PORTFOLIO

서버프로그래머_권구환_포트폴리오

이름	권 구 환
생년월일	1996. 06. 05
이메일	rnjs3486@gmail.com
전화번호	010-3579-8750
주소	경기도 하남시 학암동 위례순환로270 6512-1303

PORTFOLIO
CONTENTS

01.	시스템 모니터링 프로그램	현장 특성상 시스템이 종료되면 안되기때문에 모니터링 시스템 개발
02.	AVC 시스템(차량 통행량 관리 서버 및 운영 소프트웨어)	차량 통행량 수집용 서버 프로그램 개발
03.	스마트 구간 단속 서버 및 운영 소프트웨어	구간 단속 서버 운영프로그램 개발
04.		
05.		
06.		

Project

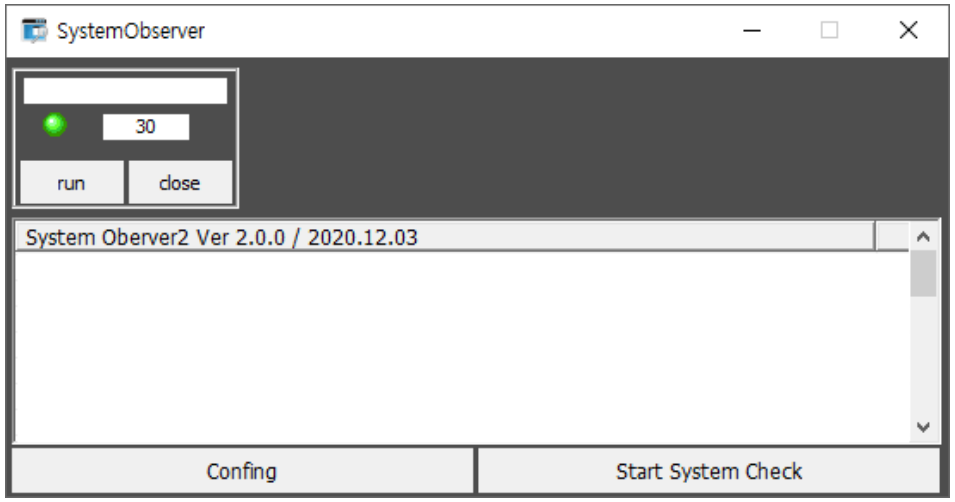
시스템 모니터링 프로그램

About project

현장 PC의 대한 시스템들이 다운이 되면 단속 데이터와 수집용 데이터가 손실이 되어 등록된 시스템들이 실행되지 않고 있으면 재실행 시켜주는 프로그램.

또한, 운영 시스템 오류가 3번 이상 일어날시 윈도우를 재실행 시키는 방법을 사용.

시스템 모니터링 프로그램



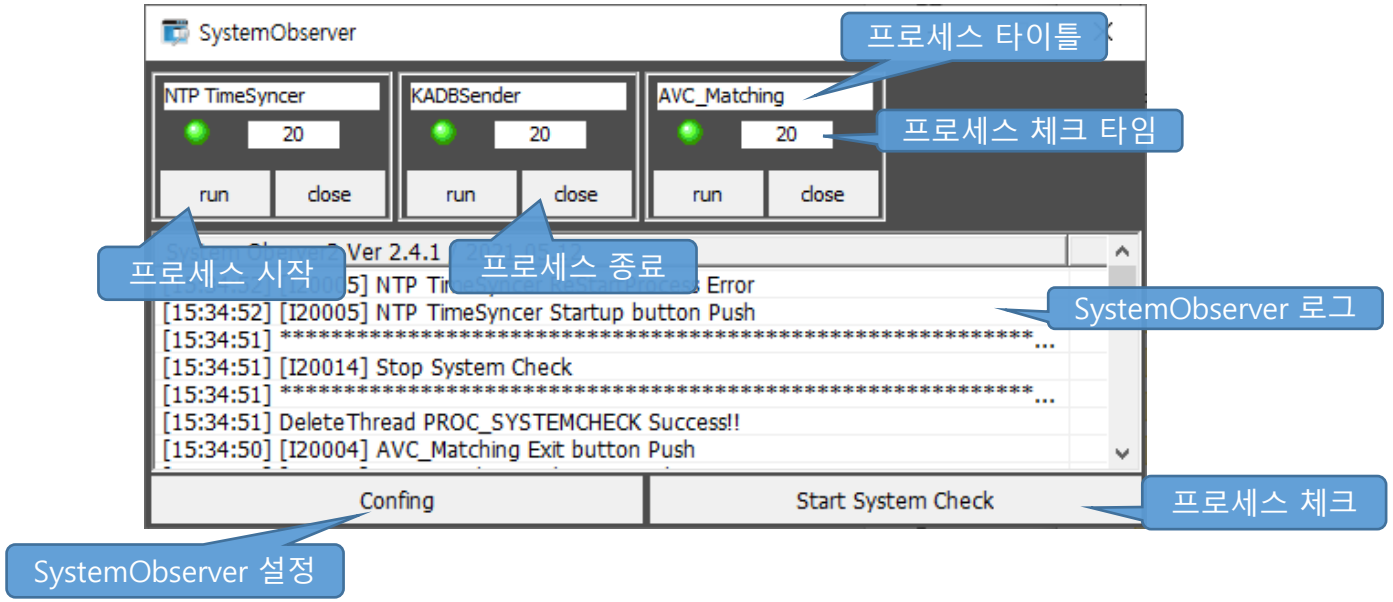
Introduce project

작업 기간	2021. 01 ~ 2021. 04 (4개월)
인력 구성(기여도)	1명
프로젝트 목적	현장 시스템 다운현상을 막기위해 개발
프로젝트 내용	현장 PC의 프로그램들을 재실행 시키는 프로그램 시스템이 강제 종료 및 무한 로딩 상대가 걸렸을때 시스템을 체크하여 재실행 시키거나 연속적인 오류 발생시 와치독을 걸어 강제 재부팅을 실행한다.
주요 업무 및 상세 역할	1) 현장 PC의 프로그램을 재실행 2) 프로그램의 종료 및 무한로딩 확인 3) 시스템 재부팅
사용언어 및 개발 환경	C / C++, STL
참고 자료	

Main work 메인 프로그램 소개

- **기능 소개**

- Run버튼을 통하여 프로세스 시작.
- Close버튼을 통하여 프로세스 종료.
- Config버튼을 통해 설정 창 띄우기.
- 프로세스 체크 버튼으로 실행

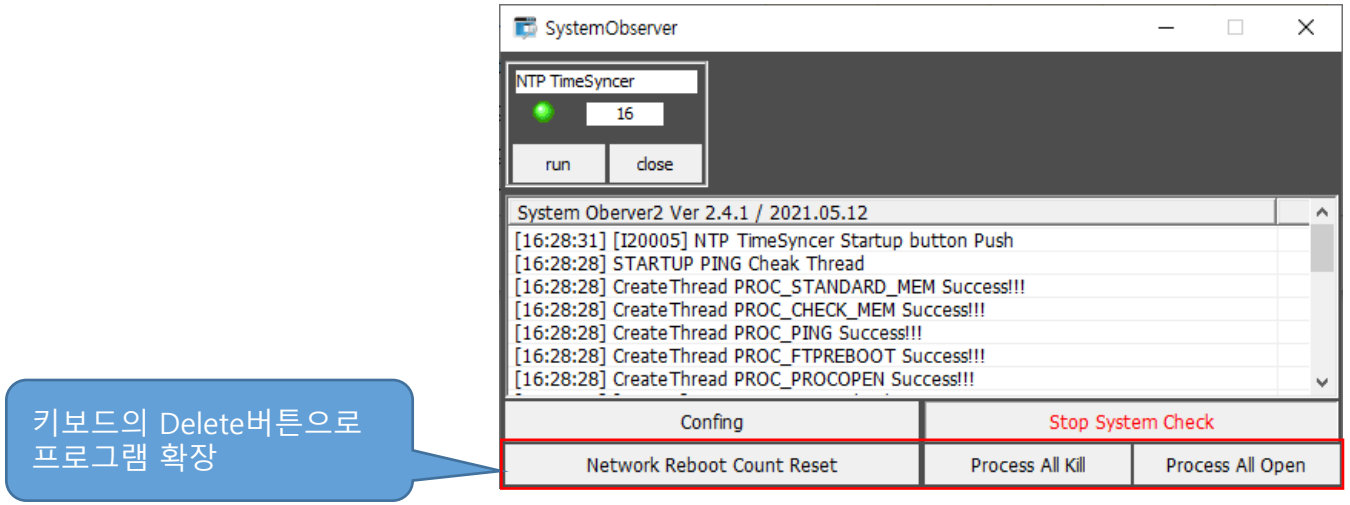


Main work 메인 프로그램 소개

- **기능 소개**

- 프로세스 체크 버튼으로 실행시 문구 변경
- 키보드의 Delete버튼 클릭으로 프로그램 확장
- Network 확인 후 연결 해제시 PC 재부팅
- 프로세스 All Kill, Open 버튼 구현

시스템 체크 동작 확인



Main work 프로세스 실행 소스

```
BOOL CCheckModuleDlg::ReStartProcess(CString _csName)
{
    CString csTmp;
    csTmp.Format(_T("%s"), (LPCSTR)this->m_csExecutePath);
    int nPos = csTmp.ReverseFind('\\') + 1;

    TCHAR strDefaultDir[MAX_PATH]; memset(strDefaultDir, 0x00, _countof(strDefaultDir));
    _tcsncpy_s(strDefaultDir, _countof(strDefaultDir), this->m_csExecutePath, nPos);
    strDefaultDir[nPos] = '\0';

    DWORD dwPriority = NORMAL_PRIORITY_CLASS;
    if (this->m_nPriority == 0)
        dwPriority = BELOW_NORMAL_PRIORITY_CLASS;
    else if (this->m_nPriority == 1)
        dwPriority = NORMAL_PRIORITY_CLASS;
    else
        dwPriority = HIGH_PRIORITY_CLASS;

    if (!FindProcess(_csName))
    {
        TCHAR strTmp[MAX_PATH];
        sprintf_s(strTmp, _countof(strTmp), _T("%s"), (LPCSTR)this->m_csExecutePath);
        STARTUPINFO StartupInfo = { 0 };
        PROCESS_INFORMATION ProcessInfo;
        StartupInfo.cb = sizeof(STARTUPINFO);
        StartupInfo.dwFlags = STARTF_USESTDHANDLES;
        StartupInfo.hStdInput = GetStdHandle(STD_INPUT_HANDLE);
        StartupInfo.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
        StartupInfo.hStdError = GetStdHandle(STD_ERROR_HANDLE);

        if (!CreateProcess(NULL,
            strTmp,
            NULL,
            NULL,
            TRUE,
            dwPriority,
            NULL,
            (LPCSTR)(LPSTR)strDefaultDir,
            &StartupInfo,
            &ProcessInfo))
        {
            return FALSE;
        }
        ::CloseHandle(ProcessInfo.hThread);
        ::CloseHandle(ProcessInfo.hProcess);
    }
    return TRUE;
}
```

