-05 And Main Logic Of The Embeded System ....
  - What Is AT Commands ....
  - AT commands that we use in our project ....
  - Main Logic ....
- FLAME SENSOR...
- MQ-4 GAS SENSOR...

SYSTEM DESİGN ....
- SCHEMATİC AND PCB...
- CODES...
  - CODES OF SLAVES...
  - CODE OF MASTER..

IMAGES OF THE SYSTEM IN OPERATING MODE ....
REFERENCES...

1.ABSTRACT:

An embedded system application that communicates 3 msp430g2553 via bluetooth modules. Using the Bluetooth module (hc-05), 2 slave and 1 master configurations are set between the msps. As a result of analog data coming from the slaves, which came from Fire sensor and MQ-4 gas sensor, the led indicators connected to the master light up.

2.BACKGROUND:

2.1 MSP430G2553:

Texas Instruments MSP430G2553 is part of the MSP430 family of ultra-low-power microcontrollers featuring different sets of peripherals. The MSP430G2553 mixed signal microcontroller features a 16-bit Semi RISC CPU, 16-bit registers, 16KB non-volatile memory, 512 bytes RAM. The G2x53 series is popular due to the low price and various package sizes. We have use 3 msp430g2553 in our project, 2 of them act as slaves and one is master.

2.2 HC-05 And Main Logic Of The Embeded System:

HC-05 Bluetooth Module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. Its communication is via serial communication which makes an easy way to interface with controller or PC. HC-05 Bluetooth module provides switching mode between master and slave mode which means it able to use neither receiving nor transmitting data. Master, slave modes and lots of modes are, arrenged with AT command of the hc-05, wich is so important for our project

a)What Is AT Commands ?

By default, the HC-05 is configured in data mode. In this mode, the module acts like a serial bridge. If you want to set an HC-05 as a master you have to put it into AT command mode. To put into AT command mode the KEY pin must be set (high). The HC-05 is now in command mode if the red LED flashes once every two seconds.

b)AT commands that we use in our project:

Set/Inquire module role:

AT+ROLE=PARAM1 → Role introduction: Slave: Passive connection; Slave-Loop: Passive connection, receive the remote Bluetooth master device data and send it back to the master device; Master: Inquire the near SPP Bluetooth slave device, build connection with it positively, and build up the transparent data transmission between master and slave device. Default: 0

Param1: module role: 0 -> Slave
1 -> Master
2 -> Slave-Loop

Set/Inquire connection mode:

AT+CMODE=Param1→ Param1: Connection mode: 0 -> Connect the module to the specified Bluetooth address. (Bluetooth address can be specified by the binding command)
1 -> Connect the module to any address (The specifying address has no effect for this mode.)
2 -> Slave-Loop
Default connection mode: 0

Set/Inquire - bind Bluetooth address:

AT+BIND=Param1 → Param1: Bluetooth address: needed to be bind

Connect Device :

AT+LINK=Param1 → Param1: Bluetooth address of remote device

AT commands of master hc-05 is:

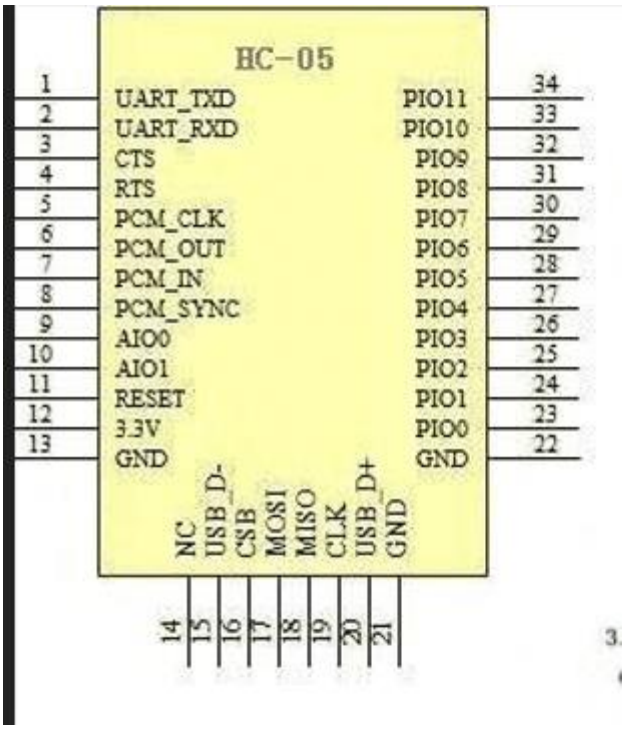
AT+ROLE=1 AT+CMODE=0

AT commands of slaves is:

AT+ROLE=0 AT+CMODE=1;

c)Main Logic:

In our project we have 2 slaves and 1 master but hc-05 is support just one slave connection to the master at a time. As a solution to this, the master was cycled between continuous in AT mode and data mode, and we achived this with switching the reset pin(11th pin) of Hc-05 with BC238 switch circuit. First master hc-05 is goes into the at mode, then the address of the 1st slave is bind to master, then it has reset and goes into data mode, and it is communicate with slave 1. With this loopish approach, we have able to control 2 slaves with a single master.



2.3 Flame Sensor:

The fire detector sensor card is a sensor card used to detect fire with a wavelength between 760 nm - 1100 nm. It has an IR receiver on it. It can be used as a fire detection sensor in fire extinguishing robots. Sensitivity can be adjusted with the trimpot on it and both analog and digital output can be output.

2.4 MQ-4 Gas Sensor:

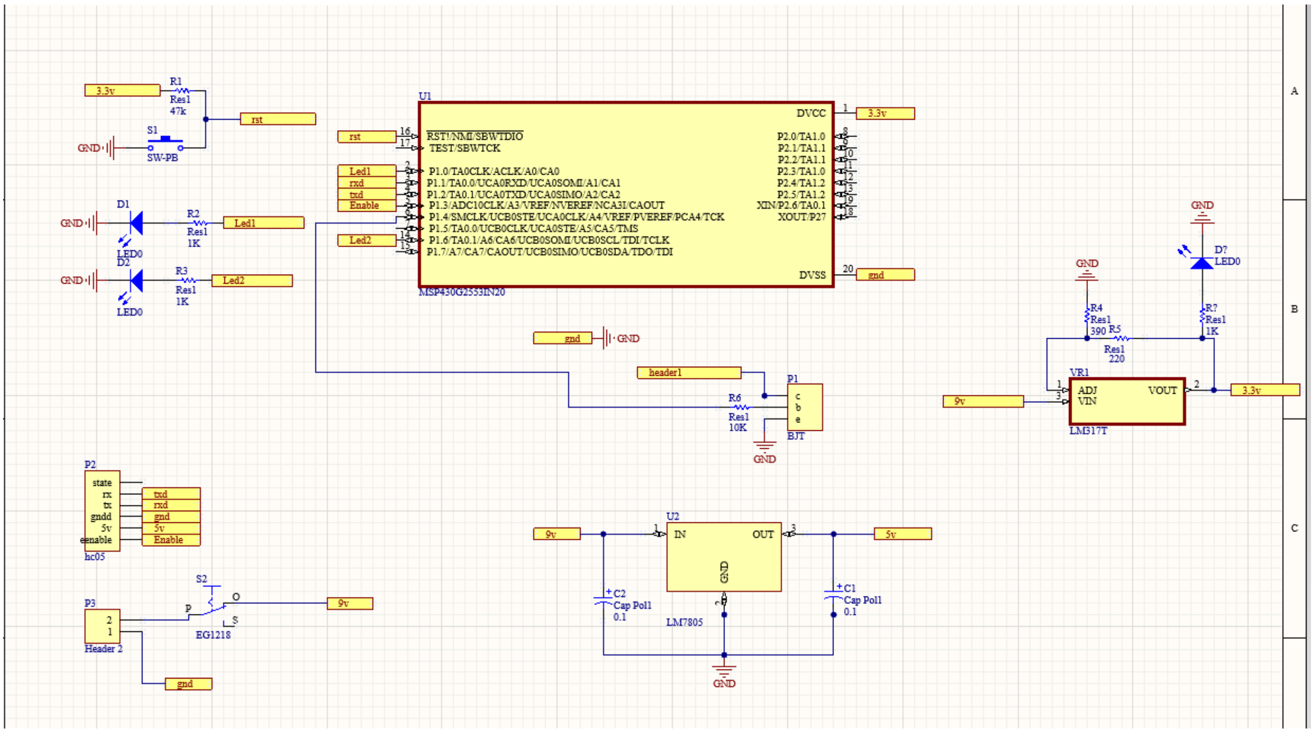
The MQ-4 methane gas sensor detects the presence of methane (CNG) natural gas in a range of concentrations suitable for gas leak detection between 300ppm and 10000ppm. Like other MQ sensors, this sensor also gives an analog voltage output according to the density of the gas. Being able to detect between 10.000ppm and 300ppm is suitable for gas leakage.

