PROJE RAPORU

TURUNCU PROJE

TAMAMLANDI

Theia

Yazar(lar):

Famura

İçerik

1. Giriş

Theia projesi, Android cihazların Windows bilgisayarlar için **ikinci bir monitör** olarak kullanılmasını sağlamak amacıyla geliştirilmiştir. Proje, hem günlük kullanım hedefiyle hem de ileride Android cihazların **VR headset** olarak kullanılmasını amaçlayan başka bir proje için deneyim kazanma amacıyla tasarlanmıştır.

Bu nedenle proje (ve dolayısıyla bu rapor) iki ana bölüme ayrılmıştır: **Windows** ve **Android**. Her bölüm kendi geliştirme sürecine, analizine ve rapor içeriğine sahiptir.

2. Windows

2.1. Proje Teknolojileri

2.1.1. Modül(ler)

- **Windows (Windows 10):**
  Projenin Windows tarafı Windows sistemler için geliştirilmiştir. Windows 7 gibi daha eski sürümlerde test edilmemiştir. Windows 11 üzerinde test edilmiş ve sorunsuz çalışmaktadır.

2.1.2. Programlama Dilleri

- **C++ (C++14 Standard) – Visual Studio 2022:**
  Windows tarafındaki tüm kodlar C++ ile yazılmıştır ve C++14 standardına göre derlenmektedir. Bu sayede modern dil özellikleri kullanılmakta ve Visual Studio 2022 ile tam uyumluluk sağlanmaktadır.

2.1.3. Libraries & External Additions

- **Winsock2.h:**
  TCP/IP tabanlı ağ iletişimi için Windows Sockets API'sini sağlar.
- **targetver.h & stdafx.h:**
  Socket programlama için gerekli temel yapılandırmaları içerir.
- **thread.h:**
  İletişimin paralel tarafında "POSIX threads" kullanımı için gereklidir.
- **shlobj.h:**
  Windows'ta özel klasörlere erişim sağlar (örneğin SHGetKnownFolderPath ile Downloads klasörü).
- **Usbmmidd_v2:**
  Sanal ekran (virtual screen) oluşturmak için kullanılan harici bir projedir. deviceinstaller64.exe üzerinden kullanılmaktadır. Referanslar mevcuttur ancak proje sahibine ulaşılamamıştır, bu nedenle rapor içerisinde detaylandırılmamıştır.
- **stb_image_write.h:**
  Bellekteki görüntü verisini PNG, JPG, BMP, TGA veya HDR gibi formatlarda dosyaya yazmak için kullanılır.
- **Janus:**
  Bu projede dosya paylaşımı için **Janus** altyapısı kullanılmıştır.

2.2. Project Architecture

Proje **modüler programlama** yaklaşımıyla geliştirilmiştir. Mimari yapı dört ana parçaya ayrılmıştır.

2.2.1. Main

Main sınıfı, uygulamanın yönetici (manager) sınıfıdır. Diğer tüm sınıflar burada başlatılır, çağrılır, analiz edilir ve belirli bir sıraya sokulur. Uygulama şu anda **console** üzerinden çalışmaktadır. Gelecekte bir frontend geliştirilirse, bu frontend de yine Main sınıfı üzerinden yönetilecektir.

2.2.1.1. Başlangıç (Initialize Everything)

```
int main() {
    Creator c;
    Recorder r;
    Wifi w;
    bool x = true;


    c.launcher();
    SOCKET send_socket = w.send_socket_create();
    SOCKET* recieve_socket = w.recieve_socket_create();
```

Öncelikle gerekli değişkenler oluşturulur. Ardından c.launcher() kullanılarak yeni bir sanal ekran oluşturulur. Sonrasında Wi-Fi modülü üzerinden socket'ler oluşturularak Janus altyapısı kullanılmaya başlanır.

2.2.1.2. Main Loop

```
while (x) {
    r.recorder();
    w.send(send_socket);
    x = w.recive(recieve_socket[0]);
}
```

Ana döngü yalnızca **3 satır koddan** oluşur: Recorder, gerekli ekran görüntüsünü alır; Wi-Fi modülü bu görüntüyü gönderir, Android cihazdan yanıt beklenir. Eğer yanıt doğru şekilde alınırsa x değişkeni true kalır ve döngü devam eder. Yanıt alınamazsa x false olur ve döngü sonlanır. İlk bakışta maliyetli bir süreç gibi görünse de yapılan testlerde günlük kullanım için tamamen yeterli olduğu görülmüştür.

