# DWIN TOUCH PANEL DOCUMENT TR

Bu doküman, Arduino Mega 2560 ile DWIN dokunmatik ekranın ve KNX haberleşme protokolünün entegre bir sistemde nasıl kullanılacağını açıklayan teknik bir dökümandır. Sistem, kullanıcının dokunmatik panel üzerinden belirlediği değerlerin KNX protokolü üzerinden bir bina otomasyon sistemine aktarılmasını sağlar.

DWIN ekranının kullanımı için öncelikle DGUS programını indir. Yeni bir proje oluşturun ve ardından hemen “ 0#word bank generating ” ile kendinize ait harf kılavuzu oluşturun ardından DGUS programının dosya konumunu açın ve sırala şeklini en yeni diye sıraladığınız zaman “0\_DWIN\_ASC.HZK” bu şekilde dosya oluşturulacaktır bu dosyayı kendi projenizin dosya konumundan DWIN\_SET klasörünün içine taşıyın. Bunları yapmadan önce touch panelde kullanıcağınız sayfalarınızı .png şeklinde indirmeniz gerekiyor bu arayüzü oluştururken paint veya daha ayrıntılı programlardan destek alabilirsiniz sonuçta .png gibi formatta olmak zorunda yoksa DGUS programı tanımaz. Harf kılavuzunu oluşturduktan sonra welcome kısmında DWIN ICL generation kısmına tıklayıp hazırda tuttuğunuz fotoğrafları, animasyon fotoğrafları veya animasyon ikonları gibi fotoğrafları oradan sisteme yükleyebilirsiniz önemli noktalar ise ; fotoğrafları 32 klasöründe kaydetmeniz gerekli, iconları ise 40 41 42 43 44 45…. 63 maksimum eğer dgus programı indirdiğiniz iconları görmüyorsa bilin ki sayıyı yanlış veya büyük koymuşsunuz. Fotoğrafları ikonları 32.icl gibi dosyada kendi projenizin içine kaydetmiş olacaktır. Unutmayın fotoğraflar 00,01,02,03 gibi başlayacak aynı şekilde animasyonlu görüntüler ve ikonlarda 00,01,02,03,04 gibi sayılarla başlamak zorunda yoksa DGUS programı onları configure edemiyor.

Fotoğrafları yani sayfaları DGUS programına indirdikten sonra artık fotoğraf düzenlemeye geçebiliriz yani dokunmatik dokunuşu, RTC display gibi öğeler eklememiz gerekiyor. Dokunmatiği “Basic Touch Module” ile nereye dokunup işlem yapıcaksak oraya sürükleyin (eğer ekranınızda pil saati için yer yoksa adresiniz 0x0010 eğer pil saati için batarya yeriniz var ise adresiniz 009C ) ve page switchingden ise hangi sayfaya gideceğini belirleyin. Geçtiğiniz sayfayı örnek olarak fan sayfası olduğunu varsayalım animasyonlu ikonunuzun olduğunu varsayalım + ve – kutupları olacak bunları da Incremental Adjusment ile methodunu belirleyebilirsiniz limitleri yani nereye kadar çıkıcağını vs ++ mı – mi olacak şekilde burada en önemli konu VP ( 0x ) dediği yer kendinize bir VP adresi atamanız gerekli o dokunmatik modülüne dokunduğunda bir şeylerin olması gerekli o yüzden örnek olarak 1200 verebilirsiniz. Data Variables Display ile de nerenin arttıcağını koyacaksınız tabi bununda VP adresini 1200 yapmayı unutmayın bunların hepsini not almak zorundasınız çünkü hava durumui klima, fan gibi öğeleri bu adreslerle iletişim halinde kalarak yönetiyorsunuz. SP var bu da system pointer bunlarla ilgili biz mühendislerin işi yok bunların sistemin parametreleri default ayarları, bizim işimiz VP adresleri ile ilgili. RTC display kısmı var benim dwin ekranımda pil bölümü yoktu ben işlemciden ekrana her bir saniyede güncelleme atıp tarihi saati güncelliyorum eğer sizin kullandığınız ekran pil saati var ise bunlara gerek yok.

Dokunmatik ayarları, VP adreslerini hallettikten sonra sol üstten file kısmına gelip, save ardından generation yapıyoruz. Bunları yaptıktan sonra config generator diyoruz, önümüze sekme çıkıcaktır. CFG Edit diye bende T5L modülü var ve burada tüm ayarlar oluyor. CRC kullanıcaksanız on veya off işaretleyin. Burada en önemlisi **Touch-sensitive variable changes update kısmını kesinlikle ve kesinlikle Auto** yapmak zorundasınız eğer yapmazsanız ekranda vp adresleri ile ilgili güncelleme yaptığınızda işlemcinize aktaramazsınız yani fan derecesini 1 arttırdığınızda konuşma için işlemcinize paket gitmesi gerekiyor yapmazsanız gitmez ve gerekli olan güncellemeyi alamazsınız. Baud rate olarak 115200 olarak geliyor fakat serial monitörde en iyi hız aralığı 9600 yapmayı unutmayın. Ses açıp kapatma butonu var istediğinizi yapabilirsiniz. Bunlara kendi ayarlarınızı verdikten sonra sağ altta **NEW CFG butonuna tıklayıp kendi projenizin klasörüne gidip DWIN\_SET klasörünün içine T5LCFG.CFG** şeklinde kaydedin.

Artık klasörünüz hazır bir şekilde bekliyor ekrana bunları aktarmak için maksimum 16gb’lik bir micro SD kart gerekiyor. SD kartı bilgisayarınıza takın ve sadece projenizin içinde ki DWIN\_SET klasörünü SD karta yükleyebilirsiniz yüklendikten sonra hafıza kartını güvenli bir şekilde çıkartın. Ekranın gücünü kesin ve SD kartınızı ekranınıza takın, ekranınıza gücü verin. Bu işlemi yaparken mavi ekranda yazılar görünecektir dosyaların indirildiği görünecektir. Ekranda indirme tamamlandı diyene kadar bekleyin. İndirdikten sonra ekranın tekrardan gücünü kesip güce bağlayın ve arayüzün indirilmesi başarıyla tamamlanmıştır.