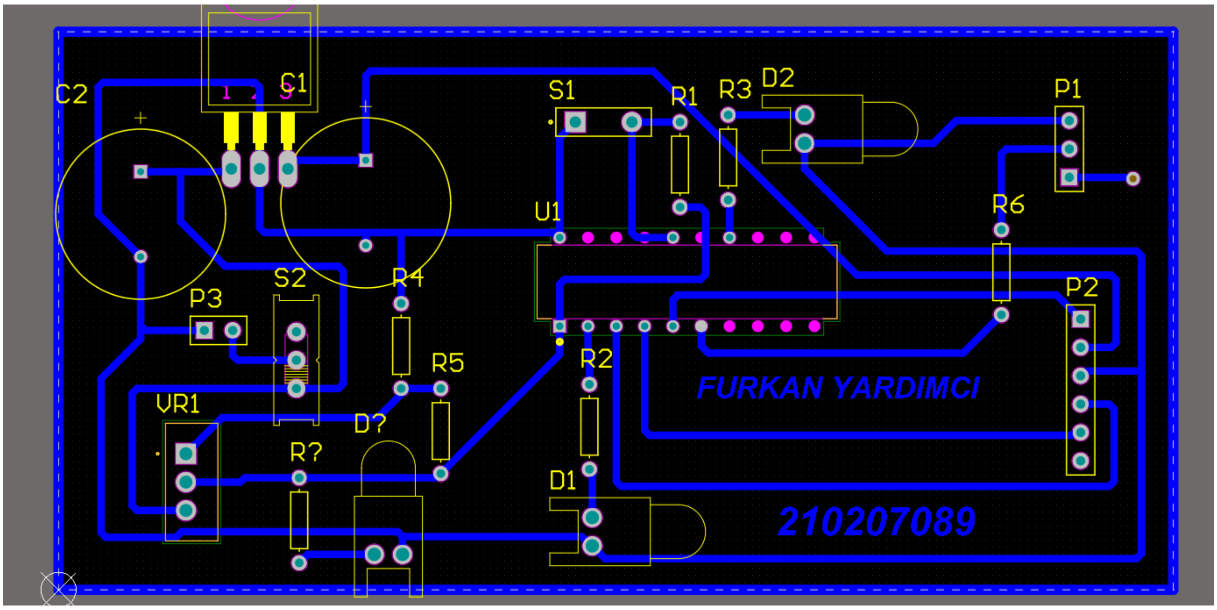
3. System Design

3.1 Schematic and PCB:

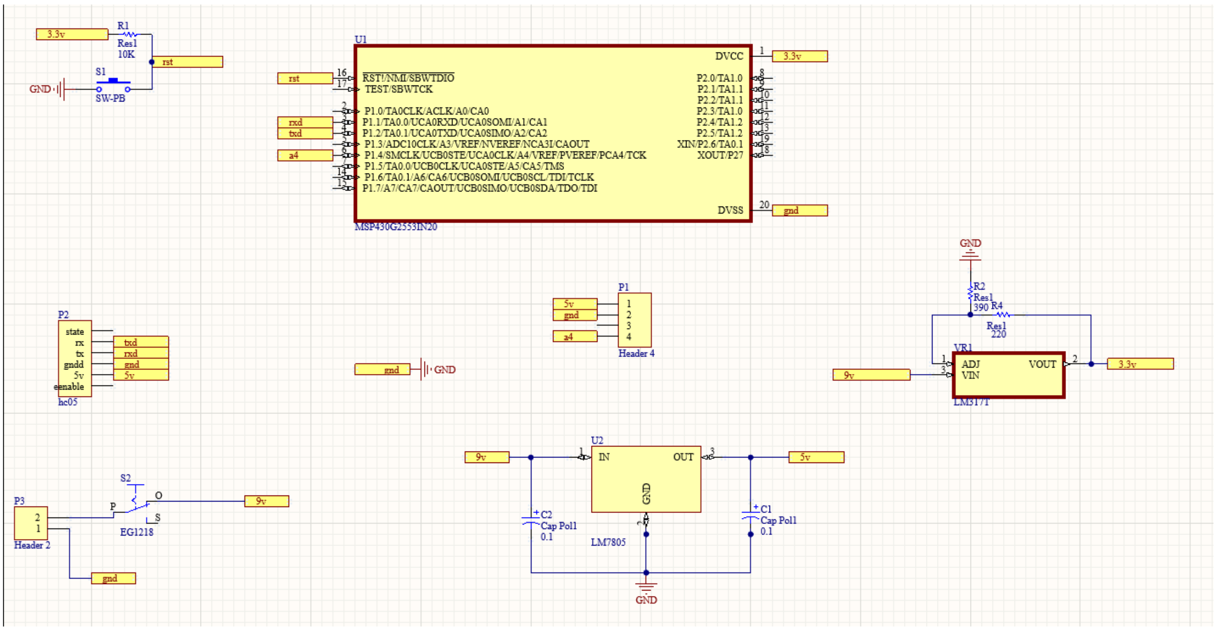
This system's schematic was designed using Altium Designer 17.0 and can be seen in the Figure below...

