EBOOK

Michael Andre Franiatte

C# Codes for Using Both Joycon and Wiimote in PC Games

WiiJoy4FPS.exe

EBOOK

C# Codes for Using Both Joycons and Wiimote in PC Games

WiiJoy4FPS.exe

Michael Franiatte
01/01/2020


The C# codes presented can simulate keyboard and mouse events to play very well PC games using both Joycon and Wiimote as a simple program and script. Information about license, EULA and contract for using these following works can be found at https://michaelfraniatte.wordpress.com.

C# Codes for Using Both Joycons and Wiimote in PC Games

Michael Franiatte*

Abstract

With these C# codes, Joycon and Wiimote on PC is the best solution to play games allowing to replace keyboard and mouse, with the same accuracy and more easy to use for a best comfortable experience of gameplay. The codes presented here allow simulating keyboard and mouse events in order to play PC games using Joycons. This paper gives 10 years of works on coding Joycons and Wiimote and coding keyboard and mouse events to have the best controls never reached by other works on it. This is the perfect solution to play PC games in a beauty manner with all codes to play in all different manner adapted to all games. Joycon and Wiimote both is very competitive with these codes which allow a perfect control without any flaw or lag for all game genres and settings. Some complementary explanations are available in other books of the same author.

Keywords: *gamepads, PC, gameplay, games, codes, Joycons*

* Author correspondence: michael.franiatte@gmail.com

1. *Win32 C++ DLL WiiJoyPairing Codes*

```
#include "stdafx.h"
#include <windows.h>
#include <bthsdpdef.h>
#include <bthdef.h>
#include <BluetoothAPIs.h>
#include <strsafe.h>
#include <iostream>
#pragma comment(lib, "Bthprops.lib")
BLUETOOTH_DEVICE_INFO btdir;
BLUETOOTH_DEVICE_INFO btdil;
BLUETOOTH_DEVICE_INFO btdiw;
bool joyconrfound = false;
bool joyconlfound = false;
bool wiimotefound = false;
#pragma warning(disable : 4995)
extern "C"
{
	__declspec(dllexport) int connect()
	{
		HBLUETOOTH_DEVICE_FIND hFind = NULL;
		HANDLE hRadios[256];
		HBLUETOOTH_RADIO_FIND hFindRadio;
		BLUETOOTH_FIND_RADIO_PARAMS radioParam;
		BLUETOOTH_RADIO_INFO radioInfo;
		BLUETOOTH_DEVICE_SEARCH_PARAMS srch;
		BLUETOOTH_DEVICE_INFO btdi;
		int nRadios = 0;
		radioParam.dwSize = sizeof(BLUETOOTH_FIND_RADIO_PARAMS);
		radioInfo.dwSize = sizeof(BLUETOOTH_RADIO_INFO);
		btdir.dwSize = sizeof(btdir);
		btdil.dwSize = sizeof(btdil);
		btdiw.dwSize = sizeof(btdiw);
		btdi.dwSize = sizeof(btdi);
		srch.dwSize = sizeof(BLUETOOTH_DEVICE_SEARCH_PARAMS);
		hFindRadio = BluetoothFindFirstRadio(&radioParam, &hRadios[nRadios++]);
		while (BluetoothFindNextRadio(hFindRadio, &hRadios[nRadios++]))
		{
			hFindRadio = BluetoothFindFirstRadio(&radioParam,
&hRadios[nRadios++]);
			BluetoothFindRadioClose(hFindRadio);
		}
		srch.fReturnAuthenticated = TRUE;
		srch.fReturnRemembered = TRUE;
		srch.fReturnConnected = TRUE;
		srch.fReturnUnknown = TRUE;
		srch.fIssueInquiry = TRUE;
		srch.cTimeoutMultiplier = 2;
		srch.hRadio = hRadios[1];
		BluetoothGetRadioInfo(hRadios[1], &radioInfo);
		WCHAR pass[6];
		DWORD pcServices = 16;
		GUID guids[16];
		pass[0] = radioInfo.address.rgBytes[0];
		pass[1] = radioInfo.address.rgBytes[1];
		pass[2] = radioInfo.address.rgBytes[2];
		pass[3] = radioInfo.address.rgBytes[3];
		pass[4] = radioInfo.address.rgBytes[4];
		pass[5] = radioInfo.address.rgBytes[5];
		for (int i = 0; i < 3; i++)
		{
			hFind = BluetoothFindFirstDevice(&srch, &btdi);
			if (hFind > 0)
			{
				do
```

```
                        {
                                if (!wcscmp(btdi.szName, L"Nintendo RVL-WBC-01") |
!wcscmp(btdi.szName, L"Nintendo RVL-CNT-01"))
                                {
                                        BluetoothAuthenticateDevice(NULL, hRadios[1],
&btdi, pass, 6);
                                        BluetoothEnumerateInstalledServices(hRadios[1],
&btdi, &pcServices, guids);
                                        BluetoothSetServiceState(hRadios[1], &btdi,
&HumanInterfaceDeviceServiceClass_UUID, BLUETOOTH_SERVICE_ENABLE);
                                        BluetoothUpdateDeviceRecord(&btdi);
                                        btdiw = btdi;
                                        wiimotefound = true;
                                }
                                if (!joyconlfound)
                                        if (!wcscmp(btdi.szName, L"Joy-Con (L)"))
                                        {
                                                BluetoothAuthenticateDevice(NULL,
hRadios[1], &btdi, pass, 6);

        BluetoothEnumerateInstalledServices(hRadios[1], &btdi, &pcServices, guids);
                                                BluetoothSetServiceState(hRadios[1],
&btdi, &HumanInterfaceDeviceServiceClass_UUID, BLUETOOTH_SERVICE_ENABLE);
                                                BluetoothUpdateDeviceRecord(&btdi);
                                                btdil = btdi;
                                                joyconlfound = true;
                                        }
                                if (!joyconrfound)
                                        if (!wcscmp(btdi.szName, L"Joy-Con (R)"))
                                        {
                                                BluetoothAuthenticateDevice(NULL,
hRadios[1], &btdi, pass, 6);

        BluetoothEnumerateInstalledServices(hRadios[1], &btdi, &pcServices, guids);
                                                BluetoothSetServiceState(hRadios[1],
&btdi, &HumanInterfaceDeviceServiceClass_UUID, BLUETOOTH_SERVICE_ENABLE);
                                                BluetoothUpdateDeviceRecord(&btdi);
                                                btdir = btdi;
                                                joyconrfound = true;
                                        }
                        } while (BluetoothFindNextDevice(hFind, &btdi));
                        BluetoothFindDeviceClose(hFind);
                }
        }
        BluetoothFindRadioClose(hFindRadio);
        if (!wiimotefound && !joyconrfound && joyconlfound)
                return 1;
        if (!joyconrfound && wiimotefound && joyconlfound)
                return 2;
        if (!joyconrfound && !joyconlfound && wiimotefound)
                return 3;
        if (!wiimotefound && joyconrfound && joyconlfound)
                return 4;
        if (!joyconlfound && wiimotefound && joyconrfound)
                return 5;
        if (!wiimotefound && !joyconlfound && joyconrfound)
                return 6;
        if (wiimotefound && joyconlfound && joyconrfound)
                return 7;
        return 0;
}
__declspec(dllexport) bool disconnect()
{
        if (wiimotefound)
                BluetoothRemoveDevice(&btdiw.Address);
        if (joyconlfound)
```

```
                BluetoothRemoveDevice(&btdil.Address);
            if (joyconrfound)
                BluetoothRemoveDevice(&btdir.Address);
            return true;
        }
}
```