Main work 프로세스 종료 소스

```
void CCheckModuleDlg::OnBnClickedCloseMfcbutton()
{
    // TODO: 여기에 컨트롤 알림 처리기 코드를 추가합니다.
    //프로그램 종료
    CSystemObserver2App* pApp = (CSystemObserver2App*)AfxGetApp();
    TCHAR strTmpLog[MAX_PATH]; memset(strTmpLog, 0x00, _countof(strTmpLog));

    if (pApp->m_appConfig.m_nCamWDTUse && this->m_csTitle == pApp->m_appConfig.m_strCamWDTProcessName)
    {
        // LiveMDS에서 강제종료시켜버리면 카메라에 와치독이 걸려버린다고한다.
        // 그래서 103코드로 전송해 카메라 와치독 해제을 걸도록 전해준다.
        SendCopyData(pApp->m_appConfig.m_strCamWDTProcessName, CMD_TYPE_CAM_WDT_STOP);
        sprintf_s(strTmpLog, _countof(strTmpLog), _T("%s CamWDT Stop OpCode Send"), (LPCSTR)this->m_csTitle);
        pApp->m_eventLog.LOGPRINT(LOG_ERR, strTmpLog);
        Sleep(50);
    }
    sprintf_s(strTmpLog, _countof(strTmpLog), _T("[I20004] %s Exit button Push"), (LPCSTR)this->m_csTitle);
    pApp->m_eventLog.LOGPRINT(LOG_ERR, strTmpLog);

    CString strTmp;
    strTmp.Format("%s", (LPCSTR)this->m_csExecutePath);
    int nPos = strTmp.ReverseFind('\\') + 1;
    strTmp = strTmp.Mid(nPos);

    // close메시지를 날린다 동작되는건 확인했고 실사용은 모르겠다.
    //SendCloseData(m_csTitle);

    int pi = FindProcessID(strTmp);

    if (pi)
    {
        TerminateProcess(pi);
    }
}
```

Project 02.

AVC (차량 교통량 조사 장비)

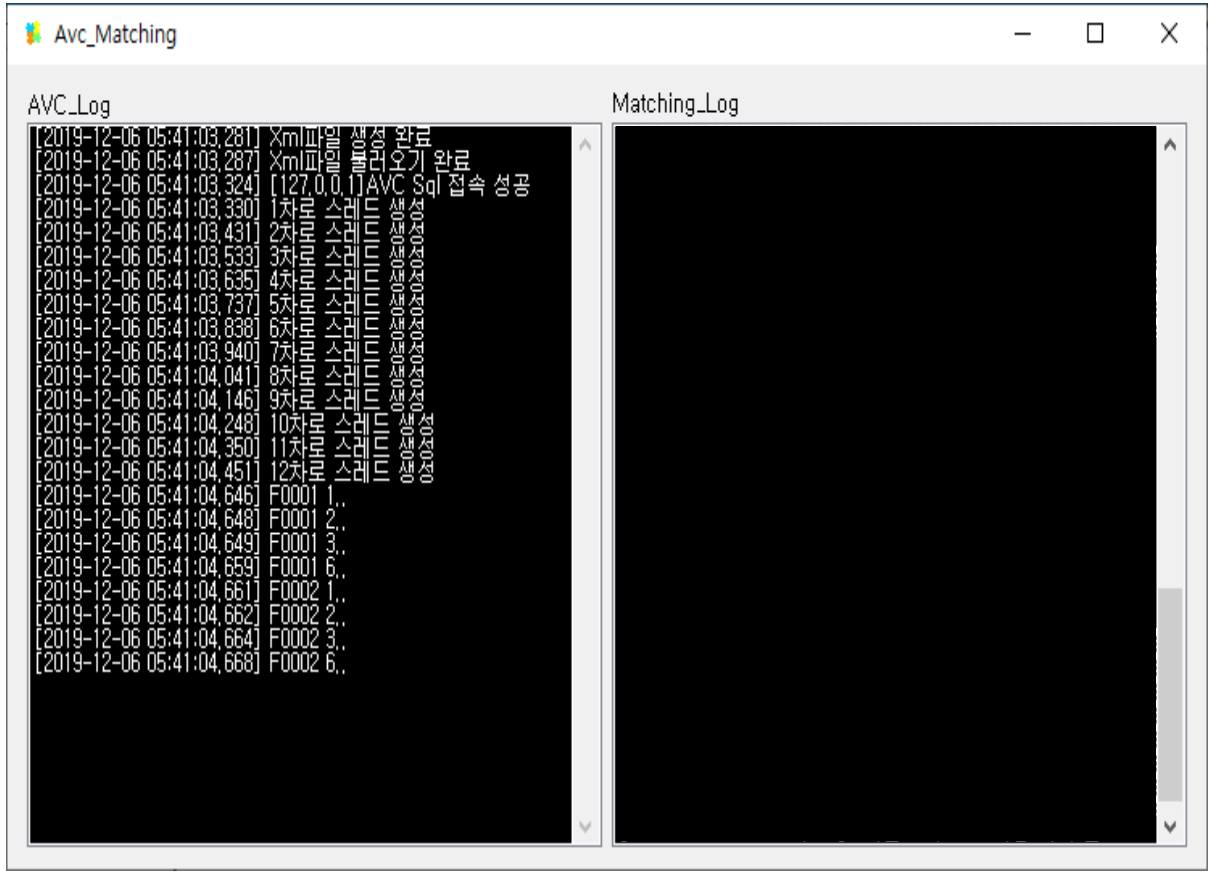
About project

특정 구간의 차량 교통량 조사를 위한 서버 개발
데이터 수집용 목적으로 개발되었습니다.

Project 02. AVC (차량 교통량 조사장비)

Introduce project

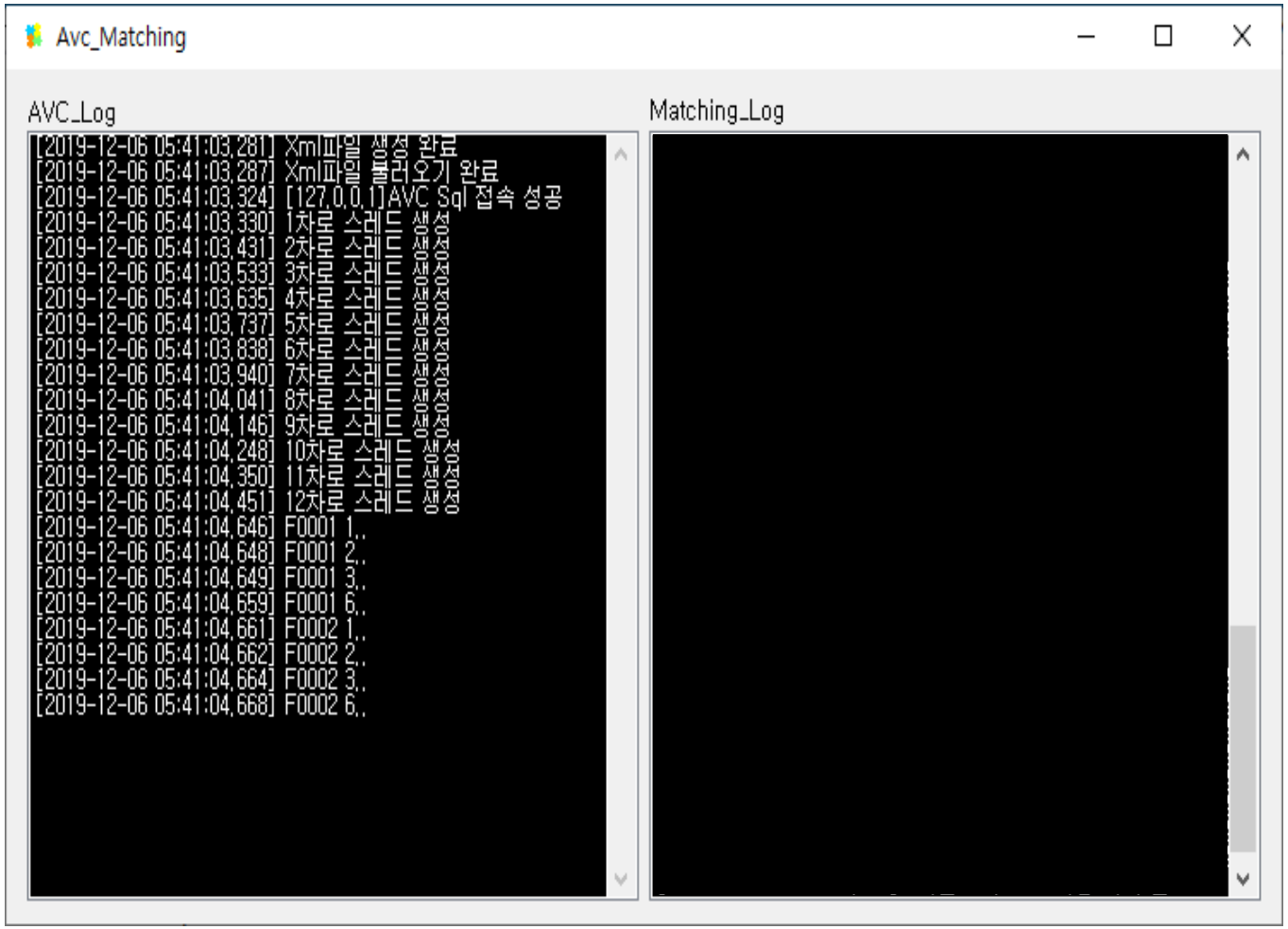
작업 기간	2019. 11 ~ 2020. 10 (12개월)
인력 구성(기여도)	1명
프로젝트 목적	교통량 조사를 위한 데이터 수집용 서버 개발
프로젝트 내용	특정 구간의 차량의 통행량을 조사하기위한 수집 목적으로 개발된 서버입니다. 차량의 데이터를 현장에서 넘겨받아 MS-SQL의 서버에 데이터를 전송하는 프로그램입니다.
주요 업무 및 상세 역할	1) 현장 데이터를 넘겨받음 2) 넘겨 받은 전방 데이터와 후방 데이터를 매칭 3) 서버 PC에 데이터를 전송 4) 저장된 데이터를 기간별로 조회
사용언어 및 개발 환경	C#, MS-SQL
참고 자료	