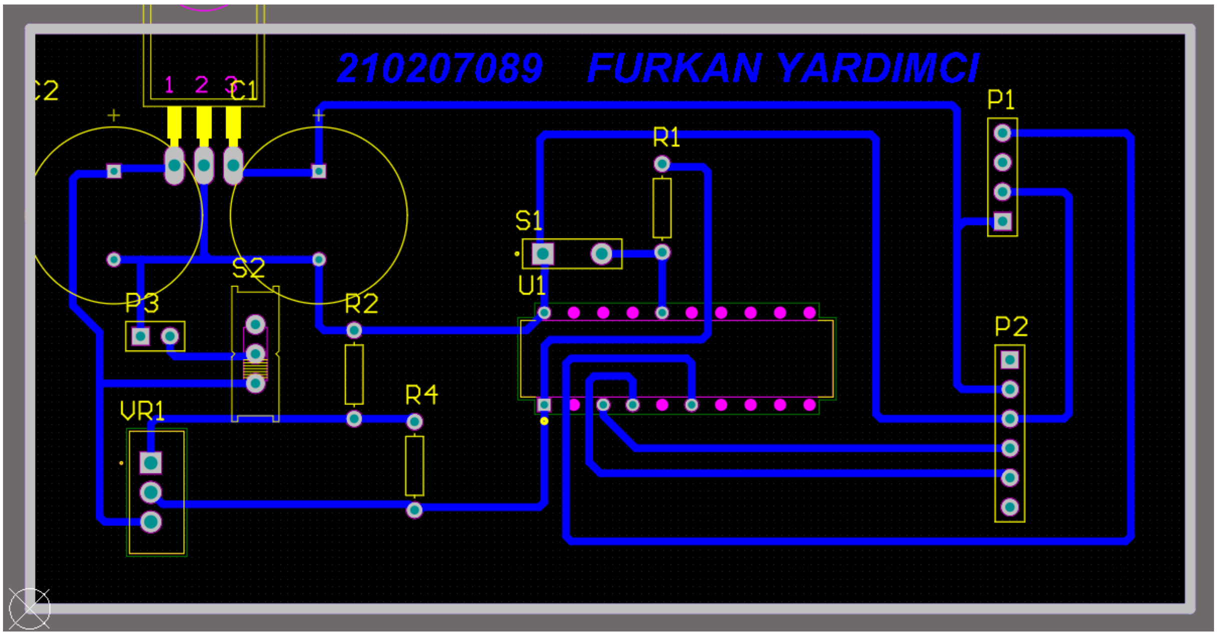
a)MASTER Circuit:



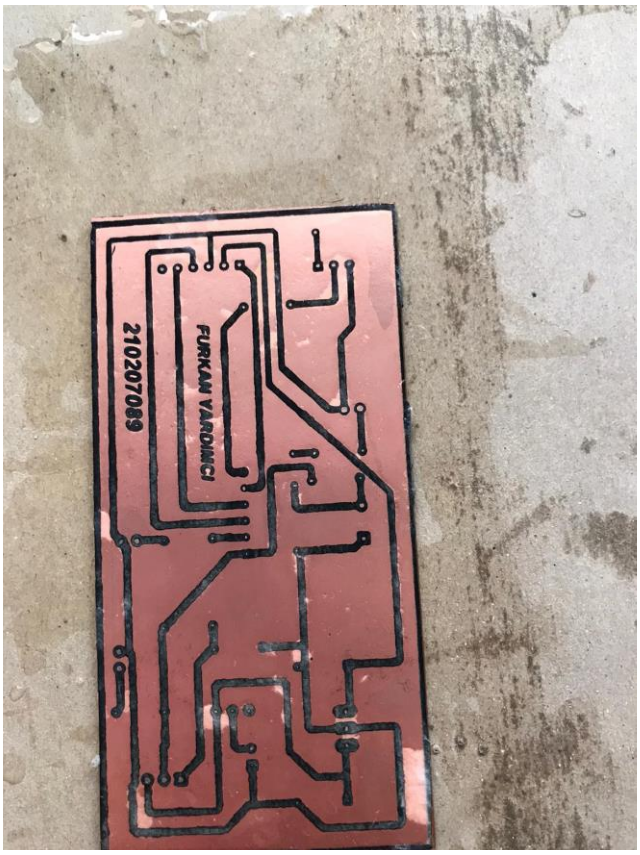


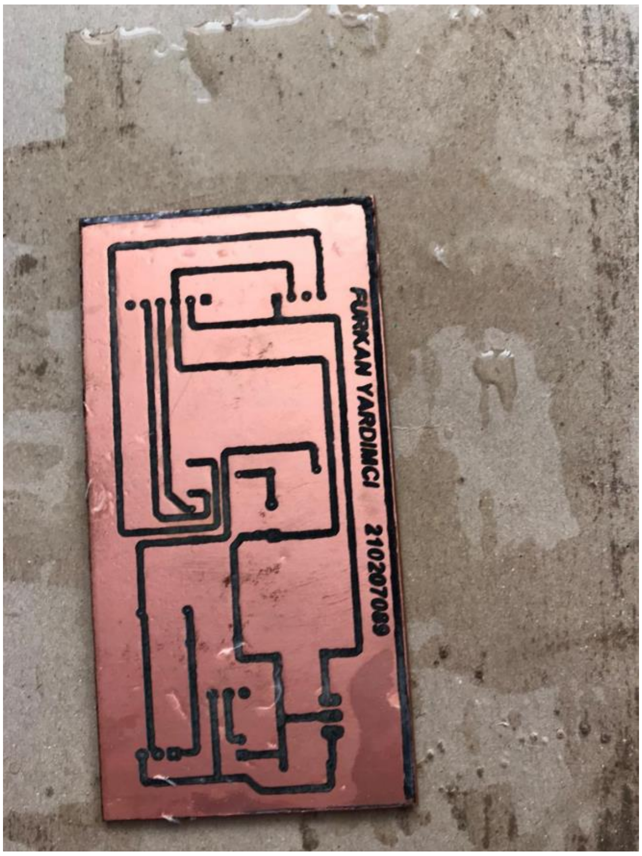
b) Slave Circuits

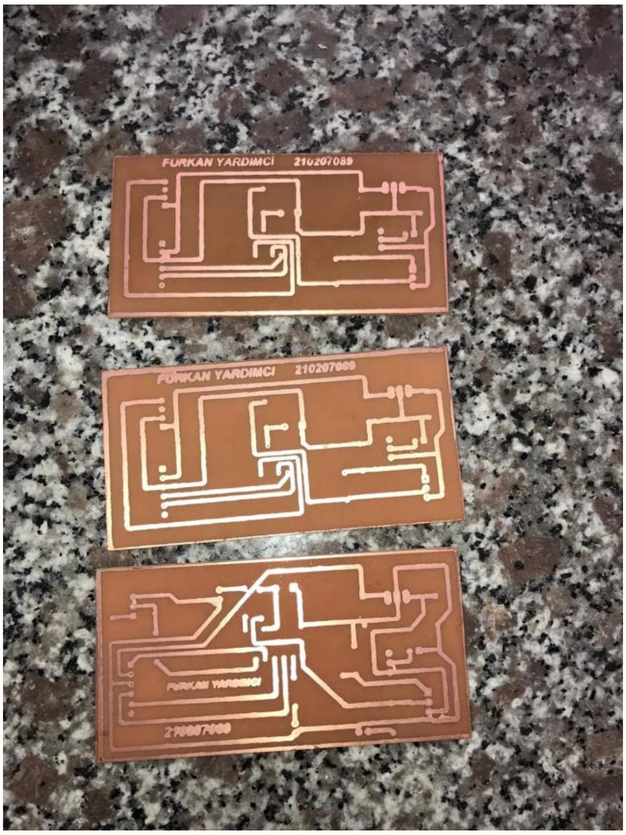


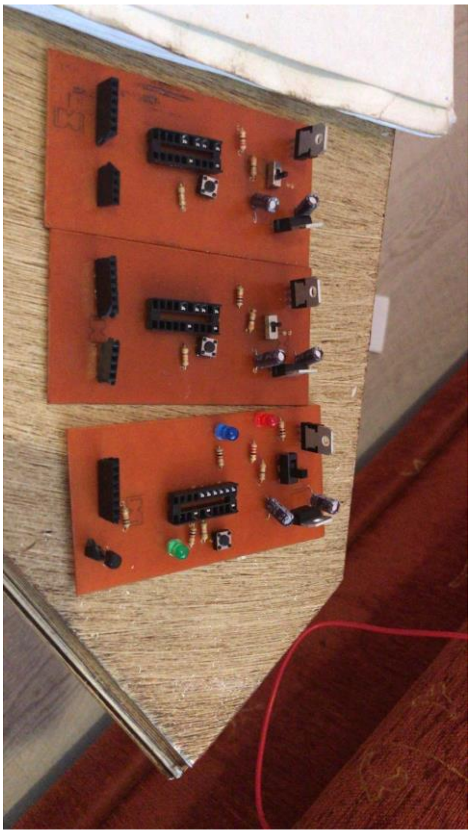


c) pcb making steps:









3.2 Codes:

This project code was written in c language and separated codes were written for master and slaves.

a.1) CODES OF SLAVES: slave codes are almost exactly the same so I'm just putting one of them in the report

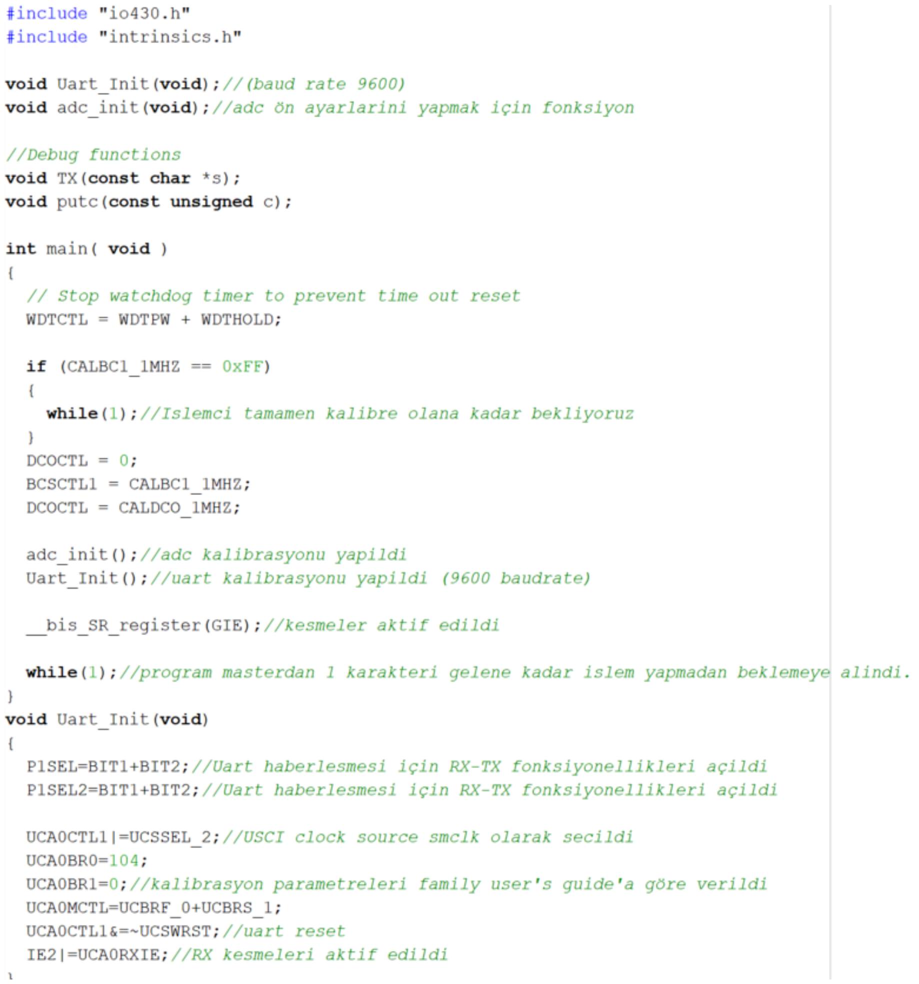
ADC Register Initializations

Register	Bit	Description
ADC10CTL0	REFON	Reference generator: ON
ADC10CTL0	SREF_1	Select reference voltage as 1.5v
ADC10CTL0	ADC10ON	ADC10 Enable: ON
ADC10CTL0	ADC10IE	ADC10 Interrupt: ENABLED
ADC10CTL0	ADC10SHT_3	Sample and holde time: 64xAdc10clk
ADC10CTL0	ENC	Enable conversion
ADC10CTL0	ADC10SC	Start Conversion
ADC10CTL1	INCH_4	Input channel A4 (P1.4)
ADC10CTL1	ADC10SSEL_0	ADC10OSC selected as a clk source

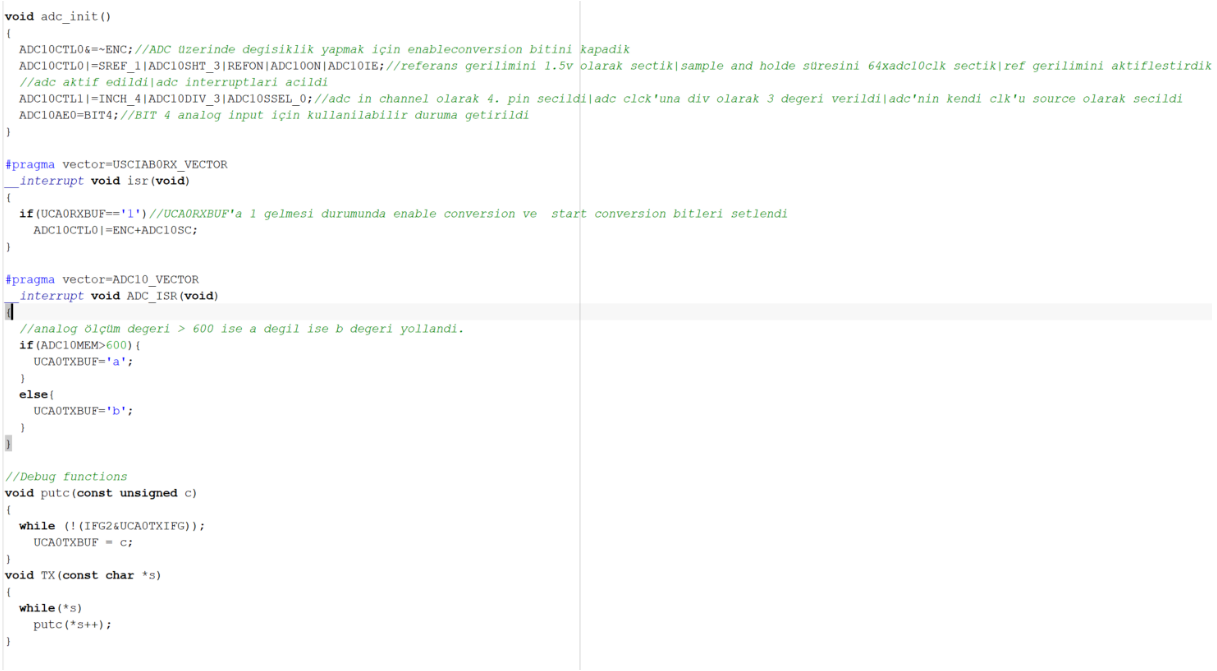
ADC10CTL1	ADC10DIV_3	ADC10OSC clk divided by 2
ADC10AE	BIT4	Analog input enable bit is set for Port 1.4

