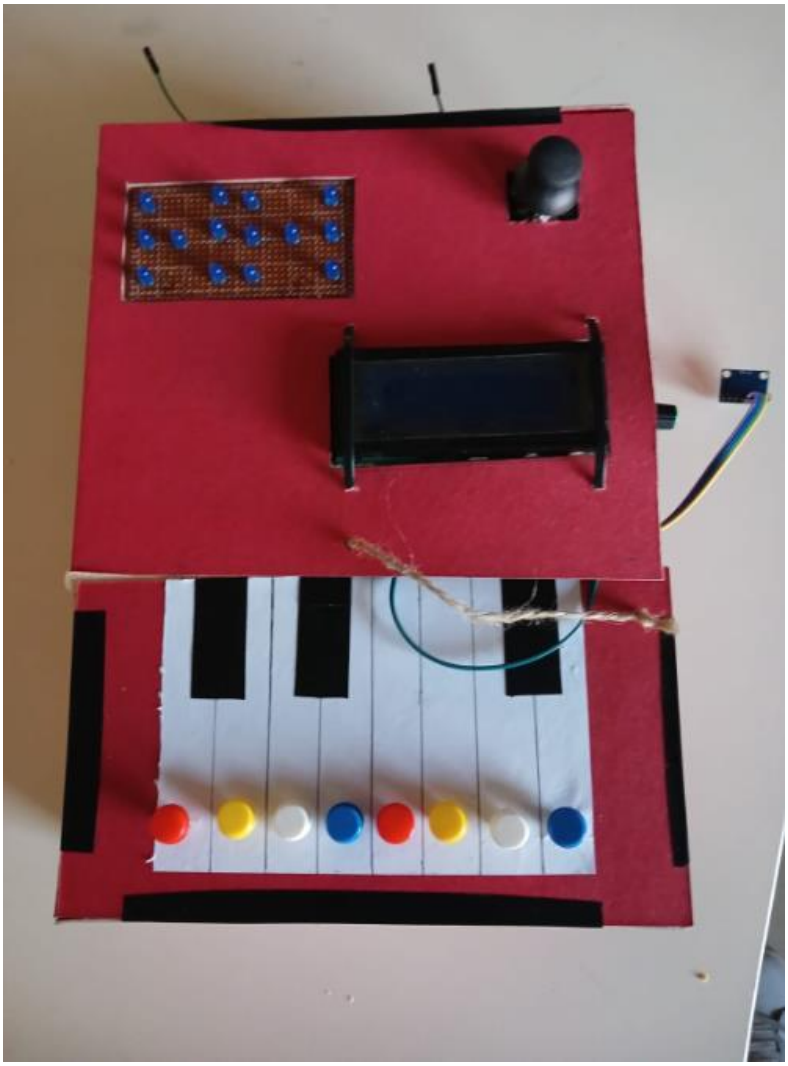
KOCAELİ ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

MEKATRONİK MÜHENDİSLİĞİ



MİKROİŞLEMCİ PROJE

Proje Adı: IMU Sensörü ile Zar Oyunu ve Piano Platformu

Emirhan Kuru

İçindekiler

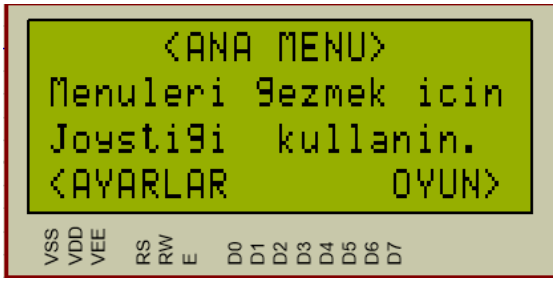
PROJE TANITIMI

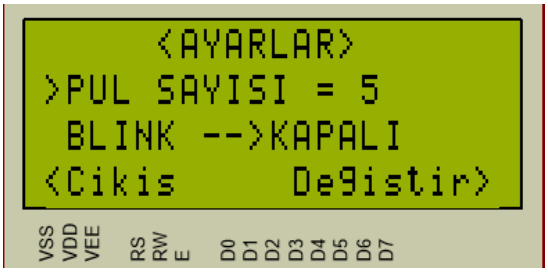
Bu projede, Joystick ile alınan ADC değerine göre LCD üzerinden bir menü üzerinde işlemler gerçekleştirilmektedir. Menü içerisinde ayarlar, oyun ve piano seçenekleri bulunmaktadır. Oyun, zarla oynanmaktadır. IMU sensöründen belirli bir değerin üzerinde ivme ölçüldüğünde, LED'lere zar değeri gönderilmektedir. Zarların rastgeleliği Timer0 ve Timer1 zamanlayıcıları kullanılarak sağlanmıştır. Projede ayrıca 2 adet PIC mikrodenetleyici kullanılmıştır.

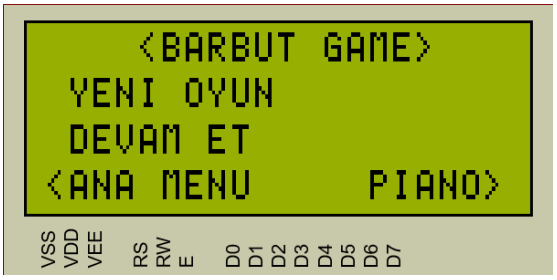
IMU'yu okuyan PIC, diğer PIC'e UART ile ivme verisini iletmektedir. Gelen veriye bağlı olarak oyun oynanmaktadır. Oyun oynandığı PIC ise diğer PIC'e, piano moduna girilip girilmediği bilgisini UART aracılığıyla göndermektedir. Kısacası, IMU ve piano işlemleri bir PIC'te gerçekleştirilirken, diğer PIC'te diğer işlemler yapılmaktadır.

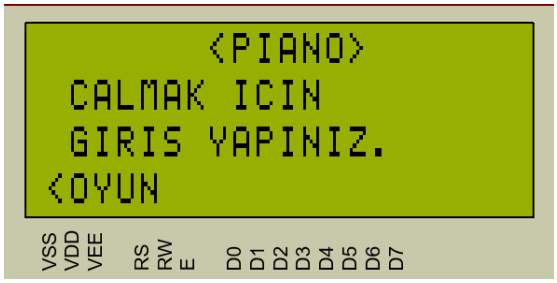
PROJE DETAY

LCD de 4 adet ana menü vardır bunlar aşağıdaki gibidir.



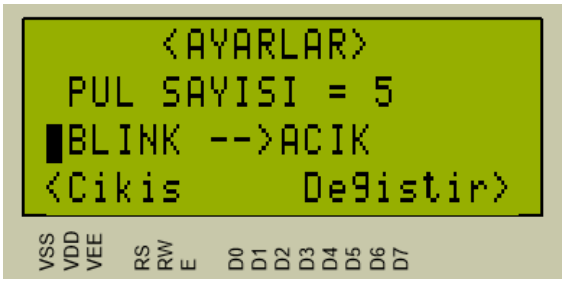






Ayarlar menüsü

Ayarlar menüsünde 2 adet değişiklik yapılabilmektedir. Pul sayısı oyuna başlangıç sayısını belirtir. Bu değer eeproma yazılarak kayıt edilmesi sağlanmıştır. Blink ise LCD nin imlecini açıp kapatmayı sağlar.



Oyun Menüsü

Oyun menüsünde 2 adet seçenek bulunmaktadır. Bunlar yeni oyun ve Devam et seçenekleri Yeni oyun seçeneği seçildiğinde Ayarlardan seçilen Pul sayısı kadar sayı ile oyuncular oyuna başlarlar. Ve her turdaki değerler eeproma kaydedilir bu sayede güç kesilmesi vs. durumlarda devam et seçeneğiyle oyuna aynı yerden devam edilebilmektedir.

ROM dolduğu için ekleyemediğim seçenek

Oyunu ilk tasarlarken ayarlar menüsüne bilgisayara karsı oyna seçeneğide eklemiştim. Ama rom dolduğu için silmek zorunda kaldım. Bilgisayın rastgeleliğini timer zamanlayıcıları ile hesaplayamayacağım için LDR sensöründen aldığım analog verdien rastgelelik ile Zar değerleri üretilecekti.

Piano Menüsü

Piano menüne girilmesi halinde LCD de Piano_animasyon oynatılmaktadır ve diğer pice uart üzerinden piano menüsü seçildiğine dair veri iletilmektedir.

EEPROM Kullanımı

Ayarlar menüsündeki başlangıç pul sayısı ve oyun sırasındaki sayılar eeproma kaydedilmiştir.

WDT kullanımı

Slave olarak kullanılan (oyunun oynandığı ve LCD kullanılan) pic de WDT kullanılmıştır. Eğer butan takılı kalırsa WDT resete gitmektedir.