Main work 매칭 프로그램

- **기능 소개**

- 현장에서 전달 받은 영상데이터와 차량 정보를 전방 데이터와 후방 데이터를 매칭하는 프로그램



Main work 차선 스레드 생성

```
private bool GetLaneListThread()
{
    DataTable DTSectionInfo = null;
    try
    {
        // 데이터베이스 SectionInfo 테이블을 읽어오는 쿼리문
        string Select_qry = "Select * From SectionInfo";

        // 작성한 쿼리문을 이용하여 데이터베이스에서 DataTable를 얻는다.
        DTSectionInfo = mssql.GetDataTable(Select_qry, ConnectionString);
    }
    catch (Exception e)
    {
        Avc_LogC.Log_Save(5, "AVC_FrmMain", "GetLaneListThread : " + e.ToString());
        return false;
    }

    LIDataList = new List<ClsLaneInfo>();
    CLList = new List<ClsVehicleMatching>();
    // SectionInfo 정보를 한개씩 불러 매칭 클래스를 생성한다.
    foreach (DataRow DRLaneInfo in DTSectionInfo.Rows)
    {
        try
        {
            ClsLaneInfo LIData = new ClsLaneInfo();
            // LIData에 차선 정보를 저장한다.
            if (!LIData.SetData(DRLaneInfo)) { continue; }
            // LIData에 NULL값이 있는지 검사한다.
            if (!LIData.DataNullCheck()) { LIData = null; continue; }

            // 차선 정보를 리스트를 관리하기 위하여 리스트에 저장한다.
            LIDataList.Add(LIData);

            // 혹시 모르니 스레드를 시작하기전에 차선 폴더를 생성해주자...
            if (Avc_LogC.LaneFolderCreate(LIData.ReceivePath, LIData.SavePath))
            {
                // 매칭 클래스 생성
                ClsVehicleMatching vehiclematching = new ClsVehicleMatching(ConnectionString, xml, LIData.ReceivePath, LIData.SavePath,
                    LIData.SectionCode, LIData.SectionNumber, LIData.SectionVector, LIData.Lane, LIData.TunnelCode, LIData.MatchFlag);

                // 클래스를 생성할때 스레드를 시작하니깐 클래스를 관리하면 될거야...
                CLList.Add(vehiclematching);

                AVC_Msg("[" + LIData.SectionCode + "] " + LIData.Lane + "차로 생성");
                Avc_LogC.Log_Save(5, LogName, "[" + LIData.SectionCode + "] " + LIData.ReceivePath + ", " + LIData.Lane + "차로 스레드 생성");
                Thread.Sleep(1);
            }
        }
        catch
        {
            continue;
        }
    }
    return true;
}
```

Main work 차량 매칭 (1)

```
public void MatchingStart()
{
    string funcname = MethodBase.GetCurrentMethod().Name;
    SetLaneMsg("[" + SectionCode + "]" + Lane + "차선 Start");

    log.Program_Log(5, SectionCode + "_" + Lane, LogName,Lane + "차선 매칭 시작");

    int failCount = 0;

    while (m_bMatching)
    {
        //IEnumerable<string> Ffile = FileRead(ReceivePath, SectionCode, Lane);
        log.Program_Log(5, SectionCode + "_" + Lane, LogName, "파일 검색 시작");

        // 폴더의 파일들의 이름을 전부 가져와 FileInfo배열에 저장
        FileInfo[] Ffile = FileRead2(ReceivePath, SectionCode, Lane);
        if (Ffile == null || Ffile.Count() < 1)
        {
            continue;
        }
        log.Program_Log(5, SectionCode + "_" + Lane, LogName, "검색 전체 파일 개수 - " + Ffile.Count().ToString());
        log.Program_Log(5, SectionCode + "_" + Lane, LogName, "파일 검색 종료");
```

```
        //FileInfo pFile = new FileInfo(Ffile.First().ToString());
        foreach (FileInfo pFile in Ffile)
        {
            SetLaneMsg(pFile.FullName.ToString());
            log.Program_Log(5, SectionCode + "_" + Lane, LogName, pFile.Name.ToString());

            if (!m_bMatching) break;

            if (pFile == null)
            {
                Thread.Sleep(50);
                continue;
            }

            // 실패 카운트가 4번 이상이면...
            if (failCount > 3)
            {
                // 에러폴더로 옮긴다...
                FileErrorMove(pFile, "Parsing");
                failCount = 0;
                Thread.Sleep(50);
                continue;
            }
```

Main work 차량 매칭 (2)

MatchingStart

```
if (!m_bMatching) break;

// 이미지 정보를 저장한다.
ClsIMGVehicledata IVData = new ClsIMGVehicledata();

if (!IsDataParseing(IVData, pFile, ref failCount))
{
    failCount++;
    Thread.Sleep(50);
    continue;
}
log.Qry_Log_Save(5, SectionCode + "_" + Lane, LogName, funcname + " - IsDataParseing Success");

if (!m_bMatching) break;

// 인식 실패, 제어기번호가 다른지 검사한다...
if (!IsFailFileJudg(IVData, pFile))
{
    Thread.Sleep(50);
    continue;
}
log.Qry_Log_Save(5, SectionCode + "_" + Lane, LogName, funcname + " - IsFailFileJudg Success");

if (!m_bMatching) break;
```

```
        if (!m_bMatching) break;

        // 매칭 시작
        if (MatchFlag == "F")
        {
            // 매칭 할 필요없는 데이터 처리
            if (!IsNotMatching(IVData, pFile))
            {
                Thread.Sleep(50);
                continue;
            }
            log.Qry_Log_Save(5, SectionCode + "_" + Lane, LogName, funcname + " - MatchFlag False : IsNotMatching Success");
        }
        else if (MatchFlag == "T")
        {
            // 매칭 해야하는 데이터 처리
            if (!IsMatching(IVData, pFile))
            {
                Thread.Sleep(50);
                continue;
            }
            log.Qry_Log_Save(5, SectionCode + "_" + Lane, LogName, funcname + " - MatchFlag True : IsMatching Success");
        }

        failCount = 0;
        //Thread.Sleep(50);

        if (!m_bMatching) break;
    }
}
```

Main work Insert 쿼리문 생성 (통과차량, 매칭차량)

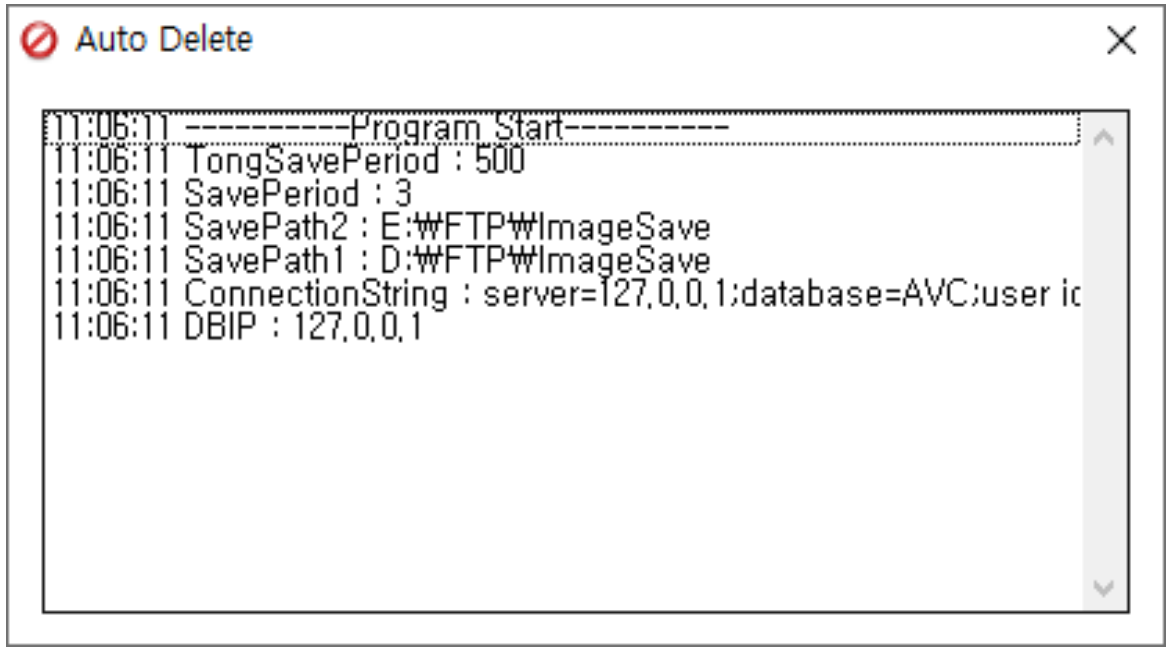
```
public string PassData_Insert(bool SpeedUse, string SiteCode, string Lane,
                              string Speed, string PassDate,
                              string VehicleNum, string FR, string MatchFlag)
{
    // 초기 작성 - kkh
    //string Insert_qry = @"EXEC InputPassData @SiteCode = '" + SiteCode + "', @
    //                                                                          +
    //                                                                          +

    // 2020_01_03 수정 - kkh
    string qry = string.Empty;
    // Ver 1.2.0 속도 컬럼 추가로 인하여 쿼리문 추가
    if (SpeedUse)
    {
        qry = @"EXEC InputPassData "
                    + "'" + SiteCode + "', "
                    + "'" + Lane + "', "
                    + "'" + Speed + "', "
                    + "'" + PassDate + "', "
                    + "'" + VehicleNum + "', "
                    + "'" + FR + "', "
                    + "'" + MatchFlag + "';";
    }
    else
    {
        qry = @"EXEC InputPassData "
                    + "'" + SiteCode + "', "
                    + "'" + Lane + "', "
                    + "'" + PassDate + "', "
                    + "'" + VehicleNum + "', "
                    + "'" + FR + "', "
                    + "'" + MatchFlag + "';";
    }
    return qry;
}
```

```
public string SectionData_Insert(bool SpeedUse, string SiteCode, string SectionNumber,
                                 string SectionVector, string TunnelCode, string isVehicleNumber,
                                 string VehicleNum_F, string VehicleNum_R,
                                 string Speed_F, string Speed_R,
                                 string PassDate_F, string PassDate_R,
                                 string Lane_F, string Lane_R)
{
    string qry = string.Empty;
    // Ver 1.2.0 속도 컬럼 추가로 인하여 쿼리문 추가
    if (SpeedUse)
    {
        // SectionData도 인덱스를 PassData처럼 잡아보기 위해 만든 프로시저인데 사용 안할거같다....
        // 혹시 모르니 지워두지말고 남겨놓자...
        //qry = @"EXEC InputSectionData "
        //        + "'" + SiteCode + "','" + SectionNumber + "','" + SectionVector + "','"
        //            + TunnelCode + "','" + isVehicleNumber + "','"
        //            + VehicleNum_F + "','" + VehicleNum_R + "','"
        //            + Speed_F + "','" + Speed_R + "','"
        //            + PassDate_F + "','" + PassDate_R + "',"
        //            + Lane_F + "," + Lane_R + ", 'N';";

        qry = @"INSERT INTO SectionData(SectionCode,SectionNumber,SectionVector,tunnelCode,isVehicleNumber,
        VehicleNumber_Front,VehicleNumber_Back,Speed_Front,Speed_Back,PassDate,PassDate_Back,Lane,Lane_Back,SendFlag) "
                    + "VALUES('" + SiteCode + "','" + SectionNumber + "','" + SectionVector + "','"
                    + TunnelCode + "','" + isVehicleNumber + "','"
                    + VehicleNum_F + "','" + VehicleNum_R + "','"
                    + Speed_F + "','" + Speed_R + "','"
                    + PassDate_F + "','" + PassDate_R + "',"
                    + Lane_F + "," + Lane_R + ", 'N');";
    }
    else
    {
        qry = @"INSERT INTO SectionData(SectionCode,SectionNumber,SectionVector,tunnelCode,isVehicleNumber,
        VehicleNumber_Front,VehicleNumber_Back,PassDate,PassDate_Back,Lane,Lane_Back,SendFlag) "
                    + "VALUES('" + SiteCode + "','" + SectionNumber + "','" + SectionVector + "','"
                    + TunnelCode + "','" + isVehicleNumber + "','"
                    + VehicleNum_F + "','" + VehicleNum_R + "','"
                    + PassDate_F + "','" + PassDate_R + "',"
                    + Lane_F + "," + Lane_R + ", 'N');";
    }
    return qry;
}
```