UART Register Initializations

Register	**Bit**	**Description**
P1SEL	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
P1SEL2	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
UCA0CTL1	UCSSEL_2	Set UART use SMCLK
UCA0BR0	104	1MHz 9600 baud
UCA0BR1	0	1MHz 9600 baud
UCA0MCTL	UCBRS_1	Modulation UCBRSx = 1
IE2	UCA0RXIE	Rx interrupts are enabled
UCA0CTL1	~UCSWRST	Resets are disabled



```
#include "io430.h"
#include "intrinsics.h"

void Uart_Init(void);//(baud rate 9600)
void adc_init(void);//adc ön ayarlarini yapmak için fonksiyon

//Debug functions
void TX(const char *s);
void putc(const unsigned c);

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  if (CALBC1_1MHZ == 0xFF)
  {
    while(1);//Islemci tamamen kalibre olana kadar bekliyoruz
  }
  DCOCTL = 0;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;

  adc_init();//adc kalibrasyonu yapildi
  Uart_Init();//uart kalibrasyonu yapildi (9600 baudrate)

  __bis_SR_register(GIE);//kesmeler aktif edildi

  while(1);//program masterdan 1 karakteri gelene kadar islem yapmadan beklemeye alindi.
}
void Uart_Init(void)
{
  P1SEL=BIT1+BIT2;//Uart haberlesmesi için RX-TX fonksiyonellikleri açildi
  P1SEL2=BIT1+BIT2;//Uart haberlesmesi için RX-TX fonksiyonellikleri açildi

  UCA0CTL1|=UCSSEL_2;//USCI clock source smclk olarak secildi
  UCA0BR0=104;
  UCA0BR1=0;//kalibrasyon parametreleri family user's guide'a göre verildi
  UCA0MCTL=UCBRF_0+UCBRS_1;
  UCA0CTL1&=~UCSWRST;//uart reset
  IE2|=UCA0RXIE;//RX kesmeleri aktif edildi
}
```



```
void adc_init()
{
  ADC10CTL0&=~ENC;//ADC üzerinde degisiklik yapmak için enableconversion bitini kapadik
  ADC10CTL0|=SREF_1|ADC10SHT_3|REFON|ADC10ON|ADC10IE;//referans gerilimini 1.5v olarak sectik|sample and holde süresini 64xadc10clk sectik|ref gerilimini aktiflestirdik
  //adc aktif edildi|adc interruptlari acildi
  ADC10CTL1|=INCH_4|ADC10DIV_3|ADC10SSEL_0;//adc in channel olarak 4. pin secildi|adc clck'una div olarak 3 degeri verildi|adc'nin kendi clk'u source olarak secildi
  ADC10AE0=BIT4;//BIT 4 analog input için kullanilabilir duruma getirildi
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void isr(void)
{
  if(UCA0RXBUF=='1')//UCA0RXBUF'a 1 gelmesi durumunda enable conversion ve  start conversion bitleri setlendi
    ADC10CTL0|=ENC+ADC10SC;
}

#pragma vector=ADC10_VECTOR
__interrupt void ADC_ISR(void)
{
  //analog ölçüm degeri > 600 ise a degil ise b degeri yollandi.
  if(ADC10MEM>600){
    UCA0TXBUF='a';
  }
  else{
    UCA0TXBUF='b';
  }
}

//Debug functions
void putc(const unsigned c)
{
  while (!(IFG2&UCA0TXIFG));
    UCA0TXBUF = c;
}
void TX(const char *s)
{
  while(*s)
    putc(*s++);
}
```

a.2) Functions:

Function	Description
void TX(const char *s)	Send a string via UART
void putc(const unsigned c)	Send a character via UART
void Uart_Init(void);	Initilazation of Uart communication
void adc_init(void);	Initilazation of ADC10

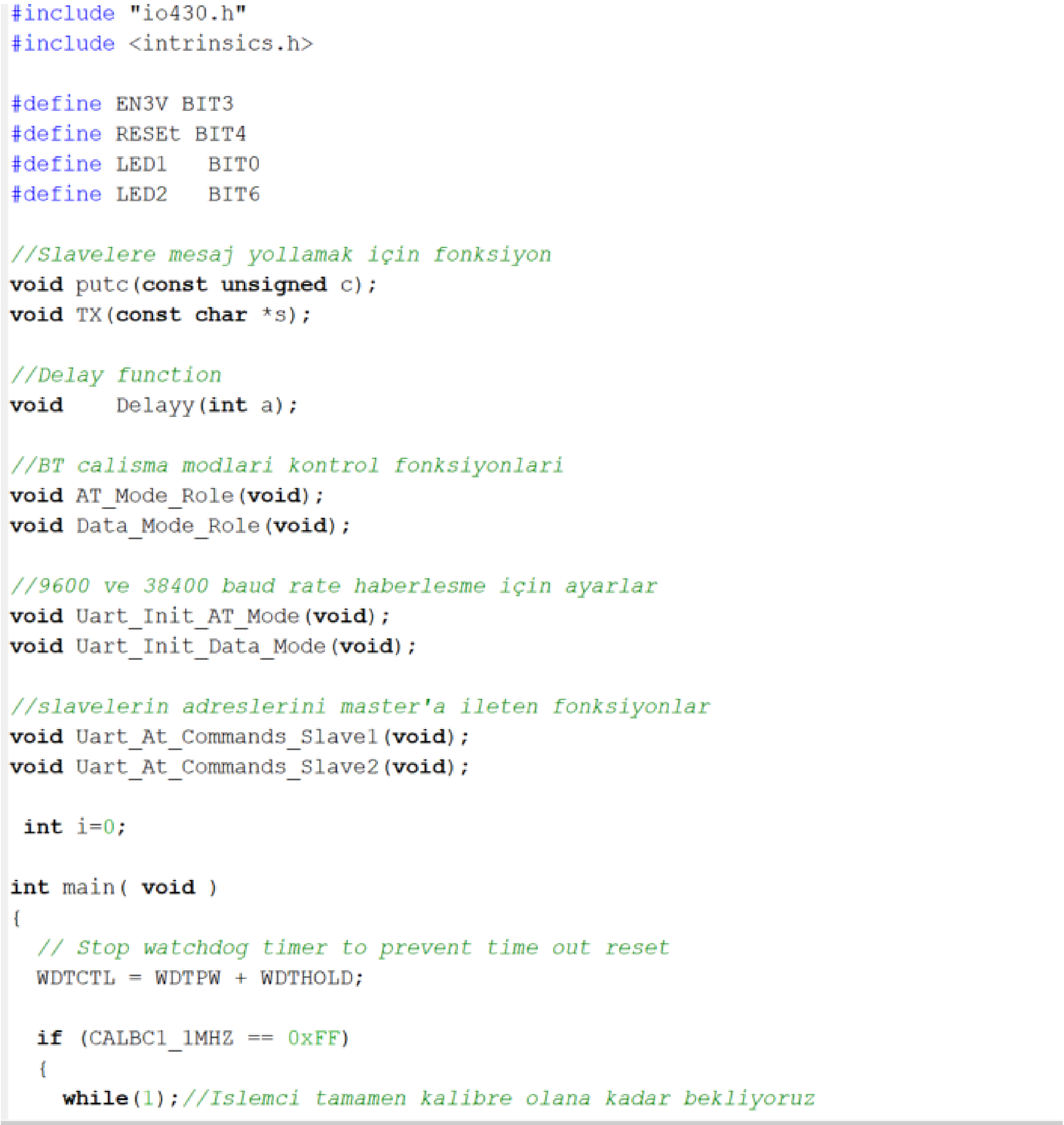
b) Code Of Master:

UART Register Initializations for 9600 baud rate. This baud rate used for BT to BT communication.

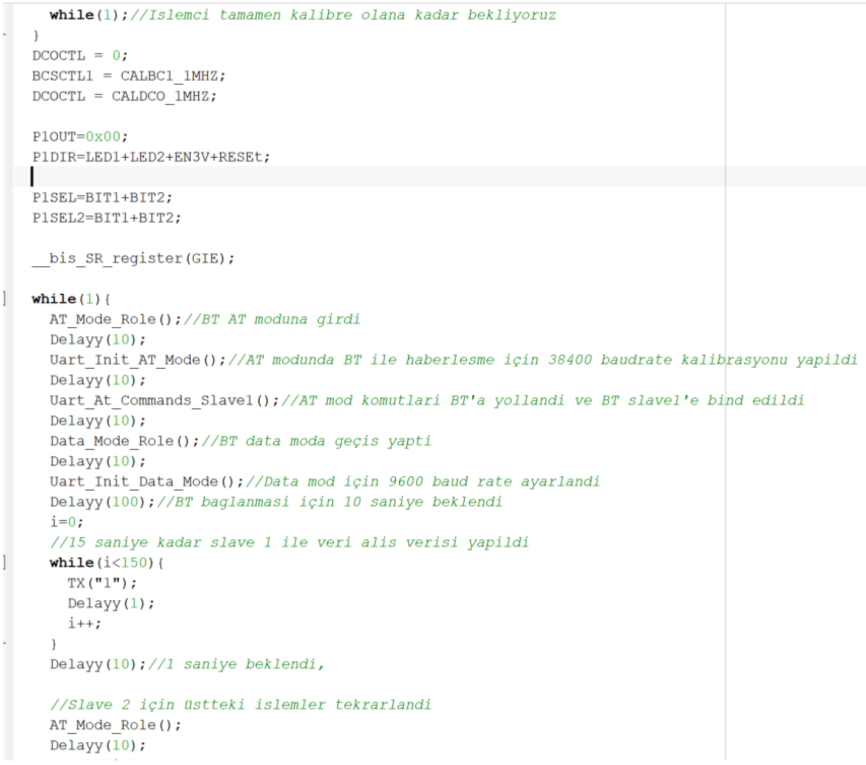
Register	Bit	Description
P1SEL	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
P1SEL2	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
UCA0CTL1	UCSSEL_2	Set UART use SMCLK
UCA0BR0	104	1MHz 9600 baud
UCA0BR1	0	1MHz 9600 baud
UCA0MCTL	UCBRS_1	Modulation UCBRSx = 1
IE2	UCA0RXIE	Rx interrupts are enabled
UCA0CTL1	~UCSWRST	Resets are disabled

UART Register Initializations for 38400 baud rate. This baud rate used for sending AT command to the module

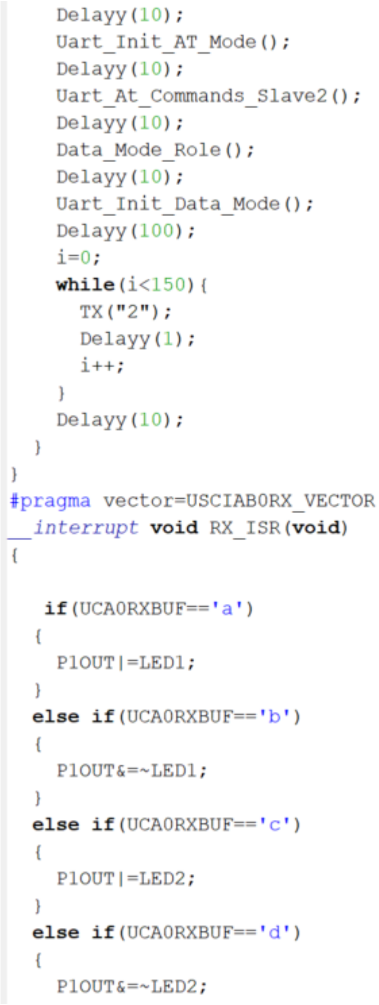
Register	Bit	Description
P1SEL	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
P1SEL2	BIT1 + BIT2	P1.1 = RXD, P1.2=TXD
UCA0CTL1	UCSSEL_2	Set UART use SMCLK
UCA0BR0	26	1MHz 9600 baud
UCA0BR1	0	1MHz 9600 baud
UCA0MCTL	UCBRS_0	Modulation UCBRSx = 0
IE2	UCA0RXIE	Rx interrupts are enabled
UCA0CTL1	~UCSWRST	Resets are disabled