SLEEP Kullanımı

Sleep fonksiyonu Master picte kullanılmıstır. Çünkü master pic piano çalmıyoken ve oyun sırasında imu sensörü göndermesi dışında hiç bir işlem yapmamaktadır. Bu sebeple bu durumlar dışında mikrodenetleyici uyku moduna alınmıştır.

Mikrodenetleyici uyku modundan çıkarmak için kesme vs gibi sinyal uygulanmalıdır. Master direkt olarak çalışınca uykuya alınır. Timer1 zamanlayıcısı Asenkron sayıcı modda kullanarak uyku sırasında timer modülü kullanılmış olur ve timer1 kesme ile uyanarak imu verisini gönderir ve sonra tekrar uyur. Yani belirli periyotla uyanıp veri gönderip tekrar uyumaktadır. Piano modunda ise zaten while döngüsünde sürekli döndüğünden pianoda uyku moduna girmez.

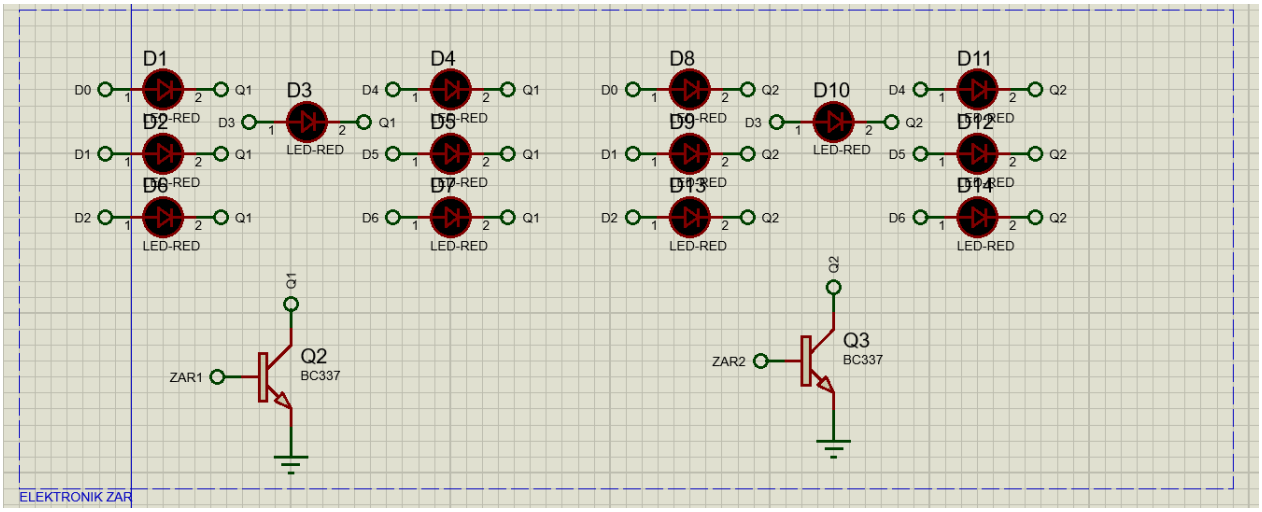
Zarlar

Zar değerleri Timer0 ve Timer1 zamanlayıcılarının değer kaydedicileri ile hesaplanmaktadır.

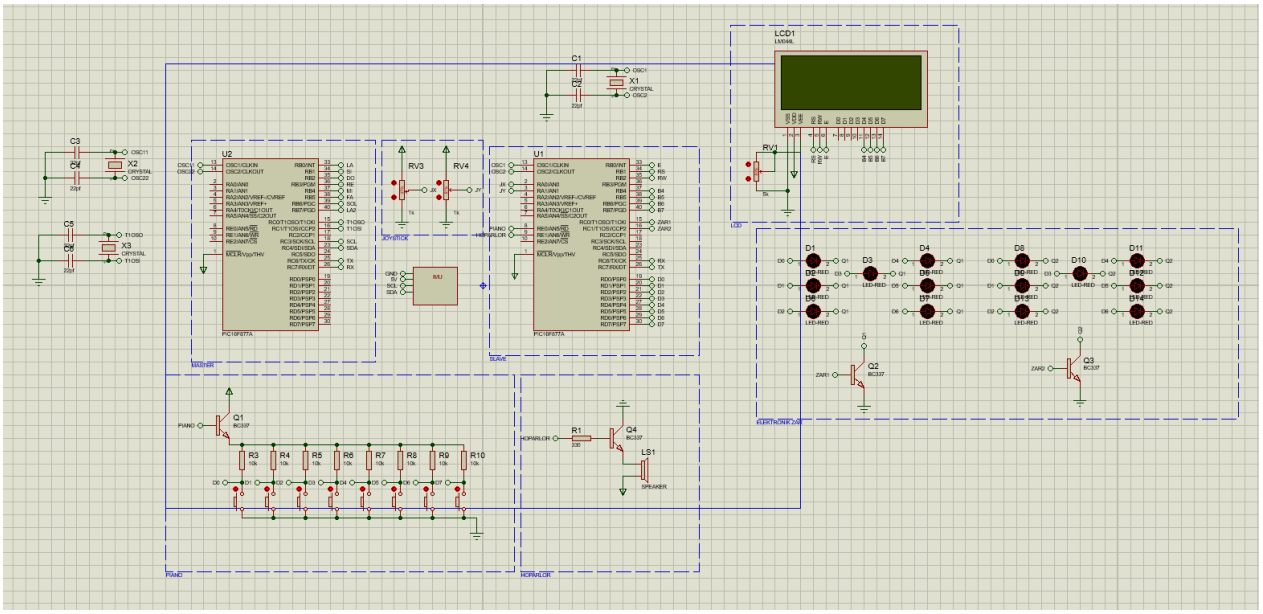
zar1 = (int)(TIMER0/50) + 1;

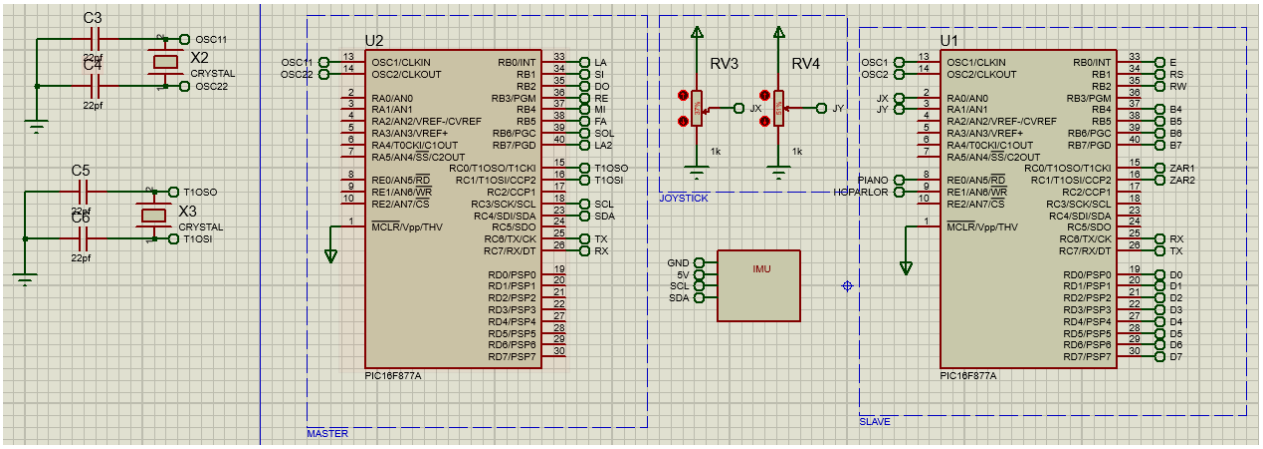
zar2 = (int)(get_timer1()/13100) + 1;

