T.C.
Ege Üniversitesi
Mühendislik Fakültesi
Elektrik Elektronik Mühendisliği Bölümü

GÖMÜLÜ SİSTEMLER DERSİ PROJE RAPORU

İKİ YILAN OYUNU

KORHAN ANIL YILMAZ 05110001004

ALİ YEŞİL 05140000536

A. Proje Hakkında

Projemizi anlatmaya geçmeden önce normalde yapmayı düşündüğümüz WiFi modülü (Xbee) ile haberleşmeyi neden yapmadığımızı açıklamak isteriz. Xbee modülü hem alıcı hem verici tarafta gerekliydi sensörden gelen veriyi işleyip Xbee modülü aracılığıyla iletecek ikinci bir kart ihtiyacı vardı. Bunun dışında Xbee modüllerinin maliyetinin çok yüksek olmasından dolayı bluetooth modülü ile haberleşecek bir proje düşündük. Daha sonra biri joystick ile biri bluetooth aracılığı ile android telefondan hareket ettirilecek iki yılandan oluşan yem yeme yarışı içinde bir oyun aklımıza geldi.

B. Geliştirme Kartı Üzerinden Kullanılan Çevre Birimleri

B.1 – RCC (Reset and Clock Control)

Kullandığımız çevre birimlerine clock sinyali vermek için kullandık. RCC üzerinden GPIO, USART, EXTI gibi birimlere clock sinyali verdik.

```
void RCC_Configuration(void)
{
 /* Enable GPIO clock */
 RCC_APB2PeriphClockCmd(USARTy_GPIO_CLK | RCC_APB2Periph_AFIO, ENABLE);
 /* Enable USARTy Clock */
 RCC_APB1PeriphClockCmd(USARTy_CLK, ENABLE);
}
```

B.2 – GPIO (General Purpose Input Output)

Kullandığımız çevre birimlerinin giriş mi, çıkış mı olacağına bu bölümde karar verdik. Joystick ve LCD için STM'nin hazırladığı hazır GPIO konfigürasyon fonksiyonları kullandık. USART için PD6 pini alternate funtion olarak ayarladık.

```
void GPIO_Configuration(void)
{
 GPIO_InitTypeDef GPIO_InitStructure;

 /* Enable the USART2 Pins Software Remapping */
 GPIO_PinRemapConfig(GPIO_Remap_USART2, ENABLE);

 /* Configure USARTy Rx as input floating */
 GPIO_InitStructure.GPIO_Pin = USARTy_RxPin;
 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
 GPIO_Init(USARTy_GPIO, &GPIO_InitStructure);

 /* Configure USARTy Tx as alternate function push-pull */
```

```
  GPIO_InitStructure.GPIO_Pin = USARTy_TxPin;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(USARTy_GPIO, &GPIO_InitStructure);
}
```

B.3 – USART(Universal Synchronous/Asynchronous Receiver/Transmitter)

USART : Universal Synchronous/Asynchronous Receiver/Transmitter olarak açılır. Açılımından da anlaşılacağı üzere senkron ve asenkron olarak çalışabilir. Temelde 2 pin kullanılır. TX ve RX. TX Transmitter(Verici) RX Receiver(Alıcı) Anlamına gelir. Herhangi bir cihaz ile bağlantı yapılırken çapraz bağlantı yapılır. Yani, TX –> RX, RX –> TX şeklinde bağlantı yapılır. Bunun sebebi diğer cihazın alıcı kısmına gönderici cihazın gönderici bacağının bağlanması gerektiğidir. Yani birinin gönderici bacağı diğerinin alıcı bacağına bağlanmakta.

USART'ın ilk yapılması gereken ayarı BAUD Rate dir. Baud Rate bir nevi hızdır. Çeşitli baudlar bulunmaktadır ve donanıma göre baud hızı arttıkça iletişimde sapmalar olabilir. Bunu kullanacağımız ürünün datasheet inden bakabiliriz.

```
  USART_InitStructure.USART_BaudRate = 19200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  /* Configure USARTy */
  USART_Init(USARTy, &USART_InitStructure);
        USART_ITConfig(USARTy, USART_IT_RXNE, ENABLE);
  NVIC_EnableIRQ(USART2_IRQn);
  /* Enable the USARTy */
  USART_Cmd(USARTy, ENABLE);
```

B.4 – EXTI (External Interrupt)