2. *C# Windows Form Codes for FPS (WiiJoy4)*

```
using Microsoft.Win32.SafeHandles;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Diagnostics;
using System.Numerics;
using AForge.Video;
using AForge.Video.DirectShow;
using AForge.Imaging.Filters;
using System.Drawing.Imaging;
using AForge;
using AForge.Imaging;
using System.Drawing.Drawing2D;
using System.Text.RegularExpressions;
using Microsoft.CSharp;
using System.CodeDom;
using System.Reflection;
namespace WiiJoy4
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
    [DllImport("system32/user32.dll")]
    public static extern bool SwitchToThisWindow(IntPtr handle, bool fAltTab);
    [DllImport("hid.dll")]
    public static unsafe extern void HidD_GetHidGuid(out Guid gHid);
    [DllImport("hid.dll")]
    public extern unsafe static bool HidD_SetOutputReport(IntPtr HidDeviceObject, byte[] lpReportBuffer, uint
ReportBufferLength);
    [DllImport("setupapi.dll")]
    public static unsafe extern IntPtr SetupDiGetClassDevs(ref Guid ClassGuid, string Enumerator, IntPtr hwndParent,
UInt32 Flags);
    [DllImport("setupapi.dll")]
    public static unsafe extern Boolean SetupDiEnumDeviceInterfaces(IntPtr hDevInfo, IntPtr devInvo, ref Guid
interfaceClassGuid, Int32 memberIndex, ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData);
    [DllImport("setupapi.dll")]
    public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr hDevInfo, ref SP_DEVICE_INTERFACE_DATA
deviceInterfaceData, IntPtr deviceInterfaceDetailData, UInt32 deviceInterfaceDetailDataSize, out UInt32 requiredSize, IntPtr
deviceInfoData);
    [DllImport("setupapi.dll")]
    public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr hDevInfo, ref SP_DEVICE_INTERFACE_DATA
deviceInterfaceData, ref SP_DEVICE_INTERFACE_DETAIL_DATA deviceInterfaceDetailData, UInt32
```

```
deviceInterfaceDetailDataSize, out UInt32 requiredSize, IntPtr deviceInfoData);
    [DllImport("Kernel32.dll")]
    public static unsafe extern SafeFileHandle CreateFile(string fileName, [MarshalAs(UnmanagedType.U4)] FileAccess
fileAccess, [MarshalAs(UnmanagedType.U4)] FileShare fileShare, IntPtr securityAttributes,
[MarshalAs(UnmanagedType.U4)] FileMode creationDisposition, [MarshalAs(UnmanagedType.U4)] uint flags, IntPtr
template);
    [DllImport("winmm.dll", EntryPoint = "timeBeginPeriod")]
    public static extern uint TimeBeginPeriod(uint ms);
    [DllImport("winmm.dll", EntryPoint = "timeEndPeriod")]
    public static extern uint TimeEndPeriod(uint ms);
    [DllImport("ntdll.dll", EntryPoint = "NtSetTimerResolution")]
    public static extern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint CurrentResolution);
    [DllImport("user32.dll")]
    public static extern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
    private int leftandright;
    public static double REGISTER_IR = 0x04b00030, REGISTER_EXTENSION_INIT_1 = 0x04a400f0,
REGISTER_EXTENSION_INIT_2 = 0x04a400fb, REGISTER_EXTENSION_TYPE = 0x04a400fa,
REGISTER_EXTENSION_CALIBRATION = 0x04a40020, REGISTER_MOTIONPLUS_INIT = 0x04a600fe, camx, camy, irx2e, iry2e,
irx3e, iry3e, irx, iry, irxc, iryc, mWSIRSensorsXcam, mWSIRSensorsYcam, mWSIRSensors0X, mWSIRSensors0Y,
mWSIRSensors1X, mWSIRSensors1Y, mWSButtonStateIRX, mWSButtonStateIRY, mWSIRSensors0Xcam,
mWSIRSensors1Xcam, mWSIRSensors0Ycam, mWSIRSensors1Ycam, MyAngle, mWSIR0notfound = 0, mWSRawValuesX,
mWSRawValuesY, mWSRawValuesZ, calibrationinit, watchM = 50, watchM1 = 2, watchM2 = 0, watchK = 50, watchK1 = 2,
watchK2 = 0, backpointX, BackpointAnglesX, InitBackpointAnglesX, posRightX, backpointY, BackpointAnglesY,
InitBackpointAnglesY, posRightY, stickviewxinit, stickviewyinit, mWSNunchuckStateRawValuesX,
mWSNunchuckStateRawValuesY, mWSNunchuckStateRawValuesZ, mWSNunchuckStateRawJoystickX,
mWSNunchuckStateRawJoystickY;
    public static bool mWSIR1foundcam, mWSIR0foundcam, mWSIR1found, mWSIR0found, mWSIRswitch,
mWSButtonStateA, mWSButtonStateB, mWSButtonStateMinus, mWSButtonStateHome, mWSButtonStatePlus,
mWSButtonStateOne, mWSButtonStateTwo, mWSButtonStateUp, mWSButtonStateDown, mWSButtonStateLeft,
mWSButtonStateRight, runningoff, Getstate, Setrecenter, notpressing1and2, camenabled, mWSNunchuckStateC,
mWSNunchuckStateZ;
    public static string processname;
    public static byte[] buff = new byte[] { 0x55 }, mBuff = new byte[22], aBuffer = new byte[22], bBuffer = new byte[22];
    public static byte Type = 0x12, IR = 0x13, WriteMemory = 0x16, ReadMemory = 0x16, IRExtensionAccel = 0x37;
    public static Guid guid = new System.Guid();
    public static uint hDevInfo, CurrentResolution = 0;
    public FilterInfoCollection CaptureDevice;
    public VideoCaptureDevice FinalFrame;
    public static Bitmap img, EditableImg;
    public static Form1 form = (Form1)Application.OpenForms["Form1"];
    private static BlobCounter blobCounter = new BlobCounter();
    public static BlobsFiltering blobfilter = new BlobsFiltering();
    public static ConnectedComponentsLabeling componentfilter = new ConnectedComponentsLabeling();
    public static Blob[] blobs;
    public static List<IntPoint> corners = new List<IntPoint>();
    public static AForge.Math.Geometry.SimpleShapeChecker shapeChecker = new
AForge.Math.Geometry.SimpleShapeChecker();
    public static BrightnessCorrection brightnessfilter = new BrightnessCorrection(-50);
    public static ColorFiltering colorfilter = new ColorFiltering();
    public static Grayscale grayscalefilter = new Grayscale(1, 0, 0);
    public static EuclideanColorFiltering eudideanfilter = new EuclideanColorFiltering();
    public VideoCapabilities[] videoCapabilities;
    public static BackgroundWorker backgroundWorkerS = new BackgroundWorker();
    public static ThreadStart threadstart;
    public static Thread thread;
    private static bool ISLEFT, ISRIGHT;
    private static Task taskDReadWiimote, taskM, taskK;
    private static bool endinvoke;
    private static Stopwatch diffM = new Stopwatch(), diffK = new Stopwatch();
    private Type mouseprogram, keyboardprogram;
    private object mouseobj, keyboardobj, objdataReadLeft, objdataReadRight, objdataReadLeftRight,
objdataReadWiimote, objdata;
    private Assembly assembly;
```

```
private System.CodeDom.Compiler.CompilerResults results;
private Microsoft.CSharp.CSharpCodeProvider provider;
private System.CodeDom.Compiler.CompilerParameters parameters;
private string addedcode, finalcode, code, mousecode, keyboardcode;

public bool this[int i]
{
   get { return _valuechanged[i]; }
   set
   {
      if (_valuechanged[i] != value)
         _Valuechanged[i] = true;
      else
         _Valuechanged[i] = false;
      _valuechanged[i] = value;
   }
}
public static bool[] _Valuechanged = new bool[2], _valuechanged = new bool[2];
private void initConfig()
{
   System.IO.StreamReader file = new System.IO.StreamReader(processname + ".txt");
   string linesofcode = "";
   string emptylines = "";
   int i = 0;
   file.ReadLine();
   file.ReadLine();
   do
   {
      emptylines = file.ReadLine();
      if (emptylines != "")
      {
         linesofcode += emptylines + " ";
         i++;
      }
      else
         break;
   }
   while (emptylines != "" | i == 0);
   file.Close();
   mousecode = linesofcode;
   file = new System.IO.StreamReader(processname + ".txt");
   linesofcode = "";
   emptylines = "";
   i = 0;
   string imgpath = file.ReadLine().Replace(".txt", ".png");
   file.ReadLine();
   do
   {
      emptylines = file.ReadLine();
      if (emptylines != "")
      {
         linesofcode += emptylines + " ";
         i++;
      }
      else
         break;
   }
   while (emptylines != "" | i == 0);
   linesofcode = "";
   emptylines = "";
   i = 0;
   file.ReadLine();
   do
```

```
    {
        emptylines = file.ReadLine();
        if (emptylines != "")
        {
            linesofcode += emptylines + " ";
            i++;
        }
        else
            break;
    }
    while (emptylines != "" | i == 0);
    file.Close();
    keyboardcode =linesofcode;
    try
    {
        System.Drawing.Image img = System.Drawing.Image.FromFile(imgpath);
        this.Size = new System.Drawing.Size(img.Width, img.Height);
        System.Drawing.Image myimage = new Bitmap(imgpath);
        this.BackgroundImage = myimage;
    }
    catch { }
}
private void Form1_Shown(object sender, EventArgs e)
{
    TimeBeginPeriod(1);
    NtSetTimerResolution(1, true, ref CurrentResolution);
    System.Diagnostics.Process process = Process.GetCurrentProcess();
    process.PriorityClass = System.Diagnostics.ProcessPriorityClass.RealTime;
    processname = Process.GetCurrentProcess().ProcessName;
    this.Text = processname;
    try
    {
        System.Diagnostics.Process processattached = System.Diagnostics.Process.Start("WiiJoy4FPS.exe");
    }
    catch { }
    backgroundWorkerS.DoWork += new DoWorkEventHandler(FormStart);
    backgroundWorkerS.RunWorkerAsync();
}
private void scriptUpdate()
{
    code = @"
        using System;
        using System.Runtime.InteropServices;
        namespace StringToCode
        {
            public class FooClass
            {
                public static uint CurrentResolution = 0;
                Input input = new Input();
                funct_driver
                public bool this[int i]
                {
                    get { return _valuechanged[i]; }
                    set
                    {
                        if (_valuechanged[i] != value)
                            _Valuechanged[i] = true;
                        else
                            _Valuechanged[i] = false;
                        _valuechanged[i] = value;
                    }
                }
                public void Open(double mWSNunchuckStateRawJoystickX, double mWSNunchuckStateRawJoystickY, double
```

```
mWSNunchuckStateRawValuesX, double mWSNunchuckStateRawValuesY, double mWSNunchuckStateRawValuesZ, bool
mWSNunchuckStateC, bool mWSNunchuckStateZ,
            double mWSButtonStateIRX, double mWSButtonStateIRY, bool mWSButtonStateA, bool mWSButtonStateB,
bool mWSButtonStateMinus, bool mWSButtonStateHome, bool mWSButtonStatePlus, bool mWSButtonStateOne, bool
mWSButtonStateTwo, bool mWSButtonStateUp, bool mWSButtonStateDown, bool mWSButtonStateLeft, bool
mWSButtonStateRight, double mWSRawValuesX, double mWSRawValuesY, double mWSRawValuesZ,
            float EulerAnglesX, float EulerAnglesY, float EulerAnglesZ, float DirectAnglesX, float DirectAnglesY, float
DirectAnglesZ, double camx, double camy, float EulerAnglesLeftX, float EulerAnglesLeftY, float EulerAnglesLeftZ, float
DirectAnglesLeftX, float DirectAnglesLeftY, float DirectAnglesLeftZ, float EulerAnglesRightX, float EulerAnglesRightY, float
EulerAnglesRightZ,
            float DirectAnglesRightX, float DirectAnglesRightY, float DirectAnglesRightZ, bool LeftButtonSHOULDER_1,
bool LeftButtonMINUS, bool LeftButtonCAPTURE, bool LeftButtonDPAD_UP, bool LeftButtonDPAD_LEFT, bool
LeftButtonDPAD_DOWN, bool LeftButtonDPAD_RIGHT, bool LeftButtonSTICK, bool RightButtonDPAD_DOWN, bool
LeftButtonSL, bool LeftButtonSR, double GetStickLeftX, double GetStickLeftY,
            bool RightButtonPLUS, bool RightButtonDPAD_RIGHT, bool RightButtonHOME, bool RightButtonSHOULDER_1,
bool RightButtonDPAD_LEFT, bool RightButtonDPAD_UP, bool RightButtonSTICK, bool RightButtonSL, bool RightButtonSR,
bool RightButtonSHOULDER_2, bool LeftButtonSHOULDER_2, double GetStickRightX, double GetStickRightY, float
GetAccelX, float GetAccelY, float GetAccelZ, float GetAccelRightX, float GetAccelRightY, float GetAccelRightZ, float
GetAccelLeftX, float GetAccelLeftY, float GetAccelLeftZ, double watchM)
            {
              TimeBeginPeriod(1);