Kodumuza gelecek olursak ben arduino mega 2560 ile Serial2 ile ekran arduino iletişimi gerçekleştirdim. DWIN ekranı UART üzerinden seri haberleşme protokolü kullanarak Arduino ile iletişim kurar. Veri paketleri aşağıdaki şekilde yapılandırılmıştır:

• Header1: 0x5A  
• Header2: 0xA5  
• Data Length: Veri uzunluğu (örneğin 0x05)  
• Komut Byte: 0x82 (yazma), 0x83 (okuma)  
• VP Address: 2 byte (örneğin 0x12 0x00)  
• Veri: 2 byte (örneğin 0x00 0x03)

Okuma komutu gönderildiğinde ekran aşağıdaki gibi bir yanıt döner:  
0x5A 0xA5 0x06 0x83 <VP Adresi High> <VP Adresi Low> <Veri High> <Veri Low>

DWIN paneli içerisindeki nesneler her biri kendine özgü VP adresine sahiptir. Bu adresler, Arduino üzerinden hangi değerlerin hangi bileşene ait olduğunu tanımlamak için kullanılır.

Örnek VP Adresleri:  
• Fan Hızı: 0x1200  
• Klima Sıcaklık: 0x1400  
• Perde: 0x1500  
• Dimmer: 0x1600  
• Bildirim: 0x1000

Kod, iki ana işlevi yerine getirir:  
1. DWIN ekranı ile veri alışverişi yaparak kullanıcının belirlediği değerleri VP adreslerinden okur.  
2. Bu değerleri uygun KNX formatına dönüştürerek KNX hattına telegram olarak gönderir.

Kullanılan kütüphaneler:  
#include <Arduino.h>  
#include <KnxDevice.h>  
#include <Cli.h>

Arduino ile DWIN ekranın haberleşmesinden sonra ekranda güncellenen verileri telegram göndermemiz gerekliydi. Bu kütüphaneleri kullanarak telegram gönderimini gerçekleştirebiliriz. Tüm kodu aşağıda bırakacağım. Unutmayın bu kodu sadece Arduino IDE çalıştırır onun dışında yazılım geliştirme platformları çalıştırmıyor kütüphaneleri yükleyemiyor!

Bu uygulama, DWIN ekranı üzerinden yapılan girişlerin KNX ağına iletilmesini mümkün kılar. Daha geniş ölçekli otomasyon sistemleri için sıcaklık, perde, ışık ve fan kontrolü gibi tüm parametreler eklenerek genişletilebilir. Sistem tasarımı yapılırken VP adreslerinin çakışmamasına dikkat edilmeli ve KNX tarafında ETS yazılımıyla eşleştirme doğru yapılmalıdır.

## Buraya örnek kodu bırakıyorum, iyi çalışmalar.DWIN Touch Panel Documentation ENG

This document is a technical guide explaining how to use the DWIN touch panel, Arduino Mega 2560, and the KNX communication protocol in an integrated system. The system allows the user to send values selected on the touch screen to a building automation system via the KNX protocol.

To use the DWIN screen, first download the **DGUS Studio** software. Create a new project, and immediately generate a custom font guide by selecting **“0#word bank generating”**. Afterward, go to the software's file location, sort the files by "date modified", and you’ll see a file named **“0\_DWIN\_ASC.HZK”**. Move this file into the **DWIN\_SET** folder inside your project directory.

Before doing this, download the pages you’ll use on the touch panel in **.png format**. You can use simple tools like Paint or more advanced software to design your interface. The images must be in .png format — otherwise, DGUS won’t recognize them.

Once your font guide is ready, in the DGUS welcome section, click on **DWIN ICL generation** to load your interface images, animations, or icons. Important points to keep in mind:

* Images must be saved in folder **32**.
* Icons should be saved in folders **40 to 63** (e.g., 40, 41, ..., 63). If DGUS doesn’t recognize the icons, you probably used an incorrect or out-of-range folder number.
* The images and icons will be saved in files like 32.icl inside your project folder.
* All file names must start with numbers like **00, 01, 02...** for both images and animated icons; otherwise, DGUS cannot configure them.

After loading your interface images into DGUS, you can begin editing the visuals and adding elements such as touch modules or RTC displays. To add a touch function, use the **Basic Touch Module** and drag it to the desired screen location. If your screen doesn’t include an RTC battery, use address **0x0010**; otherwise, use **0x009C**. Then, under **Page Switching**, define which page should be opened when that touch area is pressed.

Let’s assume you're on a "Fan" page and you have a fan animation icon with "+" and "–" buttons. You can configure these using the **Incremental Adjustment** module and define the behavior, limit values, and whether it should increment or decrement. The most important part here is the **VP address** — for example, you can assign it to **0x1200**. This address allows Arduino to interact with the touch module. You also need to use the **Data Variable Display** to show the changing value, and this should use the same VP address. Keep note of all these VP addresses, as elements like temperature, fan, and curtain controls will communicate based on them.

There is also something called **SP (System Pointer)**, which refers to internal system parameters. These are not of concern to engineers — we only deal with **VP addresses**.

If your screen does not include an internal RTC battery, you can program Arduino to update the time every second via serial communication. If your screen includes a battery-powered RTC, this step is unnecessary.

After assigning all VP addresses and setting up the touch modules, go to **File > Save**, then click **Generation**. Next, select **Config Generator**. A new window will appear — in my case, it was the **T5L module**. Make all necessary settings here. If you're using **CRC**, toggle it on or off depending on your preference.