Kesmelerin kurulumu için : İnterrupt line konfigure edilmeli ve enable olmalı, Interrupt Mask Register (IMR) kullanılmalı, Trigger Selection içi (RTSR, FTSR) kullanılmalıdır. PR bir nevi bayrak görevi görmektedir. 20 adet kesme hattı NVIC_IRQ ile kontrol edilebilir. Kullabdığımız dış kesmenin GPIO karşılığı EXTI10_15'tir. EXTI15_10, 10 dan 15 e kadar olan pinler için kullanılmaktadır. Diğer hatlar tek kanaldan kullanılır. EXTI birimi APB2'ye bağlıdır. Kesme rutini (handler) stm3210f10x_it.c içerisinde tanımlanmıştır. Kesme önceliğinde, düşük değerler daha önceliklidir.

Kesmenin kesilmesi için Preamption priority değeri, seçilen grup numarasına uygun şekilde belirlenmelir. Düşük preamption priority değerine sahip bir kesme diğerini kesebilir.

```
void EXTI15_10_IRQHandler(void)
{
 /* Checks whether the IOE EXTI line is asserted or not */
 if(EXTI_GetITStatus(IOE_IT_EXTI_LINE) != RESET)
 {
  if (IOE_GetGITStatus(IOE_2_ADDR, IOE_GIT_GPIO))
  {
   static JOY_State_TypeDef JoyState = JOY_NONE;

   /* Get the Joystick State */
   JoyState = IOE_JoyStickGetState();
                        /* Oyun bittiyse */
                        if(Game_State == GAME_OVER)
                        {
                                Game_Init();
                                Game_State = !GAME_OVER;
                        }
   switch (JoyState)
   {
   case JOY_RIGHT:
                                if(snake_direction != MOVE_DOWN)
                                        snake_direction = MOVE_UP;
    break;
   case JOY_LEFT:
                                if(snake_direction != MOVE_UP)
```

```
                                        snake_direction = MOVE_DOWN;
      break;
    case JOY_UP: //
                                    if(snake_direction != MOVE_RIGHT)
                                        snake_direction = MOVE_LEFT;
      break;
    case JOY_DOWN:
                                    if(snake_direction != MOVE_LEFT)
                                        snake_direction = MOVE_RIGHT;
      break;
    }

    /* Clear the interrupt pending bits */
    IOE_ClearGITPending(IOE_2_ADDR, IOE_GIT_GPIO);
    IOE_ClearIOITPending(IOE_2_ADDR, IOE_JOY_IT);
  }
  /* CLear all pending interrupt */
  IOE_ClearGITPending(IOE_2_ADDR, ALL_IT);
  IOE_ClearIOITPending(IOE_2_ADDR, IOE_JOY_IT);

  /* CLear all pending interrupt */
  IOE_ClearGITPending(IOE_1_ADDR, ALL_IT);

  EXTI_ClearITPendingBit(IOE_IT_EXTI_LINE);
 }
}
```

C. Kullanılan LCD Fonksiyonları

Başlangıç olarak LCD kullanımını araştırdık ve örnekleri inceledik ilk aşama olarak LCD üzerinde bir piksel yakmak yeterli olacaktı daha sonra gerekli kodlarla istediğimiz işlemleri yapabilecektik. Bunun için LCD kütüphanesinden kullanacağımız komutları inceledik.

C.1 - LCD_Clear(LCD_COLOR_WHITE) Fonsiyonu

Bu komut LCD ekranını temizlemek için kullanılıyor. İçinde yazan renk neyse LCD o renkle dolduruluyor. White ekranın temizlendiğinde beyaz olması için kullanılıyor

C.2 - LCD_SetFont(&Font16x24) Fonksiyonu

İçindeki referans ile ekrana yazılacak karakterlerin yazı boyutunu ayarlıyor. Fakat tanımlanan değerler dışında istediğimiz boyutu ayarlamak için kütüphaneye ek yapmamız gerekebiliyor. Burada tanımlanan boyutlar 8x8 8x12 12x12 16x24

C.3 - LCD_SetTextColor(LCD_COLOR_RED)

Yazı rengini değiştirmek için kullanılıyor ve eğer renk değiştirilmezse en son ayarlanan renk değeri hafızada tutuluyor bu sebeple rengi değiştirmeden başka bir yazı yazdırılırsa en son renk neyse o renkle yazmaya devam ediyor. Programlama sırasında en son rengi beyaz ayarlamıştık ve yazı yazdıramadığımızı düşünüyorduk daha sonradan bunu fark ettik ve kodumuzu düzelttik.