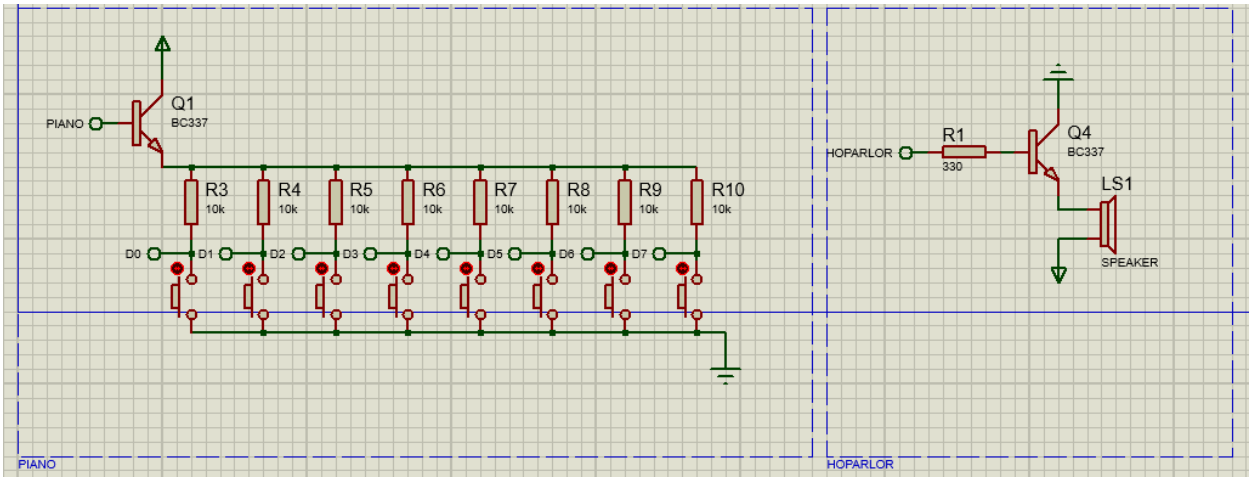
ile hesaplanılmıştır. Ayrıca her zar için 7 adet led olduğundan ledler tarama metodu ile sürülmüştüri