              NtSetTimerResolution(1, true, ref CurrentResolution);
              input.KeyboardFilterMode = KeyboardFilterMode.All;
              input.MouseFilterMode = MouseFilterMode.All;
              input.Load();
            }
            public void Close(double mWSNunchuckStateRawJoystickX, double mWSNunchuckStateRawJoystickY, double
mWSNunchuckStateRawValuesX, double mWSNunchuckStateRawValuesY, double mWSNunchuckStateRawValuesZ, bool
mWSNunchuckStateC, bool mWSNunchuckStateZ,
            double mWSButtonStateIRX, double mWSButtonStateIRY, bool mWSButtonStateA, bool mWSButtonStateB,
bool mWSButtonStateMinus, bool mWSButtonStateHome, bool mWSButtonStatePlus, bool mWSButtonStateOne, bool
mWSButtonStateTwo, bool mWSButtonStateUp, bool mWSButtonStateDown, bool mWSButtonStateLeft, bool
mWSButtonStateRight, double mWSRawValuesX, double mWSRawValuesY, double mWSRawValuesZ,
            float EulerAnglesX, float EulerAnglesY, float EulerAnglesZ, float DirectAnglesX, float DirectAnglesY, float
DirectAnglesZ, double camx, double camy, float EulerAnglesLeftX, float EulerAnglesLeftY, float EulerAnglesLeftZ, float
DirectAnglesLeftX, float DirectAnglesLeftY, float DirectAnglesLeftZ, float EulerAnglesRightX, float EulerAnglesRightY, float
EulerAnglesRightZ,
            float DirectAnglesRightX, float DirectAnglesRightY, float DirectAnglesRightZ, bool LeftButtonSHOULDER_1,
bool LeftButtonMINUS, bool LeftButtonCAPTURE, bool LeftButtonDPAD_UP, bool LeftButtonDPAD_LEFT, bool
LeftButtonDPAD_DOWN, bool LeftButtonDPAD_RIGHT, bool LeftButtonSTICK, bool RightButtonDPAD_DOWN, bool
LeftButtonSL, bool LeftButtonSR, double GetStickLeftX, double GetStickLeftY,
            bool RightButtonPLUS, bool RightButtonDPAD_RIGHT, bool RightButtonHOME, bool RightButtonSHOULDER_1,
bool RightButtonDPAD_LEFT, bool RightButtonDPAD_UP, bool RightButtonSTICK, bool RightButtonSL, bool RightButtonSR,
bool RightButtonSHOULDER_2, bool LeftButtonSHOULDER_2, double GetStickRightX, double GetStickRightY, float
GetAccelX, float GetAccelY, float GetAccelZ, float GetAccelRightX, float GetAccelRightY, float GetAccelRightZ, float
GetAccelLeftX, float GetAccelLeftY, float GetAccelLeftZ, double watchM)
            {
              TimeEndPeriod(1);
              input.Unload();
            }
            [DllImport(""user32.dll"")]
            publicstaticextern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
            [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseTo"", CallingConvention =
CallingConvention.Cdecl)]
            publicstaticextern void MoveMouseTo(int x, int y);
            [DllImport(""InputSending.dll"", EntryPoint = ""MoveMouseBy"", CallingConvention =
CallingConvention.Cdecl)]
            publicstaticextern void MoveMouseBy(int x, int y);
            [DllImport(""InputSending.dll"", EntryPoint = ""SendKey"", CallingConvention = CallingConvention.Cdecl)]
            publicstaticextern void SendKey(UInt16 bVk, UInt16 bScan);
            [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyF"", CallingConvention = CallingConvention.Cdecl)]
            publicstaticextern void SendKeyF(UInt16 bVk, UInt16 bScan);
            [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrows"", CallingConvention =
```

```
CallingConvention.Cdecl)]
        publicstaticextern void SendKeyArrows(UInt16 bVk, UInt16 bScan);
        [DllImport(""InputSending.dll"", EntryPoint = ""SendKeyArrowsF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendKeyArrowsF(UInt16 bVk, UInt16 bScan);
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonLeft"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonLeft();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonLeftF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonLeftF();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonRight"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonRight();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonRightF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonRightF();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonMiddle"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonMiddle();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonMiddleF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonMiddleF();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonWheelUp"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonWheelUp();
        [DllImport(""InputSending.dll"", EntryPoint = ""SendMouseEventButtonWheelDown"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SendMouseEventButtonWheelDown();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyDown"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SimulateKeyDown(UInt16 keyCode, UInt16 bScan);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyUp"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SimulateKeyUp(UInt16 keyCode, UInt16 bScan);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyDownArrows"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SimulateKeyDownArrows(UInt16 keyCode, UInt16 bScan);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""SimulateKeyUpArrows"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void SimulateKeyUpArrows(UInt16 keyCode, UInt16 bScan);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseMW3"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void MouseMW3(int x, int y);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MouseBrink"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void MouseBrink(int x,int y);
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClick"", CallingConvention = CallingConvention.Cdecl)]
        publicstaticextern void LeftClick();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""LeftClickF"", CallingConvention = CallingConvention.Cdecl)]
        publicstaticextern void LeftClickF();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClick"", CallingConvention = CallingConvention.Cdecl)]
        publicstaticextern void RightClick();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""RightClickF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void RightClickF();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClick"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void MiddleClick();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""MiddleClickF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void MiddleClickF();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelDownF"", CallingConvention =
```

```
CallingConvention.Cdecl)]
        publicstaticextern void WheelDownF();
        [DllImport(""SendInputLibrary.dll"", EntryPoint = ""WheelUpF"", CallingConvention =
CallingConvention.Cdecl)]
        publicstaticextern void WheelUpF();
        [DllImport(""user32.dll"")]
        publicstaticextern void SetPhysicalCursorPos(int X, int Y);
        [DllImport(""user32.dll"")]
        publicstaticextern void SetCaretPos(int X, int Y);
        [DllImport(""user32.dll"")]
        publicstaticextern void SetCursorPos(int X, int Y);
        [DllImport(""winmm.dll"", EntryPoint = ""timeBeginPeriod"")]
        publicstaticextern uint TimeBeginPeriod(uint ms);
        [DllImport(""winmm.dll"", EntryPoint = ""timeEndPeriod"")]
        publicstaticextern uint TimeEndPeriod(uint ms);
        [DllImport(""ntdll.dll"", EntryPoint = ""NtSetTimerResolution"")]
        publicstaticextern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint
CurrentResolution);
        [DllImport(""system32/user32.dll"")]
        publicstaticextern uint MapVirtualKey(uint uCode, uint uMapType);
        publicstatic ushort VK_LBUTTON = (ushort)0x01;
        publicstatic ushort VK_RBUTTON = (ushort)0x02;
        publicstatic ushort VK_CANCEL = (ushort)0x03;
        publicstatic ushort VK_MBUTTON = (ushort)0x04;
        publicstatic ushort VK_XBUTTON1 = (ushort)0x05;
        publicstatic ushort VK_XBUTTON2 = (ushort)0x06;
        publicstatic ushort VK_BACK = (ushort)0x08;
        publicstatic ushort VK_Tab = (ushort)0x09;
        publicstatic ushort VK_CLEAR = (ushort)0x0C;
        publicstatic ushort VK_Return = (ushort)0x0D;
        publicstatic ushort VK_SHIFT = (ushort)0x10;
        publicstatic ushort VK_CONTROL = (ushort)0x11;
        publicstatic ushort VK_MENU = (ushort)0x12;
        publicstatic ushort VK_PAUSE = (ushort)0x13;
        publicstatic ushort VK_CAPITAL = (ushort)0x14;
        publicstatic ushort VK_KANA = (ushort)0x15;
        publicstatic ushort VK_HANGEUL = (ushort)0x15;
        publicstatic ushort VK_HANGUL = (ushort)0x15;
        publicstatic ushort VK_JUNJA = (ushort)0x17;
        publicstatic ushort VK_FINAL = (ushort)0x18;
        publicstatic ushort VK_HANJA = (ushort)0x19;
        publicstatic ushort VK_KANJI = (ushort)0x19;
        publicstatic ushort VK_Escape = (ushort)0x1B;
        publicstatic ushort VK_CONVERT = (ushort)0x1C;
        publicstatic ushort VK_NONCONVERT = (ushort)0x1D;
        publicstatic ushort VK_ACCEPT = (ushort)0x1E;
        publicstatic ushort VK_MODECHANGE = (ushort)0x1F;
        publicstatic ushort VK_Space = (ushort)0x20;
        publicstatic ushort VK_PRIOR = (ushort)0x21;
        publicstatic ushort VK_NEXT = (ushort)0x22;
        publicstatic ushort VK_END = (ushort)0x23;
        publicstatic ushort VK_HOME = (ushort)0x24;
        publicstatic ushort VK_LEFT = (ushort)0x25;
        publicstatic ushort VK_UP = (ushort)0x26;
        publicstatic ushort VK_RIGHT = (ushort)0x27;
        publicstatic ushort VK_DOWN = (ushort)0x28;
        publicstatic ushort VK_SELECT = (ushort)0x29;
        publicstatic ushort VK_PRINT = (ushort)0x2A;
        publicstatic ushort VK_EXECUTE = (ushort)0x2B;
        publicstatic ushort VK_SNAPSHOT = (ushort)0x2C;
        publicstatic ushort VK_INSERT = (ushort)0x2D;
        publicstatic ushort VK_DELETE = (ushort)0x2E;
        publicstatic ushort VK_HELP = (ushort)0x2F;
```

```
publicstatic ushort VK_APOSTROPHE = (ushort)0xDE;
publicstatic ushort VK_0 = (ushort)0x30;
publicstatic ushort VK_1 = (ushort)0x31;
publicstatic ushort VK_2 = (ushort)0x32;
publicstatic ushort VK_3 = (ushort)0x33;
publicstatic ushort VK_4 = (ushort)0x34;
publicstatic ushort VK_5 = (ushort)0x35;
publicstatic ushort VK_6 = (ushort)0x36;
publicstatic ushort VK_7 = (ushort)0x37;
publicstatic ushort VK_8 = (ushort)0x38;
publicstatic ushort VK_9 = (ushort)0x39;
publicstatic ushort VK_A = (ushort)0x41;
publicstatic ushort VK_B = (ushort)0x42;
publicstatic ushort VK_C = (ushort)0x43;
publicstatic ushort VK_D = (ushort)0x44;
publicstatic ushort VK_E = (ushort)0x45;
publicstatic ushort VK_F = (ushort)0x46;
publicstatic ushort VK_G = (ushort)0x47;
publicstatic ushort VK_H = (ushort)0x48;
publicstatic ushort VK_I = (ushort)0x49;
publicstatic ushort VK_J = (ushort)0x4A;
publicstatic ushort VK_K = (ushort)0x4B;
publicstatic ushort VK_L = (ushort)0x4C;
publicstatic ushort VK_M = (ushort)0x4D;
publicstatic ushort VK_N = (ushort)0x4E;
publicstatic ushort VK_O = (ushort)0x4F;
publicstatic ushort VK_P = (ushort)0x50;
publicstatic ushort VK_Q = (ushort)0x51;
publicstatic ushort VK_R = (ushort)0x52;
publicstatic ushort VK_S = (ushort)0x53;
publicstatic ushort VK_T = (ushort)0x54;
publicstatic ushort VK_U = (ushort)0x55;
publicstatic ushort VK_V = (ushort)0x56;
publicstatic ushort VK_W = (ushort)0x57;
publicstatic ushort VK_X = (ushort)0x58;
publicstatic ushort VK_Y = (ushort)0x59;
publicstatic ushort VK_Z = (ushort)0x5A;
publicstatic ushort VK_LWIN = (ushort)0x5B;
publicstatic ushort VK_RWIN = (ushort)0x5C;
publicstatic ushort VK_APPS = (ushort)0x5D;
publicstatic ushort VK_SLEEP = (ushort)0x5F;
publicstatic ushort VK_NUMPAD0 = (ushort)0x60;
publicstatic ushort VK_NUMPAD1 = (ushort)0x61;