🚨 One very important setting is:  
Set **“Touch-sensitive variable changes update”** to **Auto**.  
If you don’t do this, the screen won’t send updates to the processor when a VP variable is changed. For example, if you increase the fan speed, a command must be sent to the Arduino. If Auto is not enabled, this update won’t happen.

By default, the **baud rate** is 115200, but I recommend setting it to **9600** for best performance with Serial Monitor.

Once all settings are complete, click **NEW CFG** in the bottom-right corner, go to your project folder, and save the config file as **T5LCFG.CFG** inside the **DWIN\_SET** folder.

Now your folder is ready to be transferred to the screen. You’ll need a **microSD card with a maximum of 16GB**. Insert the SD card into your computer and copy **only the DWIN\_SET folder** onto it. After copying, safely eject the SD card.

Power off your DWIN screen, insert the SD card, then power the screen back on. You will see a blue screen with installation messages. Wait until the installation is complete. Then power off and back on again — the new interface will be successfully loaded.

### Arduino – DWIN Communication

In my setup, I used **Serial2** on the Arduino Mega 2560 to establish communication with the DWIN screen using **UART (serial protocol)**. The data packet format is as follows:

• Header1: 0x5A

• Header2: 0xA5

• Data Length: e.g., 0x05

• Command Byte: 0x82 (write), 0x83 (read)

• VP Address: 2 bytes (e.g., 0x12 0x00)

• Data: 2 bytes (e.g., 0x00 0x03)

When a **read command** is sent, the screen responds as follows:

0x5A 0xA5 0x06 0x83 <VP Address High> <VP Address Low> <Data High> <Data Low>

Each component inside the DWIN panel has its own **VP address** that allows Arduino to recognize and handle the related data.

### Example VP Addresses:

* Fan Speed: 0x1200
* AC Temperature: 0x1400
* Curtain: 0x1500
* Dimmer: 0x1600
* Notification: 0x1000

### Arduino Code Functionality

The Arduino code performs two main tasks:

1. It communicates with the DWIN screen to read user-defined values from VP addresses.
2. It converts those values into valid KNX telegrams and sends them over the KNX bus.

### Required Libraries:

#include <Arduino.h>

#include <KnxDevice.h>

#include <Cli.h>

After establishing communication between Arduino and the DWIN panel, it’s essential to send the updated screen data as KNX telegrams. These libraries allow that.

⚠️ **Note:** This code only works in **Arduino IDE**. Other software development environments cannot compile it or load the necessary libraries.

### Final Remarks

This system enables user inputs from the DWIN screen to be transmitted into a KNX network. It can be expanded to control lighting, temperature, curtains, and fans for larger-scale automation projects. While designing the system:

* Ensure **VP addresses do not conflict**.
* Use **ETS software** on the KNX side to correctly map and link addresses.

Below is a sample code block for your reference.

**Good luck and happy building!**

#include <Arduino.h>

#include <Cli.h> // https://github.com/franckmarini/Cli

#include <KnxDevice.h>

const byte TX\_PIN = 16;  // Arduino Mega TX2 pini

const byte RX\_PIN = 17;  // Arduino Mega RX2 pini

#define DWIN\_HEADER1 0x5A

#define DWIN\_HEADER2 0xA5

#define DWIN\_WRITE\_CMD 0x82      // Yazma komutu

#define DWIN\_READ\_CMD 0x83       // Okuma komutu

#define DWIN\_OK\_RESPONSE\_O 0x4F  // 'O'

#define DWIN\_OK\_RESPONSE\_K 0x4B  // 'K'

// DWIN VP adresleri

#define VP\_NOTIFICATIONS 0x1000   // Bildirim paneli

#define VP\_WEATHER 0x1100        // Hava durumu

#define VP\_FAN\_SPEED 0x1200      // Fan hızı

#define VP\_RGB\_DIMMER 0x1300     // RGB dimmer

#define VP\_AC\_TEMP 0x1400        // Klima

#define VP\_SHUTTER 0x1500        // Perde/Panjur

#define VP\_DIMMER 0x1600         // Dimmer

// Sistem min-max değerleri

const int DIMMER\_MIN = 0;

const int DIMMER\_MAX = 10;

const int RGB\_MIN = 0;

const int RGB\_MAX = 20;

const int SHUTTER\_MIN = 0;

const int SHUTTER\_MAX = 10;

const int AC\_TEMP\_MIN = 16;

const int AC\_TEMP\_MAX = 30;

const int FAN\_MIN = 0;

const int FAN\_MAX = 5;

const int WEATHER\_MIN = -15;

const int WEATHER\_MAX = 55;

#define VP\_RTC 0x0010  // Donanımsal RTC olmadan kullanılacak adres

#define MAX\_BUFFER\_SIZE 64 //veya 32 yapabilirsiniz

#define MIN\_BUFFER\_SIZE 6

unsigned long lastUserChange = 0;

const unsigned long USER\_PRIORITY\_TIME = 3000;  // Kullanıcı önceliği süresi (3 saniye) feedback ile ekrandan girilen değer karışmasın diye bu var

struct RTCData

{

  uint8\_t year;    // Yıl (00-99)

  uint8\_t month;   // Ay (01-12)

  uint8\_t day;     // Gün (01-31)

  uint8\_t week;    // Haftanın günü (0-6, 0=Pazar, 1=Pazartesi, ..., 6=Cumartesi)

  uint8\_t hour;    // Saat (00-23)

  uint8\_t minute;  // Dakika (00-59)

  uint8\_t second;  // Saniye (00-59)

};

unsigned long lastRTCUpdate = 0;  // Son RTC güncellemesi zamanı

const unsigned long RTC\_UPDATE\_INTERVAL = 1000;  // 1 saniye (milisaniye cinsinden)