C.4 - LCD_DisplayStringLine(Line0, (uint8_t*)"YAZI.")

Ekranda belli bir satırda yazı yazdırmak için kullanılıyor. Fakat dikkat edilmesi gereken şey ekran boyutu dikkate alınarak ve ayarlanan yazı boyutu da dikkate alınarak oluşan satır sayısını bilebiliriz. En alt satır için girilmesi gereken değer böyle bulunabiliyor.

D. Oyun Algoritması

D.1 – Oyun Altyapısının Oluştrulması

Ekranda bir pikseli yaktıktan sonra yılan çizimi ve hareket kısımlarını ayarlamak geliyor. Burada yılanın her bir parçasının yapılan denemeler ile 4x4 olmasına karar verdik yeterli bir boyut oluyordu. Daha sonra oyun alanımızı ayarlamak vardı bunun için ekran boyutumuzun 240x320 olduğunu bildiğimiz için her taraftan 20 piksel boşluk bırakmanın uygun olacağını düşündük. Bu sebeple bir çizgi çizdirdik ve sınırlar 19-219 ile 19-299 arası oldu sonuçta her taraftan 20 piksel uzaklıkta bir çizgi çizdirmiş olduk.

```
for(i = 19; i < 220; ++i)
 PutPixel(i, 19);
for(i = 19; i < 220; ++i)
 PutPixel(i, 299);
for(i = 19; i < 300; ++i)
 PutPixel(219, i);
for(i = 19; i < 300; ++i)
 PutPixel(19, i);
```

Sınırları ayarladıktan sonra üstte yılanların skorlarını yazdırmayı ve altta da isimlerimizi yazdırmayı düşündük ve oyun tasarımına geçtik.

Ekranda 4x4 pikseli her seferinde ayrı bir kodla yapmamak için 4*4 pikseli istediğimiz renge boyayacak ve gerekirse renk beyaz ayarlanarak silecek bir kod yazdık.

```
void Put_4px_Block(uint16_t x, uint16_t y, uint8_t sr, uint16_t color)
{
if(sr == PIXEL_SET)
{
 LCD_SetTextColor(color);
}
else if(sr == PIXEL_RESET)
{
 LCD_SetTextColor(LCD_COLOR_WHITE);
}
PutPixel(x, y);
PutPixel(x, y + 1);
```

```
PutPixel(x, y + 2);
PutPixel(x, y + 3);
PutPixel(x + 1, y);
PutPixel(x + 1, y + 1);
PutPixel(x + 1, y + 2);
PutPixel(x + 1, y + 3);
PutPixel(x + 2, y);
PutPixel(x + 2, y + 1);
PutPixel(x + 2, y + 2);
PutPixel(x + 2, y + 3);
PutPixel(x + 3, y);
PutPixel(x + 3, y + 1);
PutPixel(x + 3, y + 2);
PutPixel(x + 3, y + 3);
}
```

Bu kodda alınan X ve Y konumundan başlayarak 3 sağa 3 yukarı pikseller istenilen renge boyanıyor. Böylece 4*4 boyutunda bir piksel oluşuyor.

Yılanın her parçasının X ve Y konumlarını tutmak için iki ayrı dizi oluşturuldu dizilerden birine snake birine Mouse dedik böylece kafamızın karışmayacağını düşündük.

D.2 – Yılanın Hareket Ettirilmesi

Yılanın konumlarını tutan dizide yılan yukarı doğru çizilecekse X değeri sabit oluyor, Y değerleri 4ün katları oranında azaltılıyor. Yılan sağa ya da sola çizilecekse Y sabit kalıyor X +- 4 ün katları oranında değişiyor. Yılanın hareketi ise dizideki tüm değerler kaydırılarak sağlanıyor en üstteki konum değeri tutuluyor daha sonra tüm konum değerleri hareket yönüne göre değiştiriliyor ve en üstteki tutulan konum temizleniyor böylece yılan hareket etmiş oluyor. Tabi bu hareketin düzenli olabilmesi için işlemlerin belli süre aralıklarla yapılması gerekiyor bunun için Timer kullanıldı ve işlemler bir for döngüsü ile belli süre aralıklarla yapıldı. Gecikmeyi sağlayan fonksiyon aşağıdadır.

```
for(i = 0; i < 0xbddff; ++i);
```