publicstatic ushort VK_NUMPAD2 = (ushort)0x62;
publicstatic ushort VK_NUMPAD3 = (ushort)0x63;
publicstatic ushort VK_NUMPAD4 = (ushort)0x64;
publicstatic ushort VK_NUMPAD5 = (ushort)0x65;
publicstatic ushort VK_NUMPAD6 = (ushort)0x66;
publicstatic ushort VK_NUMPAD7 = (ushort)0x67;
publicstatic ushort VK_NUMPAD8 = (ushort)0x68;
publicstatic ushort VK_NUMPAD9 = (ushort)0x69;
publicstatic ushort VK_MULTIPLY = (ushort)0x6A;
publicstatic ushort VK_ADD = (ushort)0x6B;
publicstatic ushort VK_SEPARATOR = (ushort)0x6C;
publicstatic ushort VK_SUBTRACT = (ushort)0x6D;
publicstatic ushort VK_DECIMAL = (ushort)0x6E;
publicstatic ushort VK_DIVIDE = (ushort)0x6F;
publicstatic ushort VK_F1 = (ushort)0x70;
publicstatic ushort VK_F2 = (ushort)0x71;
publicstatic ushort VK_F3 = (ushort)0x72;
publicstatic ushort VK_F4 = (ushort)0x73;
publicstatic ushort VK_F5 = (ushort)0x74;
publicstatic ushort VK_F6 = (ushort)0x75;
```

```
publicstatic ushort VK_F7 = (ushort)0x76;
publicstatic ushort VK_F8 = (ushort)0x77;
publicstatic ushort VK_F9 = (ushort)0x78;
publicstatic ushort VK_F10 = (ushort)0x79;
publicstatic ushort VK_F11 = (ushort)0x7A;
publicstatic ushort VK_F12 = (ushort)0x7B;
publicstatic ushort VK_F13 = (ushort)0x7C;
publicstatic ushort VK_F14 = (ushort)0x7D;
publicstatic ushort VK_F15 = (ushort)0x7E;
publicstatic ushort VK_F16 = (ushort)0x7F;
publicstatic ushort VK_F17 = (ushort)0x80;
publicstatic ushort VK_F18 = (ushort)0x81;
publicstatic ushort VK_F19 = (ushort)0x82;
publicstatic ushort VK_F20 = (ushort)0x83;
publicstatic ushort VK_F21 = (ushort)0x84;
publicstatic ushort VK_F22 = (ushort)0x85;
publicstatic ushort VK_F23 = (ushort)0x86;
publicstatic ushort VK_F24 = (ushort)0x87;
publicstatic ushort VK_NUMLOCK = (ushort)0x90;
publicstatic ushort VK_SCROLL = (ushort)0x91;
publicstatic ushort VK_LeftShift = (ushort)0xA0;
publicstatic ushort VK_RightShift = (ushort)0xA1;
publicstatic ushort VK_LeftControl = (ushort)0xA2;
publicstatic ushort VK_RightControl = (ushort)0xA3;
publicstatic ushort VK_LMENU = (ushort)0xA4;
publicstatic ushort VK_RMENU = (ushort)0xA5;
publicstatic ushort VK_BROWSER_BACK = (ushort)0xA6;
publicstatic ushort VK_BROWSER_FORWARD = (ushort)0xA7;
publicstatic ushort VK_BROWSER_REFRESH = (ushort)0xA8;
publicstatic ushort VK_BROWSER_STOP = (ushort)0xA9;
publicstatic ushort VK_BROWSER_SEARCH = (ushort)0xAA;
publicstatic ushort VK_BROWSER_FAVORITES = (ushort)0xAB;
publicstatic ushort VK_BROWSER_HOME = (ushort)0xAC;
publicstatic ushort VK_VOLUME_MUTE = (ushort)0xAD;
publicstatic ushort VK_VOLUME_DOWN = (ushort)0xAE;
publicstatic ushort VK_VOLUME_UP = (ushort)0xAF;
publicstatic ushort VK_MEDIA_NEXT_TRACK = (ushort)0xB0;
publicstatic ushort VK_MEDIA_PREV_TRACK = (ushort)0xB1;
publicstatic ushort VK_MEDIA_STOP = (ushort)0xB2;
publicstatic ushort VK_MEDIA_PLAY_PAUSE = (ushort)0xB3;
publicstatic ushort VK_LAUNCH_MAIL = (ushort)0xB4;
publicstatic ushort VK_LAUNCH_MEDIA_SELECT = (ushort)0xB5;
publicstatic ushort VK_LAUNCH_APP1 = (ushort)0xB6;
publicstatic ushort VK_LAUNCH_APP2 = (ushort)0xB7;
publicstatic ushort VK_OEM_1 = (ushort)0xBA;
publicstatic ushort VK_OEM_PLUS = (ushort)0xBB;
publicstatic ushort VK_OEM_COMMA = (ushort)0xBC;
publicstatic ushort VK_OEM_MINUS = (ushort)0xBD;
publicstatic ushort VK_OEM_PERIOD = (ushort)0xBE;
publicstatic ushort VK_OEM_2 = (ushort)0xBF;
publicstatic ushort VK_OEM_3 = (ushort)0xC0;
publicstatic ushort VK_OEM_4 = (ushort)0xDB;
publicstatic ushort VK_OEM_5 = (ushort)0xDC;
publicstatic ushort VK_OEM_6 = (ushort)0xDD;
publicstatic ushort VK_OEM_7 = (ushort)0xDE;
publicstatic ushort VK_OEM_8 = (ushort)0xDF;
publicstatic ushort VK_OEM_102 = (ushort)0xE2;
publicstatic ushort VK_PROCESSKEY = (ushort)0xE5;
publicstatic ushort VK_PACKET = (ushort)0xE7;
publicstatic ushort VK_ATTN = (ushort)0xF6;
publicstatic ushort VK_CRSEL = (ushort)0xF7;
publicstatic ushort VK_EXSEL = (ushort)0xF8;
publicstatic ushort VK_EREOF = (ushort)0xF9;
```

```
publicstaticushort VK_PLAY = (ushort)0xFA;
publicstaticushort VK_ZOOM = (ushort)0xFB;
publicstaticushort VK_NONAME = (ushort)0xFC;
publicstaticushort VK_PA1 = (ushort)0xFD;
publicstaticushort VK_OEM_CLEAR = (ushort)0xFE;
publicstaticushort S_LBUTTON = (ushort)MapVirtualKey(0x01, 0);
publicstaticushort S_RBUTTON = (ushort)MapVirtualKey(0x02, 0);
publicstaticushort S_CANCEL = (ushort)MapVirtualKey(0x03, 0);
publicstaticushort S_MBUTTON = (ushort)MapVirtualKey(0x04, 0);
publicstaticushort S_XBUTTON1 = (ushort)MapVirtualKey(0x05, 0);
publicstaticushort S_XBUTTON2 = (ushort)MapVirtualKey(0x06, 0);
publicstaticushort S_BACK = (ushort)MapVirtualKey(0x08, 0);
publicstaticushort S_Tab = (ushort)MapVirtualKey(0x09, 0);
publicstaticushort S_CLEAR = (ushort)MapVirtualKey(0x0C, 0);
publicstaticushort S_Return = (ushort)MapVirtualKey(0x0D, 0);
publicstaticushort S_SHIFT = (ushort)MapVirtualKey(0x10, 0);
publicstaticushort S_CONTROL = (ushort)MapVirtualKey(0x11, 0);
publicstaticushort S_MENU = (ushort)MapVirtualKey(0x12, 0);
publicstaticushort S_PAUSE = (ushort)MapVirtualKey(0x13, 0);
publicstaticushort S_CAPITAL = (ushort)MapVirtualKey(0x14, 0);
publicstaticushort S_KANA = (ushort)MapVirtualKey(0x15, 0);
publicstaticushort S_HANGEUL = (ushort)MapVirtualKey(0x15, 0);
publicstaticushort S_HANGUL = (ushort)MapVirtualKey(0x15, 0);
publicstaticushort S_JUNJA = (ushort)MapVirtualKey(0x17, 0);
publicstaticushort S_FINAL = (ushort)MapVirtualKey(0x18, 0);
publicstaticushort S_HANJA = (ushort)MapVirtualKey(0x19, 0);
publicstaticushort S_KANJI = (ushort)MapVirtualKey(0x19, 0);
publicstaticushort S_Escape = (ushort)MapVirtualKey(0x1B, 0);
publicstaticushort S_CONVERT = (ushort)MapVirtualKey(0x1C, 0);
publicstaticushort S_NONCONVERT = (ushort)MapVirtualKey(0x1D, 0);
publicstaticushort S_ACCEPT = (ushort)MapVirtualKey(0x1E, 0);
publicstaticushort S_MODECHANGE = (ushort)MapVirtualKey(0x1F, 0);
publicstaticushort S_Space = (ushort)MapVirtualKey(0x20, 0);
publicstaticushort S_PRIOR = (ushort)MapVirtualKey(0x21, 0);
publicstaticushort S_NEXT = (ushort)MapVirtualKey(0x22, 0);
publicstaticushort S_END = (ushort)MapVirtualKey(0x23, 0);
publicstaticushort S_HOME = (ushort)MapVirtualKey(0x24, 0);
publicstaticushort S_LEFT = (ushort)MapVirtualKey(0x25, 0);
publicstaticushort S_UP = (ushort)MapVirtualKey(0x26, 0);
publicstaticushort S_RIGHT = (ushort)MapVirtualKey(0x27, 0);
publicstaticushort S_DOWN = (ushort)MapVirtualKey(0x28, 0);
publicstaticushort S_SELECT = (ushort)MapVirtualKey(0x29, 0);
publicstaticushort S_PRINT = (ushort)MapVirtualKey(0x2A, 0);
publicstaticushort S_EXECUTE = (ushort)MapVirtualKey(0x2B, 0);
publicstaticushort S_SNAPSHOT = (ushort)MapVirtualKey(0x2C, 0);
publicstaticushort S_INSERT = (ushort)MapVirtualKey(0x2D, 0);
publicstaticushort S_DELETE = (ushort)MapVirtualKey(0x2E, 0);
publicstaticushort S_HELP = (ushort)MapVirtualKey(0x2F, 0);
publicstaticushort S_APOSTROPHE = (ushort)MapVirtualKey(0xDE, 0);
publicstaticushort S_0 = (ushort)MapVirtualKey(0x30, 0);
publicstaticushort S_1 = (ushort)MapVirtualKey(0x31, 0);
publicstaticushort S_2 = (ushort)MapVirtualKey(0x32, 0);
publicstaticushort S_3 = (ushort)MapVirtualKey(0x33, 0);
publicstaticushort S_4 = (ushort)MapVirtualKey(0x34, 0);
publicstaticushort S_5 = (ushort)MapVirtualKey(0x35, 0);
publicstaticushort S_6 = (ushort)MapVirtualKey(0x36, 0);
publicstaticushort S_7 = (ushort)MapVirtualKey(0x37, 0);
publicstaticushort S_8 = (ushort)MapVirtualKey(0x38, 0);
publicstaticushort S_9 = (ushort)MapVirtualKey(0x39, 0);
publicstaticushort S_A = (ushort)MapVirtualKey(0x41, 0);
publicstaticushort S_B = (ushort)MapVirtualKey(0x42, 0);
publicstaticushort S_C = (ushort)MapVirtualKey(0x43, 0);
publicstaticushort S_D = (ushort)MapVirtualKey(0x44, 0);
```

```
publicstaticushort S_E = (ushort)MapVirtualKey(0x45, 0);
publicstaticushort S_F = (ushort)MapVirtualKey(0x46, 0);
publicstaticushort S_G = (ushort)MapVirtualKey(0x47, 0);
publicstaticushort S_H = (ushort)MapVirtualKey(0x48, 0);
publicstaticushort S_I = (ushort)MapVirtualKey(0x49, 0);
publicstaticushort S_J = (ushort)MapVirtualKey(0x4A, 0);
publicstaticushort S_K = (ushort)MapVirtualKey(0x4B, 0);
publicstaticushort S_L = (ushort)MapVirtualKey(0x4C, 0);
publicstaticushort S_M = (ushort)MapVirtualKey(0x4D, 0);
publicstaticushort S_N = (ushort)MapVirtualKey(0x4E, 0);
publicstaticushort S_O = (ushort)MapVirtualKey(0x4F, 0);
publicstaticushort S_P = (ushort)MapVirtualKey(0x50, 0);
publicstaticushort S_Q = (ushort)MapVirtualKey(0x51, 0);
publicstaticushort S_R = (ushort)MapVirtualKey(0x52, 0);
publicstaticushort S_S = (ushort)MapVirtualKey(0x53, 0);
publicstaticushort S_T = (ushort)MapVirtualKey(0x54, 0);
publicstaticushort S_U = (ushort)MapVirtualKey(0x55, 0);
publicstaticushort S_V = (ushort)MapVirtualKey(0x56, 0);
publicstaticushort S_W = (ushort)MapVirtualKey(0x57, 0);
publicstaticushort S_X = (ushort)MapVirtualKey(0x58, 0);
publicstaticushort S_Y = (ushort)MapVirtualKey(0x59, 0);
publicstaticushort S_Z = (ushort)MapVirtualKey(0x5A, 0);
publicstaticushort S_LWIN = (ushort)MapVirtualKey(0x5B, 0);
publicstaticushort S_RWIN = (ushort)MapVirtualKey(0x5C, 0);
publicstaticushort S_APPS = (ushort)MapVirtualKey(0x5D, 0);
publicstaticushort S_SLEEP = (ushort)MapVirtualKey(0x5F, 0);
publicstaticushort S_NUMPAD0 = (ushort)MapVirtualKey(0x60, 0);
publicstaticushort S_NUMPAD1 = (ushort)MapVirtualKey(0x61, 0);
publicstaticushort S_NUMPAD2 = (ushort)MapVirtualKey(0x62, 0);
publicstaticushort S_NUMPAD3 = (ushort)MapVirtualKey(0x63, 0);
publicstaticushort S_NUMPAD4 = (ushort)MapVirtualKey(0x64, 0);
publicstaticushort S_NUMPAD5 = (ushort)MapVirtualKey(0x65, 0);
publicstaticushort S_NUMPAD6 = (ushort)MapVirtualKey(0x66, 0);
publicstaticushort S_NUMPAD7 = (ushort)MapVirtualKey(0x67, 0);
publicstaticushort S_NUMPAD8 = (ushort)MapVirtualKey(0x68, 0);
publicstaticushort S_NUMPAD9 = (ushort)MapVirtualKey(0x69, 0);
publicstaticushort S_MULTIPLY = (ushort)MapVirtualKey(0x6A, 0);
publicstaticushort S_ADD = (ushort)MapVirtualKey(0x6B, 0);
publicstaticushort S_SEPARATOR = (ushort)MapVirtualKey(0x6C, 0);
publicstaticushort S_SUBTRACT = (ushort)MapVirtualKey(0x6D, 0);
publicstaticushort S_DECIMAL = (ushort)MapVirtualKey(0x6E, 0);
publicstaticushort S_DIVIDE = (ushort)MapVirtualKey(0x6F, 0);
publicstaticushort S_F1 = (ushort)MapVirtualKey(0x70, 0);
publicstaticushort S_F2 = (ushort)MapVirtualKey(0x71, 0);
publicstaticushort S_F3 = (ushort)MapVirtualKey(0x72, 0);
publicstaticushort S_F4 = (ushort)MapVirtualKey(0x73, 0);
publicstaticushort S_F5 = (ushort)MapVirtualKey(0x74, 0);
publicstaticushort S_F6 = (ushort)MapVirtualKey(0x75, 0);
publicstaticushort S_F7 = (ushort)MapVirtualKey(0x76, 0);
publicstaticushort S_F8 = (ushort)MapVirtualKey(0x77, 0);
publicstaticushort S_F9 = (ushort)MapVirtualKey(0x78, 0);
publicstaticushort S_F10 = (ushort)MapVirtualKey(0x79, 0);
publicstaticushort S_F11 = (ushort)MapVirtualKey(0x7A, 0);
publicstaticushort S_F12 = (ushort)MapVirtualKey(0x7B, 0);
publicstaticushort S_F13 = (ushort)MapVirtualKey(0x7C, 0);
publicstaticushort S_F14 = (ushort)MapVirtualKey(0x7D, 0);
publicstaticushort S_F15 = (ushort)MapVirtualKey(0x7E, 0);
publicstaticushort S_F16 = (ushort)MapVirtualKey(0x7F, 0);
publicstaticushort S_F17 = (ushort)MapVirtualKey(0x80, 0);
publicstaticushort S_F18 = (ushort)MapVirtualKey(0x81, 0);
publicstaticushort S_F19 = (ushort)MapVirtualKey(0x82, 0);
publicstaticushort S_F20 = (ushort)MapVirtualKey(0x83, 0);
publicstaticushort S_F21 = (ushort)MapVirtualKey(0x84, 0);
```

```
    public static ushort S_F22 = (ushort)MapVirtualKey(0x85, 0);
    public static ushort S_F23 = (ushort)MapVirtualKey(0x86, 0);
    public static ushort S_F24 = (ushort)MapVirtualKey(0x87, 0);
    public static ushort S_NUMLOCK = (ushort)MapVirtualKey(0x90, 0);