RTCData currentRTC;  // Mevcut RTC değerlerini tutacak global değişken

struct VPValues

{

    uint16\_t address;

    uint16\_t lastValue;  // int'ten uint16\_t'ye değiştirildi

    bool isChanged;

};

struct DynamicBuffer

{

    byte\* data;

    int size;

    int index;

    int expectedLength;

};

DynamicBuffer buffer =

{

    .data = new byte[MAX\_BUFFER\_SIZE],

    .size = MAX\_BUFFER\_SIZE,

    .index = 0,

    .expectedLength = 0

};

VPValues vpValues[] =

{

    {VP\_FAN\_SPEED, 0, false},      // lastValue 0 olarak başlatıldı

    {VP\_WEATHER, 0, false},

    {VP\_NOTIFICATIONS, 0, false},

    {VP\_RGB\_DIMMER, 0, false},

    {VP\_AC\_TEMP, 0, false},

    {VP\_SHUTTER, 0, false},

    {VP\_DIMMER, 0, false}

};

void setRTCFromSerial();

void printHexData(const char\* message, byte\* data, int length);

void printDebugMessage(const char\* message);

void clearBuffer();

void recoverFromError();

VPValues\* findVPValue(uint16\_t vpAddress);

bool writeToVP(uint16\_t vpAddress, uint16\_t value);

bool readFromVP(uint16\_t vpAddress, uint16\_t& value);

bool setRTC(const RTCData& rtc);

bool initializeRTC();

void processIncomingByte(byte incomingByte);

void processSerialInput();

void clicommand();

void clicommand ();

void processReceivedData();

void knxEvents(byte index);

struct VPValues\* findVPValue(uint16\_t vpAddress);

void ReadFanFeedback(void);

Cli cli = Cli(Serial);

// Definition of the Communication Objects attached to the device

KnxComObject KnxDevice::\_comObjectsList[] =

{

  /\* Index 0 - Fan Hızı Sensör \*/

  KnxComObject(G\_ADDR(0,0,10), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_SENSOR),

  /\* Index 1 - Fan Hızı Logic IN \*/

  KnxComObject(G\_ADDR(0,0,14), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_LOGIC\_IN),

  /\* Index 2 - RGB Dimmer Parlaklık - Sensör \*/

  KnxComObject(G\_ADDR(1,1,20), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_SENSOR),

  /\* Index 3 - RGB Dimmer Parlaklık - Logic IN \*/

  KnxComObject(G\_ADDR(1,1,21), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_LOGIC\_IN),

  /\* Index 4 - Klima Sıcaklık Sensör \*/

  KnxComObject(G\_ADDR(1,1,40), KNX\_DPT\_9\_001 /\* 9.001 F16 DPT\_Temperature \*/, COM\_OBJ\_SENSOR),

  /\* Index 5 - Klima Sıcaklık Logic IN \*/

  KnxComObject(G\_ADDR(1,1,41), KNX\_DPT\_9\_001 /\* 9.001 F16 DPT\_Temperature \*/, COM\_OBJ\_LOGIC\_IN),

  /\* Index 6 - Perde Konumu Sensör \*/

  KnxComObject(G\_ADDR(1,1,50), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_SENSOR),

  /\* Index 7 - Perde Konumu Logic IN \*/

  KnxComObject(G\_ADDR(1,1,51), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_LOGIC\_IN),

  /\* Index 8 - Dimmer Değeri Sensör \*/

  KnxComObject(G\_ADDR(1,1,60), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_SENSOR),

  /\* Index 9 - Dimmer Değeri Logic IN \*/

  KnxComObject(G\_ADDR(1,1,61), KNX\_DPT\_5\_001 /\* 5.001 U8 DPT\_Scaling \*/, COM\_OBJ\_LOGIC\_IN),

};

const byte KnxDevice::\_comObjectsNb = sizeof(\_comObjectsList) / sizeof(KnxComObject); // do no change this code

// Callback function to handle com objects updates

void knxEvents(byte index)

{

  switch(index) {

case 1: // Fan hızı feedback - Logic IN (0/0/14)

  {

    byte fanValue;

    Knx.read(1, fanValue);

    // Yüzdelik değeri 0-5 arası fan hızına dönüştür

    byte fanLevel;

    if (fanValue <= 10) fanLevel = 0;

    else if (fanValue <= 30) fanLevel = 1;

    else if (fanValue <= 50) fanLevel = 2;

    else if (fanValue <= 70) fanLevel = 3;

    else if (fanValue <= 90) fanLevel = 4;

    else fanLevel = 5;

    Serial.print("KNX'ten Fan feedback değeri alındı: %");

    Serial.print(fanValue);

    Serial.print(" -> Fan seviyesi: ");

    Serial.println(fanLevel);

    writeToVP(VP\_FAN\_SPEED, fanLevel);

  }

  break;

    // case 3: // Klima - Logic IN

    //   {

    //     float tempValue;

    //     Knx.read(3, tempValue);

    //     Serial.print("KNX'ten Sıcaklık değeri alındı: ");

    //     Serial.println(tempValue);

    //     writeToVP(VP\_AC\_TEMP, (uint16\_t)tempValue);

    //   }

    //   break;

    // case 5: // Perde - Logic IN

    //   {

    //     byte shutterValue;

    //     Knx.read(5, shutterValue);

    //     int mappedShutter = (shutterValue \* SHUTTER\_MAX) / 100; // 0-100% arası değeri 0-10 arası değere dönüştür

    //     Serial.print("KNX'ten Perde değeri alındı: ");

    //     Serial.println(mappedShutter);

    //     writeToVP(VP\_SHUTTER, mappedShutter);

    //   }

    //   break;

    // case 7: // Dimmer - Logic IN

    //   {

    //     byte dimmerValue;

    //     Knx.read(7, dimmerValue);

    //     int mappedDimmer = (dimmerValue \* DIMMER\_MAX) / 100; // 0-100% arası değeri 0-10 arası değere dönüştür