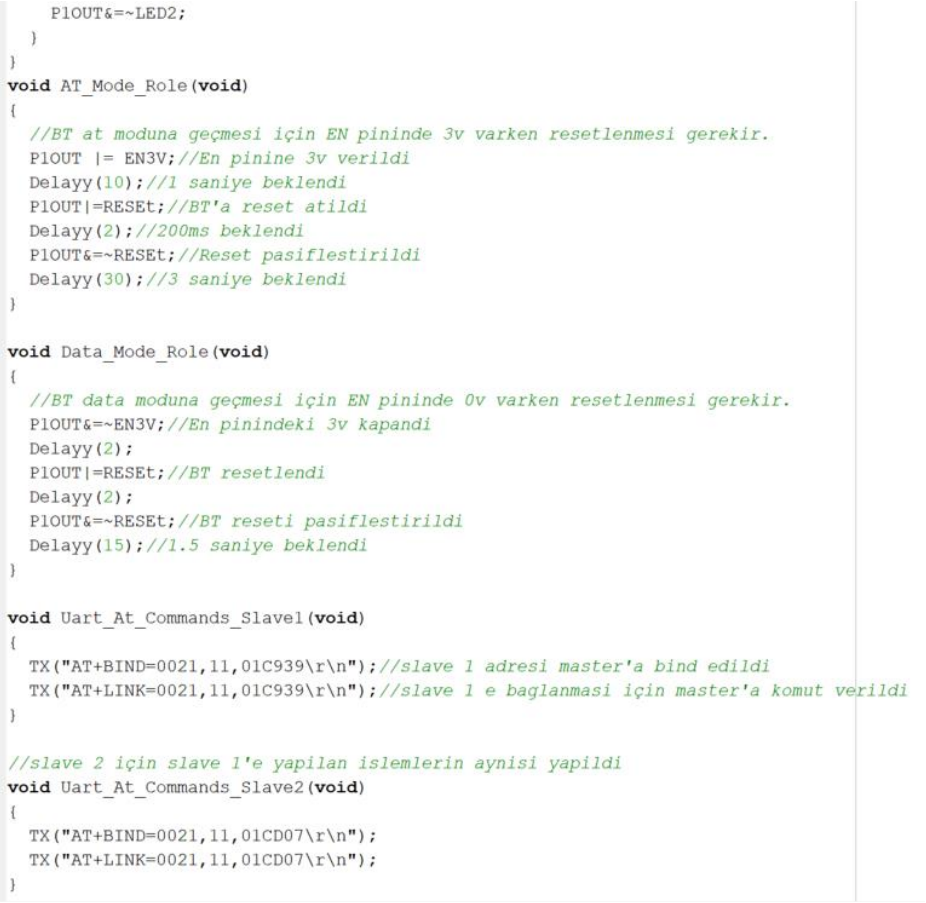
```
#include "io430.h"
#include <intrinsics.h>

#define EN3V BIT3
#define RESEt BIT4
#define LED1   BIT0
#define LED2   BIT6

//Slavelere mesaj yollamak için fonksiyon
void putc(const unsigned c);
void TX(const char *s);

//Delay function
void    Delayy(int a);

//BT calisma modlari kontrol fonksiyonlari
void AT_Mode_Role(void);
void Data_Mode_Role(void);

//9600 ve 38400 baud rate haberlesme için ayarlar
void Uart_Init_AT_Mode(void);
void Uart_Init_Data_Mode(void);

//slavelerin adreslerini master'a ileten fonksiyonlar
void Uart_At_Commands_Slave1(void);
void Uart_At_Commands_Slave2(void);

 int i=0;

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  if (CALBC1_1MHZ == 0xFF)
  {
    while(1);//Islemci tamamen kalibre olana kadar bekliyoruz
```



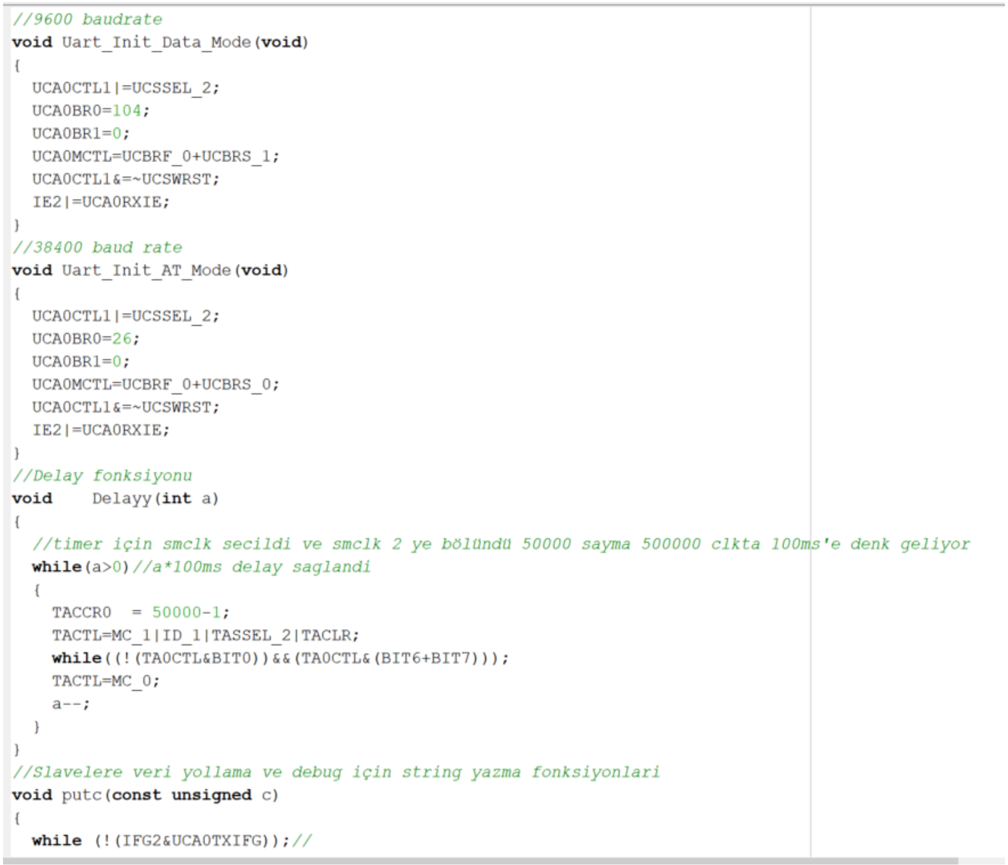
```
    while(1);//Islemci tamamen kalibre olana kadar bekliyoruz
  }
  DCOCTL = 0;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;

  P1OUT=0x00;
  P1DIR=LED1+LED2+EN3V+RESEt;

  P1SEL=BIT1+BIT2;
  P1SEL2=BIT1+BIT2;

  __bis_SR_register(GIE);

  while(1){
    AT_Mode_Role();//BT AT moduna girdi
    Delayy(10);
    Uart_Init_AT_Mode();//AT modunda BT ile haberlesme için 38400 baudrate kalibrasyonu yapildi
    Delayy(10);
    Uart_At_Commands_Slave1();//AT mod komutlari BT'a yollandi ve BT slave1'e bind edildi
    Delayy(10);
    Data_Mode_Role();//BT data moda geçis yapti
    Delayy(10);
    Uart_Init_Data_Mode();//Data mod için 9600 baud rate ayarlandi
    Delayy(100);//BT baglanmasi için 10 saniye beklendi
    i=0;
    //15 saniye kadar slave 1 ile veri alis verisi yapildi
    while(i<150){
      TX("1");
      Delayy(1);
      i++;
    }
    Delayy(10);//1 saniye beklendi,

    //Slave 2 için üstteki islemler tekrarlandi
    AT_Mode_Role();
    Delayy(10);
```



