T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Kurnaz Alarm

Ömer ÇEVİK

Danışman
Dr. Öğr. Üyesi Alp Arslan BAYRAKÇİ

Ocak, 2020
Gebze, KOCAELİ

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Kurnaz Alarm

Ömer ÇEVİK

Danışman
Dr. Öğr. Üyesi Alp Arslan BAYRAKÇİ

Ocak, 2020
Gebze, KOCAELİ

Bu çalışma 15/01/2020 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümünde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	**Dr. Öğr. Üyesi Alp Arslan BAYRAKÇİ**	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	
Jüri Adı	**Prof. Dr. Erkan ZERGEROĞLU**	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

ÖNSÖZ

Bu raporun ilk taslaklarının hazırlanmasında emeği geçenlere, raporun son halini almasında yol gösterici olan Sayın Dr. Öğr. Üyesi Alp Arslan BAYRAKÇİ hocama ve bu çalışmayı destekleyen Gebze Teknik Üniversitesi'ne teşekkürlerimi sunarım.

Ayrıca bu zamana kadar geçirdiğim süre içerisinde bana sürekli destek veren aileme, dostlarıma ve beni bilgileri ile aydınlatıp akademik anlamda beni dolduran tüm hocalarıma saygı ve sevgilerimi sunarım.

Ocak, 2020 **Ömer ÇEVİK**

İÇİNDEKİLER

ŞEKİL LİSTESİ

KISALTMA LİSTESİ

IDE : Integrated Development Environment
LCD : Liquid Crystal Display
ST : Società Generale Semiconduttori & Thomson Semiconducteurs
DC : Direct Current

SEMBOL LİSTESİ

ÖZET

Günümüzde insanlar sabahları uyanmakta zorluk çekmektedirler ve bu sorunu alarmlarla, alarmlı saatlerle çözmeye girişmişlerdir. Ancak normalde kurulumu yapıldıktan sonra bir tuşla durdurulabilen veya ertelenebilen bu alarmlı saatler ihtiyacı karşılamamakla birlikte istenildiği sonuçlar oluşturmamaktadır. Bu nedenle donanımsal olarak daha kurnaz işler çeviren bir alarmlı saate ihtiyaç duyulmaktadır. Bu alarm çalmaya başladığı zaman kaçmaya da başlamakta ve belirli süre aralıklarla melodisi değişmekte ve durdurulabilmesi için üzerinde olan oyunu oynayabilmesi ve bitirebilmesi gerekmektedir. Bu şekilde uyuyan insan uyanmaktan başka çaresi kalmayacaktır.

SUMMARY

Today people have difficulty waking up in the morning and have attempted to solve this problem with alarms and alarm clocks. However, these alarm clocks, which can normally be stopped or postponed with a key after installation, do not meet the need, but do not produce the desired results. For this reason, there is a need for an alarm clock which makes hardware more cunning. When this alarm starts to sound, it starts to run away and its melody changes periodically and it must be able to play and end the game to stop it. In this way, sleeping people will have no choice to wake up.

1. GİRİŞ

Günümüzde insanlar sabahları uyanmakta zorluk çekmektedirler ve bu sorunu alarmlarla, alarmlı saatlerle çözmeye girişmişlerdir. Ancak normalde kurulumu yapıldıktan sonra bir tuşla durdurulabilen veya ertelenebilen bu alarmlı saatler ihtiyacı karşılamamakla birlikte istenildiği sonuçlar oluşturmamaktadır. Bu nedenle alarmların evrimleşmesi yönünde bir girişim ortaya çıkması gerekmekteydi. Kurnaz Alarm projesi ile alarmın susturulması kolay olmamakla beraber sabah uyandırmak için bire bir çözüm sunmaktadır. Alarmın çalmaya başladıktan sonra susturulacak veya erteleyebilecek bir tuşunun olmaması, alarm çalmaya başladıktan sonra hareket edip kaçmaya başlaması ve alarmın peşinden koşup yakalanılırsa bile üzerinde başlanılmış olan oyunun tamamlanması koşuluyla ancak susturulabilmesi tamamen amaçlanan uyandırma işlemini doğrular niteliktedir.

2. ARKAPLAN

Projede alarmın çalması ve hareket edebilmesi için mikroişlemci aracılığına ihtiyaç duyulmaktadır. Araştırmalar sonucunda STM32 Nucleo F401RE mikroişlemci kitine karar verildi ve STM32 mikroişlemci kitinde yer alan pinlerle istenilen modüller bağlanıldı. Gerçek zamanlı saat olarak STM32 içerisinde yer alan RTC modülü kullanıldı.