Main work 데이터 삭제

- **기능 소개**

- 일정 기간의 데이터만 저장되도록 데이터를 삭제해 주는 프로그램



Main work 데이터 삭제 (1)

```
public void DeleteThread()
{
    DateTime sDateTime = DateTime.Parse("2000-01-01 00:00:00");

    string SelectString = "SELECT MIN(PassDate) AS MinPassDate FROM PassData";

    SqlDataReader result = cDB.CreateReader(SelectString, _Init.ConnectionString);

    if (result != null)
    {
        if (result.HasRows)
        {
            result.Read();

            if (result["MinPassDate"].ToString() != "")
            {
                sDateTime = sDateTime.AddHours(Convert.ToInt32(result["MinPassDate"].ToString()));

                if (sDateTime.CompareTo(DateTime.Now.AddDays(_Init.SavePeriod * -1 - 1)) < 0)
                {
                    string strsql = "DELETE FROM SectionData WHERE PassDate <= '" + sDateTime.ToString("yyyy-MM-dd 23:59:59.999") + "'";

                    int DelCnt = cDB.CreateCommand(strsql, _Init.ConnectionString);

                    DSetLog("매칭 데이터 삭제 [" + sDateTime.ToString("yyyy-MM-dd") + "] " + DelCnt.ToString() + "건");

                    strsql = "DELETE FROM PassData WHERE PassDate <= CAST(DATEDIFF(hour, '2000-01-01 00:00:00', '" + sDateTime.ToString("yyyy-MM-dd 23:59:59") + "') AS int)";

                    DelCnt = cDB.CreateCommand(strsql, _Init.ConnectionString);

                    DSetLog("통과 데이터 삭제 [" + sDateTime.ToString("yyyy-MM-dd") + "] " + DelCnt.ToString() + "건");
```

Main work 데이터 삭제 (2)

```
                if (_Init.ImageDelete == 1)
                {
                    try
                    {
                        for (int i = 0; i < _Init.rowcount; i++)
                        {
                            DirectoryInfo di = new DirectoryInfo(_Init.SavePath[i] + @"\" + sDateTime.ToString("yyyy") + @"\" + sDateTime.ToString("MM") + @"\" + sDateTime.ToString("dd"));
                            if (di.Exists && _Init.SavePath[i] != "")
                            {
                                DSetLog("영상삭제시작 : " + _Init.SavePath[i] + @"\" + sDateTime.ToString("yyyy") + @"\" + sDateTime.ToString("MM") + @"\" + sDateTime.ToString("dd"));
                                di.Delete(true);
                                DSetLog("영상삭제완료 : " + _Init.SavePath[i] + @"\" + sDateTime.ToString("yyyy") + @"\" + sDateTime.ToString("MM") + @"\" + sDateTime.ToString("dd"));
                            }
                        }
                    }
                    catch (System.IO.IOException e)
                    {
                        DSetLog("영상삭제실패 [" + sDateTime.ToString("yyyy-MM-dd") + "] : " + e.Message);
                    }
                    catch (Exception e)
                    {
                        DSetLog("영상삭제실패 [" + sDateTime.ToString("yyyy-MM-dd") + "] : " + e.Message);
                    }
                }
            }
        }
    }
    else
    {
        DSetLog("데이터 없음");
    }
}
else
{
    DSetLog("데이터 삭제 조회 실패..");
}

DeleteThreadFlag = false;
```

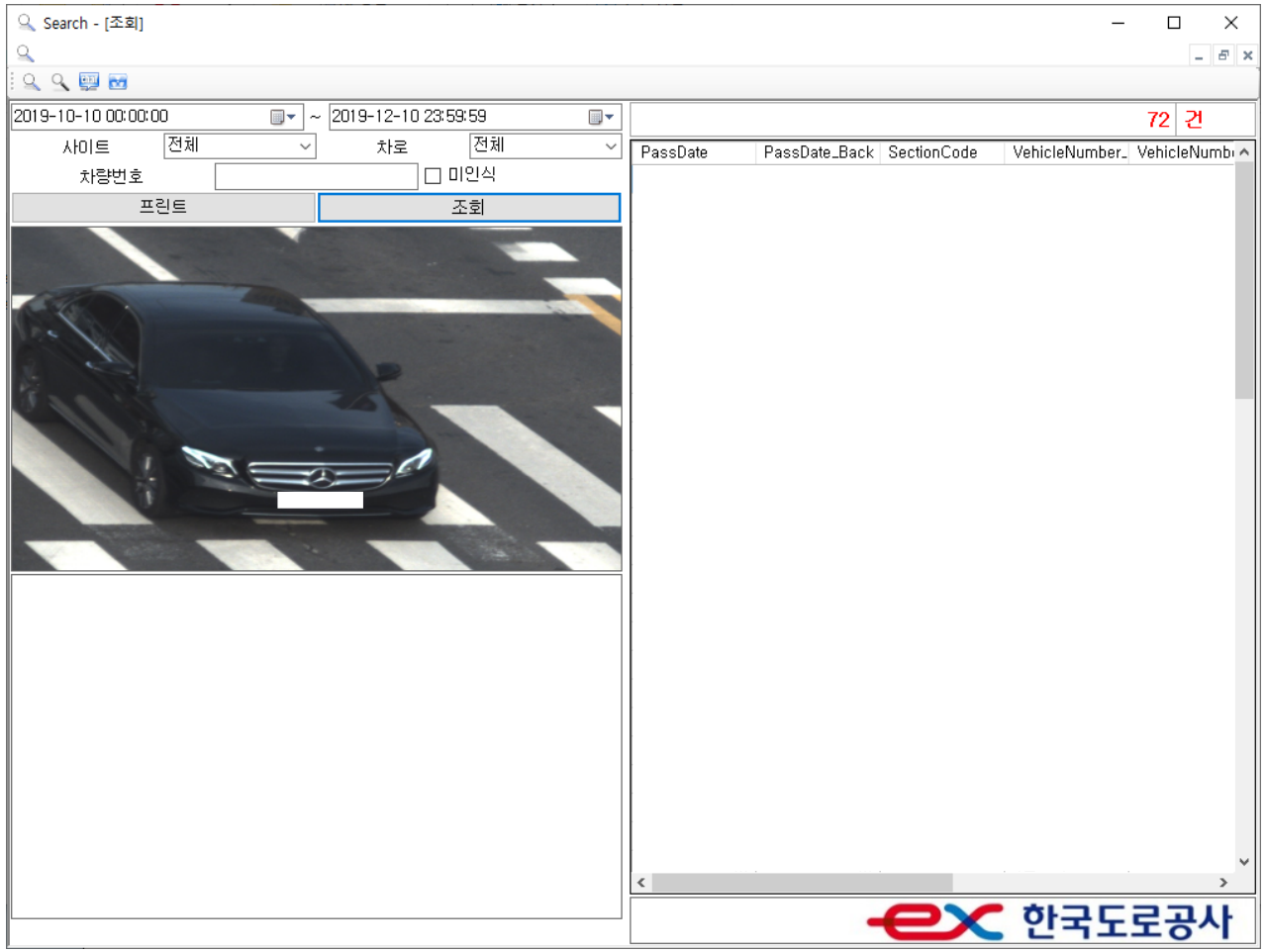
Main work 데이터 조회

- **기능 소개**

- 시간대별, 사이트별, 차로별, 전후면 조회

- 차량번호 및 미인식 결과 조회

- 선택된 영상 프린터 출력



Main work 전송 시스템

- **기능 소개**

- 도로 공단 서버로 매칭된 데이터를 전송

Main work 데이터 전송

```
private void sendRun()
{
    ArrayList arrMag = null;

    while (true)
    {
        try
        {
            if (client.Connected && clientFile.Connected)
            {

                if (!isTest)
                    arrMag = violationMag.getMessageList();
                else
                {
                    arrMag = violationMag.getTestMessageList();
                    isTest = false;
                }

                if (arrMag != null)
                {
                    sendMage(arrMag);

                    for (int index = 1; index < arrMag.Count; index += 2)
                    {

                        Array.Clear(((byte[])arrMag[index]), 0, ((byte[])arrMag[index]).Length);
                        arrMag[index] = null;
                    }
                    arrMag.Clear();
                    arrMag = null;

                    Thread.Sleep(500);
                }
                else
                {
                    SystemMessage("정보 없음....");

                    Thread.Sleep(5000);
                }


            }
            else
                Thread.Sleep(5000);
        }
        catch (ThreadInterruptedException ex)
        {
            log.Error(0,"sendRun(" + type + ")(ThreadInterruptedException) : " + ex);
            break;
        }
        catch (Exception ex)
        {
            log.Error(0,"sendRun(" + type + ") : " + ex);
        }
    }
}
```