    public static ushort S_SCROLL = (ushort)MapVirtualKey(0x91, 0);
    public static ushort S_LeftShift = (ushort)MapVirtualKey(0xA0, 0);
    public static ushort S_RightShift = (ushort)MapVirtualKey(0xA1, 0);
    public static ushort S_LeftControl = (ushort)MapVirtualKey(0xA2, 0);
    public static ushort S_RightControl = (ushort)MapVirtualKey(0xA3, 0);
    public static ushort S_LMENU = (ushort)MapVirtualKey(0xA4, 0);
    public static ushort S_RMENU = (ushort)MapVirtualKey(0xA5, 0);
    public static ushort S_BROWSER_BACK = (ushort)MapVirtualKey(0xA6, 0);
    public static ushort S_BROWSER_FORWARD = (ushort)MapVirtualKey(0xA7, 0);
    public static ushort S_BROWSER_REFRESH = (ushort)MapVirtualKey(0xA8, 0);
    public static ushort S_BROWSER_STOP = (ushort)MapVirtualKey(0xA9, 0);
    public static ushort S_BROWSER_SEARCH = (ushort)MapVirtualKey(0xAA, 0);
    public static ushort S_BROWSER_FAVORITES = (ushort)MapVirtualKey(0xAB, 0);
    public static ushort S_BROWSER_HOME = (ushort)MapVirtualKey(0xAC, 0);
    public static ushort S_VOLUME_MUTE = (ushort)MapVirtualKey(0xAD, 0);
    public static ushort S_VOLUME_DOWN = (ushort)MapVirtualKey(0xAE, 0);
    public static ushort S_VOLUME_UP = (ushort)MapVirtualKey(0xAF, 0);
    public static ushort S_MEDIA_NEXT_TRACK = (ushort)MapVirtualKey(0xB0, 0);
    public static ushort S_MEDIA_PREV_TRACK = (ushort)MapVirtualKey(0xB1, 0);
    public static ushort S_MEDIA_STOP = (ushort)MapVirtualKey(0xB2, 0);
    public static ushort S_MEDIA_PLAY_PAUSE = (ushort)MapVirtualKey(0xB3, 0);
    public static ushort S_LAUNCH_MAIL = (ushort)MapVirtualKey(0xB4, 0);
    public static ushort S_LAUNCH_MEDIA_SELECT = (ushort)MapVirtualKey(0xB5, 0);
    public static ushort S_LAUNCH_APP1 = (ushort)MapVirtualKey(0xB6, 0);
    public static ushort S_LAUNCH_APP2 = (ushort)MapVirtualKey(0xB7, 0);
    public static ushort S_OEM_1 = (ushort)MapVirtualKey(0xBA, 0);
    public static ushort S_OEM_PLUS = (ushort)MapVirtualKey(0xBB, 0);
    public static ushort S_OEM_COMMA = (ushort)MapVirtualKey(0xBC, 0);
    public static ushort S_OEM_MINUS = (ushort)MapVirtualKey(0xBD, 0);
    public static ushort S_OEM_PERIOD = (ushort)MapVirtualKey(0xBE, 0);
    public static ushort S_OEM_2 = (ushort)MapVirtualKey(0xBF, 0);
    public static ushort S_OEM_3 = (ushort)MapVirtualKey(0xC0, 0);
    public static ushort S_OEM_4 = (ushort)MapVirtualKey(0xDB, 0);
    public static ushort S_OEM_5 = (ushort)MapVirtualKey(0xDC, 0);
    public static ushort S_OEM_6 = (ushort)MapVirtualKey(0xDD, 0);
    public static ushort S_OEM_7 = (ushort)MapVirtualKey(0xDE, 0);
    public static ushort S_OEM_8 = (ushort)MapVirtualKey(0xDF, 0);
    public static ushort S_OEM_102 = (ushort)MapVirtualKey(0xE2, 0);
    public static ushort S_PROCESSKEY = (ushort)MapVirtualKey(0xE5, 0);
    public static ushort S_PACKET = (ushort)MapVirtualKey(0xE7, 0);
    public static ushort S_ATTN = (ushort)MapVirtualKey(0xF6, 0);
    public static ushort S_CRSEL = (ushort)MapVirtualKey(0xF7, 0);
    public static ushort S_EXSEL = (ushort)MapVirtualKey(0xF8, 0);
    public static ushort S_EREOF = (ushort)MapVirtualKey(0xF9, 0);
    public static ushort S_PLAY = (ushort)MapVirtualKey(0xFA, 0);
    public static ushort S_ZOOM = (ushort)MapVirtualKey(0xFB, 0);
    public static ushort S_NONAME = (ushort)MapVirtualKey(0xFC, 0);
    public static ushort S_PA1 = (ushort)MapVirtualKey(0xFD, 0);
    public static ushort S_OEM_CLEAR = (ushort)MapVirtualKey(0xFE, 0);
}
public class Input
{
    private IntPtr context;
    public KeyboardFilterMode KeyboardFilterMode { get; set; }
    public MouseFilterMode MouseFilterMode { get; set; }
    public bool IsLoaded { get; set; }
    public Input()
    {
        context = IntPtr.Zero;
```

```
    KeyboardFilterMode = KeyboardFilterMode.None;
    MouseFilterMode = MouseFilterMode.None;
}
public bool Load()
{
    context = InterceptionDriver.CreateContext();
    return true;
}
public void Unload()
{
    InterceptionDriver.DestroyContext(context);
}
public void SendKey(Keys key, KeyState state, int keyboardId)
{
    Stroke stroke = new Stroke();
    KeyStroke keyStroke = new KeyStroke();
    keyStroke.Code = key;
    keyStroke.State = state;
    stroke.Key = keyStroke;
    InterceptionDriver.Send(context, keyboardId, ref stroke, 1);
}
public void SendKey(Keys key, int keyboardId)
{
    SendKey(key, KeyState.Down, keyboardId);
}
public void SendKeyF(Keys key, int keyboardId)
{
    SendKey(key, KeyState.Up, keyboardId);
}
public void SendMouseEvent(MouseState state, int mouseId)
{
    Stroke stroke = new Stroke();
    MouseStroke mouseStroke = new MouseStroke();
    mouseStroke.State = state;
    if (state == MouseState.ScrollUp)
    {
        mouseStroke.Rolling = 120;
    }
    else if (state == MouseState.ScrollDown)
    {
        mouseStroke.Rolling = -120;
    }
    stroke.Mouse = mouseStroke;
    InterceptionDriver.Send(context, mouseId, ref stroke, 1);
}
public void SendLeftClick(int mouseId)
{
    SendMouseEvent(MouseState.LeftDown, mouseId);
}
public void SendRightClick(int mouseId)
{
    SendMouseEvent(MouseState.RightDown, mouseId);
}
public void SendLeftClickF(int mouseId)
{
    SendMouseEvent(MouseState.LeftUp, mouseId);
}
public void SendRightClickF(int mouseId)
{
    SendMouseEvent(MouseState.RightUp, mouseId);
}
public void SendMiddleClick(int mouseId)
{
```

```
        SendMouseEvent(MouseState.MiddleDown, mouseId);
    }
    public void SendMiddleClickF(int mouseId)
    {
        SendMouseEvent(MouseState.MiddleUp, mouseId);
    }
    public void SendWheelUp(int mouseId)
    {
        SendMouseEvent(MouseState.ScrollUp, mouseId);
    }
    public void SendWheelDown(int mouseId)
    {
        SendMouseEvent(MouseState.ScrollDown, mouseId);
    }
    public void MoveMouseBy(int deltaX, int deltaY, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = deltaX;
        mouseStroke.Y = deltaY;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveRelative;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
    public void MoveMouseTo(int x, int y, int mouseId)
    {
        Stroke stroke = new Stroke();
        MouseStroke mouseStroke = new MouseStroke();
        mouseStroke.X = x;
        mouseStroke.Y = y;
        stroke.Mouse = mouseStroke;
        stroke.Mouse.Flags = MouseFlags.MoveAbsolute;
        InterceptionDriver.Send(context, mouseId, ref stroke, 1);
    }
}
[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
public delegate int Predicate(int device);
[Flags]
public enum KeyState : ushort
{
    Down = 0x00,
    Up = 0x01,
    E0 = 0x02,
    E1 = 0x04,
    TermsrvSetLED = 0x08,
    TermsrvShadow = 0x10,
    TermsrvVKPacket = 0x20
}
[Flags]
public enum KeyboardFilterMode : ushort
{
    None = 0x0000,
    All = 0xFFFF,
    KeyDown = KeyState.Up,
    KeyUp = KeyState.Up << 1,
    KeyE0 = KeyState.E0 << 1,
    KeyE1 = KeyState.E1 << 1,
    KeyTermsrvSetLED = KeyState.TermsrvSetLED << 1,
    KeyTermsrvShadow = KeyState.TermsrvShadow << 1,
    KeyTermsrvVKPacket = KeyState.TermsrvVKPacket << 1
}
[Flags]
public enum MouseState : ushort
```

```
{
   LeftDown = 0x01,
   LeftUp = 0x02,
   RightDown = 0x04,
   RightUp = 0x08,
   MiddleDown = 0x10,
   MiddleUp = 0x20,
   LeftExtraDown = 0x40,
   LeftExtraUp = 0x80,
   RightExtraDown = 0x100,
   RightExtraUp = 0x200,
   ScrollVertical = 0x400,
   ScrollUp = 0x400,
   ScrollDown = 0x400,
   ScrollHorizontal = 0x800,
   ScrollLeft = 0x800,
   ScrollRight = 0x800,
}
[Flags]
public enum MouseFilterMode : ushort
{
   None = 0x0000,
   All = 0xFFFF,
   LeftDown = 0x01,
   LeftUp = 0x02,
   RightDown = 0x04,
   RightUp = 0x08,
   MiddleDown = 0x10,
   MiddleUp = 0x20,
   LeftExtraDown = 0x40,
   LeftExtraUp = 0x80,
   RightExtraDown = 0x100,
   RightExtraUp = 0x200,
   MouseWheelVertical = 0x400,
   MouseWheelHorizontal = 0x800,
   MouseMove = 0x1000,
}
[Flags]
public enum MouseFlags : ushort
{
   MoveRelative = 0x000,
   MoveAbsolute = 0x001,
   VirtualDesktop = 0x002,
   AttributesChanged = 0x004,
   MoveWithoutCoalescing = 0x008,
   TerminalServicesSourceShadow = 0x100
}
[StructLayout(LayoutKind.Sequential)]
public struct MouseStroke
{
   public MouseState State;
   public MouseFlags Flags;
   public Int16 Rolling;
   public Int32 X;
   public Int32 Y;
   public UInt16 Information;
}
[StructLayout(LayoutKind.Sequential)]
public struct KeyStroke
{
   public Keys Code;
   public KeyState State;
   public UInt32 Information;
```

```
        }
        [StructLayout(LayoutKind.Explicit)]
        public struct Stroke
        {
            [FieldOffset(0)]
            public MouseStroke Mouse;
            [FieldOffset(0)]
            public KeyStroke Key;
        }
        public static class InterceptionDriver
        {
            [DllImport(""interception.dll"", EntryPoint = ""interception_create_context"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern IntPtr CreateContext();
            [DllImport(""interception.dll"", EntryPoint = ""interception_destroy_context"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern void DestroyContext(IntPtr context);
            [DllImport(""interception.dll"", EntryPoint = ""interception_get_precedence"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern void GetPrecedence(IntPtr context, Int32 device);
            [DllImport(""interception.dll"", EntryPoint = ""interception_set_precedence"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern void SetPrecedence(IntPtr context, Int32 device, Int32 Precedence);
            [DllImport(""interception.dll"", EntryPoint = ""interception_get_filter"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern void GetFilter(IntPtr context, Int32 device);
            [DllImport(""interception.dll"", EntryPoint = ""interception_set_filter"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern void SetFilter(IntPtr context, Predicate predicate, Int32 keyboardFilterMode);
            [DllImport(""interception.dll"", EntryPoint = ""interception_wait"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 Wait(IntPtr context);
            [DllImport(""interception.dll"", EntryPoint = ""interception_wait_with_timeout"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 WaitWithTimeout(IntPtr context, UInt64 milliseconds);
            [DllImport(""interception.dll"", EntryPoint = ""interception_send"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 Send(IntPtr context, Int32 device, ref Stroke stroke, UInt32 numStrokes);
            [DllImport(""interception.dll"", EntryPoint = ""interception_receive"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 Receive(IntPtr context,Int32 device, ref Stroke stroke, UInt32 numStrokes);
            [DllImport(""interception.dll"", EntryPoint = ""interception_get_hardware_id"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 GetHardwareId(IntPtr context, Int32 device, String hardwareIdentifier, UInt32
sizeOfString);
            [DllImport(""interception.dll"", EntryPoint = ""interception_is_invalid"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 IsInvalid(Int32 device);
            [DllImport(""interception.dll"", EntryPoint = ""interception_is_keyboard"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 IsKeyboard(Int32 device);
            [DllImport(""interception.dll"", EntryPoint = ""interception_is_mouse"", CallingConvention =
CallingConvention.Cdecl)]
            public static extern Int32 IsMouse(Int32 device);
        }
        public class KeyPressedEventArgs : EventArgs
        {
            public Keys Key { get; set; }
            public KeyState State { get; set; }
            public bool Handled { get;set; }
        }
        public enum Keys : ushort
        {
```