    //     Serial.print("KNX'ten Dimmer değeri alındı: ");

    //     Serial.println(mappedDimmer);

    //     writeToVP(VP\_DIMMER, mappedDimmer);

    //   }

    //   break;

  }

}

void ReadFanFeedback(void)

{

    Knx.update(1); //indekslerinize göre burayı düzenleyin

}

void updateRTC()

{

  // Saniye artır

  currentRTC.second++;

  // Saniye taşması

  if (currentRTC.second >= 60) {

      currentRTC.second = 0;

      currentRTC.minute++;

      // Dakika taşması

      if (currentRTC.minute >= 60) {

          currentRTC.minute = 0;

          currentRTC.hour++;

          // Saat taşması

          if (currentRTC.hour >= 24) {

              currentRTC.hour = 0;

              currentRTC.day++;

              currentRTC.week = (currentRTC.week + 1) % 7;

              // Ay kontrolü

              uint8\_t daysInMonth;

              switch(currentRTC.month) {

                  case 4: case 6: case 9: case 11:

                      daysInMonth = 30;

                      break;

                  case 2:

                      // Basit artık yıl kontrolü

                      daysInMonth = ((currentRTC.year % 4) == 0) ? 29 : 28;

                      break;

                  default:

                      daysInMonth = 31;

              }

              // Gün taşması

              if (currentRTC.day > daysInMonth) {

                  currentRTC.day = 1;

                  currentRTC.month++;

                  // Ay taşması

                  if (currentRTC.month > 12) {

                      currentRTC.month = 1;

                      currentRTC.year++;

                      if (currentRTC.year > 99) currentRTC.year = 0;

                  }

              }

          }

      }

  }

  setRTC(currentRTC);

}

bool sendRTCData(const RTCData& rtc)

{

  if (!Serial2) {

      printDebugMessage("Hata: Serial2 bağlantısı yok!");

      return false;

  }

  byte sendBuffer[12] = {

      DWIN\_HEADER1,    // 0x5A

      DWIN\_HEADER2,    // 0xA5

      0x0B,            // 11 byte veri

      DWIN\_WRITE\_CMD,  // 0x82

      0x00,            // VP 0x009C (RTC)

      0x9C,

      rtc.year,

      rtc.month,

      rtc.day,

      rtc.hour,

      rtc.minute,

      rtc.second

  };

  printHexData("RTC verisi gönderiliyor: ", sendBuffer, 12);

  Serial2.write(sendBuffer, 12);

  Serial2.flush();

  unsigned long startTime = millis();

  while (millis() - startTime < 1000) {

      if (Serial2.available() >= 6) {

          byte response[6];

          Serial2.readBytes(response, 6);

          if (response[0] == DWIN\_HEADER1 &&

              response[1] == DWIN\_HEADER2 &&

              response[2] == 0x03 &&

              response[3] == DWIN\_WRITE\_CMD &&

              response[4] == DWIN\_OK\_RESPONSE\_O &&

              response[5] == DWIN\_OK\_RESPONSE\_K) {

              printDebugMessage("RTC verisi başarıyla gönderildi");

              return true;

          }

      }

      delay(1);

  }

  printDebugMessage("Hata: RTC verisi gönderilemedi!");

  return false;

}

void printHexData(const char\* message, byte\* data, int length)

{

    Serial.print(message);

    for(int i = 0; i < length; i++) {

        if(data[i] < 0x10) Serial.print("0");

        Serial.print(data[i], HEX);

        Serial.print(" ");

    }

    Serial.println();

}

void printDebugMessage(const char\* message)

{

    Serial.println(message);

}

void clearBuffer()

{

    buffer.index = 0;

    buffer.expectedLength = 0;

    memset(buffer.data, 0, buffer.size);

    printDebugMessage("Buffer temizlendi");

}

void recoverFromError()

{

    printDebugMessage("Hata kurtarma başlatılıyor...");

    clearBuffer();

    Serial2.end();

    delay(100);

    Serial2.begin(9600);

    for (size\_t i = 0; i < sizeof(vpValues)/sizeof(VPValues); i++) {

        vpValues[i].lastValue = 0;  // 0 olarak sıfırla

        vpValues[i].isChanged = false;

    }

    printDebugMessage("Hata kurtarma tamamlandı");

}

VPValues\* findVPValue(uint16\_t vpAddress)

{

    for (size\_t i = 0; i < sizeof(vpValues)/sizeof(VPValues); i++) {

        if (vpValues[i].address == vpAddress) {

            return &vpValues[i];

        }

    }

    return NULL;

}

bool writeToVP(uint16\_t vpAddress, uint16\_t value)

{

  if (!Serial2) {

      printDebugMessage("Hata: Serial2 bağlantısı yok!");

      return false;

  }

  // Değeri doğru formata dönüştür

  uint16\_t formattedValue = value;

  switch(vpAddress) {

      case VP\_FAN\_SPEED:

          formattedValue = value & 0x000F;  // Sadece ilk 4 biti kullan

          break;

      case VP\_RGB\_DIMMER:

          formattedValue = value & 0x001F;  // Sadece ilk 5 biti kullan

          break;

      case VP\_AC\_TEMP:

          formattedValue = value & 0x00FF;  // Sadece ilk 8 biti kullan

          break;

      case VP\_SHUTTER:

      case VP\_DIMMER:

          formattedValue = value & 0x000F;  // Sadece ilk 4 biti kullan

          break;

  }

  byte sendBuffer[8] =

  {

      DWIN\_HEADER1,           // 0x5A

      DWIN\_HEADER2,           // 0xA5

      0x05,                   // Veri uzunluğu (5 byte)