Main work 전송 플래그 업데이트

```
public ArrayList getMessageList()
{
    ArrayList arrMag = null;
    try
    {
        DataTable data = getViolationData();

        if (data != null && data.Rows.Count > 0)
        {
            if (type.Trim() == "rank")
            {
                updateSendFlag(data.Rows[0]["rankDrivingDataID"].ToString().Trim(), "C");
                arrMag = makeRankVioMessage(data);
            }
            else
            {
                updateSendFlag(data.Rows[0]["laneViolationDataID"].ToString().Trim(), "C");
                arrMag = makeLaneVioMessage(data);
            }
            if (arrMag != null && arrMag.Count < 1)
            {
                arrMag.Clear();
                arrMag = null;

                if (type.Trim() == "rank")
                    updateSendFlag(data.Rows[0]["rankDrivingDataID"].ToString().Trim(), "F");
                else
                    updateSendFlag(data.Rows[0]["laneViolationDataID"].ToString().Trim(), "F");
            }
        }
        else
        {
            SystemMessage("위반 정보가 없음");
            log.Info(6, "getMessageList(" + type + ") : violatio data not existence");
        }
    }
    catch (System.Exception ex)
    {
        log.Error(0,"getMessageList(" + type + ") ", ex);
    }

    return arrMag;
}
```

Main work 전송 데이터 가져오기

```
private DataTable getViolationData()
{
    DataTable res = null;
    try
    {
        String sql = "";
        String tableName = "";

        if (type.Trim() == "rank")
        {
            sql = "SELECT data.rankDrivingDataID as rankDrivingDataID,SectionCode, " +
                        "SectionNumber,SectionVector,tunnelCode,isVehicleNumber,VehicleNumber_Front, " +
                        "VehicleNumber_Back,Lane,PassDate,FullImageFileName,ImageFileSize, " +
                        "videoClipName,videoClipSize,videoClipFlag,SendFlag " +
                    "FROM rankDrivingData as data ,rankDrivingVideoClipData as videoClip " +
                    "WHERE data.rankDrivingDataID = videoClip.rankDrivingDataID " +
                        "AND data.rankDrivingDataID=(SELECT TOP 1 rankDrivingDataID FROM rankDrivingData " +
                            "WHERE videoClipFlag='Y' AND SendFlag='N' GROUP BY rankDrivingDataID);";
            tableName = "rankDrivingData";
        }
        else
        {
            sql = "SELECT TOP 1 laneViolationDataID,SectionCode,SectionNumber,SectionVector " +
                        ",tunnelCode,isVehicleNumber,VehicleNumber_Front,VehicleNumber_Back,Lane " +
                        ",PassDate,videoClipName,videoClipSize,FullImageFileName,ImageFileSize " +
                        ",videoClipFlag,SendFlag " +
                    "FROM laneViolationData " +
                    "WHERE videoClipFlag = 'Y' AND SendFlag='N'";
            tableName = "laneViolationData";
        }
        log.Info(6, "getViolationData(" + type + ") : " + sql);
        res = DBlib.SelectDataTable(sql, tableName);

        if (res != null && res.Rows.Count < 1)
        {
            res.Clear();
            res = null;
        }
    }
    catch (System.Exception ex)
    {
        log.Error(0,"getPassData(" + type + ") ", ex);
    }

    return res;
}
```

Project 03.

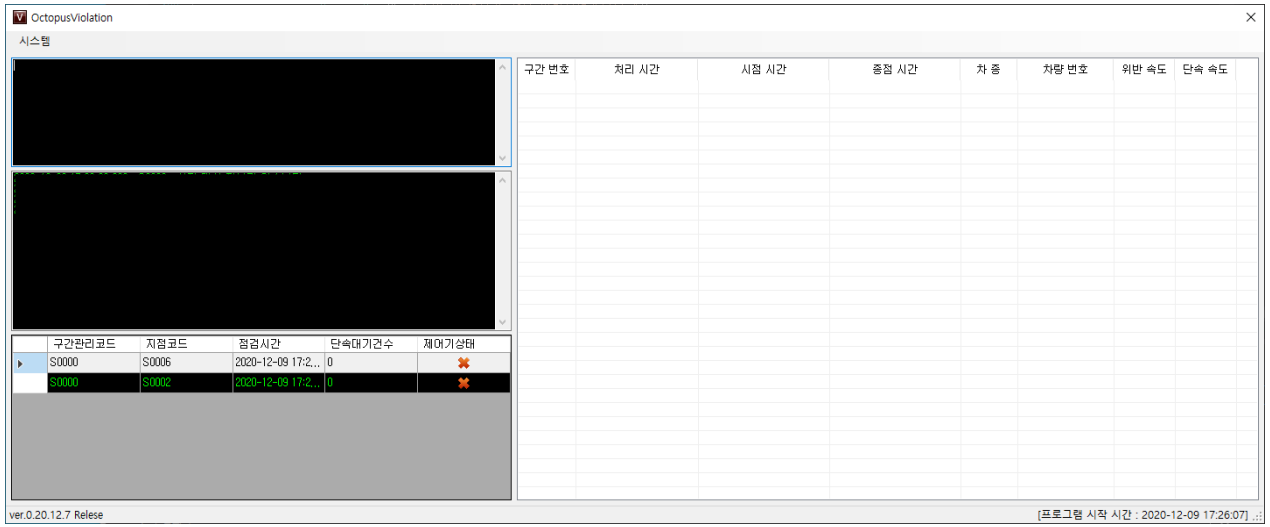
스마트 구간 서버

About project

구간 단속을 위한 프로그램.
속도 위반 차량의 데이터를 경찰청 서버에 전송한다.

Introduce project

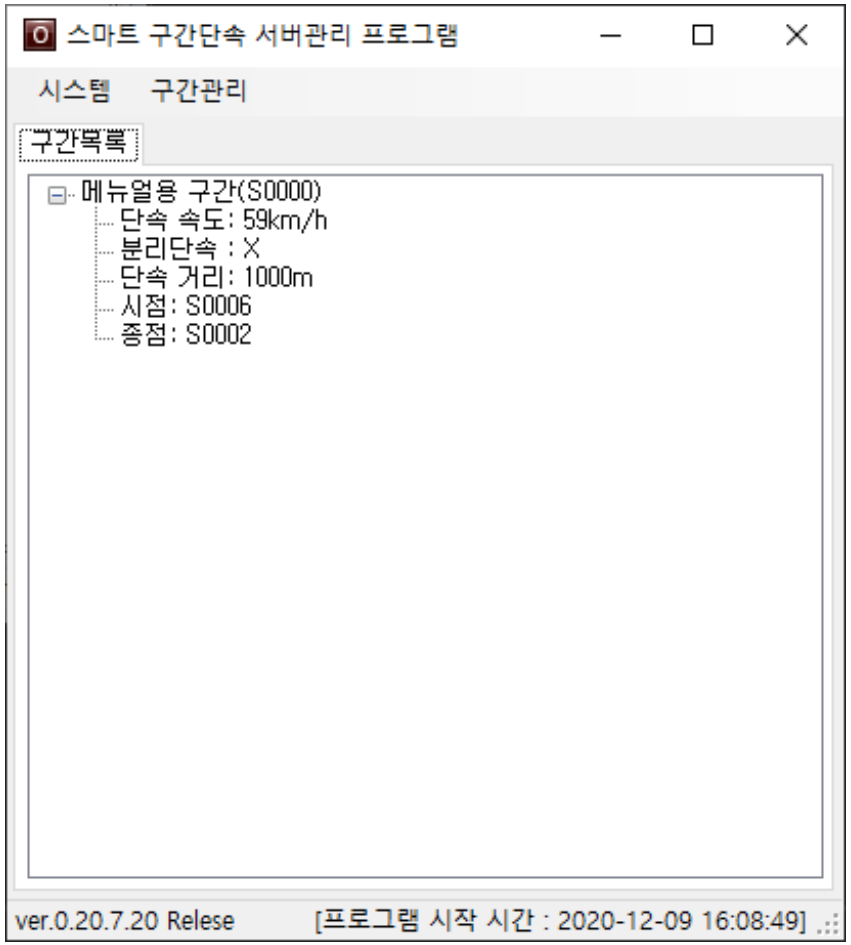
작업 기간	2019.10 ~ 2020.08 (약 10개월)
인력 구성(기여도)	1명
프로젝트 목적	구간단속을 목적으로 한 서버 프로그램
프로젝트 내용	현장 PC에서 서버로 데이터를 전송해주면 시점과 종점을 기준으로 속도를 계산하여 속도위반 데이터를 영상과 함께 경찰청 서버로 전송해준다.
주요 업무 및 상세 역할	1) 현장에서 들어온 영상과 데이터를 저장. 2) 데이터를 가지고 구간 속도를 계산 3) 지점 과속 데이터를 생성 4) 구간 과속 데이터를 생성 5) 경찰청 서버에 데이터를 전송
사용언어 및 개발 환경	C#, MS-SQL
참고 자료	



Main work 시종점 등록 프로그램

- **기능 소개**

- 시 종점 사이트의 정보를 입력할 수 있다.
- 구간 정보를 입력하여 구간 단속 범위를 생성한다.
- 생성한 사이트 정보, 구간 정보를 MS-Server에 저장한다.



Main work 시종점 정보 등록

```
private bool InsertSiteInfo(SITEINFO target)
{
    try
    {
        if (target == null)
        {
            log.Error(0,"InsertSiteInfo : target is null");
            return false;
        }

        //생성한 테이블에 정보 입력
        String qry = "INSERT INTO SiteInfo (SiteCode,SiteName,SiteKind,CrackDownKind,SiteCrackDownFlag,LimitSpeed,POS"
                    +",LocalPos,Memo,FTPIP,FTPPORT,FTPID,FTPPW,EqCode) VALUES "
                    +"('@NEWSITECODE','@SITENAME','@SITEKIND','@CRACKDOWNKIND', @SITECRACKDOWNFLAG,@LIMITSPEED,@POS,'@LOCALPOS',"
                    +"'@MEMO','@FTPIP','@FTPPORT','@FTPID','@FTPPW','@EQCODE')";

        qry = qry.Replace("@NEWSITECODE", target.SiteCode);
        qry = qry.Replace("@SITENAME", target.SiteName);
        qry = qry.Replace("@SITEKIND", target.SiteKind.Code);
        qry = qry.Replace("@CRACKDOWNKIND", target.CrackDownKind.Code);
        qry = qry.Replace("@SITECRACKDOWNFLAG", target.CrackDownFlag.ToString());
        qry = qry.Replace("@LIMITSPEED", target.LimitSpeed.ToString());
        if (target.POS!=null)
        {
            qry = qry.Replace("@POS", target.POS.GetPointQry());
        }
        else
        {
            qry = qry.Replace("@POS", "NULL");
        }
        qry = qry.Replace("@LOCALPOS", target.LocalPos);
        qry = qry.Replace("@MEMO", target.Memo);
        qry = qry.Replace("@FTPIP", target.FTPInfo.FtpPath);
        qry = qry.Replace("@FTPPORT", target.FTPInfo.port.ToString());
        qry = qry.Replace("@FTPID", target.FTPInfo.UserID);
        qry = qry.Replace("@FTPPW", target.FTPInfo.UserPW);
        qry = qry.Replace("@EQCODE", target.EQCode);

        return DBLib.QureyData(qry);
    }
    catch (System.Exception ex)
    {
        log.Error(0,"InsertSiteInfo Exception : " + ex.Message);
        return false;
    }
}
```