Bu modüllerden başlıca olarak buzzer modülü STM32 mikroişlemci kitine entegrasyonu sağlanılarak ses çıkartılması yönünde ilk işlemler tamamlandı.

Bir başka modül olarak ileride hareket halinde olacak olan alarmın yön değiştirebilir halde olabilmesi için gerekli olan bir adet uzaklık sensör modülü de gerekli olan araştırmalar ve implementasyonlar sonucunda STM32 mikroişlemci kitine bağlanıldı.

Saatin gösterilmesi ve alarmın durdurulabilmesi amacıyla kullanıcıya bir arayüz oluşturulması yönünde LCD 16 x 2 ekran modülü i2c protokolüyle STM32 mikroişlemci kitine bağlanıldı.

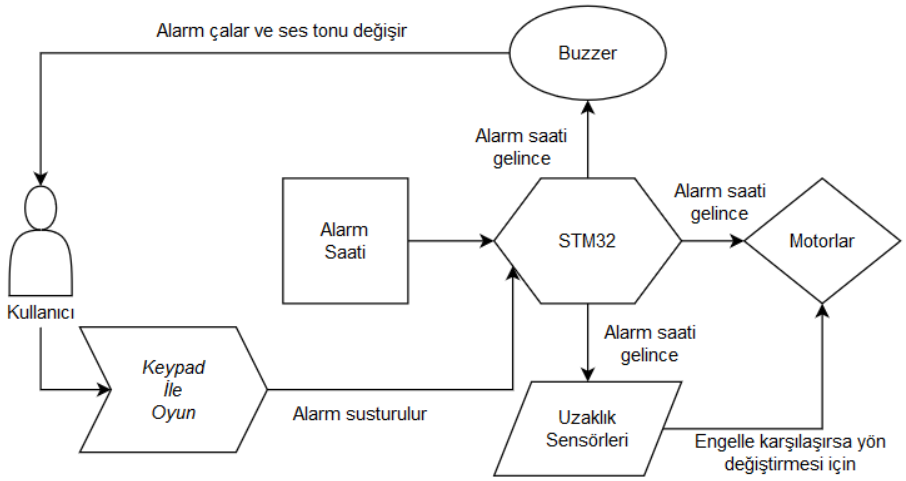
Alarm çalmaya başladığında LCD ekranda beliren basit bir toplama işlemi oyununun oynanabilmesi için keypad modülü STM32 mikroişlemci kitine entegre edildi.

Alarmın çaldığı anda kaçabilmesi için gerekli olan iki adet DC motor, DC motor sürücü kartı ve tank paletleri kullanıldı. Mikroişlemci ile motorlar DC motor sürücü kartı ile haberleşmektedir.

Yapılan bütün implementasyonlar Windows 10 işletim sistemi üzerinde, Keil µVision 5 ve STM32CubeMX IDEleri kullanılarak C programlama diliyle gerçekleşmiştir.

3. METOT

Bütün modüllerin aynı anda STM32 üzerinde çalışmasından önce modül modül tasarlanan implementasyonlar teker teker STM32 üzerinde çalıştırılması yönünde aşama kaydedildi.



ŞEKİL 1: Tasarım Planı

Tasarım planında (Şekil 1) belirtildiği üzere STM32 üzerinde bulunan RTC ile saat kurulumu ve alarm kurulumu yapılarak Alarm Saati olarak belirtilen kısımda LCD ekranda kullanıcıya ara yüz sağlanmaktadır. Kurulmuş olan alarm saatinin vakti geldiğinde buzzer, uzaklık sensörü ve DC motorları (2 tane) aktif hale gelir ve buzzer modülünde melodiler çalarken kurnaz alarmın motorları sayesinde alarm kaçmaya başlamaktadır. Alarmın ön kısmında bulunan uzaklık sensörü sayesinde eğer karşısına 20 santimetre mesafeyle bir şey çıkarsa sağa yönelimli yön değiştirilmesi sağlanmaktadır. Alarm çalmaya başladığı anda LCD ekranda yer alan saat ibaresi kaybolmakta ve yerine bir oyun gelmektedir. Bu oyun bitirilmediği takdirde alarm hiçbir zaman susmayacak, durmayacak ve kaçabildiği kadar kaçacaktır.