CANCEL = 70,
BACK = 14,
TAB = 15,
CLEAR = 76,
RETURN = 28,
SHIFT = 42,
CONTROL = 29,
MENU = 56,
CAPITAL = 58,
ESCAPE = 1,
SPACE = 57,
PRIOR = 73,
NEXT = 81,
END = 79,
HOME = 71,
LEFT = 101,
UP = 100,
RIGHT = 103,
DOWN = 102,
SNAPSHOT = 84,
INSERT = 91,
NUMPADDEL = 83,
NUMPADINSERT = 82,
HELP = 99,
APOSTROPHE = 41,
BACKSPACE = 14,
PAGEDOWN = 97,
PAGEUP = 93,
FIN = 96,
MOUSE = 105,
A = 16,
B = 48,
C = 46,
D = 32,
E = 18,
F = 33,
G = 34,
H = 35,
I = 23,
J = 36,
K = 37,
L = 38,
M = 39,
N = 49,
O = 24,
P = 25,
Q = 30,
R = 19,
S = 31,
T = 20,
U = 22,
V = 47,
W = 44,
X = 45,
Y = 21,
Z = 17,
LWIN = 91,
RWIN = 92,
APPS = 93,
DELETE = 95,
NUMPAD0 = 82,
NUMPAD1 = 79,
NUMPAD2 = 80,