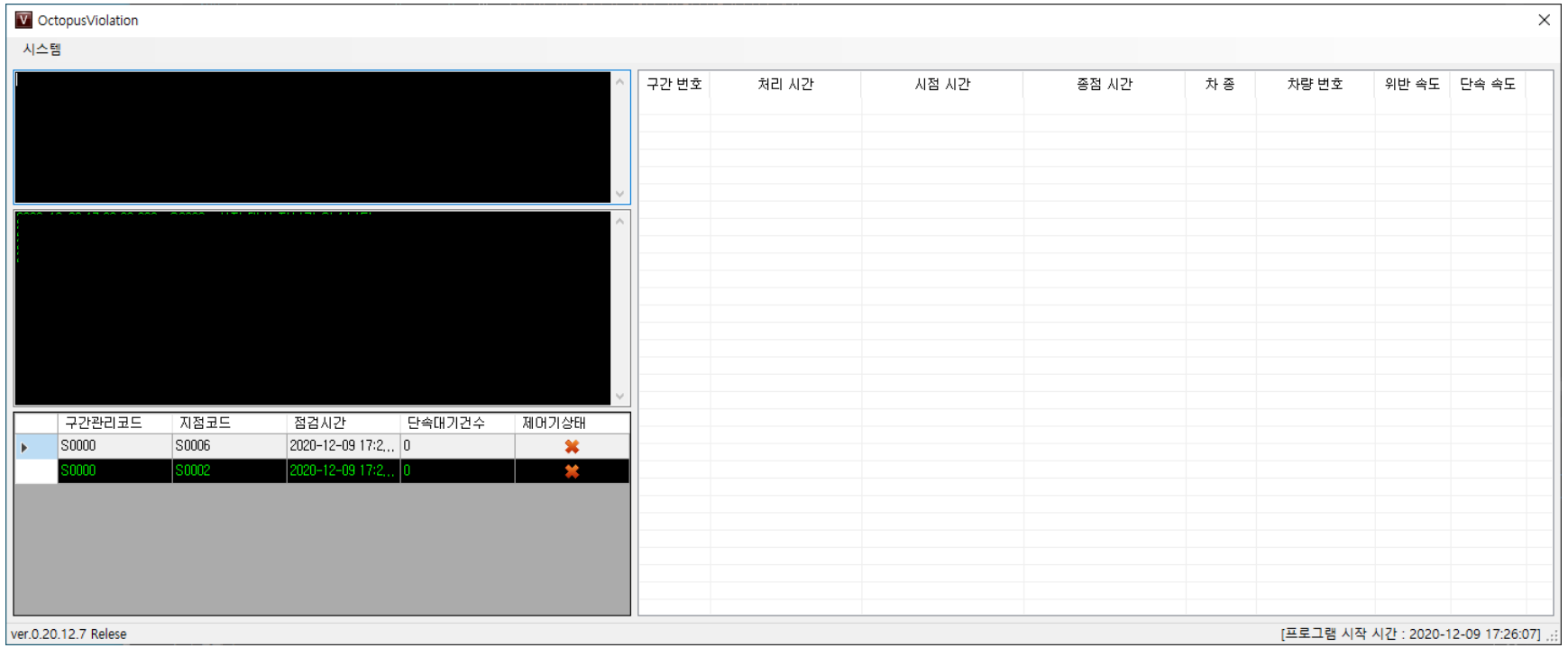
Main work 구간 정보 등록

```
private bool InsertSectionInfo()
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    try
    {
        if (!CheckSectionBaseInfo())
        {
            return false;
        }

        if (!CheckCrackDownInfo())
        {
            return false;
        }
        // kkh : 2020-02-20 분리단속 쿼리 생성
        string qry = "INSERT INTO SectionInfo (SectionNm,StartLocalNo,EndLocalNo,SectionLocalNo,Distance,LimitTravelTime "
                        + ",LimitSpeed,BigLimitSpeed"
                        + ",CrackDownFlag,DivCrackDownFlag,SiteCrackDownFlag"
                        + ",GISInfo)"
                        + " VALUES ('@SECTIONNAME','@STARTSITECODE','@ENDSITECODE','@SECTIONCODE',@DISTANCE,@LIMITTRAVELTIME"
                        + ",@LIMITSPEED, @BIGLIMITSPEED"
                        + ",@CRACKDOWNFLAG, @DIVCRACKDOWNFLAG, @SITECRACKDOWNFLAG"
                        + ",@GISPOS)";

        qry = SetSectionInfoQry(qry);
        if (!DBLib.QureyData(qry))
        {
            log.Error(0, funcname + " : " + new ArgumentException("Failed Insert " + qry));
            return false;
        }
        else
        {
            if (!AddtSectionSiteInfo())
            {
                log.Error(0, funcname + ": AddtSectionSiteInfo retrun false");
                return false;
            }
            else
            {
                if (!CreateSiteTable())
                {
                    log.Error(0, funcname + ": CreateSiteTable return false");
                    return false;
                }
                //구간 정보 테이블 생성
                if (!CreateSectionTable(txbSectionCode.Text))
                {
                    log.Error(0, funcname + ": CreateSectionTable return false");
                    return false;
                }
            }
        }
        return true;
    }
    catch (System.Exception ex)
    {
        log.Error(0, "InsertSectionInfo Exception : " + ex.Message);
        return false;
    }
}
```

Main work 단속 시스템

- **기능 소개**

- 저장한 사이트 정보와 구간 정보를 기준으로 단속을 시작한다.
- 현장에서 전달 받은 데이터와 영상을 사용하여 지점 단속과 구간단속 모두를 진행한다.
- 운영 모드와 시험 모드로 변경할 수 있다.



Main work 지점 단속(1)

```
public void EnforcementSite(PASSCARINFO target, SECTIONINFO sectionvalue)
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    String dbviolationflag = ConfigInfo.GetValue("CrackDownDBFlag", "ViolationFlag");
    String dbfailviolationflag = ConfigInfo.GetValue("CrackDownDBFlag", "FailViolationFlag");
    String dbnotviolationflag = ConfigInfo.GetValue("CrackDownDBFlag", "NotViolationFlag");
    try
    {
        //해당 차종이 대형인지 소형인지 확인한다.
        int passspeed = -1;
        int limitspeed = -1;
        //log.Debug(funcname + " : 지점단속 시작, " + violationinfo);
        bool violationflag = CheckViolation(target, ref passspeed, ref limitspeed);

        if (violationflag)
        {
            //위반
            if (ProcViolationVechicle(target, sectionvalue))
            {
                String violationinfo = target.SiteCode + ", " + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss")
                    + ", 차종: " + target.VehicleKind
                + ", 위반속도: " + passspeed.ToString() + " > " + limitspeed.ToString()
                + ", 차량번호: " + target.VehicleNumber;

                //파일 생성
                UpdateCrackDownResult(target, sectionvalue, dbviolationflag);

                string siteviomsg = target.SiteCode + "_" + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss")
                    + "_" + target.VehicleKind
                    + "_" + target.VehicleNumber + "_" + passspeed.ToString() + "_" + limitspeed.ToString();

                OnCrackDownStatus("지점위반, " + violationinfo + " 단속");
            }
            else
            {
                String violationinfo = target.SiteCode + ", " + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss")
                    + ", 차종: " + target.VehicleKind
                + ", 위반속도: " + passspeed.ToString() + " > " + limitspeed.ToString()
                + ", 차량번호: " + target.VehicleNumber;

                UpdateCrackDownResult(target, sectionvalue, dbfailviolationflag);
                OnCrackDownStatus("지점위반, " + violationinfo + " 단속 실패");
            }
        }
        else
        {
            //정상
            String violationinfo = target.SiteCode + ", " + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss")
                + ", 차종: " + target.VehicleKind
            + ", 위반속도: " + passspeed.ToString() + " < " + limitspeed.ToString()
            + ", 차량번호: " + target.VehicleNumber;

            //svioflag=2로 변경하고 sendflag=P로 변경한다.
            UpdateCrackDownResult(target, sectionvalue, dbnotviolationflag);
            log.Debug(0, funcname + " : 지점단속 완료, " + violationinfo);
        }
    }
    catch (System.Exception ex)
    {
        log.Error(0, funcname + " : " + ex.ToString());
    }
}
```

Main work 지점 단속(2)

```
private bool CheckViolation(PASSCARINFO target, ref int passspeed, ref int limitspeed)
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    try
    {
        //해당 차종이 대형인지 소형인지 확인한다.
        //1:소형,2:버스/승합,3:대형화물,4:소형화물,5:특장
        int carkind = Convert.ToInt32(target.VehicleKind);
        int HeavyTruckType = Convert.ToInt32(ConfigInfo.GetValue("CrackDownVehicleType", "HeavyTruck"));
        int Specialvehicle = Convert.ToInt32(ConfigInfo.GetValue("CrackDownVehicleType", "SpecialVehicle"));
        int madmaxspeed = Convert.ToInt32(ConfigInfo.GetValue("SystemInfo", "MadMax"));
        bool violationflag = false;
        if (target.StandardTimeState != 255 && target.Speed > 0)
        {
            if (carkind == HeavyTruckType || carkind == Specialvehicle)
            {
                //대형이면 대형 단속 속도와 통과 속도를 비교한다.3:대형트럭, 5:특수차
                passspeed = target.Speed;
                limitspeed = target.LVechicleLimitSpeed;
                if (target.LVechicleLimitSpeed != madmaxspeed && target.Speed > target.LVechicleLimitSpeed)
                {
                    //위반
                    violationflag = true;
                }
            }
            else
            {
                //소형이면 소형 단속 속도와 통과 속도를 비교한다.1:승용차,2:버스/승합,4:소형트럭
                passspeed = target.Speed;
                limitspeed = target.VechicleLimitSpeed;
                if (target.VechicleLimitSpeed != madmaxspeed && target.Speed > target.VechicleLimitSpeed)
                {
                    //위반
                    violationflag = true;
                }
            }
        }
        else
        {
            String msg = target.SiteCode + ", " + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss.fff")
                + ", 표준시간상태: " + target.StandardTimeState.ToString()
                + ", 차량번호: " + target.VehicleNumber + ", 속도: " + target.Speed.ToString();
            if (target.StandardTimeState != 255)
            {
                log.Debug(0, funcname + " : 표준시간 동기화 실패 단속대상 아님, " + msg);
            }
            else
            {
                log.Debug(0, funcname + " : 속도 0, 단속대상 아님, " + msg);
            }

            OnCrackDownStatus(msg);
        }

        return violationflag;
    }
    catch (System.Exception ex)
    {
        log.Error(0, funcname + " : " + ex.ToString());
        return false;
    }
}
```