4 Buzzer Modülü

Buzzer modülünü kullanmak için STMCubeMX IDE'si ve Keil µVision 5 IDE'si aracılığıyla C kodunda timerlar kullanılarak belirli frekanslarla tonun başlatılıp durdurularak melodi elde edilmesi sağlanıldı. Bu tonun hızlılığı ve yavaşlığına göre farklı melodiler oluşturulması elde edildi. ToneFreq dizisi içerisinde 8 farklı frekansa ait değerler saklanmaktadır. Buzzerın çalması melody integer değişkeni ile başta yavaş daha sonra hızlı bir ritimle sağlanmaktadır.

```
int melody = 100;
void buzzerModule(void)
{
  int index = 0;
  float freq = 0;

  while(freq != -1)
  {
    freq = ToneFreq[index++];
    startTone(freq);
    HAL_Delay(melody);
    stopTone();
    HAL_Delay(melody);
  }
  index = 0;
  freq = 0;
}
```

Şekil 2: Buzzer Modülü

5 Uzaklık Sensörü Modülü

Uzaklık sensörü, belirlenmiş trigger pinini set ve reset ederken echo pininden okunan ses dalgası değerlerini ses hızıyla çarpıp mesafe ölçmek için kullanılır. Bu sensör hareket halindeki alarmın motorlarının durup yön değiştirebilmesi için gereklidir ve sayısı bir adettir. Echo pini kullanılarak dalganın çarpıp dönmesiyle oluşan değer okunmaktadır. Hesaplanılan uzaklık değeri 20 santimetreden küçük ise motorların durdurulması sonra döndürülmesi ve daha sonra ileri yönde tekrar hareketi sağlanılmaktadır.

```
// Speed of sound in cm/us
const float speedOfSound = 0.0343/2;
float distance = 0;

void ultraSonicModule()
{
    uint32_t numTicks;
    HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_RESET);
    usDelay(3);

    HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_SET);
    usDelay(10);
    HAL_GPIO_WritePin(TRIG_GPIO_Port, TRIG_Pin, GPIO_PIN_RESET);

    while(HAL_GPIO_ReadPin(ECHO_GPIO_Port, ECHO_Pin) == GPIO_PIN_RESET);

    numTicks = 0;
    while(HAL_GPIO_ReadPin(ECHO_GPIO_Port, ECHO_Pin) == GPIO_PIN_SET)
    {
      ++numTicks;
      usDelay(2); //2.8usec
    };

    distance = (numTicks + 0.0f)*2.8f*speedOfSound;

    if(distance < 20)
    {
      // Stop Motors. Turn the motor. Start motors.
      turnMotors();
    }
}
```

Şekil 3: Uzaklık Sensörü Modülü

6 LCD Modülü

LCD ekranı kullanabilmek için i2c protokolüyle ekrana belirlenen karakterler yazdırılabilmektedir. 16 x 2 LCD ekran için yapılan implementasyonlarda ilk satıra yazmak için 0x80, ikinci satıra yazmak için 0xc0 komutları ile ayarlamalar yapılabilmektedir. HAL kütüphanesinde yer alan I2C haberleşme protokolü sayesinde LCD ekrana gönderilmesi istenen değerler başarıyla sağlanmaktadır.

```
void lcd_send_cmd (char cmd)
{
  char data_u, data_l;
  uint8_t data_t[4];
  data_u = (cmd&0xf0);
  data_l = ((cmd<<4)&0xf0);
  data_t[0] = data_u|0x0C;  //en=1, rs=0
  data_t[1] = data_u|0x08;  //en=0, rs=0
  data_t[2] = data_l|0x0C;  //en=1, rs=0
  data_t[3] = data_l|0x08;  //en=0, rs=0
  HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);
}

void lcd_send_data (char data)
{
  char data_u, data_l;
  uint8_t data_t[4];
  data_u = (data&0xf0);
  data_l = ((data<<4)&0xf0);
  data_t[0] = data_u|0x0D;  //en=1, rs=0
  data_t[1] = data_u|0x09;  //en=0, rs=0
  data_t[2] = data_l|0x0D;  //en=1, rs=0
  data_t[3] = data_l|0x09;  //en=0, rs=0
  HAL_I2C_Master_Transmit (&hi2c1, SLAVE_ADDRESS_LCD,(uint8_t *) data_t, 4, 100);
}

void lcd_init (void)
{
  lcd_send_cmd (0x02);
  lcd_send_cmd (0x28);
  lcd_send_cmd (0x0c);
  lcd_send_cmd (0x80);
}

void lcd_send_string (char *str)
{
  while (*str) lcd_send_data (*str++);
}
```