PROTEUS ÇİZİMİ

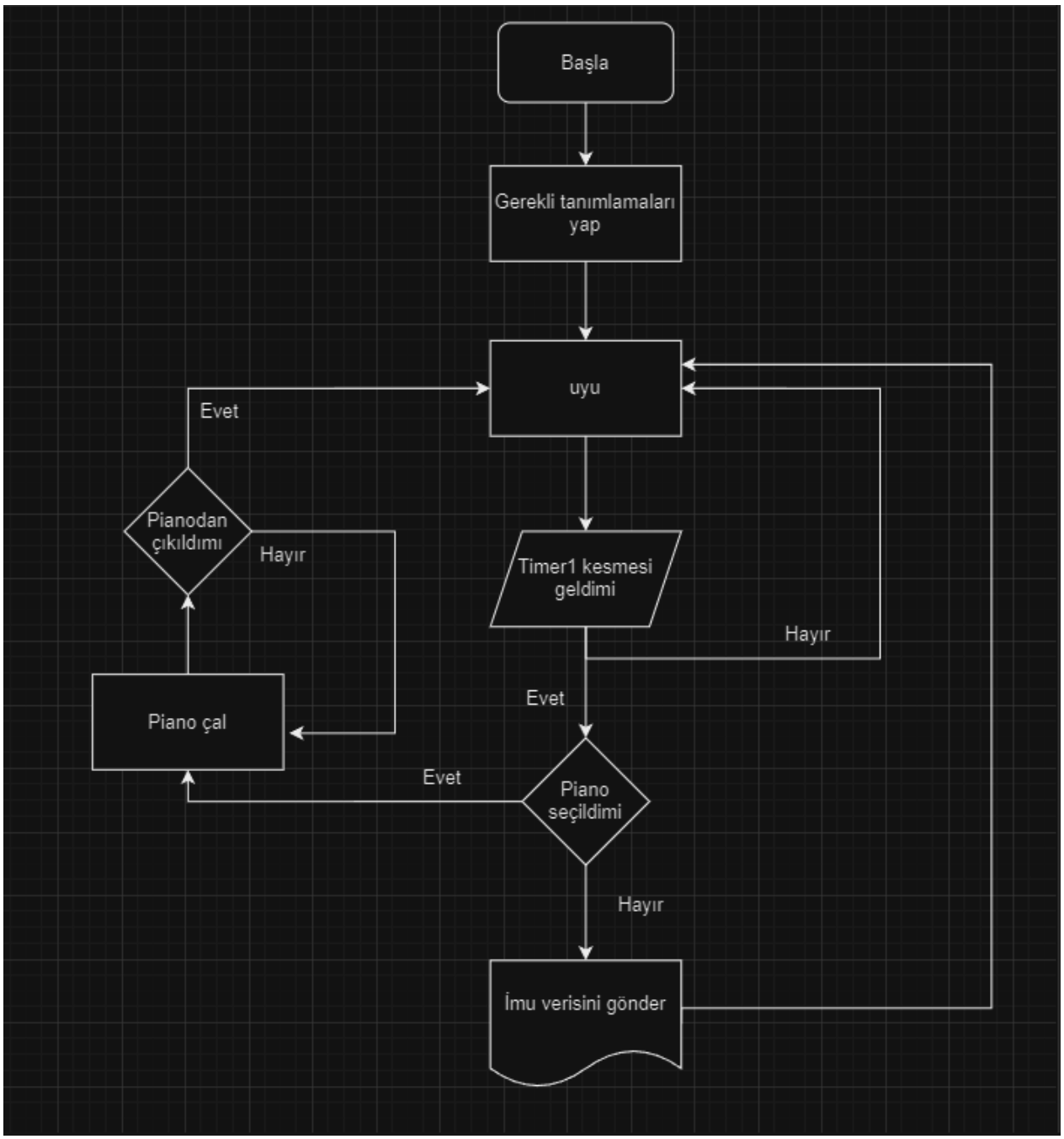




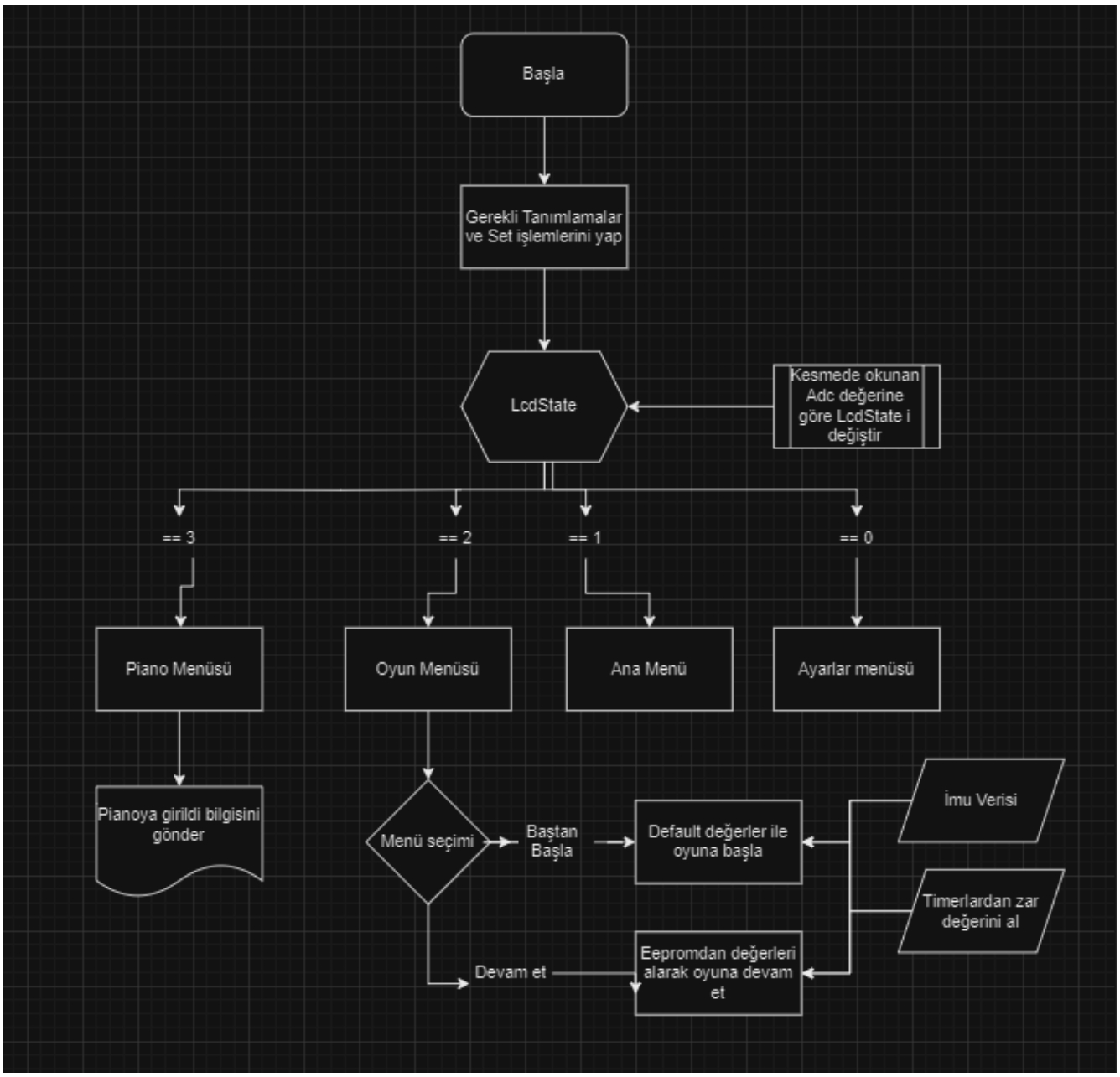


Akış Diyagramı

Master.c Akış diyagramı



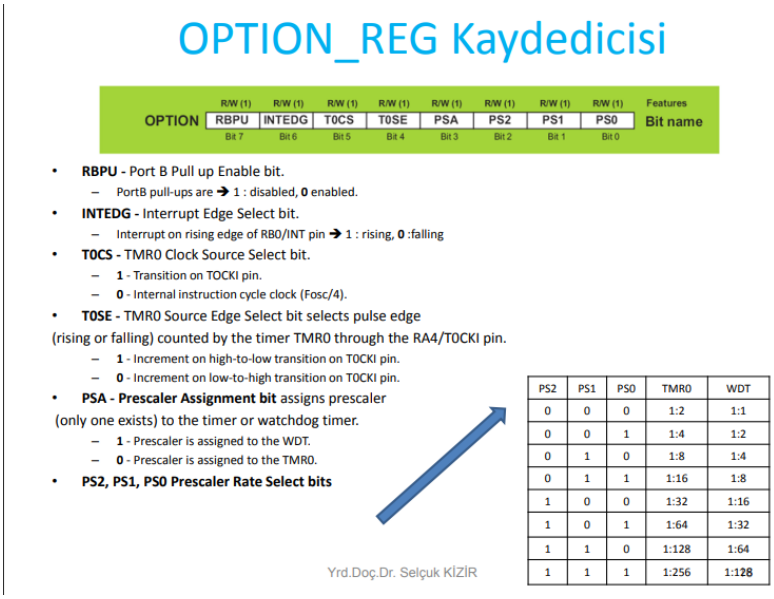
Slave.c Akış Diyagramı



Register Tanımlamaları

Slave pic için (LCD ve oyunun oynandığı) ;

OPTION_REG (0x81)



Optıon registeri Tımer 0 zamanlayıcı ayarlarının yapıldığı registerdır. Prescaler 256 ve zamanlayıcı modda kullanılmak istendiğinden dolayı 0xC7 degeri veilmesi gerekmedtedir. Default olarak 1 değerinin aldığı için byte sal ve (&&) operatörü ile bu değer verilebilir.

TIMER0 (0x01)

Timer 0 değer kaydedicisidir. 0-255 arası sayması istendiğinden dolayı başlangıçta 0 degeri veilmişitir.

T1CON (0x10)

T1con kaydedicisi timer1 zamanlayıcısının kontrol kaydedicisidir. T1 zamanlayısıcı asenkron zamanlayıcı modda çalışması istendiği ve prescaler değeri 8 olarak seçildiğinden dolayı 0x35 değerini almalıdır. T1CON default 0 aldığından dolayı byte sal veya (||) kullanılarak bu değer kaydediciye atanabilir.

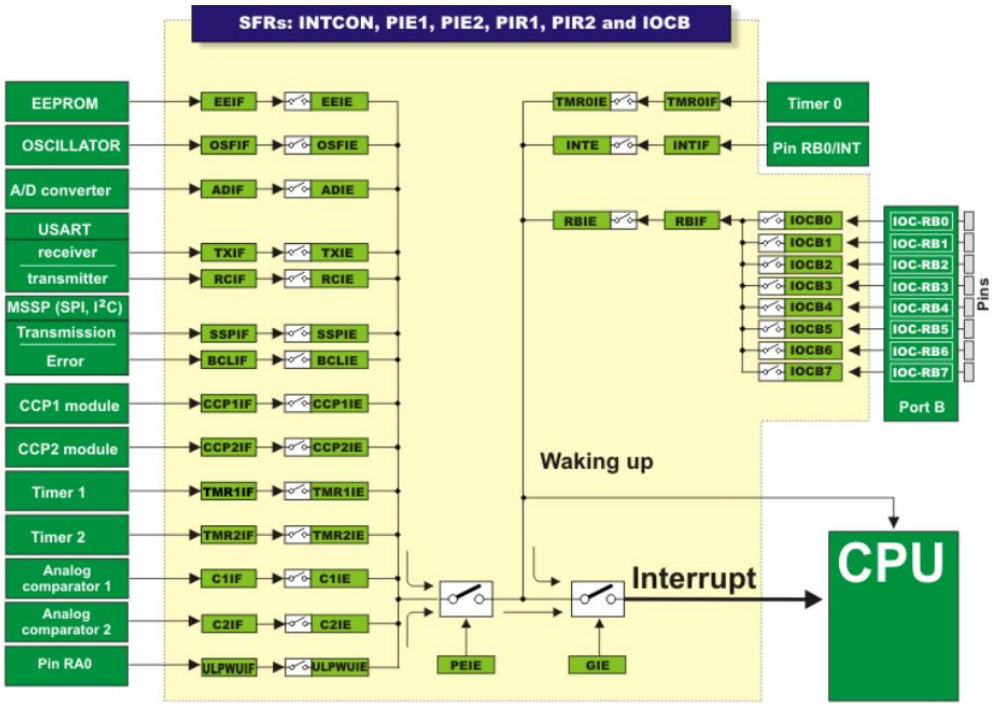
T1CON Kaydedicisi

- **7:6** Kullanılmayan bitlerdir. '0' okunur.
- **T1CKPS1:T1CKPS0 –** Bölme Oranı (Prescaler) Seçim Bitleri
- **T1OSCEN –** T1 Osilatörü Yetkilendirme Biti
  - **1 –** Osilatör Aktif
  - **0 –** Osilatör Pasif
- **T1SYNC –** Harici Saat Sinyal Girişi Senkronizasyon Seçim Biti
  - TMR1CS = 1 olduğunda:
    - **1 –** Senkronizasyon pasif
    - **0 –** Senkronizasyon aktif
  - TMR1CS = 0 olduğunda:
    - Dahili saat sinyalleri aktif olduğundan senkronizasyon devre dışı.
- **TMR1CS**: Timer1 Clock Source Select bit
  - 1 = External clock from pin T1OSO/T1CKI (on the rising edge)
  - 0 = Internal clock (FOSC/4)
- **TMR1ON**: Timer1 On bit
  - 1 = Enables Timer1
  - 0 = Stops Timer1

PS2	PS1	TMR1
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

INTCON(0xD1)

INTCON kaydedicisi genel kesme ayarlarının yapıldığı kaydedicidir. Kodda 2 adet kesme kullanılmaktadır. TIMER0 Ve RDA kesmesi TIMER0 için sadece GIE açılması yeterli olmaktadır ama RDA kesme için PEIE nında açılması gerekmektedir. Buna göre başlangıçta INTCON registeri 0xD1 değerini almalıdır. Fakat her kesme oluştuğunda GIE PEIE TOIF ve TOIE gibi bitler de değişiklik olmaktadır. Her kesmeye gidildiğinde bu bitlerin durumu ayarlanmalıdır. Aksi halde kesme fonksiyonu istenilen gibi çalışmaz.



INTCON Kaydedicisi

	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (x)	Features
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

- **GIE** – Genel Kesme Yetkilendirme biti
  - GIE = 0 ➔ disabled, GIE = 1 ➔ enabled.
- **PEIE** – Çevresel Kesme Yetkilendirme Biti
  - PEIE = 0 ➔ disabled, PEIE = 1 ➔ enabled.
- **T0IE – Timer0 Kesme Yetkilendirme Biti**
  - **T0IE = 0 ➔ disabled, T0IE = 1 ➔ enabled.**
- **INTE** – Harici Kesme(RB0/INT) Yetkilendirme Biti
  - INTE = 0 ➔ disabled, INTE = 1 ➔ enabled.
- **RBIE** – RB4-RB7 Pinlerindeki Değişim Kesmesi Yetkilendirme Biti
  - RBIE = 0 ➔ disabled, RBIE = 1 ➔ enabled.
- **T0IF – Timer0 Taşma Kesmesi Bayrağı**
  - **T0IF = 0 ➔ Taşma yok, T0IF = 1 ➔ Taşma var.**
- **INTF** – Harici Kesme(RB0/INT) Bayrağı
  - INTF = 0 ➔ Harici kesme yok, INTF = 1 ➔ Harici kesme var.
- **RBIF** – PORTB<7:4> Değişim Kesme Bayrağı
  - RBIF = 0 ➔ Değişim yok, RBIF = 1 ➔ Değişim var.