Main work 지점 단속(3)

```
private bool ProcViolationVechicle(PASSCARINFO target, SECTIONINFO sectionvalue)
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    String msg = String.Empty;
    try
    {
        if (target == null)
        {
            log.Error(0, funcname + (" : target is null"));
            return false;
        }
        msg = target.SiteCode + ", " + target.PassDate.ToString("yyyy-MM-dd HH:mm:ss.fff") + ", "
                + target.VehicleNumber + " 이미지 다운로드에 실패하였습니다.";
        //위반이면 해당 영상을 FTP로 다운로드 한다.
        target.ViolationImageInfo = null;

        if (!DownLoadSiteImage(target, ref target.ViolationImageInfo, sectionvalue))
        {
            log.Error(0, funcname + (" : DownLoadSiteImage retrun false"));
            log.Error(0, funcname + " : " + msg);
            OnCrackDownStatus(msg);
            return false;
        }
        else
        {
            //해당 파일을 생성한다.
            if (!CreateViolationFile(target))
            {
                log.Error(0, funcname + (" : CreateViolationFile retrun false"));
                log.Error(0, funcname + " : " + msg);
                OnCrackDownStatus(msg);
                return false;
            }

            //이미지 해당 정보를 DB에 갱신한다.
            if (!UpdateViolationImageInfo(target, sectionvalue))
            {
                //실패
                log.Error(0, funcname + (" : UpdateViolationImageInfo retrun false"));
                log.Error(0, funcname + " : " + msg);
                OnCrackDownStatus(msg + " Image don't Update");
                return false;
            }
        }

        return true;
    }
    catch (System.Exception ex)
    {
        log.Error(0, funcname + " : " + ex.ToString());
        return false;
    }
}
```

Main work 지점 단속(4)

```
private bool UpdateViolationImageInfo(PASSCARINFO info, SECTIONINFO sectionvalue)
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    try
    {
        if (info == null)
        {
            log.Error(0, funcname + (" : info is null"));
            return false;
        }

        if (info.ViolationImageInfo == null)
        {
            log.Error(0, funcname + (" : imageinfo is null"));
            return false;
        }
        String qry = "UPDATE @TABLENAME SET [FullImageFileName] = '@FULLIMAGENAME',[PlateImageFileName] = '@PLATEIMAGENAME' "
                   + "WHERE PassDate ='@PASSDATEVALUE' and VehicleNumber='@CARNUM' and msec=@MSECVALUE";
        String tablename = ConfigInfo.GetValue("SystemInfo", "PassCarInfoTable");
        tablename += sectionvalue.SectionLocalNo + info.LocationFlag;
        qry = qry.Replace("@TABLENAME", tablename);
        qry = qry.Replace("@FULLIMAGENAME", info.ViolationImageInfo.FullImageInfo.ImageFullName);
        qry = qry.Replace("@PLATEIMAGENAME", info.ViolationImageInfo.PlateImageInfo.ImageFullName);
        qry = qry.Replace("@CARNUM", info.VehicleNumber);
        qry = qry.Replace("@PASSDATEVALUE", info.PassDate.ToString("yyyy-MM-dd HH:mm:ss"));
        qry = qry.Replace("@MSECVALUE", info.msec.ToString());
        if (DBLib.QureyData(qry))
        {
            return true;
        }
        else
        {
            log.Error(0, funcname + " : QureyData retrun false, " + qry);
            return false;
        }
    }
    catch (System.Exception ex)
    {
        log.Error(0, funcname + " : " + ex.ToString());
        return false;
    }
}
```

Main work 구간 단속(1)

```
private bool ProcCrackDown(DataTable target, bool startsiteflag)
{
    String funcname = MethodBase.GetCurrentMethod().Name;

    String nowVehicleInfo = String.Empty;
    PASSCARINFO cheksectioninfo = null;
    String locationflag = "2";
    if (startsiteflag)
        locationflag = "1";

    try
    {
        if (target == null)
        {
            log.Error(0, funcname + (" : target == null"));
            return false;
        }
        if (target.Rows.Count < 1)
        {
            log.Error(0, funcname + (" : target.Rows.Count < 1"));
            return false;
        }
```

```
for (int i = 0; i < target.Rows.Count; i++)
{
    int TestMode = Convert.ToInt32(ConfigInfo.GetValue("TestModeInfo", "TestMode"));

    String sectiondataid = String.Empty;
    DataRow checkinfo = target.Rows[i];
    String nowinfo = NowSectionInfo.SectionLocalNo + " : " + checkinfo["VehicleNumber"].ToString();
    log.Debug(0, funcname + " : checkinfo = " + nowinfo);
    cheksectioninfo = new PASSCARINFO(checkinfo);

    if (cheksectioninfo != null && !IsViolationInfo(cheksectioninfo, startsiteflag))
    {

        //단속 작업 시작
        log.Debug(0, funcname + " : Start " + nowinfo);
        Thread.Sleep(1);
        if (startsiteflag)
        {
            cheksectioninfo.LocationFlag = "1";
        }
        else
        {
            cheksectioninfo.LocationFlag = "2";
        }
        //지점 단속 모드이면 지점 단속 진행
        if (NowSectionInfo.SiteCrackDownFlag >= 1)
        {
            log.Debug(0, funcname + " : Site CrackDown Start " + nowinfo);
            StartEnforcementSite(cheksectioninfo);
        }
```

Main work 구간 단속(2)

```
//다른 지점정보의 테이블에 시간, 밀리초, 차량번호로 해당 차량이 존재하는지 확인한다.
PASSCARINFO othersiteinfo = IsSiteCarPassInfo(cheksectioninfo, startsiteflag, NowSectionInfo);
Thread.Sleep(1);
if (othersiteinfo != null)
{
    log.Debug(1, funcname + " : ProcViolation 2 Violation Start " + nowinfo);
    //존재하면 단속 시작
    TRAVELINFO travelresult = new TRAVELINFO();
    log.Debug(1, "ProcViolation 2 : VehicleNumber = " + cheksectioninfo.VehicleNumber);
    travelresult.CarNumber = cheksectioninfo.VehicleNumber;
    PASSCARINFO startpassinfo = null;
    PASSCARINFO endpassinfo = null;
    //단속
    //단속된 데이터를 SectionData에 입력
    if (startsiteflag)
    {
        //시점인 경우
        startpassinfo = cheksectioninfo;
        endpassinfo = othersiteinfo;
    }
    else
    {
        //종점인경우
        startpassinfo = othersiteinfo;
        endpassinfo = cheksectioninfo;
    }
    Thread.Sleep(1);
```

```
if (CheckViolation(startpassinfo, endpassinfo, ref travelresult))
{
    //위반 차량
    log.Debug(1, "ProcViolation 3 : SendAliveMessage " + nowinfo);
    Thread.Sleep(1);
    //속도 계산하여 미친속도면 단속에서 제외
    String sendflag = CheckMadMaxSpeed(travelresult, nowinfo);
    String violationmsg = string.Empty;
    string viosectionmsg = string.Empty;

    violationmsg = GetViolationMsg(NowSectionInfo.DivCrackDownFlag, startpassinfo, endpassinfo, cheksectioninfo, travelresult);
    viosectionmsg = GetVioSectionMsg(NowSectionInfo.DivCrackDownFlag, startpassinfo, endpassinfo, cheksectioninfo, travelresult);

    Thread.Sleep(1);
    sectiondataid = InsertCrackDownViolation(startpassinfo, endpassinfo, travelresult, sendflag);

    if (sectiondataid != String.Empty)
    {
        Thread.Sleep(1);
        log.Debug(1, "ProcViolation 6 : SendAliveMessage " + nowinfo);
        OnSectionVehicleData(viosectionmsg);

        if (TestMode == 0)
        {
            if (!CreateSectionViolationFile(startpassinfo, endpassinfo, travelresult, sectiondataid))
            {
                log.Error(0, funcname + (" : CreateSectionViolationFile return false " + nowinfo));
                OnCrackDownStatus("Don't Create Violation File... " + violationmsg);
                Thread.Sleep(1);
            }
        }
    }
    else
    {
        log.Error(0, funcname + (" : InsertCrackDownViolation return false " + nowinfo));
        Thread.Sleep(1);
    }
}
```

Main work 구간 단속(3)

```
else
{
    //정상 통과 차량, 입력하고 vioflag=2
    log.Debug(1, "ProcViolation 7 : Not Violation " + nowinfo);
    Thread.Sleep(1);
    //통과차량 정보만 입력
    sectiondataid = InsertSectionPassInfo(startpassinfo, endpassinfo, travelresult);
    if (sectiondataid != String.Empty)
    {
        log.Debug(1, "ProcViolation 8 : SendAliveMessage " + nowinfo);
        Thread.Sleep(1);
    }
    else
    {
        log.Debug(0, funcname + " : InsertSectionPassInfo return String.Empty, " + nowinfo);
    }
}
othersiteinfo.LocationFlag = "1";

if (startsiteflag)
{
    othersiteinfo.LocationFlag = "2";
}
```

```
    //다른 지점 정보를 갱신하여 다시 확인하지 않도록 설정
    if (!UpdateViolationFlagPassCarInfo(othersiteinfo, NowSectionInfo.SectionLocalNo,
        othersiteinfo.LocationFlag))
    {
        log.Error(0, "ProcViolation : " + othersiteinfo.SiteCode
            + " , " + othersiteinfo.VehicleNumber + " , "
            + othersiteinfo.PassDate.ToString("yyyy-MM-dd HH:mm:ss.fff ") + nowinfo);
        log.Error(0, "ProcViolation : UpdateViolationFlagPassCarInfo return false " + nowinfo);
        log.Debug(1, "ProcViolation 9 : SendAliveMessage " + nowinfo);
        Thread.Sleep(1);
    }

    //지점 단속 모드이면 지점 단속 진행
    if (NowSectionInfo.SiteCrackDownFlag >= 1)
    {
        log.Debug(0, funcname + " : 9-1 Site CrackDown Start " + nowinfo);
        StartEnforcementSite(othersiteinfo);
        Thread.Sleep(1);
    }
}
else
{
    log.Debug(1, funcname + " : ProcViolation 10 " + nowinfo);
    log.Error(0, funcname + (" : OtherSiteInfo is null " + nowinfo));
    Thread.Sleep(1);
}
```