NUMPAD3 = 81,
NUMPAD4 = 75,
NUMPAD5 = 76,
NUMPAD6 = 77,
NUMPAD7 = 71,
NUMPAD8 = 72,
NUMPAD9 = 73,
MULTIPLY = 55,
ADD = 78,
SUBTRACT = 74,
DECIMAL = 83,
PRINTSCREEN = 84,
DIVIDE = 53,
F1 = 59,
F2 = 60,
F3 = 61,
F4 = 62,
F5 = 63,
F6 = 64,
F7 = 65,
F8 = 66,
F9 = 67,
F10 = 68,
F11 = 87,
F12 = 88,
NUMLOCK = 69,
SCROLLLOCK = 70,
LEFTSHIFT = 42,
RIGHTSHIFT = 54,
LEFTCONTROL = 29,
RIGHTCONTROL = 99,
LEFTALT = 56,
RIGHTALT = 98,
BROWSER_BACK = 106,
BROWSER_FORWARD = 105,
BROWSER_REFRESH = 103,
BROWSER_STOP = 104,
BROWSER_SEARCH = 101,
BROWSER_FAVORITES = 102,
BROWSER_HOME = 50,
VOLUME_MUTE = 32,
VOLUME_DOWN = 46,
VOLUME_UP = 48,
MEDIA_NEXT_TRACK = 25,
MEDIA_PREV_TRACK = 16,
MEDIA_STOP = 36,
MEDIA_PLAY_PAUSE = 34,
LAUNCH_MAIL = 108,
LAUNCH_MEDIA_SELECT = 109,
LAUNCH_APP1 = 107,
LAUNCH_APP2 = 33,
OEM_1 = 27,
OEM_PLUS = 13,
OEM_COMMA = 50,
OEM_MINUS = 0,
OEM_PERIOD = 51,
OEM_2 = 52,
OEM_3 = 40,
OEM_4 = 12,
OEM_5 = 43,
OEM_6 = 26,
OEM_7 = 41,
OEM_8 = 53,

```
OEM_102 = 86,
EREOF = 93,
ZOOM = 98,
Escape = 1,
One = 2,
Two = 3,
Three = 4,
Four = 5,
Five = 6,
Six = 7,
Seven = 8,
Eight = 9,
Nine = 10,
Zero = 11,
DashUnderscore = 12,
PlusEquals = 13,
Backspace = 14,
Tab = 15,
OpenBracketBrace = 26,
CloseBracketBrace = 27,
Enter = 28,
Control = 29,
SemicolonColon = 39,
SingleDoubleQuote = 40,
Tilde = 41,
LeftShift = 42,
BackslashPipe = 43,
CommaLeftArrow = 51,
PeriodRightArrow = 52,
ForwardSlashQuestionMark = 53,
RightShift = 54,
RightAlt = 56,
Space = 57,
CapsLock = 58,
Up = 72,
Down = 80,
Right = 77,
Left = 75,
Home = 71,
End = 79,
Delete = 83,
PageUp = 73,
PageDown = 81,
Insert = 82,
PrintScreen = 55,
NumLock = 69,
ScrollLock = 70,
Menu = 93,
WindowsKey = 91,
NumpadDivide = 53,
NumpadAsterisk = 55,
Numpad7 = 71,
Numpad8 = 72,
Numpad9 = 73,
Numpad4 = 75,
Numpad5 = 76,
Numpad6 = 77,
Numpad1 = 79,
Numpad2 = 80,
Numpad3 = 81,
Numpad0 = 82,
NumpadDelete = 83,
NumpadEnter = 28,
```

```
            NumpadPlus = 78,
            NumpadMinus = 74,
        }
        public class MousePressedEventArgs : EventArgs
        {
            public MouseState State { get; set; }
            public bool Handled { get; set; }
            public int X { get; set; }
            public int Y { get; set; }
            public short Rolling { get; set; }
        }
        public enum ScrollDirection
        {
            Down,
            Up
        }
    }
    ";
    addedcode = @"public static bool[] _Valuechanged = new bool[2], _valuechanged = new bool[2];
        public double mousex, mousey, mousexp, mouseyp, irx, iry, deadzone = 30f, zoning = 250f, hardness = 330f, center = 200f, max = 1900f;
        public void Main(double mWSNunchuckStateRawJoystickX, double mWSNunchuckStateRawJoystickY, double mWSNunchuckStateRawValuesX, double mWSNunchuckStateRawValuesY, double mWSNunchuckStateRawValuesZ, bool mWSNunchuckStateC, bool mWSNunchuckStateZ,
        double mWSButtonStateIRX, double mWSButtonStateIRY, bool mWSButtonStateA, bool mWSButtonStateB, bool mWSButtonStateMinus, bool mWSButtonStateHome, bool mWSButtonStatePlus, bool mWSButtonStateOne, bool mWSButtonStateTwo, bool mWSButtonStateUp, bool mWSButtonStateDown, bool mWSButtonStateLeft, bool mWSButtonStateRight, double mWSRawValuesX, double mWSRawValuesY, double mWSRawValuesZ,
        float EulerAnglesX, float EulerAnglesY, float EulerAnglesZ, float DirectAnglesX, float DirectAnglesY, float DirectAnglesZ, double camx, double camy, float EulerAnglesLeftX, float EulerAnglesLeftY, float EulerAnglesLeftZ, float DirectAnglesLeftX, float DirectAnglesLeftY, float DirectAnglesLeftZ, float EulerAnglesRightX, float EulerAnglesRightY, float EulerAnglesRightZ,
        float DirectAnglesRightX, float DirectAnglesRightY, float DirectAnglesRightZ, bool LeftButtonSHOULDER_1, bool LeftButtonMINUS, bool LeftButtonCAPTURE, bool LeftButtonDPAD_UP, bool LeftButtonDPAD_LEFT, bool LeftButtonDPAD_DOWN, bool LeftButtonDPAD_RIGHT, bool LeftButtonSTICK, bool RightButtonDPAD_DOWN, bool LeftButtonSL, bool LeftButtonSR, double GetStickLeftX, double GetStickLeftY,
        bool RightButtonPLUS, bool RightButtonDPAD_RIGHT, bool RightButtonHOME, bool RightButtonSHOULDER_1, bool RightButtonDPAD_LEFT, bool RightButtonDPAD_UP, bool RightButtonSTICK, bool RightButtonSL, bool RightButtonSR, bool RightButtonSHOULDER_2, bool LeftButtonSHOULDER_2, double GetStickRightX, double GetStickRightY, float GetAccelX, float GetAccelY, float GetAccelZ, float GetAccelRightX, float GetAccelRightY, float GetAccelRightZ, float GetAccelLeftX, float GetAccelLeftY, float GetAccelLeftZ, double watchM)
        {
            irx = (mWSButtonStateIRX >= 0 ? Scale(mWSButtonStateIRX, 0f, 1360f, 0f, 1500f) : Scale(mWSButtonStateIRX, -1360f, 0f, -1500f, 0f));
            iry = (mWSButtonStateIRY + center >= 0 ? Scale(mWSButtonStateIRY + center, 0f, 768f + center, 0f, max) : Scale(mWSButtonStateIRY + center, -768f + center, 0f, -max, 0f));
            mousex = Math.Pow(irx > 0 ? irx : -irx, zoning / 100f) * (1500f / Math.Pow(1500f, zoning / 100f)) * (irx > 0 ? 1f : -1f) * (-deadzone / 100f + 1f);
            mousey = Math.Pow(iry > 0 ? iry : -iry, zoning * (1500f / max) / 100f) * (max / Math.Pow(max, zoning * (1500f / max) / 100f)) * (iry > 0 ? 1f : -1f) * (-deadzone / 100f + 1f);
            mousexp += mousex * watchM / 40f;
            mouseyp += mousey * watchM / 40f;
            input.MoveMouseTo((int)(32767.5f - (mousex * hardness) / 100f - mousexp), (int)((mousey * hardness) / 100f + mouseyp + 32767.5f), 12);
        }
        private double Scale(double value, double min, double max, double minScale, double maxScale)
        {
            double scaled = minScale + (double)(value - min) / (max - min) * (maxScale - minScale);
            return scaled;
        }";
    string keyboardaddedcode = @"public static bool[] _Valuechanged = new bool[36], _valuechanged = new bool[36];
        public bool foraorcison, mWSButtonStateAio, randA, ApressIO = false, HomeFTG = false;
        public void Main(double mWSNunchuckStateRawJoystickX, double mWSNunchuckStateRawJoystickY, double
```

```
mWSNunchuckStateRawValuesX, double mWSNunchuckStateRawValuesY, double mWSNunchuckStateRawValuesZ, bool
mWSNunchuckStateC, bool mWSNunchuckStateZ,
        double mWSButtonStateIRX, double mWSButtonStateIRY, bool mWSButtonStateA, bool mWSButtonStateB, bool
mWSButtonStateMinus, bool mWSButtonStateHome, bool mWSButtonStatePlus, bool mWSButtonStateOne, bool
mWSButtonStateTwo, bool mWSButtonStateUp, bool mWSButtonStateDown, bool mWSButtonStateLeft, bool
mWSButtonStateRight, double mWSRawValuesX, double mWSRawValuesY, double mWSRawValuesZ,
        float EulerAnglesX, float EulerAnglesY, float EulerAnglesZ, float DirectAnglesX, float DirectAnglesY, float
DirectAnglesZ, double camx, double camy, float EulerAnglesLeftX, float EulerAnglesLeftY, float EulerAnglesLeftZ, float
DirectAnglesLeftX, float DirectAnglesLeftY, float DirectAnglesLeftZ, float EulerAnglesRightX, float EulerAnglesRightY, float
EulerAnglesRightZ,
        float DirectAnglesRightX, float DirectAnglesRightY, float DirectAnglesRightZ, bool LeftButtonSHOULDER_1, bool
LeftButtonMINUS, bool LeftButtonCAPTURE, bool LeftButtonDPAD_UP, bool LeftButtonDPAD_LEFT, bool
LeftButtonDPAD_DOWN, bool LeftButtonDPAD_RIGHT, bool LeftButtonSTICK, bool RightButtonDPAD_DOWN, bool
LeftButtonSL, bool LeftButtonSR, double GetStickLeftX, double GetStickLeftY,
        bool RightButtonPLUS, bool RightButtonDPAD_RIGHT, bool RightButtonHOME, bool RightButtonSHOULDER_1,
bool RightButtonDPAD_LEFT, bool RightButtonDPAD_UP, bool RightButtonSTICK, bool RightButtonSL, bool RightButtonSR,
bool RightButtonSHOULDER_2, bool LeftButtonSHOULDER_2, double GetStickRightX, double GetStickRightY, float
GetAccelX, float GetAccelY, float GetAccelZ, float GetAccelRightX, float GetAccelRightY, float GetAccelRightZ, float
GetAccelLeftX, float GetAccelLeftY, float GetAccelLeftZ, double watchM)
    {
        this[2] = LeftButtonSHOULDER_2;
        if (_Valuechanged[2] & this[2])
            SimulateKeyDown(VK_LeftControl, S_LeftControl);
        if (_Valuechanged[2] & !this[2])
            SimulateKeyUp(VK_LeftControl, S_LeftControl);
        this[3] = LeftButtonMINUS;
        if (_Valuechanged[3] & this[3])
            SimulateKeyDown(VK_Return, S_Return);
        if (_Valuechanged[3] & !this[3])
            SimulateKeyUp(VK_Return, S_Return);
        this[4] = (GetAccelLeftX > 1.5f | GetAccelLeftX < -1.5f) & !((mWSRawValuesZ > 0 ? mWSRawValuesZ : -
mWSRawValuesZ) >= 40f & (mWSRawValuesY > 0 ? mWSRawValuesY : -mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ?
mWSRawValuesX : -mWSRawValuesX) >= 40f);
        if (_Valuechanged[4] & this[4])
            SimulateKeyDown(VK_V, S_V);
        if (_Valuechanged[4] & !this[4])
            SimulateKeyUp(VK_V, S_V);
        this[27] = LeftButtonCAPTURE;
        if (_Valuechanged[27] & this[27])
            SimulateKeyDown(VK_P, S_P);
        if (_Valuechanged[27] & !this[27])
            SimulateKeyUp(VK_P, S_P);
        this[5] = LeftButtonDPAD_UP;
        if (_Valuechanged[5] & this[5])
            SimulateKeyDownArrows(VK_UP, S_UP);
        if (_Valuechanged[5] & !this[5])
            SimulateKeyUpArrows(VK_UP, S_UP);
        this[6] = LeftButtonDPAD_LEFT;
        if (_Valuechanged[6] & this[6])
            SimulateKeyDownArrows(VK_LEFT, S_LEFT);
        if (_Valuechanged[6] & !this[6])
            SimulateKeyUpArrows(VK_LEFT, S_LEFT);
        this[7] = LeftButtonDPAD_DOWN;
        if (_Valuechanged[7] & this[7])
            SimulateKeyDownArrows(VK_DOWN, S_DOWN);
        if (_Valuechanged[7] & !this[7])
            SimulateKeyUpArrows(VK_DOWN, S_DOWN);
        this[8] = LeftButtonDPAD_RIGHT;
        if (_Valuechanged[8] & this[8])
            SimulateKeyDownArrows(VK_RIGHT, S_RIGHT);
        if (_Valuechanged[8] & !this[8])