D.3 – Yılanın Oyuncu Tarafından Kontrolü

Yılanın hareketi de çözüldükten sonra komutların nasıl işleme alınacağı konusu var bunun için de oyunda problem olmaması ve kesilme olmaması için hareketleri kesmeler ile almayı düşündük Joystickten gelen kesme EXTI15_10 ile Bluetooth'tan gelen ve USART2 ile alınan bilgi de usart kesmesi ile işleme sokuldu. USART2 kesmesinin sebebi Bluetooth modülünü USART2 nin Rx bacağı olan PD6 pinine bağlamış olmamızdır.

Daha sonra kesme ayarları gelmekte yazılmış olan bir program normal akışında devam ederken, kurulu olan herhangi bir olaydan dolayı oluşan özel durum için; o an yapmakta olan işini bırakıp, bu özel durumun belirtmiş olduğu fonksiyona giderek, o fonksiyonu yerine getirmesi ve daha sonra kaldığı yerden normal akışına devam etmesine kesme denir.

Örneğin; bir otelde ışıkları kontrol eden bir otomasyon sistemi olduğunu düşünelim. Bu otomasyon sistemine bir de duman sensörü kesme olarak tanımlansın. Normal durumlarda bu sistem, sadece oteldeki ışıkları kontrol etmekte. Ne zamanki otelde yangın çıktı ve duman sensörü bu dumanı algılayıp işlemcimize haber verdi, işte o anda işlemcinin işi ne olursa olsun, o işi bırakıp alarmı çalıştırmak için belirtilen fonksiyona gider ve alarmı çalıştırır. Bir başka örnekte; bir aracın elektronik kısımlarını kontrol eden bir işlemci, ne zaman bir kaza yapsa; o anda işini bırakıp hava yastıklarının şişmesini sağlamakta.

Kesmeler bir sistemin belkemiğidir diyebiliriz. Çok önemli olayları, hiç bir aksaklık yaşamadan yerine getirmek için sistemlerimizde kesme kullanmamız şart. Biz de yılan hareketlerinin programda uygulanmasını kesme ile sağladık. Joystick ya da bluetoothtan veri geldiğinde kesmeye giriliyor ve yapılan karşılaştırma sonucunda hareket sağlanıyor

Kesmelerle gelen hareket bilgisinden sonra yılanın hareketlerini ayarlama kısmı geliyor. Kesme ile yön değiştirme kısmından önce yılanın hareket ettirilmesini anlatacağız. Yılan hareketi yılanın yönü için gerekli kontrolleri yaptıktan sonra yılanın adreslerinin tutulduğu dizideki değerlerin yılanın hareketini sağlayacak şekilde kaydırılması sonrasında yeni adres bilgileri ile yılanın çizilmesi ve en sondaki 4*4'lük pikselin silinmesi ile yapılmaktadır. Bu sayede yılan hareket ediyor gibi görünmektedir. Yılanın yön değişikliği Joystick ya da telefondan kesme ile yön bilgisi alındığında seçme(switch(d)) kısmında ilgili kod dizini sayesinde yılanın başlangıç adresi alınan yönü sağlayacak şekilde X veya Y değerlerinin 4 azaltılması ya da 4 arttırılması ile sağlanmaktadır Aşağıda bunu sağlayan kod vardır.

```
void Snake_Move(uint16_t d)
{
   uint16_t i, temp[2];
   temp[0] = snake[snake_length - 1][0];
   temp[1] = snake[snake_length - 1][1];

   /* Snake dizisinde birer eleman kaydirma */
   for(i = 1; i < snake_length; ++i)
   {
      snake[snake_length - i][0] = snake[snake_length - i - 1][0];
      snake[snake_length - i][1] = snake[snake_length - i - 1][1];
   }
/* Yon tayini */
   switch(d)
   {
      case MOVE_UP:
         snake[0][1] -= 4; break;
      case MOVE_DOWN:
         snake[0][1] += 4; break;
      case MOVE_LEFT:
```

```
            snake[0][0] -= 4; break;
        case MOVE_RIGHT:
            snake[0][0] += 4; break;
    }
    /* Yilani ciz */
    for(i = 0; i < snake_length; ++i)
        Put_4px_Block(snake[i][0], snake[i][1], PIXEL_SET, LCD_COLOR_GREEN);


    /* Yilanin son parcasini sil */
    Put_4px_Block(temp[0], temp[1], PIXEL_RESET, LCD_COLOR_WHITE);
}
```