Main work 구간 단속(4)

```
    else
    {
        if (cheksectioninfo == null)
        {
            log.Error(0, funcname + " : cheksectioninfo is null, " + nowinfo);
        }
        else
        {
            log.Error(0, funcname + " : IsViolationInfo retrun ture, " + nowinfo);
        }
    }

    Thread.Sleep(1);

    //통과차량 정보 갱신
    if (!UpdateViolationFlagPassCarInfo(cheksectioninfo, NowSectionInfo.SectionLocalNo, cheksectioninfo.LocationFlag))
    {
        Thread.Sleep(1);
        log.Error(0, "ProcViolation : " + cheksectioninfo.SiteCode + " , " + cheksectioninfo.VehicleNumber + " , "
            + cheksectioninfo.PassDate.ToString("yyyy-MM-dd HH:mm:ss.fff ") + nowinfo);
        log.Error(1, "ProcViolation : UpdateViolationFlagPassCarInfo return false " + nowinfo);
        log.Debug(1, "ProcViolation 13 : SendAliveMessage " + nowinfo);
    }

    Thread.Sleep(1);
}
return true;
```

Main work 구간 통과차량 등록(1)

```
private String InsertSectionPassInfo(PASSCARINFO startpassinfo,
             PASSCARINFO endpassinfo, TRAVELINFO traveltarget)
{
    String funcname = MethodBase.GetCurrentMethod().Name;
    String sectiondataid = String.Empty;
    try
    {
        log.Debug(0, funcname + " : Start " + startpassinfo.VehicleNumber);
        if (startpassinfo == null)
        {
            log.Error(0, funcname + (" : startpassinfo is null"));
            return String.Empty;
        }

        if (endpassinfo == null)
        {
            log.Error(0, funcname + (" : endpassinfo is null"));
            return String.Empty;
        }

        if (traveltarget == null)
        {
            log.Error(0, funcname + (" : traveltarget is null"));
            return String.Empty;
        }
```

```
        log.Debug(1, funcname + " : DIV msec " + startpassinfo.VehicleNumber);
        //by hong2da_20180109, Add, 밀리초를 초와 밀리초로 변환하는 부분 함수로 변환
        String msecvalue = "0";
        String secvalue = "0";
        DivTravelTimesecmsec(traveltarget.TravelTime.TotalMilliseconds, ref secvalue, ref msecvalue);
        //log.Debug(funcname + " : Create SectionDataID Start " + startpassinfo.VehicleNumber);
        sectiondataid = CreateSectionDataID(endpassinfo, NowSectionInfo);
        //log.Debug(funcname + " : Create SectionDataID End" + startpassinfo.VehicleNumber);
        //log.Debug(funcname + " : SectionDataID= " + sectiondataid + ", " + startpassinfo.VehicleNumber);

        String qry = "INSERT INTO @SECTABLE ([SectionDataID],[SectionCode],[StartSiteCode],[StartTimeState]"
                   + ",[StartLane],[StartDate],[Startmsec],[StartVehicleKind],[StartSpeed]"
                   + ",[EndSiteCode],[EndTimeState],[EndLane],[EndDate],[Endmsec],[EndSpeed],[EndVehicleKind]"
                   + ",[VehicleNumber],[TravelTime],[mTravelTime],[Distance],[Speed],[SpeedLimit] "
                   + ",[StartPlateX],[StartPlateY],[StartPlateW],[StartPlateH]"
                   + ",[EndPlateX],[EndPlateY],[EndPlateW],[EndPlateH], [SendFlag] "
                   + ",[VioFlag],[VioCheckDateTime]) VALUES "
                   + "(@SECTIONDATEID,@SECTIONCODE,@STARTSITECODE,@STARTTIMESTATE,@STARTLANE,@STARTDATE"
                   + ",@STARTMSEC,@STARTVEHICLEKIND,@STARTSPEED "
                   + ",@ENDSITECODE,@ENDTIMESTATE,@ENDLANE,@ENDDATE,@ENDSEC,@ENDSPEED,@ENDVEHICLEKIND,@VEHICLENUMBER "
                   + ",@TRAVELTIME,@MTRAVELTIME,@DISTANCE,@SPEED,@SPEEDLIMIT"
                   + ",@STARTPLATEX,@STARTPLATEY,@STARTPLATEW,@STARTPLATEH "
                   + ",@ENDPLATEX,@ENDPLATEY,@ENDPLATEW,@ENDPLATEH,@SENDFLAG "
                   + ",@VIOFLAG,@VIOCHECKDATETIME)";
```

Main work 구간 통과차량 등록(2)

```
String tablename = ConfigInfo.GetValue("SystemInfo", "SectionDataTable") + NowSectionInfo.SectionLocalNo;
qry = qry.Replace("@SECTABLE", tablename);
log.Debug(1, funcname + " : Create Query " + startpassinfo.VehicleNumber);
SqlCommand cmd = new SqlCommand(qry, DBLib.NowConnection);
cmd.Parameters.AddWithValue("@SECTIONDATEID", sectiondataid);
cmd.Parameters.AddWithValue("@SECTIONCODE", NowSectionInfo.SectionLocalNo);
cmd.Parameters.AddWithValue("@STARTSITECODE", startpassinfo.SiteCode);
cmd.Parameters.AddWithValue("@STARTTIMESTATE", startpassinfo.StandardTimeState);
cmd.Parameters.AddWithValue("@STARTLANE", startpassinfo.Lane);
cmd.Parameters.AddWithValue("@STARTDATE", startpassinfo.PassDate);
cmd.Parameters.AddWithValue("@STARTMSEC", startpassinfo.msec);
cmd.Parameters.AddWithValue("@STARTVEHICLEKIND", startpassinfo.VehicleKind);
cmd.Parameters.AddWithValue("@STARTSPEED", startpassinfo.Speed);

cmd.Parameters.AddWithValue("@ENDSITECODE", endpassinfo.SiteCode);
cmd.Parameters.AddWithValue("@ENDTIMESTATE", endpassinfo.StandardTimeState);
cmd.Parameters.AddWithValue("@ENDLANE", endpassinfo.Lane);
cmd.Parameters.AddWithValue("@ENDDATE", endpassinfo.PassDate);
cmd.Parameters.AddWithValue("@ENDSEC", endpassinfo.msec);
cmd.Parameters.AddWithValue("@ENDSPEED", endpassinfo.Speed);
cmd.Parameters.AddWithValue("@ENDVEHICLEKIND", endpassinfo.VehicleKind);

cmd.Parameters.AddWithValue("@VEHICLENUMBER", endpassinfo.VehicleNumber);
cmd.Parameters.AddWithValue("@TRAVELTIME", secvalue);
cmd.Parameters.AddWithValue("@MTRAVELTIME", msecvalue);
cmd.Parameters.AddWithValue("@DISTANCE", NowSectionInfo.Distance);
cmd.Parameters.AddWithValue("@SPEED", traveltarget.TravelSpeed);
```

```
// kkh : 2020-02-20 분리단속 플래그가 1이면 실행 0이면 일반 단속
if (NowSectionInfo.DivCrackDownFlag == 1)
{
    // kkh : 2020-02-20 분리단속 속도 추가
    // kkh : 대형,특장 이면 BigLimitSpeed를 SpeedLimit에 넣는다.
    if (endpassinfo.VehicleKind == "3" || endpassinfo.VehicleKind == "5")
        cmd.Parameters.AddWithValue("@SPEEDLIMIT", NowSectionInfo.BigLimitSpeed);
    // kkh : 2020-02-20 분리단속 속도 추가
    // kkh : 대형,특장이 아닌 모든차는 LimitSpeed를 넣는다.
    else cmd.Parameters.AddWithValue("@SPEEDLIMIT", NowSectionInfo.LimitSpeed);
}
else cmd.Parameters.AddWithValue("@SPEEDLIMIT", NowSectionInfo.LimitSpeed);

cmd.Parameters.AddWithValue("@STARTPLATEX", startpassinfo.PlateX);
cmd.Parameters.AddWithValue("@STARTPLATEY", startpassinfo.PlateY);
cmd.Parameters.AddWithValue("@STARTPLATEW", startpassinfo.PlateW);
cmd.Parameters.AddWithValue("@STARTPLATEH", startpassinfo.PlateH);

cmd.Parameters.AddWithValue("@ENDPLATEX", endpassinfo.PlateX);
cmd.Parameters.AddWithValue("@ENDPLATEY", endpassinfo.PlateY);
cmd.Parameters.AddWithValue("@ENDPLATEW", endpassinfo.PlateW);
cmd.Parameters.AddWithValue("@ENDPLATEH", endpassinfo.PlateH);
```

Main work 구간 통과차량 등록(3)

```
        String violationflag = "N";

        //표준시간이 없을 경우
        if (startpassinfo.StandardTimeState > 1)
        {
            violationflag = "NULL";
        }
        if (endpassinfo.StandardTimeState > 1)
        {
            violationflag = "NULL";
        }

        if (violationflag != "NULL")
        {
            cmd.Parameters.AddWithValue("@SENDFLAG", violationflag);
        }
        else
        {
            cmd.Parameters.AddWithValue("@SENDFLAG", DBNull.Value);
        }
            cmd.Parameters.AddWithValue("@VIOFLAG", "2");
            cmd.Parameters.AddWithValue("@VIOCHECKDATETIME", DateTime.Now);
            int rowcount = cmd.ExecuteNonQuery();
            if (rowcount > 0)
            {
                return sectiondataid;
            }
            else
            {
                log.Error(0, funcname + " : " + new ArgumentException("ExecuteNonQuery return RowCount<0"));
                return String.Empty;
            }
        }
        catch (System.Exception ex)
        {
            log.Error(0, funcname + " : " + ex.ToString());
            return String.Empty;
        }
    }
```

Main work 구간 통과차량 등록(4)

```
        String violationflag = "N";

        //표준시간이 없을 경우
        if (startpassinfo.StandardTimeState > 1)
        {
            violationflag = "NULL";
        }
        if (endpassinfo.StandardTimeState > 1)
        {
            violationflag = "NULL";
        }

        if (violationflag != "NULL")
        {
            cmd.Parameters.AddWithValue("@SENDFLAG", violationflag);
        }
        else
        {
            cmd.Parameters.AddWithValue("@SENDFLAG", DBNull.Value);
        }
            cmd.Parameters.AddWithValue("@VIOFLAG", "2");
            cmd.Parameters.AddWithValue("@VIOCHECKDATETIME", DateTime.Now);
            int rowcount = cmd.ExecuteNonQuery();
            if (rowcount > 0)
            {
                return sectiondataid;
            }
            else
            {
                log.Error(0, funcname + " : " + new ArgumentException("ExecuteNonQuery return RowCount<0"));
                return String.Empty;
            }
        }
        catch (System.Exception ex)
        {
            log.Error(0, funcname + " : " + ex.ToString());
            return String.Empty;
        }
    }
```

End of Document