ADCON0(0x1F)

ADCCON0 Kaydedicisi

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
	bit 7							bit 0

- **CHS2 – CHS1 – CHS0 :** Analog-dijital kanal seçim bitleri

ADCCON0 <CHS2>	ADCCON0 <CHS1>	ADCCON0 <CHS0>	ANALOG KANAL SEÇİMİ
0	0	0	Channel 0 (AN0)
0	0	1	Channel 1 (AN1)
0	1	0	Channel 2 (AN2)
0	1	1	Channel 3 (AN3)
1	0	0	Channel 4 (AN4)
1	0	1	Channel 5 (AN5)
1	1	0	Channel 6 (AN6)
1	1	1	Channel 7 (AN7)

- **GO/DONE' :** Analog çevrim tamamlandı biti. Analog çevrim tamamlandığında otomatik olarak sıfırlanır.
  - **0 :** Analog Çevrim tamamlandı.
  - **1 :** Analog çevrim işlem sürecinde.
- **ADON :** Analog-Dijital Çevirici modülü açma-kapama biti.
  - **0 :** ADC modülü pasif
  - **1 :** ADC modülü aktif

11

ADCON0 kaydedicisi kanal seçim bitinin yapıldığı resiterdır. 2 adet kanal kullanıldığından her kanal değiştiğinde CHS0 biti değişicektir. Bu biti #bit operatörü ile SET_ADC değerine atadım. Her kanal değişimi olunca bu biti değiştirerek kanal değişimi yapıyorum. Ayrıca ADON biti aktif edilmeli ve 3. Bit ise kendisi değişmektedir. Bu bite müdehale etmedim. ADCS bitleri ise ADCON1 kaydedicisindeki bir bit ile beraber aşağıdaki tabloda görülmektedir. Clock olarak dahili (INTERNAL) osilatör kullanılmak istendiğinden dolayı 6. Ve 7. Bitleri set ettim.

ADCCON0 Kaydedicisi

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
	bit 7							bit 0

- **ADCS1 – ADCS0 :** Analog-dijital çevirici clock seçim bitleri

ADCCON1 <ADCS2>	ADCCON0 <ADCS1>	ADCCON0 <ADCS0>	CLOCK CONVERSION
0	0	0	F_osc / 2
0	0	1	F_osc / 8
0	1	0	F_osc / 32
0	1	1	F_RC : Saat kaynağı dahili RC osilatör
1	0	0	F_osc / 4
1	0	1	F_osc / 16
1	1	0	F_osc / 64
1	1	1	F_RC : Saat kaynağı dahili RC osilatör

10

ADCON1 (0x9F)

ADCON1 kaydedicisi ise hangi pinlerin analog veya dijital olduğu seçimini yapar. Projede 2 adet analog pin kullanılmaktadır. Buna en uygun olanı AN0,AN1, ve AN3 pinlerinin analog diğerlerinin diital olduğu 0100 seçeneğidir. Bundan dolayı ADCON1 = 0x04 değeri atanmıştır.

ADCCON1 Kaydedicisi

	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
	bit 7							bit 0

- **ADFM :** Sonuç kaydedicileri bilgisi ne tarafa dayalı olacağı belirlenir.
  - **0 :** Sonuçlar sola dayalı.
  - **1 :** Sonuçlar sağa dayalı
- **ADCS2 :** Analog-dijital çevirici clock seçim bitleri. ADCCON0<ADCS1:ADCS0> ile birlikte kullanılır.
- **PCFG3 – PCFG2 – PCFG1 – PCFG0 :** ADC port konfigurasyon kontrol bitleri.

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2/1
011x	D	D	D	D	D	D	D	D	—	—	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1/2

12

PIE1 (0x8C)

Uart haberleşmesi ile alınan veri RDA kesmesinde alındığı için PIE1 registerının RCIE biti set edilmelidir. BU bit uart ile veri alınınca kesme oluşturur.

PIE1 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

PSPIE: Parallel Slave Port Read/Write Interrupt Enable bit**(1)**

1 = Enables the PSP to read/write interrupt

0 = Disables the PSP read/write interrupt

Note **(1)**: PSPIE is reserved on PIC16F873A/876A devices; always maintain this bit clear.

ADIE: A/D Converter Interrupt Enable bit

1 = Enables the A/D converter interrupt

0 = Disables the A/D converter interrupt

RCIE: USART Receive Interrupt Enable bit

1 = Enables the USART to receive interrupt

0 = Disables the USART receive interrupt

TXIE: USART Transmit Interrupt Enable bit

1 = Enables the USART to transmit interrupt

0 = Disables the USART transmit interrupt

SSPIE: Synchronous Serial Port Interrupt Enable bit

1 = Enables the SSP interrupt

0 = Disables the SSP interrupt

CCP1IE: CCP1 Interrupt Enable bit

1 = Enables the CCP1 interrupt

0 = Disables the CCP1 interrupt

TMR2IE: TMR2 to PR2 Match Interrupt Enable bit

1 = Enables the TMR2 to PR2 match interrupt

0 = Disables the TMR2 to PR2 match interrupt

TMR1IE: TMR1 Overflow Interrupt Enable bit

1 = Enables the TMR1 overflow interrupt

0 = Disables the TMR1 overflow interrupt

Master pic için (imu ve piano);

T1CON (0x10);

- **7:6** Kullanılmayan bitlerdir. '0' okunur.
- **T1CKPS1:T1CKPS0 –** Bölme Oranı (Prescaler) Seçim Bitleri
- **T1OSCEN –** T1 Osilatörü Yetkilendirme Biti
  - **1 –** Osilatör Aktif
  - **0 –** Osilatör Pasif
- **T1SYNC –** Harici Saat Sinyal Girişi Senkronizasyon Seçim Biti
  - TMR1CS = 1 olduğunda:
    - **1 –** Senkronizasyon pasif
    - **0 –** Senkronizasyon aktif
  - TMR1CS = 0 olduğunda:
    - Dahili saat sinyalleri aktif olduğundan senkronizasyon devre dışı.
- **TMR1CS**: Timer1 Clock Source Select bit
  - 1 = External clock from pin T1OSO/T1CKI (on the rising edge)
  - 0 = Internal clock (FOSC/4)
- **TMR1ON**: Timer1 On bit
  - 1 = Enables Timer1
  - 0 = Stops Timer1

PS2	PS1	TMR1
0	0	1:1
0	1	1:2
1	0	1:4
1	1	1:8

master sürekli uyuyup veri göndereceği zaman uyanması istenmektedir. Bunu yapmanın bir yolu timer1 zamanlayıcısını asenkron zamanlayıcı modunda kullanıp dışarıdan düşük hertz 10Khz-300Khz mertebelerinde kristal takarak timer1 kesmesinde uyandırmakla yapılabilir. Bunun için senkronizasyonu kapatarak dışarıdan kristal bağlamak istendiğinden T1CON = 0x07 yapılmıştır.

20khz kristal ile 100ms taşma süresi istendiğinden prescaler 1 seçilip kaydedici 63535 den başlatılmıştır.

INTCON

Burdaki herhangi bir kesme kullanılmayacağı için sadece GIE ve PIEI set edilmiştir. Bu yüzden 0xC0 değeri atanmıştır.

INTCON Kaydedicisi

- **GIE** – Genel Kesme Yetkilendirme biti
  - GIE = 0 ➔ disabled, GIE = 1 ➔ enabled.
- **PEIE** – Çevresel Kesme Yetkilendirme Biti
  - PEIE = 0 ➔ disabled, PEIE = 1 ➔ enabled.
- **T0IE – Timer0 Kesme Yetkilendirme Biti**
  - **T0IE = 0 ➔ disabled, T0IE = 1 ➔ enabled.**
- **INTE** – Harici Kesme(RB0/INT) Yetkilendirme Biti
  - INTE = 0 ➔ disabled, INTE = 1 ➔ enabled.
- **RBIE** – RB4-RB7 Pinlerindeki Değişim Kesmesi Yetkilendirme Biti
  - RBIE = 0 ➔ disabled, RBIE = 1 ➔ enabled.
- **T0IF – Timer0 Taşma Kesmesi Bayrağı**
  - **T0IF = 0 ➔ Taşma yok, T0IF = 1 ➔ Taşma var.**
- **INTF** – Harici Kesme(RB0/INT) Bayrağı
  - INTF = 0 ➔ Harici kesme yok, INTF = 1 ➔ Harici kesme var.
- **RBIF** – PORTB<7:4> Değişim Kesme Bayrağı
  - RBIF = 0 ➔ Değişim yok, RBIF = 1 ➔ Değişim var.

PIE1

Rda ve timer1 kesmesi oluşması için 0x21 değeri atanmıştır.

PIR1

Pır1 in 0 biti olan TMR1IF ve 5. biti RCIF taşma bayrakları her kesmede resetlenmelidir.

İ2C İMU sensörü

ACCEL_CONFİG (0x1C)

Bu register acceloremeter ayarlarının ayarlandığı kısımdır. Zar için sadece X yönünde gelen veri kullanıldığından dolayı sadece bu set edilmeli y ve z 0 yapılabilir. Afs ise çözünürlük yüksek olması için 11 olarak set edilmiştir bunu sonucunda 0x91 değerini almıştır.