D.4 – Rastgele Yem Oluşturulması

Ekranda rastgele yem çizilmesi de yem yenildiğinde sayaçtan alınan sayının istediğimiz aralıkları sağlayacak şekilde ayarlanması sonrasında oluşan konumdaki yerin(4*4'lük alanın) mavi renge boyanması ile yapılmaktadır. Yılanların yemi yeme kontrolü de yılanların başlangıç adresleri ile yem adreslerinin karşılaştırılması ile yapılmaktadır. Yemi yiyen yılanın boyu ve skoru bir artmaktadır ve yeni yem ekranda belirmektedir. Yeni yem koyulurken koyulacağı konumun yılanların üzerine gelip gelmediği kontrolü yapılmaktadır. Bu kontrol yem adresinin yılanların tüm adresleri ile karşılaştırılması ile yapılmaktadır.

```
void Put_A_Bait(uint16_t *x, uint16_t *y)
{
    uint16_t i;
    /* Rastgele yem yeri */
    *x = (count * 4) % 192 + 27;
    *y = (count * 4) % 272 + 27;


    /*Yem yilanin uzerine mi geldi */
    for(i = 0; i < snake_length; ++i)
    {
        if(*x == snake[i][0] && snake[i][1] == *y)
        {
            *x = (count * 4) % 192 + 27;
            *y = (count * 4) % 272 + 27;
            i = 0;
        }
    }
```

D.5 – Kazananın Belirlenmesi

Bu bölümde verilen kodlarda yazan loser değişkeni 1 veya 2 değerini almaktadır. Eğer değer 1 ise oyunu kazananın yeşil yılan, 2 ise de oyunu kazananın mor yılan olduğu anlaşılmaktadır. Oyun bitişinde bu karşılaştırmalar ile kazanan oyuncu ekrana yazılmaktadır.

D.5.1 – Yılanın Kendi Kendine Çarpması

Yılanın kendi kendine çarpması da kontrol edilmektedir bu kontrol de yılanın başlangıç adresinin yılanın tüm adresleri ile karşılaştırılması ile yapılmaktadır. Burada kendi kendine çarpan yılan da yenilmiş sayılmaktadır. Bunları sağlayan kodlar aşağıdadır.

```
for(i = 1; i < snake_length; ++i)
{
    if(snake[0][0] == snake[i][0] && snake[0][1] == snake[i][1] && Game_State != GAME_OVER)
    {
      loser = 2;
      Game_Over();
      Game_State = GAME_OVER;
    }
}
```

D.5.2 – Yılanın Sınırlara Çarpması

Yılanın sınırlara çarpma kontrolü yılanın başlangıç adresi ve sınır çizgi adreslerinin karşılaştırılmasıyla gerçekleştirilir. Burada en sağ ve en aşağı sınırlar çizgi sınırlarından dört piksel aşağıdadır. Bunun sebebi 4*4 piksel çizdirmede çizimi sağ ve aşağı doğru yapmamızdır.

```
        /* Yilanin cerceveye carpmasi */
        if((snake[0][0] == 19 || snake[0][0] == 215 || snake[0][1] == 19 || snake[0][1] == 295) &&
Game_State != GAME_OVER)
        {
            loser = 2;
            Game_Over();
            Game_State = GAME_OVER;
        }
```

D.5.3 – Yılanın Skorunun On Olması

Yılanın yem yeme kontrolü oluşturulan yem ile yılanın başlangıç adresinin karşılaştırılması ile yapılmaktadır. Eğer yılanın başlangıç X ve Y adresleri yem ile aynı ise yemi yiyen yılanın skoru bir arttırılmaktadır. Yediği yem sayısı on olan yılan oyunu kazanmaktadır.

```
/* Yilanin yemi yemesi */
if(snake[0][0] == bait[0] && snake[0][1] == bait[1] && Game_State != GAME_OVER)
{
        Score_Table(++snake_score, mouse_score);
        Snake_Add(++snake_length);
        Put_A_Bait(&bait[0], &bait[1]);
}
if(snake_score > 9)
{
        loser = 1;
        Game_Over();
        Game_State = GAME_OVER;
}
```