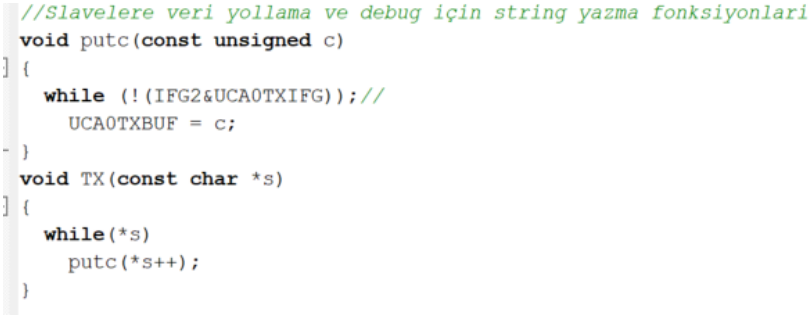
```
    Delayy(10);
    Uart_Init_AT_Mode();
    Delayy(10);
    Uart_At_Commands_Slave2();
    Delayy(10);
    Data_Mode_Role();
    Delayy(10);
    Uart_Init_Data_Mode();
    Delayy(100);
    i=0;
    while(i<150){
      TX("2");
      Delayy(1);
      i++;
    }
    Delayy(10);
  }
}
#pragma vector=USCIAB0RX_VECTOR
__interrupt void RX_ISR(void)
{

   if(UCA0RXBUF=='a')
  {
    P1OUT|=LED1;
  }
  else if(UCA0RXBUF=='b')
  {
    P1OUT&=~LED1;
  }
  else if(UCA0RXBUF=='c')
  {
    P1OUT|=LED2;
  }
  else if(UCA0RXBUF=='d')
  {
    P1OUT&=~LED2;
```



```
    P1OUT&=~LED2;
  }
}
void AT_Mode_Role(void)
{
  //BT at moduna geçmesi için EN pininde 3v varken resetlenmesi gerekir.
  P1OUT |= EN3V;//En pinine 3v verildi
  Delayy(10);//1 saniye beklendi
  P1OUT|=RESEt;//BT'a reset atildi
  Delayy(2);//200ms beklendi
  P1OUT&=~RESEt;//Reset pasiflestirildi
  Delayy(30);//3 saniye beklendi
}

void Data_Mode_Role(void)
{
  //BT data moduna geçmesi için EN pininde 0v varken resetlenmesi gerekir.
  P1OUT&=~EN3V;//En pinindeki 3v kapandi
  Delayy(2);
  P1OUT|=RESEt;//BT resetlendi
  Delayy(2);
  P1OUT&=~RESEt;//BT reseti pasiflestirildi
  Delayy(15);//1.5 saniye beklendi
}

void Uart_At_Commands_Slave1(void)
{
  TX("AT+BIND=0021,11,01C939\r\n");//slave 1 adresi master'a bind edildi
  TX("AT+LINK=0021,11,01C939\r\n");//slave 1 e baglanmasi için master'a komut verildi
}

//slave 2 için slave 1'e yapilan islemlerin aynisi yapildi
void Uart_At_Commands_Slave2(void)
{
  TX("AT+BIND=0021,11,01CD07\r\n");
  TX("AT+LINK=0021,11,01CD07\r\n");
}
```



```
//9600 baudrate
void Uart_Init_Data_Mode(void)
{
  UCA0CTL1|=UCSSEL_2;
  UCA0BR0=104;
  UCA0BR1=0;
  UCA0MCTL=UCBRF_0+UCBRS_1;
  UCA0CTL1&=~UCSWRST;
  IE2|=UCA0RXIE;
}
//38400 baud rate
void Uart_Init_AT_Mode(void)
{
  UCA0CTL1|=UCSSEL_2;
  UCA0BR0=26;
  UCA0BR1=0;
  UCA0MCTL=UCBRF_0+UCBRS_0;
  UCA0CTL1&=~UCSWRST;
  IE2|=UCA0RXIE;
}
//Delay fonksiyonu
void    Delayy(int a)
{
  //timer için smclk secildi ve smclk 2 ye böländü 50000 sayma 500000 clkta 100ms'e denk geliyor
  while(a>0)//a*100ms delay saglandi
  {
    TACCR0  = 50000-1;
    TACTL=MC_1|ID_1|TASSEL_2|TACLR;
    while((!(TA0CTL&BIT0))&&(TA0CTL&(BIT6+BIT7)));
    TACTL=MC_0;
    a--;
  }
}
//Slavelere veri yollama ve debug için string yazma fonksiyonlari
void putc(const unsigned c)
{
  while (!(IFG2&UCA0TXIFG));//
```

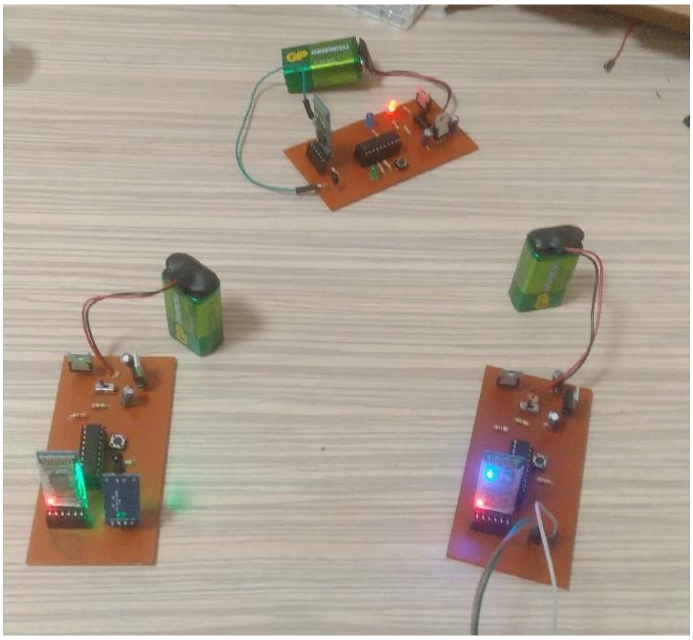


```
//Slavelere veri yollama ve debug için string yazma fonksiyonlari
void putc(const unsigned c)
{
  while (!(IFG2&UCA0TXIFG));//
    UCA0TXBUF = c;
}
void TX(const char *s)
{
  while(*s)
    putc(*s++);
}
```

b.2) Functions:

Function	Description
void TX(const char *s)	Send a string via UART
void putc(const unsigned c)	Send a character via UART
void Uart_Init(void);	Initilazation of Uart communication
void adc_init(void);	Initilazation of ADC10
void Delayy(int a);	A function that produces a delay of a*100ms
void AT_Mode_Role(void);	Function that makes the necessary bias setting for Bluetooth to switch to at mode
void Data_Mode_Role(void);	Function that makes the necessary bias setting for Bluetooth to switch to data mode
void Uart_Init_AT_Mode(void);	Function that initialize uart setting for at mode of the Bluetooth module.(38400 baud rate)
void Uart_Init_Data_Mode(void);	Function that initialize uart setting for data mode of the Bluetooth module.(9600 baud rate)
Void Uart_At_Commands_Slave1(void);	Function that bind and connect master to slave 1
void Uart_At_Commands_Slave2(void);	Function that bind and connect master to slave 2

4. Images Of The System In Operation:



5.REFERENCES:

https://www.ti.com/lit/ug/slau772a/slau772a.pdf?HQS=ti-null-null-verifimanuf_df-manu-pf-octopart-wwe&ts=1648994984108&ref_url=https%253A%252F%252Foctopart.com%252F

https://www.ti.com/lit/ug/slau144j/slau144j.pdf

https://www.saibatudomt.com.br/2018/01/conectando-3-dispositivos-arduino-utilizando-o-modulo-bluetooth-hc-05/

https://forum.arduino.cc/t/hc05-bluetooth-with-multi-slaves/274151

https://forum.arduino.cc/t/hc-05-master-and-slave-at-the-same-time/535604

https://wiki.analog.com/university/courses/eps/bjt-switch

https://www.electronicwings.com/ti-launchpad/hc-05-bluetooth-module-interfacing-with-msp-exp430g2-ti-launchpad