4.5 Register 28 – Accelerometer Configuration ACCEL_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		-		

Description:

This register is used to trigger accelerometer self test and configure the accelerometer full scale range. This register also configures the Digital High Pass Filter (DHPF).

Accelerometer self-test permits users to test the mechanical and electrical portions of the accelerometer. The self-test for each accelerometer axis can be activated by controlling the *XA_ST*, *YA_ST*, and *ZA_ST* bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation simulates an external force. The actuated sensor, in turn, will produce a corresponding output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each accelerometer axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

AFS_SEL selects the full scale range of the accelerometer outputs according to the following table.

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

CONFIG (0x1A)

Config registerı filtre ayarlarının yapıldığı ve asenkron mu yoksa sekron bir şekildemi verilerin alınacağının seçildiği registerdır.

4.3 Register 26 – Configuration

CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1A	26	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		

Description:

This register configures the external Frame Synchronization (FSYNC) pin sampling and the Digital Low Pass Filter (DLPF) setting for both the gyroscopes and accelerometers.

An external signal connected to the FSYNC pin can be sampled by configuring *EXT_SYNC_SET*.

Signal changes to the FSYNC pin are latched so that short strobes may be captured. The latched FSYNC signal will be sampled at the Sampling Rate, as defined in register 25. After sampling, the latch will reset to the current FSYNC signal state.

The sampled value will be reported in place of the least significant bit in a sensor data register determined by the value of *EXT_SYNC_SET* according to the following table.

EXT_SYNC_SET	FSYNC Bit Location
0	Input disabled
1	TEMP_OUT_L[0]
2	GYRO_XOUT_L[0]
3	GYRO_YOUT_L[0]
4	GYRO_ZOUT_L[0]
5	ACCEL_XOUT_L[0]
6	ACCEL_YOUT_L[0]
7	ACCEL_ZOUT_L[0]

The DLPF is configured by *DLPF_CFG*. The accelerometer and gyroscope are filtered according to the value of *DLPF_CFG* as shown in the table below.

DLPF_CFG	Accelerometer (F_s = 1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Bit 7 and bit 6 are reserved.

Parameters:

EXT_SYNC_SET 3-bit unsigned value. Configures the FSYNC pin sampling.
DLPF_CFG 3-bit unsigned value. Configures the DLPF setting.