      DWIN\_WRITE\_CMD,         // 0x82

      (byte)(vpAddress >> 8), // VP adres high byte

      (byte)(vpAddress),      // VP adres low byte

      (byte)(formattedValue >> 8),     // Veri high byte

      (byte)(formattedValue)           // Veri low byte

  };

  printHexData("Gonderilen veri: ", sendBuffer, 8);

  Serial2.write(sendBuffer, 8);

  Serial2.flush();

  unsigned long startTime = millis();

  while (millis() - startTime < 1000) {

      if (Serial2.available() >= 6) {

          byte response[6];

          Serial2.readBytes(response, 6);

          printHexData("Alınan yanıt: ", response, 6);

          if (response[0] == DWIN\_HEADER1 &&

              response[1] == DWIN\_HEADER2 &&

              response[2] == 0x03 &&

              response[3] == DWIN\_WRITE\_CMD &&

              response[4] == DWIN\_OK\_RESPONSE\_O &&

              response[5] == DWIN\_OK\_RESPONSE\_K) {

              String message = "Yazma başarılı: " + String(formattedValue);

              printDebugMessage(message.c\_str());

              return true;

          }

      }

      delay(1);

  }

  printDebugMessage("Hata: Yazma yanıtı alınamadı!");

  return false;

}

bool readFromVP(uint16\_t vpAddress, uint16\_t& value)

{

    if (!Serial2) {

        printDebugMessage("Hata: Serial2 bağlantısı yok!");

        return false;

    }

    byte sendBuffer[7] =

    {

        DWIN\_HEADER1,           // 0x5A

        DWIN\_HEADER2,           // 0xA5

        0x04,                   // Veri uzunluğu (4 byte)

        DWIN\_READ\_CMD,          // 0x83

        (byte)(vpAddress >> 8), // VP adres high byte

        (byte)(vpAddress),      // VP adres low byte

        0x01                    // 1 word oku

    };

    printHexData("Okuma komutu: ", sendBuffer, 7);

    Serial2.write(sendBuffer, 7);

    Serial2.flush();

    unsigned long startTime = millis();

    while (millis() - startTime < 1000) {

        if (Serial2.available() >= 9) {

            byte response[9];

            Serial2.readBytes(response, 9);

            printHexData("Okuma yanıtı: ", response, 9);

            if (response[0] == DWIN\_HEADER1 &&

                response[1] == DWIN\_HEADER2 &&

                response[2] == 0x06 &&

                response[3] == DWIN\_READ\_CMD &&

                response[4] == (byte)(vpAddress >> 8) &&

                response[5] == (byte)(vpAddress)) {

                value = (response[7] << 8) | response[8];

                String message = "Okuma başarılı: " + String(value);

                printDebugMessage(message.c\_str());

                return true;

            }

        }

        delay(1);

    }

    printDebugMessage("Hata: Okuma yanıtı alınamadı!");

    return false;

}

void processIncomingByte(byte incomingByte)

{

    static unsigned long lastByteTime = 0;

    unsigned long currentTime = millis();

    if (currentTime - lastByteTime > 1000 && buffer.index > 0) {

        printDebugMessage("Veri alımı zaman aşımı!");

        recoverFromError();

        return;

    }

    lastByteTime = currentTime;

    if (buffer.index >= buffer.size) {

        printDebugMessage("Buffer taşması!");

        recoverFromError();

        return;

    }

    printHexData("Alınan byte: ", &incomingByte, 1);

    if (buffer.index == 0) {

        if (incomingByte == DWIN\_HEADER1) {

            buffer.data[buffer.index++] = incomingByte;

            printDebugMessage("Header1 alındı");

        } else {

            printDebugMessage("Hata: Geçersiz Header1!");

            recoverFromError();

        }

        return;

    }

    if (buffer.index == 1) {

        if (incomingByte == DWIN\_HEADER2) {

            buffer.data[buffer.index++] = incomingByte;

            printDebugMessage("Header2 alındı");

        } else {

            printDebugMessage("Hata: Geçersiz Header2!");

            recoverFromError();

        }

        return;

    }

    if (buffer.index == 2) {

        buffer.expectedLength = incomingByte + 3;

        if (buffer.expectedLength > buffer.size ||

            buffer.expectedLength < MIN\_BUFFER\_SIZE) {

            printDebugMessage("Hata: Geçersiz paket uzunluğu!");

            recoverFromError();

            return;

        }

        buffer.data[buffer.index++] = incomingByte;

        return;

    }

    buffer.data[buffer.index++] = incomingByte;

    if (buffer.index == buffer.expectedLength) {

        printDebugMessage("Veri paketi tamamlandı");

        processReceivedData();

        clearBuffer();

    }

}

bool setRTC(const RTCData& rtc) {

  if (!Serial2) {

      printDebugMessage("Hata: Serial2 bağlantısı yok!");

      return false;

  }

  // Değerleri kontrol et

  if (rtc.year > 99 || rtc.month < 1 || rtc.month > 12 ||

      rtc.day < 1 || rtc.day > 31 || rtc.hour > 23 ||

      rtc.minute > 59 || rtc.second > 59 ||

      rtc.week > 6) {  // Haftanın günü kontrolü eklendi

      printDebugMessage("Hata: Geçersiz RTC değerleri!");

      return false;

  }

  byte sendBuffer[14] = {

      DWIN\_HEADER1,    // 0x5A

      DWIN\_HEADER2,    // 0xA5

      0x0B,            // 11 byte veri

      DWIN\_WRITE\_CMD,  // 0x82

      0x00,            // VP 0x0010 (RTC)

      0x10,

      rtc.year,        // YY - Yıl (0-99)

      rtc.month,       // MM - Ay (1-12)

      rtc.day,         // DD - Gün (1-31)

      rtc.week,        // WW - Haftanın günü (0-6)

      rtc.hour,        // HH - Saat (0-23)

      rtc.minute,      // MM - Dakika (0-59)

      rtc.second,      // SS - Saniye (0-59)