2.2.1.3. Bitiş (Close Everything)

Program sonlandığında sanal ekran ve socket'ler kapatılır. Böylece sistem kaynaklarının gereksiz yere kullanılmasının önüne geçilir.

```
w.socket_close(send_socket);
w.socket_close(recieve_socket[1]);
c.closer();

return 0;
```

2.2.2. Screen Creator

Adı yalnızca "creator" olsa da bu sınıf hem **sanal ekranı oluşturur** hem de **kapatır**. Bu işlemler .bat dosyaları üzerinden gerçekleştirilir.

2.2.2.1. Reach Opener & Opener

Bu bölümde opener.bat dosyası kullanılır.

```
deviceinstaller64 install usbmmidd.inf usbmmidd
deviceinstaller64 enableidd 1
```

Kod, bu .bat dosyasına erişir:

```
void launchHiddenCMDInSubfolder() {
    std::string basePath = getExecutablePath();
    std::string targetPath = basePath + "\\usbmmidd_v2";
    std::cout << "Target Path: " << targetPath << std::endl;

    // CMD'yi gizli aç ve orada bir komut çalıştır (örneğin "dir")
    std::string command = "cmd /K \"cd /d " + targetPath + " && opener.bat\"";
```

Ve arka planda gizli bir **command prompt** oluşturarak dosyayı çalıştırır. Böylece kullanıcı deneyimi daha temiz hale getirilir.

```
STARTUPINFO si;
PROCESS_INFORMATION pi;

ZeroMemory(&si, sizeof(si));
si.cb = sizeof(si);
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = SW_HIDE; // CMD'yi gizle

ZeroMemory(&pi, sizeof(pi));

if (CreateProcessA(nullptr, (LPSTR)command.c_str(), nullptr,
        nullptr, FALSE, CREATE_NO_WINDOW, nullptr, nullptr, &si, &pi)) {
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
else {
    std::cerr << "CMD başlatılamadı!" << std::endl;
}
```

2.2.2.2. Reach Closer & Closer

Benzer şekilde, sanal ekranı kapatmak için closer.bat dosyası kullanılır:

```
deviceinstaller64 stop usbmmidd
deviceinstaller64 remove usbmmidd
```

Kod önce .bat dosyasına ulaşır, ardından gizli bir cmd üzerinden çalıştırır:

```
void closeHiddenCMDInSubfolder() {
    std::string basePath = getExecutablePath();
    std::string targetPath = basePath + "\\usbmmidd_v2";
    std::cout << "Target Path: " << targetPath << std::endl;

    // CMD'yi gizli aç ve orada bir komut çalıştır (örneğin "dir")
    std::string command = "cmd /K \"cd /d " + targetPath + " && closer.bat\"";
```

Bu işlem sonucunda sanal ekran kaldırılır:

```
STARTUPINFO si;
PROCESS_INFORMATION pi;

ZeroMemory(&si, sizeof(si));
si.cb = sizeof(si);
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = SW_HIDE; // CMD'yi gizle

ZeroMemory(&pi, sizeof(pi));

if (CreateProcessA(nullptr, (LPSTR)command.c_str(), nullptr,
        nullptr, FALSE, CREATE_NO_WINDOW, nullptr, nullptr, &si, &pi)) {
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
}
else {
    std::cerr << "CMD başlatılamadı!" << std::endl;
}
```

2.2.2.3. Path Helper

Her iki işlemde de Janus projesinde kullanılan getExecutablePath() metodundan faydalanılır. Bu metod, C++'ın verilen dosya yolunu string fonksiyonu olarak değil gerçek bir **filepath** olarak algılamasını sağlar. Bu metodun detaylı açıklaması **Janus raporunun 4. sayfasında** bulunabilir.

```
std::string getExecutablePath() {
    char path[MAX_PATH];
    GetModuleFileNameA(nullptr, path, MAX_PATH);
    std::string fullPath(path);

    size_t pos = fullPath.find_last_of("\\/");
    return (pos != std::string::npos) ? fullPath.substr(0, pos) : fullPath;
}
```

2.2.3. Screen Recorder

Bu sınıfın temel amacı, Windows kütüphanelerini kullanarak ekran verisini **Bitmap** olarak almak ve bunu bir görüntü dosyasına dönüştürmektir. Bu görüntü daha sonra Android cihazda gösterilmek üzere gönderilir. Bu süreç beş ana aşamaya ayrılmıştır:

2.2.3.1. Initialize & Monitor Bulma

Kod, Windows'un kendi kütüphanelerini kullanarak hedef monitörü tespit eder. Ardından ekranın genişlik ve yükseklik bilgileri alınır.

```
void captureScreenAsImage(const char* outputFileName, const char* format) {
    RECT targetMonitorRect = { 0 };
    EnumDisplayMonitors(nullptr, nullptr, MonitorEnumProc, (LPARAM)&targetMonitorRect);

    int screenWidth = targetMonitorRect.right - targetMonitorRect.left;
    int screenHeight = targetMonitorRect.bottom - targetMonitorRect.top;

    if (screenWidth <= 0 || screenHeight <= 0) {
        std::cerr << "Failed to find the specified monitor." << std::endl;
        return;
    }
```

2.2.3.2. Device Contexti Oluşturma

Monitör bilgileri alındıktan sonra, ekran içeriği BitMap'e dönüştürülebilecek şekilde hazırlanır.

```
// Create a device context for the monitor
HDC hScreenDC = GetDC(nullptr);
HDC hMemoryDC = CreateCompatibleDC(hScreenDC);
HBITMAP hBitmap = CreateCompatibleBitmap(hScreenDC, screenWidth, screenHeight);
SelectObject(hMemoryDC, hBitmap);

// Copy the monitor's screen content
BitBlt(hMemoryDC, 0, 0, screenWidth, screenHeight,
       hScreenDC, targetMonitorRect.left, targetMonitorRect.top, SRCCOPY);
```

2.2.3.3. Bitmap Olarak Veriyi Alma

Bu aşamada ekran görüntüsü fotoğraf olarak kaydedilir.

```
BITMAP bmp;
GetObject(hBitmap, sizeof(BITMAP), &bmp);
int dataSize = bmp.bmWidth * bmp.bmHeight * 4;
unsigned char* pixelData = new unsigned char[dataSize];
GetBitmapBits(hBitmap, dataSize, pixelData);

// Save the image with the desired extension
if (strcmp(format, "png") == 0) {
    stbi_write_png(outputFileName, bmp.bmWidth,
        bmp.bmHeight, 4, pixelData, bmp.bmWidth * 4);
}
else {
    std::cerr << "Unsupported format: " << format << std::endl;
}
```

2.2.3.4. Cleanup

İşlem tamamlandıktan sonra tüm kaynaklar temizlenir ve sistem kaynaklarının gereksiz kullanımı engellenir..

```
delete[] pixelData;
DeleteObject(hBitmap);
DeleteDC(hMemoryDC);
ReleaseDC(nullptr, hScreenDC);
```

2.2.4. Wi-Fi Handler

Bu bölüm tamamen **Janus** altyapısını kullanır. Detaylar için Janus raporuna bakılabilir:

Janus.docx

2.3. Proje Analizi

2.3.1. Test

Tüm testler Windows bilgisayarlar üzerinde yapılmıştır ve başarılı sonuçlar elde edilmiştir. Ölçeklenebilirlik açısından kod umut vericidir. Testlerde saniyede yaklaşık **80 görüntü** gönderilebildiği gözlemlenmiştir..

2.3.2. Upgradable Things

Tüm testler Windows bilgisayarlar üzerinde yapılmıştır ve başarılı sonuçlar elde edilmiştir. Ölçeklenebilirlik açısından kod umut vericidir. Testlerde saniyede yaklaşık **80 görüntü** gönderilebildiği gözlemlenmiştir.

3. Android

3.1. Proje Teknolojileri

3.1.1. Modül(ler)

- **Android (Android 13+):**
  Android tarafı Android 13 ve üzeri sürümler için tasarlanmıştır. Android 7+ sürümlerde çalışması muhtemeldir ancak test edilemediği için kesinlik yoktur.

3.1.2. Programlama Dilleri

- **Java (Java 15) – Android Studio:**
  Tüm Android kodları Java ile yazılmıştır ve Java 15 hedeflenmiştir. Modern dil özellikleri kullanılarak Android Studio ile uyumlu ve performanslı bir yapı sağlanmıştır.

3.1.3. Kütüphaneler & Harici Eklentiler

- **java.net.ServerSocket & java.net.Socket:**
  TCP/IP tabanlı iletişim sağlar. ServerSocket bağlantıları dinler, Socket veri alışverişini yönetir.