CCS C Kodları

Slave.c

```
#include <16F877A.h>

#device ADC=8
```

```
#FUSES WDT                  //Watch Dog Timer

#FUSES NOBROWNOUT           //No brownout reset

#FUSES NOLVP                //No low voltage prgming, B3(PIC16) or
B5(PIC18) used for I/O


#use delay(crystal=4000000)

#use rs232(baud=9600, xmit=pin_c6, rcv= pin_C7, parity = N, stop
= 1)


//LCD ayarları

#define use_portb_lcd TRUE

#include "LCD420.c"


#use fast_io(D)

//Register

#byte OPTION_REG = 0x81

#byte TIMER0 = 0x01

#byte T1CON = 0x10

#byte INTCON = 0x0B

#bit GIE = INTCON.7

#bit PEIE = INTCON.6

#bit TOIE = INTCON.5

#bit TOIF = INTCON.2

#byte ADCON0 = 0x1F

#byte ADCON1 = 0x9F

#bit SET_ADC = ADCON0.3


#byte PIE1 = 0x8C

#byte PIR1 = 0x0C

#bit  RCIF = PIR1.5

#bit  TXIF = PIR1.4

#define zarbir PIN_C0

#define zariki PIN_C1



int zar_tablo[7] = {0x00,0x08,0x22,0x1C,0x55,0x5D,0x77};

int pul_sayisi = 5;

int pul_sayisi_1,pul_sayisi_2;

int count_timer0 =0;

char imu =0;

unsigned int adc_value_x = 128;

unsigned int adc_value_y = 128;

int zar1 = 0,zar2 =0;

int devamet =0,loop;

int Cik = 0,zar_timer=1;

char lcdState = 1;


void kazanan()

{

        Lcd_gotoxy(1,1);

        Lcd_putc("xxxxxxxxxxxxxxxxxxxx");

        Lcd_gotoxy(1,2);

        Lcd_putc("x    PLAYER      x");

        Lcd_gotoxy(1,3);

        Lcd_putc("x    KAZANDI     x");

        Lcd_gotoxy(1,4);

        Lcd_putc("xxxxxxxxxxxxxxxxxxxx");

  if(pul_sayisi_1 == 0)

  int kazanan = 2;
```

```
  if(pul_sayisi_2 == 0)
  kazanan = 1;
  Lcd_gotoxy(14,2);
  printf(lcd_putc,"%d",kazanan);

}
void play_piano()
{
  Lcd_gotoxy(1,1);
  Lcd_putc("   #     #     ");
  Lcd_gotoxy(1,2);
  Lcd_putc("  ##  ## # # # ");
  Lcd_gotoxy(1,3);
  Lcd_putc(" # ##  ## ## # # ");
  Lcd_gotoxy(1,4);
  Lcd_putc("## ## # ### ##### # ");
  delay_ms(100);
  Lcd_gotoxy(1,1);
  Lcd_putc("    # # #     ");
  Lcd_gotoxy(1,2);
  Lcd_putc("#  ## # ###    #");
  Lcd_gotoxy(1,3);
  Lcd_putc("##  ## ## ### # # ##");
  Lcd_gotoxy(1,4);
  Lcd_putc("### ##### ### ### ##");
  delay_ms(100);
  Lcd_gotoxy(1,1);
  Lcd_putc("   #     #     ");
  Lcd_gotoxy(1,2);
  Lcd_putc("  ##  ## # # # ");
  Lcd_gotoxy(1,3);
  Lcd_putc(" # ##  ## ## # # ");
  Lcd_gotoxy(1,4);
  Lcd_putc("## ## # ### ##### # ");
  delay_ms(100);
  Lcd_gotoxy(1,1);
  Lcd_putc("               ");
  Lcd_gotoxy(1,2);
  Lcd_putc("               ");
  Lcd_gotoxy(1,3);
  Lcd_putc("    #    #    # ");
  Lcd_gotoxy(1,4);
  Lcd_putc("### ## #  ### ##  ##");
  delay_ms(100);
}
#priority rda,timer0

#int_rda
void seri_kesme()
{

  imu = getc();
  if(4 == lcdState)
  restart_wdt();
  RCIF = 0;
  GIE = 1;
}

#int_timer0
void kesme_timer0()
{
  INTCON = 0x65;
```

```
  count_timer0++;
  if(2 == count_timer0)
  {
  SET_ADC = 0;
  delay_us(20);
  adc_value_x = read_adc();
  delay_us(20);
  SET_ADC = 1;
  delay_us(20);
  adc_value_y = read_adc();
  delay_us(20);

  count_timer0=0;

   if(lcdState == 4)
   {
    if(adc_value_x < 50)
    {
    zar_timer=0;
    loop =0;
    lcdState = 2;

    }
   }
  }
  GIE  = 1;
  TOIF = 0;
}
void main()
{
  lcd_init();
  ADCON0 = 0xC5;
  ADCON1 = 0x04;

  OPTION_REG = OPTION_REG && 0xD7;
  TIMER0 = 0;

  T1CON = T1CON || 0x35;
  set_timer1(0);

  INTCON = 0xE1;
  PIE1 = 0x20;
  setup_wdt(WDT_2304MS);

  /*
   *   0 = ayarlar
   *   1 = ana menu
   *   2 = oyun
   *   3 = piano
   *   4 = oyun basla
  */

  while(TRUE)
  {
    switch(lcdState)
    {
    case 0: //ayarlar
      {
      int choose = 1, pul_count = 0, cursor = 0, loop = 1, writeLCD =
1;
        Lcd_gotoxy(1,1);
```

```
Lcd_putc("    <AYARLAR>     ");
Lcd_gotoxy(1,2);
Lcd_putc(" PUL SAYISI =      ");
Lcd_gotoxy(1,3);
Lcd_putc(" BLINK -->       ");
Lcd_gotoxy(1,4);
Lcd_putc("<Cikis    Degistir>");
while (loop)
  {restart_wdt();
    if (writeLCD)
    {
      writeLCD = 0;
      if (choose == 1)
      {
        Lcd_gotoxy(1,2);
        Lcd_putc(">");
        Lcd_gotoxy(12,2);
        if (pul_count)
        {
          printf(lcd_putc," = %d    ",pul_sayisi);
        }
        else
        {
          printf(lcd_putc," = %d    ",pul_sayisi);
        }
        Lcd_gotoxy(1,3);
        Lcd_putc(" ");
        Lcd_gotoxy(11,3);
        if (cursor) Lcd_putc("KAPALI    ");
        else Lcd_putc("ACIK     ");
        Lcd_gotoxy(1,2);
      }
      if (choose == 2)
      {
        Lcd_gotoxy(1,2);
        Lcd_putc(" ");
        Lcd_gotoxy(12,2);
        if (pul_count) printf(lcd_putc," = %d
",pul_sayisi);
        else printf(lcd_putc," = %d    ",pul_sayisi);
        Lcd_gotoxy(1,3);
        Lcd_putc(">");
        Lcd_gotoxy(11,3);
        if (cursor) Lcd_putc("KAPALI    ");
        else Lcd_putc("ACIK     ");
        Lcd_gotoxy(1,3);
      }
    }
    if (adc_value_x > 200)
    {
      while(adc_value_x > 200)
      {
        delay_ms(10);
      }
      if (choose == 1) choose = 2;
      else choose = 1;
      writeLCD = 1;
    }

    if (adc_value_y < 50)
    {
      while(adc_value_y < 50)
```

```
{
   delay_ms(10);
}

if (choose == 1)
{
   if (pul_count)
   {
      pul_count = 0;
      if(pul_sayisi<=20)
      pul_sayisi++;
      if(pul_sayisi>20)
      pul_sayisi=5;
   }
   else
   {
      pul_count = 1;

   }
}
if (choose == 2)
{
   if (cursor)
   {
      cursor = 0;
      lcd_setcursor_vb(TRUE, TRUE);
   }
   else
   {
      cursor = 1;
      lcd_setcursor_vb(FALSE, FALSE);
         }
      }
      writeLCD = 1;
    }

    if (adc_value_x < 50)
    {
      while(adc_value_x < 50)
      {
        delay_ms(10);
      }
      loop = 0;
      writeLCD = 0;
      lcdState = 1;

    }
    write_eeprom(0,pul_sayisi);
  }
  break;
} //case0
case 1: //ANA MENU
{
  short loop = 1;
  Lcd_gotoxy(1,1);
  Lcd_putc("    <ANA MENU>    ");
  Lcd_gotoxy(1,2);
  Lcd_putc("Menuleri gezmek icin");
  Lcd_gotoxy(1,3);
  Lcd_putc("Joystigi  kullanin. ");
  Lcd_gotoxy(1,4);
  Lcd_putc("<AYARLAR      OYUN>");
```

```
  while (loop)
    {
      if (adc_value_x > 200)
      {
        while(adc_value_x > 200)
        {
          delay_ms(10);
        }
        loop = 0;
        lcdState = 2;
      }
      if (adc_value_x < 50)
      {
        while(adc_value_x < 50)
        {
          delay_ms(10);
        }
        loop = 0;
        lcdState = 0;
      }
    }
    break;
} //case 1
case 2:
{
int choose = 1, loop = 1, writeLCD = 1, enter =0;
  Lcd_gotoxy(1,1);
  Lcd_putc("   <BARBUT GAME>   ");
  Lcd_gotoxy(1,2);
  Lcd_putc(" YENI OYUN         ");
  Lcd_gotoxy(1,3);
  Lcd_putc(" DEVAM ET          ");
  Lcd_gotoxy(1,4);
  Lcd_putc("<ANA MENU    PIANO>");
  while (loop){
  restart_wdt();
  if(0 == Cik)
  {
      if (adc_value_x > 200)
      {
        while(adc_value_x > 200)
        {
          delay_ms(10);
        }
        loop = 0;
        lcdState = 3;
      }

      if (adc_value_x < 50)
      {
        while(adc_value_x < 50)
        {
          delay_ms(10);
        }
        loop = 0;
        lcdState = 1;
      }
      if(adc_value_y < 50)
      {
       Cik =1;
       enter = 1;
      }
```

```
}
if(Cik == 1)
{
if (writeLCD)
    {
      writeLCD = 0;
      if (choose == 1)
      {
        Lcd_gotoxy(1,2);
        Lcd_putc(">YENI OYUN         ");
        Lcd_gotoxy(1,3);
        Lcd_putc(" DEVAM ET          ");
        Lcd_gotoxy(1,2);
      }
      if (choose == 2)
      {
        Lcd_gotoxy(1,2);
        Lcd_putc(" YENI OYUN         ");
        Lcd_gotoxy(1,3);
        Lcd_putc(">DEVAM ET          ");
        Lcd_gotoxy(1,3);
      }
      /*if(choose == 3)
      {
        Lcd_gotoxy(1,2);
        Lcd_putc(">AYARLAR           ");
        Lcd_gotoxy(1,3);
        Lcd_putc("                   ");
      }*/
    }
    if (adc_value_x > 200)
    {
      while(adc_value_x > 200)
      {
        delay_ms(10);
      }
      /*if (choose == 1) choose = 2;
      else choose = 1;*/
      choose++;
      if(3 == choose)
      choose =1;
      writeLCD = 1;
      enter = 0;
    }
    if (adc_value_x < 50)
    {
      while(adc_value_x < 50)
      {
        delay_ms(10);
      }
      loop = 0;
      writeLCD = 0;
      lcdState = 2;
      Cik = 0;
    }
    if(adc_value_y < 50 && enter == 0)
    {
      while(adc_value_y < 50)
      {
        delay_ms(10);
      }
      loop = 0;
```

```
            writeLCD = 1;
            Cik = 0;
            /*if(3 == choose)
            lcdState = 7;*/
            if(2 == choose)
            devamet=1;
            else devamet=0;
            lcdState = 4;
          }
           }
            }
      break;
} //case2
case 3:   //piano
{restart_wdt();
       int loop = 1, writeLCD = 1, Cik =0;
  Lcd_gotoxy(1,1);
  Lcd_putc("      <PIANO>      ");
  Lcd_gotoxy(1,2);
  Lcd_putc(" CALMAK ICIN      ");
  Lcd_gotoxy(1,3);
  Lcd_putc(" GIRIS YAPINIZ.    ");
  Lcd_gotoxy(1,4);
  Lcd_putc("<OYUN             ");
   while (loop){
   restart_wdt();

   if(0 == Cik)
   {
        if (adc_value_x < 50)
        {
            while(adc_value_x < 50)
            {
              delay_ms(10);
            }
            loop = 0;
            lcdState = 2;
        }
        if(adc_value_y < 50)
        {
         Cik =1;
        }
}
if(Cik == 1)
{
 play_piano();
 putc(1);
if (writeLCD)
      {
         writeLCD = 0;
      }

      if (adc_value_x < 50)
      {
         while(adc_value_x < 50)
         {
           delay_ms(10);
         }
         loop = 0;
         writeLCD = 0;
         lcdState = 3;
         Cik = 0;
```

```
                putc(0);
              }
               }
                }
            break;
      }
        case 4: // oyuna basla
        {restart_wdt();
         set_tris_d(0x00);
         output_d(0);
         //int choose = 1, pul_count = 0, cursor = 0, loop = 1,
writeLCD = 1;
         int oyuncu=1,game=0;
         loop =1,zar_timer=1;
         if(devamet ==0 || pul_sayisi_1 ==0 || pul_sayisi_2 ==0)
         {
         pul_sayisi_1 = read_eeprom(0);
        pul_sayisi_2 = read_eeprom(0);
         }
         else if(devamet == 1)
         {
        pul_sayisi_1 = read_eeprom(1);
        pul_sayisi_2 = read_eeprom(2);
        oyuncu      = read_eeprom(3);
        if(oyuncu == 2) oyuncu = 1;
        else oyuncu = 2;
         }
        Lcd_gotoxy(1,1);
        Lcd_putc("xxxxxxxxxxxxxxxxxxx");
        Lcd_gotoxy(1,2);
        Lcd_putc("x     BARBUT     x");
        Lcd_gotoxy(1,3);
        Lcd_putc("x      GAME      x");
        Lcd_gotoxy(1,4);
        Lcd_putc("xxxxxxxxxxxxxxxxxxx");
        for(int i =0;i<8;i++)
        {
        if (adc_value_x < 50)
            {
               while(adc_value_x < 50)
               {
                 delay_ms(10);
               }
               loop = 0;
               lcdState = 2;
            }
        Lcd_gotoxy(1,1);
        Lcd_putc("xxxxxxxxxxxxxxxxxxx");
        Lcd_gotoxy(1,4);
        Lcd_putc("xxxxxxxxxxxxxxxxxxx");
        delay_ms(50);
        Lcd_gotoxy(1,1);
        Lcd_putc("////////////////////");
        Lcd_gotoxy(1,4);
        Lcd_putc("////////////////////");
        delay_ms(50);
        restart_wdt();
        }
        Lcd_gotoxy(1,1);
        Lcd_putc("xxxxxxxxxxxxxxxxxxx");
```

```
        Lcd_gotoxy(2,2);
        Lcd_putc("1 PLAYER   2 PLAYER");
        Lcd_gotoxy(1,3);
        printf(lcd_putc,"x   %d        %d
x",pul_sayisi_1,pul_sayisi_2);
        Lcd_gotoxy(1,4);
        Lcd_putc("xxxxxxxxxxxxxxxxxxxx");


          while (loop)
          {
           //game function
           restart_wdt();
           Lcd_gotoxy(1,3);
           printf(lcd_putc,"    %d        %d
",pul_sayisi_1,pul_sayisi_2);


           if(oyuncu == 1)
           {
           lcd_gotoxy(2,3);
           Lcd_putc(">");
           lcd_gotoxy(14,3);
           Lcd_putc(" ");
           }
           else
           {
           lcd_gotoxy(14,3);
           Lcd_putc(">");
           lcd_gotoxy(2,3);
           Lcd_putc(" ");
           }
```

```
 if (adc_value_x < 50)
 {
    while(adc_value_x < 50)
    {
      delay_ms(10);
    }
    loop = 0;
    lcdState = 2;
 }
if(imu >= 15)//imu)
{


  zar1 = (int)(TIMER0/50) + 1;
  zar2 = (int)(get_timer1()/13100) + 1;


  if(1 == oyuncu)
  {
    if(zar1 == zar2)
    {
      if(zar1 == 1 || zar1 == 2 || zar1 == 4)
      {
      pul_sayisi_2 += pul_sayisi_1;
      pul_sayisi_1 =0;
      }
      else
      {
      pul_sayisi_1 += pul_sayisi_2;
      pul_sayisi_2 =0;
      }
      } // eşit zar
     else
```

```
    {
     if((zar1 + zar2)% 2==1)
     {
     pul_sayisi_1++;
     pul_sayisi_2--;}
     else
     {
     pul_sayisi_1--;
     pul_sayisi_2++;}
    }
   } //player 1
   if(2 == oyuncu)
  {
   if(zar1 == zar2)
   {
     if(zar1 == 1 || zar1 == 2 || zar1 == 4)
     {
     pul_sayisi_1 += pul_sayisi_2;
     pul_sayisi_2 =0;
     }
     else
     {
     pul_sayisi_2 += pul_sayisi_1;
     pul_sayisi_1 =0;
     }
     } // eşit zar
     else
    {
     if((zar1 + zar2)% 2==1)
     {
     pul_sayisi_2++;
     pul_sayisi_1--;}
     else
     {
     pul_sayisi_2--;
     pul_sayisi_1++;}
    }


   } //player 2
   for(int r =0;r<18;r++)
  {
 output_high(zarbir);
 output_d(0xFF);
 delay_ms(20);
 output_low(zarbir);

 output_high(zariki);
 output_d(0xFF);
 delay_ms(20);
 output_low(zariki);

 output_high(zarbir);
 output_d(0x00);
 delay_ms(20);
 output_low(zarbir);

 output_high(zariki);
 output_d(0x00);
 delay_ms(20);
 output_low(zariki);
 restart_wdt();
```

```
  if(0 == zar_timer)
  break;
  }

  for(r =0;r<75;r++)
  {
  output_high(zarbir);
  output_d(zar_tablo[zar1]);
  delay_ms(20);
  output_low(zarbir);

  output_high(zariki);
  output_d(zar_tablo[zar2]);
  delay_ms(20);
  output_low(zariki);
  imu =0;
  if(oyuncu == 1)
  oyuncu = 2;
  else oyuncu = 1;
  restart_wdt();
  if(0 == zar_timer)
  break;
  }
  write_eeprom(1,pul_sayisi_1);
  write_eeprom(2,pul_sayisi_2);
  write_eeprom(3,oyuncu);
  } //imu

 if(pul_sayisi_1 == 0 || pul_sayisi_2 == 0)
{
  if(0 == zar_timer)
  break;
  kazanan();
  for(int u =0;u<30;u++)
  {
  restart_wdt();
  delay_ms(10);
  }
  loop = 0;
  lcdState = 2;
  game=1;
  Cik = 1;
  }
  }
  break;
  }
 } //switch
 }
}
```