      0x00            // Sonlandırıcı byte

  };

  printHexData("RTC verisi gönderiliyor: ", sendBuffer, 14);

  Serial2.write(sendBuffer, 14);

  Serial2.flush();

  delay(100);

  unsigned long startTime = millis();

  while (millis() - startTime < 2000) {

      if (Serial2.available() >= 6) {

          byte response[6];

          Serial2.readBytes(response, 6);

          printHexData("RTC yanıtı: ", response, 6);

          if (response[0] == DWIN\_HEADER1 &&

              response[1] == DWIN\_HEADER2 &&

              response[2] == 0x03 &&

              response[3] == DWIN\_WRITE\_CMD &&

              response[4] == DWIN\_OK\_RESPONSE\_O &&

              response[5] == DWIN\_OK\_RESPONSE\_K) {

              printDebugMessage("RTC ayarlama başarılı");

              return true;

          }

      }

      delay(10);

  }

  printDebugMessage("Hata: RTC ayarlama yanıtı alınamadı!");

  return false;

}

void processReceivedData() {

  if (buffer.index < MIN\_BUFFER\_SIZE) {

      printDebugMessage("Hata: Yetersiz veri!");

      return;

  }

//   uint16\_t vpAddress = (buffer.data[4] << 8) | buffer.data[5];

//   uint16\_t value = (buffer.data[6] << 8) | buffer.data[7];

uint16\_t vpAddress = (buffer.data[4] << 8) | buffer.data[5];

uint16\_t value = buffer.data[8]; // Son byte'ta fan hızı değeri var (00-05)

    // Kullanıcı değişikliği zamanını güncelle

    lastUserChange = millis();

  if (vpAddress == VP\_FAN\_SPEED) {

    // Son byte'taki değeri KNX fan nesnesine yaz (indeks 0)

    byte fanValue = value; // 0-5 arası değer

    // KNX için 0,20,40,60,80,100 değerlerine dönüştür ets parametrelerinden kendiniz ayarlayabilirsiniz

    byte scaledValue = ((fanValue \* 20)\*255)/100; // Doğrudan 0,20,40,60,80,100 değerlerini verir

    Knx.write(0, scaledValue); //indeks 0 write işlemi

    Serial.print("Fan hızı KNX'e gönderildi: ");

    Serial.print(fanValue);

    Serial.print(" -> KNX değeri: %");

    Serial.println(scaledValue);

}

  // else if (vpAddress == VP\_AC\_TEMP) {

  //     // Klima sıcaklığı - KNX indeksi 4

  //     float tempValue = (float)value;

  //     Knx.write(4, tempValue);

  //     Serial.print("Klima sıcaklığı KNX'e gönderildi: ");

  //     Serial.println(tempValue);

  // }

  // else if (vpAddress == VP\_SHUTTER) {

  //     // Perde konumu - KNX indeksi 6

  //     byte scaledValue = (value \* 100) / SHUTTER\_MAX;

  //     Knx.write(6, scaledValue);

  //     Serial.print("Perde konumu KNX'e gönderildi: ");

  //     Serial.println(scaledValue);

  // }

  // else if (vpAddress == VP\_DIMMER) {

  //     // Dimmer değeri - KNX indeksi 8

  //     byte scaledValue = (value \* 100) / DIMMER\_MAX;

  //     Knx.write(8, scaledValue);

  //     Serial.print("Dimmer değeri KNX'e gönderildi: ");

  //     Serial.println(scaledValue);

  // } bunları regülasyon yapmanız gerekli

//   switch(vpAddress) {

//       case VP\_FAN\_SPEED:

//           value = value & 0x000F;  // Sadece ilk 4 biti al (0-5 arası)

//           break;

//       case VP\_RGB\_DIMMER:

//           value = value & 0x001F;  // Sadece ilk 5 biti al (0-20 arası)

//           break;

//       case VP\_AC\_TEMP:

//           value = value & 0x00FF;  // Sadece ilk 8 biti al (16-30 arası)

//           break;

//       case VP\_SHUTTER:

//       case VP\_DIMMER:

//           value = value & 0x000F;  // Sadece ilk 4 biti al (0-10 arası)

//           break;

//   } bu bölüm regülasyona ugratırsanız kullanabilirsiniz.

  VPValues\* vp = findVPValue(vpAddress);

  if (vp != NULL && vp->lastValue != value) {

      vp->lastValue = value;

      vp->isChanged = true;

      String message = "Değer değişti - Adres: 0x" +

                      String(vpAddress, HEX) +

                      " Değer: " + String(value);

      printDebugMessage(message.c\_str());

  }

}

void clicommand ()

{

    cli.RegisterCmd("read",&ReadFanFeedback);

    Serial.begin(9600);

    Serial1.begin(9600); // KNX transceiver

    Serial2.begin(9600); // DWIN ekranı

    if (Knx.begin(Serial1, P\_ADDR(1,1,202)) == KNX\_DEVICE\_ERROR) {

        Serial.println("knx init ERROR, stop here!!");

        while(1);

    }

    Serial.println("knx started...");

    printDebugMessage("\nDWIN Test Programı - Komutlar:");

    printDebugMessage("-----------------------------");

    printDebugMessage("read - Fan seviyesi geri bildirimini oku");

    printDebugMessage("-----------------------------");

}

void setRTCFromSerial()