- **java.io.DataInputStream & java.io.DataOutputStream:**
  Binary veri aktarımı için kullanılır. Görüntü boyutu + byte dizisi gibi özel bir protokol oluşturulmasını sağlar.

- **android.graphics.BitmapFactory:**
  Byte array'den Bitmap oluşturur. PNG ve JPEG gibi formatları destekler.

- **android.widget.ImageView & android.widget.TextView:**
  Görüntü ve durum mesajlarını UI üzerinde göstermek için kullanılır.

- **androidx.appcompat.app.AppCompatActivity:**
  Activity yaşam döngüsünü yönetir (onCreate, onDestroy).

- **android.os.Bundle:**
  Oluşum parametrelerini göndermek ve oluşturma sırasında tekrar elde etmek için kullanılır.

- **Thread (Java/Kotlin standard):**
  Server işlemlerinin UI thread'i bloklamaması için arka planda çalışmasını sağlar.

- **@Volatile annotation:**
  Thread'ler arasında running değişkeninin görünürlüğünü garanti eder.

- **R.layout.activity_main, R.id.imageView, R.id.statusView:**
  XML tanımlı kullanıcı arayüzü düzenine ve bileşenlerine referanslar. Java kodunu görsel öğelere bağlamak için kullanılır.

3.2. Proje Mimarisi

Android tarafındaki mimari oldukça basittir. Asıl iş yükü Windows tarafında olduğu için Android tarafı daha hafif bir yapıya sahiptir.

3.2.1. Activity Lifecycle Manager

Uygulamanın giriş noktasıdır. UI bileşenlerini başlatır ve arka planda server thread'ini çalıştırır. Activity yaşam döngüsü boyunca server'ın temiz şekilde açılıp kapanmasını sağlar.

```
class MainActivity : AppCompatActivity() {
    private lateinit var imageView : ImageView
        private lateinit var statusView : TextView

        private val port = 5002
        @Volatile private var running = true
        private var serverSocket : ServerSocket ? = null

        override fun onCreate(savedInstanceState : Bundle ? ) {
        super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)
            imageView = findViewById(R.id.imageView)
            statusView = findViewById(R.id.statusView)
            Thread{ runServer() }.start()
    }
```

3.2.2. Server Handler

Port **5002** üzerinden ServerSocket oluşturur ve Windows uygulamasından gelen bağlantıları dinler.
Her bağlantı için: UI üzerinde bağlantı durumu gösterilir / Görüntü işleme handleClient() fonksiyonuna devredilir / İletişim tamamlandıktan sonra socket kapatılır

```
private fun runServer() {
    try {
        serverSocket = ServerSocket(port)
            runOnUiThread{ statusView.text = "Listening on port $port" }

            while (running) {
                val client = serverSocket!!.accept()
                    runOnUiThread{ statusView.text = "Client connected: ${client.inetAddress.hostAddress}" }
                    handleClient(client)
                    try { client.close() }
                catch (_: Exception) {}
                runOnUiThread{ statusView.text = "Client disconnected" }
            }
    }
    catch (e: Exception) {
        e.printStackTrace()
            runOnUiThread{ statusView.text = "Server error: ${e.message}" }
    }
    finally {
        try { serverSocket ? .close() }
        catch (_: Exception) {}
    }
}
```

3.2.3. Client Communication & Image Decoder

Socket üzerinden özel bir protokol ile veri okunur:

1. Önce 4 byte'lık görüntü boyutu okunur
2. Ardından görüntü byte'ları alınır
3. BitmapFactory ile decode edilir
4. ImageView üzerinde gösterilir
5. Göndericiye 1 byte'lık onay sinyali (0x01) gönderilir

Hata durumları yönetilir ve socket'lerin düzgün şekilde kapanması sağlanır.

```
private fun handleClient(socket: Socket) {
    try {
        val input = DataInputStream(socket.getInputStream())
            val output = DataOutputStream(socket.getOutputStream())

            while (running) {
                // Protocol: 4-byte big-endian length, then that many image bytes
                val len = try { input.readInt() }
                catch (e: Exception) { break }
                if (len <= 0) break

                    val buf = ByteArray(len)
                    var read = 0
                    while (read < len) {
                        val n = input.read(buf, read, len - read)
                            if (n < 0) throw Exception("Stream closed")
                                read += n
                    }

                val bmp = BitmapFactory.decodeByteArray(buf, 0, len)
                    if (bmp != null) {
                        runOnUiThread{ imageView.setImageBitmap(bmp) }
                    }
                    else {
                        runOnUiThread{ statusView.text = "Decode failed (len=$len)" }
                    }

                // Send single-byte ack (0x01)
                output.writeByte(1)
                    output.flush()
            }
```