Şekil 4: LCD Modülü

7 Keypad Modülü

Keypad modülü ile alarm çaldığı zaman LCD ekranda beliren oyunun oynanması sağlanmaktadır. 4 x 4 keypad için yapılan implementasyonda her bir sütun için girdi olarak kullanıcı belirlemekte ve her bir sütunun hangi satırda olduğunu kod içerisinde her satıra set ve reset ederek bulunabilmektedir.

```
char KeyPad_WaitForKeyGetChar(uint32_t  Timeout_ms)
{
  switch(KeyPad_WaitForKey(Timeout_ms))
  {
    case 0x0000:
      return 0;
    case 0x0101:
      return '1';
    case 0x0201:
      return '2';
    case 0x0401:
      return '3';
    case 0x0102:
      return '4';
    case 0x0202:
      return '5';
    case 0x0402:
      return '6';
    case 0x0104:
      return '7';
    case 0x0204:
      return '8';
    case 0x0404:
      return '9';
    case 0x0208:
      return '0';

    default:
      return 0;
  }
}
```

Şekil 5.1: Keypad Modülü

```
uint16_t  KeyPad_WaitForKey(uint32_t  Timeout_ms)
{
  uint16_t  keyRead;
  while(Timeout_ms==0)
  {
    keyRead = KeyPad_Scan();
    ultraSonicModule();
      buzzerModule();
    if(keyRead!=0)
    {
      KeyPad.LastKey = keyRead;
      return keyRead;
    }
    HAL_Delay(_KEYPAD_DEBOUNCE_TIME_MS);
  }
  KeyPad.LastKey=0;
  return 0;
}
```

Şekil 5.2: Keypad Modülü Karakter Okunması

Şekil 5.1'de gösterilmiş olan fonksiyon oyun içerisinde çağrılmakta ve kullanıcının keypad üzerindeki tuşlara basmasıyla birlikte KeyPad_WaitForKey() fonksiyonu basılan 0-9 aralığındaki değerleri return ederek LCD ekrana taşımaktadır.

Şekil 5.2'de gösterilmiş olan fonksiyon keypadden değer okunması bitene kadar KeyPad_Scan() fonksiyonuyla değerler okumakta ve interrupt olarak yapılmayan bu

sistem içerisinde buzzer ve uzaklık sensörlerinin çalışması bu döngü içerisinde sürekli olarak çağırılarak sağlanmaktadır.

```
uint16_t  KeyPad_Scan(void)
{
  uint16_t key=0;
  for(uint8_t c=0 ; c<KeyPad.RowSize ; c++)
  {
    for(uint8_t i=0 ; i<KeyPad.RowSize ; i++)
      HAL_GPIO_WritePin((GPIO_TypeDef*)_KEYPAD_ROW_GPIO_PORT[i],_KEYPAD_ROW_GPIO_PIN[i],GPIO_PIN_SET);
    HAL_GPIO_WritePin((GPIO_TypeDef*)_KEYPAD_ROW_GPIO_PORT[c],_KEYPAD_ROW_GPIO_PIN[c],GPIO_PIN_RESET);
    HAL_Delay(5);
    for(uint8_t r=0 ; r<KeyPad.ColumnSize ; r++)
    {
      if(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_COLUMN_GPIO_PORT[r],_KEYPAD_COLUMN_GPIO_PIN[r])==GPIO_PIN_RESET)
      {
        HAL_Delay(_KEYPAD_DEBOUNCE_TIME_MS);
        if(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_COLUMN_GPIO_PORT[r],_KEYPAD_COLUMN_GPIO_PIN[r])==GPIO_PIN_RESET)
        {
          key |= 1<<c;
          key |= 1<<(r+8);
          while(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_COLUMN_GPIO_PORT[r],_KEYPAD_COLUMN_GPIO_PIN[r])==GPIO_PIN_RESET)
            HAL_Delay(5);
          return key;
        }
      }
    }
  }
  return key;
}
```

Şekil 5.3: Keypad Modülü Pinlerin Set ve Reset Edilerek Okunması

Şekil 5.3 de belirtilen fonksiyon HAL kütüphanesinin yardımıyla keypade basılan bir tuş haricindeki satırları set edip basılan tuş satırını reset ettikten sonra bu tuşun hangi sütunda olduğunu read ederek basıldığı anda o tuş, o tuşa ait değerler return edilmektedir.