Master.c

```
#include <16F877A.h>
#device ADC=8

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES NOBROWNOUT            //No brownout reset
#FUSES NOLVP                 //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O

#use delay(crystal=4000000)
#use rs232(baud=9600, xmit=pin_c6, rcv= pin_C7, parity = N, stop = 1)
```

```
#define La PIN_B0
#define Si PIN_B1
#define D0 PIN_B2
#define Re PIN_B3
#define Mi PIN_B4
#define Fa PIN_B5
#define Sol PIN_B6
#define La2 PIN_B7

#define Hoparlor PIN_E0

#byte T1CON = 0x10
#byte TMR1L = 0x0E
#byte TMR1H = 0x0F

#byte INTCON = 0x0B
#bit GIE = INTCON.7
#bit PEIE = INTCON.6

#byte PIE1   = 0x8C
#byte PIR1   = 0x0C
//#bit  TMR1E  = PIE1.0
#bit  TMR1IF = PIR1.0
#bit  RCIF   = PIR1.5
#bit  TXIF   = PIR1.4

//Imu registers
#use i2c(Master,Fast,sda=PIN_C4,scl=PIN_C3)
#define W_DATA       0xD0
#define R_DATA       0xD1
#define PWR_MGMT_1    0x6B
#define PWR_MGMT_2    0x6C
#define CONFIG       0x1A
#define ACCEL_CONFIG  0x1C
#define ACCEL_XOUT_H  0x3B
#define ACCEL_XOUT_L  0x3C

void MPU6050_write(int add, int data)
{
  i2c_start();
  i2c_write(W_DATA);
  i2c_write(add);
  i2c_write(data);
  i2c_stop();
}

int16 MPU6050_read(int add)
{
  int retval;
  i2c_start();
  i2c_write(W_DATA);
  i2c_write(add);
  i2c_start();
  i2c_write(R_DATA);
  retval = i2c_read(0);
  i2c_stop();
  return retval;
}
```

```
void MPU6050_init()
{
  MPU6050_write(PWR_MGMT_1, 0x80);
  delay_ms(100);
  MPU6050_write(PWR_MGMT_2 , 0x00);
  delay_ms(100);
  MPU6050_write(CONFIG     , 0x01);
  delay_ms(10);
  MPU6050_write(ACCEL_CONFIG, 0x98);
}

float MPU6050_get_Ax()
{
  signed int8 A_data_x[2];
  signed int16 accel_value_x;
  A_data_x[0] = MPU6050_read(ACCEL_XOUT_H);
  A_data_x[1] = MPU6050_read(ACCEL_XOUT_L);
  accel_value_x = make16(A_data_x[0], A_data_x[1]);
  float acx = (float)accel_value_x/(float)16384;
  return acx;
}

int Ax,gelen =0;

#priority rda,timer1

#int_timer1
void kesme_timer1()
{
  set_timer1(63535);
  TMR1IF = 0;
  GIE = 1;
}

#int_rda
void serial_kesme()
{
  gelen = getc();
  RCIF = 0;
  GIE = 1;
}

void main()
{
  MPU6050_init();
  delay_us(20);

  T1CON = T1CON || 0x07;
  set_timer1(63535);

  INTCON = INTCON || 0xC0;
  PIE1    = PIE1  || 0x21;

  while(TRUE)
  {
  sleep();
  Ax = MPU6050_get_Ax()
  putc(Ax);

   while(gelen)
   {
```

```
if(input(La)==0)
{
  output_high(Hoparlor);
  delay_us(1140); //1136
  output_low(Hoparlor);
  delay_us(1140);
}
if(input(Si) == 0)
{

  output_high(Hoparlor);
  delay_us(1020); //1012
  output_low(Hoparlor);
  delay_us(1020);
}
if(input(D0) == 0)
{
  output_high(Hoparlor);
  delay_us(960);//956
  output_low(Hoparlor);
  delay_us(960);
}
if(input(Re)== 0)
{
  output_high(Hoparlor);
  delay_us(860);//852
  output_low(Hoparlor);
  delay_us(860);
}
if(input(Mi)== 0)
{
  output_high(Hoparlor);
  delay_us(760);//768
  output_low(Hoparlor);
  delay_us(760);
}
if(input(Fa)==0)
{
  output_high(Hoparlor);
  delay_us(720);//716
  output_low(Hoparlor);
  delay_us(720);
}
if(input(Sol)== 0)
{
  output_high(Hoparlor);
  delay_us(640);
  output_low(Hoparlor);
  delay_us(640);
}
if(input(La2)==0)
{
  output_high(Hoparlor);
  delay_us(560);//568
  output_low(Hoparlor);
  delay_us(560);
}
output_low(Hoparlor);

 }
 }
}
```