3.3. Proje Analizi

3.3.1. Test

Android tarafı yalnızca Android 13+ cihazlarda test edilmiştir ve sorunsuz çalışmıştır. Testlerde dakikada yaklaşık **400 görüntü** gösterilebildiği gözlemlenmiştir. Ancak Windows ve Wi-Fi performansı hesaba katıldığında pratikte **dakikada ~60 görüntü** elde edilmiştir.

3.3.2. Geliştirilebilir Alanlar

Bu taraf da temelde ekran görüntüsü alıp göndermeye dayanır. Daha gelişmiş ve optimize bir streaming yöntemi ile ciddi performans kazanımları sağlanabilir.

4. Proje Yorumları

4.1. Possible Usages

Proje esas olarak **günlük ofis kullanımı** için tasarlanmıştır. Oyun için uygun değildir.

4.1.1. Kullanım Sırasında Yapılması Gerekenler

Projeyi geliştirmek isteyenlerin deviceinstaller64.exe konusunda **son derece dikkatli** olması gerekir. Yanlış kullanımda tüm sürücülerin devre dışı kalmasına yol açabilir. Ne yazık ki bu bizzat tecrübe edilmiştir. (Bilgisayarım Emily, bu tecrübe nedeniyle SSD değişimine kadar asla çözülmeyecek sorunlarla karşılaştı.)

4.2. Geliştirici Yorumları

Bu proje, Android cihazların VR headset olarak kullanılmasına giden yolda yalnızca bir adım olarak değerlendirilirse çok küçük bir adım sayılır. Daha Streaming konusunda öğrenilecek ve geliştirilecek çok şey var malesef. Proje başlangıçta iki kişi (Famura ve Aybüke) için planlanmıştır. Ancak geliştirme sürecinin ortasında (beklendik şekilde) Aybüke projeden ayrıldı ve projeyi tek başıma tamamladım.

Hooah

-Famura

5. Credits

- Sadece Famura Projesi

6. Referanslar

- Microsoft. (n.d.). Winsock2 Reference. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/winsock/windows-sockets-start-page-2
- ISO/IEC. (2014). ISO/IEC 14882:2014 – Programming Languages – C++ (C++14 Standard). International Organization for Standardization. https://www.iso.org/standard/64029.html
- Microsoft. (n.d.). SHGetKnownFolderPath function (shlobj_core.h). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/shlobj_core/nf-shlobj_core-shgetknownfolderpath
- Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson.
- Stevens, W. R., Fenner, B., & Rudoff, A. M. (2004). Unix Network Programming, Volume 1: The Sockets Networking API (3rd ed.). Addison-Wesley.
- Microsoft. (n.d.). InetPton function (ws2tcpip.h). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/ws2tcpip/nf-ws2tcpip-inetpton
- How To Create a Virtual Monitor - Windows [Video]. (2023, Mart ?). YouTube. https://www.youtube.com/watch?v=ybHKFZjSkVY
- Oracle. (n.d.). ServerSocket (Java Platform SE 15). Oracle Documentation. https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/net/ServerSocket.html
- Oracle. (n.d.). DataInputStream (Java Platform SE 15). Oracle Documentation. https://docs.oracle.com/en/java/javase/15/docs/api/java.base/java/io/DataInputStream.html
- Android Developers. (n.d.). BitmapFactory. Android Developer Guide. https://developer.android.com/reference/android/graphics/BitmapFactory
- Android Developers. (n.d.). AppCompatActivity. Android Developer Guide. https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity
- Microsoft. (n.d.). GetDC function (wingdi.h). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdc
- Microsoft. (n.d.). BitBlt function (wingdi.h). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-bitblt
- Microsoft. (n.d.). SHGetKnownFolderPath function (shlobj_core.h). Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/api/shlobj_core/nf-shlobj_core-shgetknownfolderpath
- GitHub. (n.d.). stb_image_write.h by nothings. GitHub Repository. https://github.com/nothings/stb/blob/master/stb_image_write.h

7. Disk İçerikleri

- Windows ve Android kaynak kodları (.rar)
- Bu raporun PDF versiyonu
- Igra Rock’en’Roll Cela Jugoslavia by Elektricini Orgazam