8 Oyun

Alarmın durdurulabilmesi için bilgisayar tarafından belirlenmiş beş rakamla kullanıcının gireceği beş rakamın birbirlerine karşılık gelen sayısal değerlerin toplamının dokuz olması gerekmektedir. Oyun amaç olarak basitçe bu işlemi gütmektedir. Alarm çalmaya başlar başlamaz playGame()(Şekil 6) fonksiyonu çağrılır ve oyunun oynanması sağlanır. Burada kullanıcı doğru değerleri girdiği takdirde LCD ekrana bu bilgi yansıtılır yanlış değerler yoksayılır. Beşte beş yapan kullanıcı alarmın susturulmasını ve motorların durdurulmasını sağlar. Saat işlevine geri dönülür.

```
char gamePC[30];
lcd_send_cmd(0x80);
sprintf(gamePC,"PC  : %d %d %d %d %d", game[0], game[1], game[2], game[3], game[4]);
lcd_send_string(gamePC);
lcd_send_cmd(0xc0);
lcd_send_string(gameUser);

while(gameContinue)
{
  while(counter < 5)
  {
    HAL_Delay(melodyChanger);
    key = KeyPad_WaitForKeyGetChar(timeout);
    if(key != ' ' && key != 0)
    {
      if(game[counter] + key - '0' == 9)
      {
        gameUser[usrIndex++] = key;
        gameUser[usrIndex++] = ' ';
        lcd_send_cmd(0xc0);
        lcd_send_string(gameUser);
        ++winner;
        ++counter;
      }
    }

    HAL_Delay(melodyChanger);
    ultraSonicModule();
    HAL_RTCEx_BKUPWrite(&hrtc, RTC_BKP_DR1, 0x32F2); // backup register
  }
  if(winner == 5)
  {
    stopMotors();
    alarmRing = false;
    gameContinue = false;
    HAL_GPIO_DeInit(Button_GPIO_Port,Button_Pin);
  }
```

Şekil 6: Oyun

9 Motorlar

Alarm çalmaya başladığı anda ekranda beliren oyunla birlikte kaçmaya çalışması gerekmektedir. Bu kaçış DC motorlara 12 volt gerilim verilerek motorların paletleri ileri doğru sürmesiyle gerçekleşmektedir. Eğer alarm karşısında bir engelle karşılaşırsa uzaklık sensörüyle haberleşen motorlar durur ve başka bir yöne dönerek hareketine devam etmektedir.

```c
#include "Motors.h"

void startForwardMotors(void)
{
  HAL_GPIO_WritePin(MOTOR_1_1_GPIO_Port,MOTOR_1_1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(MOTOR_1_2_GPIO_Port,MOTOR_1_2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_2_1_GPIO_Port,MOTOR_2_1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(MOTOR_2_2_GPIO_Port,MOTOR_2_2_Pin,GPIO_PIN_RESET);
}

void stopMotors(void)
{
  HAL_GPIO_WritePin(MOTOR_1_1_GPIO_Port,MOTOR_1_1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_1_2_GPIO_Port,MOTOR_1_2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_2_1_GPIO_Port,MOTOR_2_1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_2_2_GPIO_Port,MOTOR_2_2_Pin,GPIO_PIN_RESET);
}

void turnMotors(void)
{
  stopMotors();
  HAL_Delay(600);

  HAL_GPIO_WritePin(MOTOR_1_1_GPIO_Port,MOTOR_1_1_Pin,GPIO_PIN_SET);
  HAL_GPIO_WritePin(MOTOR_1_2_GPIO_Port,MOTOR_1_2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_2_1_GPIO_Port,MOTOR_2_1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(MOTOR_2_2_GPIO_Port,MOTOR_2_2_Pin,GPIO_PIN_SET);

  HAL_Delay(600);
  startForwardMotors();
}
```

Şekil 7: Motorlar

Şekil 7'de motorların ileri doğru hareket edilmesi startForwardMotors() fonksiyonuyla iki motorun ilk pinlerini set diğer iki pinlerini reset ederek sağlanmaktadır. Motorların durdurulması stopMotors() fonksiyonu ile tüm pinlerin reset edilmesiyle sağlanılmaktadır. Motorların sağa dönebilmesi için önce tüm motorların durdurulması daha sonra ilk motor ileri doğru diğer motor geri doğru hareket etmesi yönünde pinler set ve reset edilir ve dönüş sağlandıktan sonra tekrar ileri yönde devam etmesi için startForwardMotors() fonksiyonu çağrılır.

KAYNAKLAR

[1] https://en.wikipedi0.org/wiki/STMicroelectronics
[2] https://www.draw.io
[3] STM32 NUCLEO F401RE Data Sheet