{

  printDebugMessage("RTC ayarlama modu başladı");

  printDebugMessage("Yıl (00-99):");

  while(!Serial.available()) delay(100);

  currentRTC.year = Serial.parseInt();

  printDebugMessage("Ay (01-12):");

  while(!Serial.available()) delay(100);

  currentRTC.month = Serial.parseInt();

  printDebugMessage("Gün (01-31):");

  while(!Serial.available()) delay(100);

  currentRTC.day = Serial.parseInt();

  printDebugMessage("Haftanın günü (0=Pazar, 1=Pazartesi, ..., 6=Cumartesi):");

  while(!Serial.available()) delay(100);

  currentRTC.week = Serial.parseInt();

  printDebugMessage("Saat (00-23):");

  while(!Serial.available()) delay(100);

  currentRTC.hour = Serial.parseInt();

  printDebugMessage("Dakika (00-59):");

  while(!Serial.available()) delay(100);

  currentRTC.minute = Serial.parseInt();

  printDebugMessage("Saniye (00-59):");

  while(!Serial.available()) delay(100);

  currentRTC.second = Serial.parseInt();

  if (setRTC(currentRTC)) {

      printDebugMessage("RTC başarıyla ayarlandı");

  } else {

      printDebugMessage("RTC ayarlama hatası!");

  }

}

void processSerialInput() {

  if (Serial.available()) {

      String input = Serial.readStringUntil('\n');

      input.trim();

      if (input == "rtc") {

          printDebugMessage("RTC Ayarlama Modu");

          setRTCFromSerial();

          return;

      }

      int spaceIndex = input.indexOf(' ');

      String command = input.substring(0, spaceIndex);

      int value = input.substring(spaceIndex + 1).toInt();

      if (command == "bildirim") {

          if (value == 0 || value == 1) {

              if (writeToVP(VP\_NOTIFICATIONS, value)) {

                  String message = "Bildirim durumu ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Bildirim değeri 0 veya 1 olmalı!");

          }

      }

      else if (command == "hava") {

          if (value >= WEATHER\_MIN && value <= WEATHER\_MAX) {

              if (writeToVP(VP\_WEATHER, value)) {

                  String message = "Hava sıcaklığı ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Hava sıcaklığı -15 ile 55 arasında olmalı!");

          }

      }

      else if (command == "fan") {

          if (value >= FAN\_MIN && value <= FAN\_MAX) {

              if (writeToVP(VP\_FAN\_SPEED, value)) {

                  String message = "Fan hızı ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Fan hızı 0-5 arasında olmalı!");

          }

      }

      else if (command == "rgb") {

          if (value >= RGB\_MIN && value <= RGB\_MAX) {

              if (writeToVP(VP\_RGB\_DIMMER, value)) {

                  String message = "RGB parlaklığı ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: RGB parlaklığı 0-20 arasında olmalı!");

          }

      }

      else if (command == "klima") {

          if (value >= AC\_TEMP\_MIN && value <= AC\_TEMP\_MAX) {

              if (writeToVP(VP\_AC\_TEMP, value)) {

                  String message = "Klima sıcaklığı ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Klima sıcaklığı 16-30 arasında olmalı!");

          }

      }

      else if (command == "perde") {

          if (value >= SHUTTER\_MIN && value <= SHUTTER\_MAX) {

              if (writeToVP(VP\_SHUTTER, value)) {

                  String message = "Perde konumu ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Perde konumu 0-10 arasında olmalı!");

          }

      }

      else if (command == "dimmer") {

          if (value >= DIMMER\_MIN && value <= DIMMER\_MAX) {

              if (writeToVP(VP\_DIMMER, value)) {

                  String message = "Dimmer değeri ayarlandı: " + String(value);

                  printDebugMessage(message.c\_str());

              }

          } else {

              printDebugMessage("Hata: Dimmer değeri 0-10 arasında olmalı!");

          }

      }

      else {

          printDebugMessage("Hata: Geçersiz komut!");

      }

  }

}

void setup()

{

    Serial.begin(9600);   // Debug için

    Serial2.begin(9600);  // DWIN ekranı için (clicommand()fonksiyonu içinde yapılıyor göstermelik ekledim  )

    pinMode(TX\_PIN, OUTPUT);

    pinMode(RX\_PIN, INPUT);

    clicommand();

    // İlk bağlantı sonrası fan geri bildirimini oku

    delay(1000);

    ReadFanFeedback();

    if (buffer.data == NULL)

    {

        printDebugMessage("Hata: Buffer oluşturulamadı!");

        while(1);

    }

    if (!Serial2)

    {

        printDebugMessage("Hata: Serial2 başlatılamadı!");

        while(1);

    }

    printDebugMessage("\nDWIN Test Programı - Komutlar:");

    printDebugMessage("-----------------------------");

    printDebugMessage("bildirim [0-1] - Bildirim durumu");

    printDebugMessage("hava [-15-55] - Hava sıcaklığı");

    printDebugMessage("fan [0-5] - Fan hızı");

    printDebugMessage("rgb [0-20] - RGB parlaklığı");

    printDebugMessage("klima [16-30] - Klima sıcaklığı");

    printDebugMessage("perde [0-10] - Perde konumu");

    printDebugMessage("dimmer [0-10] - Dimmer değeri");

    printDebugMessage("rtc - RTC ayarlama modu");

    printDebugMessage("-----------------------------");

}

void loop()

{

    Knx.task();

    cli.Run();

     // DWIN ekranından gelen verileri işle

      while (Serial2.available()) {

          processIncomingByte(Serial2.read());

      }

  unsigned long currentMillis = millis();

  if (currentMillis - lastRTCUpdate >= RTC\_UPDATE\_INTERVAL)

  {

      lastRTCUpdate = currentMillis;

      updateRTC();

  }

  processSerialInput();

  while (Serial2.available())

  {

      processIncomingByte(Serial2.read());

  }

  for (size\_t i = 0; i < sizeof(vpValues)/sizeof(VPValues); i++)

  {

      if (vpValues[i].isChanged) {

          uint16\_t currentValue;

          if (readFromVP(vpValues[i].address, currentValue)) {

              if (currentValue != vpValues[i].lastValue) {

                  writeToVP(vpValues[i].address, vpValues[i].lastValue);

              }

          }

          vpValues[i].isChanged = false;

      }

  }

}