            SimulateKeyUpArrows(VK_RIGHT, S_RIGHT);
        this[9] = LeftButtonSTICK;
```

```
            if (_Valuechanged[9] & this[9])
                SimulateKeyDown(VK_LeftShift, S_LeftShift);
            if (_Valuechanged[9] & !this[9])
                SimulateKeyUp(VK_LeftShift, S_LeftShift);
            this[10] = LeftButtonSHOULDER_1;
            if (_Valuechanged[10] & this[10])
                SimulateKeyDown(VK_Space, S_Space);
            if (_Valuechanged[10] & !this[10])
                SimulateKeyUp(VK_Space, S_Space);
            this[29] = LeftButtonSL;
            if (_Valuechanged[29] & this[29])
                SimulateKeyDown(VK_B, S_B);
            if (_Valuechanged[29] & !this[29])
                SimulateKeyUp(VK_B, S_B);
            this[28] = LeftButtonSR;
            if (_Valuechanged[28] & this[28])
                SimulateKeyDown(VK_N, S_N);
            if (_Valuechanged[28] & !this[28])
                SimulateKeyUp(VK_N, S_N);
            this[16] = GetStickLeftX > 0.33f;
            this[17] = GetStickLeftX < -0.33f;
            this[18] = GetStickLeftY > 0.33f;
            this[19] = GetStickLeftY < -0.33f;
            if (_Valuechanged[16] & this[16])
                SimulateKeyDown(VK_D, S_D);
            if (_Valuechanged[16] & !this[16])
                SimulateKeyUp(VK_D, S_D);
            if (_Valuechanged[17] & this[17])
                SimulateKeyDown(VK_Q, S_Q);
            if (_Valuechanged[17] & !this[17])
                SimulateKeyUp(VK_Q, S_Q);
            if (_Valuechanged[18] & this[18])
                SimulateKeyDown(VK_Z, S_Z);
            if (_Valuechanged[18] & !this[18])
                SimulateKeyUp(VK_Z, S_Z);
            if (_Valuechanged[19] & this[19])
                SimulateKeyDown(VK_S, S_S);
            if (_Valuechanged[19] & !this[19])
                SimulateKeyUp(VK_S, S_S);
            this[30] = DirectAnglesLeftY <= -0.666f;
            if (_Valuechanged[30] & this[30])
                SimulateKeyDown(VK_A, S_A);
            if (_Valuechanged[30] & !this[30])
                SimulateKeyUp(VK_A, S_A);
            this[31] = DirectAnglesLeftY >= 0.666f;
            if (_Valuechanged[31] & this[31])
                SimulateKeyDown(VK_E, S_E);
            if (_Valuechanged[31] & !this[31])
                SimulateKeyUp(VK_E, S_E);
            this[13] = (mWSRawValuesZ > 0 ? mWSRawValuesZ : -mWSRawValuesZ) >= 40f & (mWSRawValuesY > 0 ?
mWSRawValuesY : -mWSRawValuesY) >= 40f & (mWSRawValuesX > 0 ? mWSRawValuesX : -mWSRawValuesX) >= 40f;
            if (_Valuechanged[13] & this[13])
                SimulateKeyDown(VK_R, S_R);
            if (_Valuechanged[13] & !this[13])
                SimulateKeyUp(VK_R, S_R);
            this[20] = mWSButtonStateOne;
            if (_Valuechanged[20] & this[20])
                SimulateKeyDown(VK_Tab, S_Tab);
            if (_Valuechanged[20] & !this[20])
                SimulateKeyUp(VK_Tab, S_Tab);
            this[21] = mWSButtonStateDown;
            if (_Valuechanged[21] & this[21])
                SimulateKeyDown(VK_C, S_C);
```

```
            if (_Valuechanged[21] & !this[21])
                SimulateKeyUp(VK_C, S_C);
            this[22] = mWSButtonStateHome;
            if (_Valuechanged[22] & this[22])
                SimulateKeyDown(VK_F, S_F);
            if (_Valuechanged[22] & !this[22])
                SimulateKeyUp(VK_F, S_F);
            this[23] = mWSButtonStateRight;
            if (_Valuechanged[23] & this[23])
                SimulateKeyDown(VK_U, S_U);
            if (_Valuechanged[23] & !this[23])
                SimulateKeyUp(VK_U, S_U);
            this[24] = mWSButtonStateLeft;
            if (_Valuechanged[24] & this[24])
                SimulateKeyDown(VK_Y, S_Y);
            if (_Valuechanged[24] & !this[24])
                SimulateKeyUp(VK_Y, S_Y);
            this[25] = mWSButtonStateUp;
            if (_Valuechanged[25] & this[25])
                SimulateKeyDown(VK_X, S_X);
            if (_Valuechanged[25] & !this[25])
                SimulateKeyUp(VK_X, S_X);
            this[26] = mWSButtonStateTwo;
            if (_Valuechanged[26] & this[26])
                SimulateKeyDown(VK_Escape, S_Escape);
            if (_Valuechanged[26] & !this[26])
                SimulateKeyUp(VK_Escape, S_Escape);
            this[14] = mWSButtonStatePlus | (HomeFTG & mWSButtonStateHome);
            if (_Valuechanged[14] & this[14])
                SimulateKeyDown(VK_G, S_G);
            if (_Valuechanged[14] & !this[14])
                SimulateKeyUp(VK_G, S_G);
            this[15] = mWSButtonStateMinus | (HomeFTG & mWSButtonStateHome);
            if (_Valuechanged[15] & this[15])
                SimulateKeyDown(VK_T, S_T);
            if (_Valuechanged[15] & !this[15])
                SimulateKeyUp(VK_T, S_T);
            this[11] = mWSButtonStateB;
            if (_Valuechanged[11] & this[11])
                LeftClick();
            if (_Valuechanged[11] & !this[11])
                LeftClickF();
            if (ApressIO)
            {
                foraorcison = (mWSButtonStateMinus | mWSButtonStatePlus | mWSButtonStateHome | ((mWSRawValuesZ >
0 ? mWSRawValuesZ : -mWSRawValuesZ) >= 40f & (mWSRawValuesY > 0 ? mWSRawValuesY : -mWSRawValuesY) >= 40f &
(mWSRawValuesX > 0 ? mWSRawValuesX : -mWSRawValuesX) >= 40f) | mWSButtonStateUp | mWSButtonStateDown |
mWSButtonStateLeft | mWSButtonStateRight);
                this[32] = mWSButtonStateA;
                if (_Valuechanged[32] & this[32])
                    if (!randA)
                    {
                        mWSButtonStateAio = true;
                        randA = true;
                    }
                    else
                        if (randA)
                        {
                            mWSButtonStateAio = false;
                            randA = false;
                        }
                if (mWSButtonStateAio & foraorcison)
                {
```

```
            mWSButtonStateAio = false;
            randA = false;
          }
          this[33] = mWSButtonStateAio | (mWSButtonStateA & foraorcison);
          if (_Valuechanged[33] & this[33])
            RightClick();
          if (_Valuechanged[33] & !this[33])
            RightClickF();
        }
        else
        {
          this[34] = mWSButtonStateA;
          if (_Valuechanged[34] & this[34])
            RightClick();
          if (_Valuechanged[34] & !this[34])
            RightClickF();
        }
      }";
try {initConfig(); }
catch
{
    using (System.IO.StreamWriter createdfile = System.IO.File.AppendText(processname + ".txt"))
    {
        createdfile.WriteLine(processname + ".txt");
        createdfile.WriteLine("//mouse control");
        createdfile.Write(addedcode);
        createdfile.WriteLine("");
        createdfile.WriteLine("");
        createdfile.WriteLine("//keyboard control");
        createdfile.Write(keyboardaddedcode);
        createdfile.WriteLine("");
        createdfile.WriteLine("");
    }
    initConfig();
}
finalcode = code.Replace("funct_driver", mousecode);
parameters = new System.CodeDom.Compiler.CompilerParameters();
parameters.GenerateExecutable = false;
parameters.GenerateInMemory = true;
parameters.ReferencedAssemblies.Add("System.Windows.Forms.dll");
parameters.ReferencedAssemblies.Add("System.Drawing.dll");
provider = new Microsoft.CSharp.CSharpCodeProvider();
results = provider.CompileAssemblyFromSource(parameters, finalcode);
if (results.Errors.HasErrors)
{
    StringBuilder sb = new StringBuilder();
    foreach (System.CodeDom.Compiler.CompilerError error in results.Errors)
    {
        sb.AppendLine(String.Format("Error ({0}) : {1}", error.ErrorNumber, error.ErrorText));
    }
    MessageBox.Show("mouse control :\n\r" +sb.ToString());
    Getstate = false;
    this.BackColor = System.Drawing.Color.Black;
    return;
}
assembly = results.CompiledAssembly;
mouseprogram = assembly.GetType("StringToCode.FooClass");
mouseobj = Activator.CreateInstance(mouseprogram);
finalcode = code.Replace("funct_driver", keyboardcode);
parameters = new System.CodeDom.Compiler.CompilerParameters();
parameters.GenerateExecutable = false;
parameters.GenerateInMemory = true;
parameters.ReferencedAssemblies.Add("System.Windows.Forms.dll");
```

```
    parameters.ReferencedAssemblies.Add("System.Drawing.dll");
    provider = new Microsoft.CSharp.CSharpCodeProvider();
    results = provider.CompileAssemblyFromSource(parameters, finalcode);
    if (results.Errors.HasErrors)
    {
        StringBuilder sb = new StringBuilder();
        foreach (System.CodeDom.Compiler.CompilerError error in results.Errors)
        {
            sb.AppendLine(String.Format("Error ({0}) : {1}", error.ErrorNumber, error.ErrorText));
        }
        MessageBox.Show("keyboard control :\n\r" + sb.ToString());
        Getstate = false;
        this.BackColor = System.Drawing.Color.Black;
        return;
    }
    assembly = results.CompiledAssembly;
    keyboardprogram = assembly.GetType("StringToCode.FooClass");
    keyboardobj = Activator.CreateInstance(keyboardprogram);
    taskM = new Task(WiiJoy_thrM);
    taskM.Start();
    taskK = new Task(WiiJoy_thrK);
    taskK.Start();
    assembly = null;
    results = null;
    provider = new Microsoft.CSharp.CSharpCodeProvider();
    parameters = new System.CodeDom.Compiler.CompilerParameters();
    finalcode = "";
    code = "";
    assembly = null;
    results = null;
    provider = new Microsoft.CSharp.CSharpCodeProvider();
    parameters = new System.CodeDom.Compiler.CompilerParameters();
    finalcode = "";
    code = "";
    mousecode = "";
    keyboardcode = "";
    this.BackColor = System.Drawing.Color.DarkGray;
}
private bool initConnected()
{
    bool connected = false;
    System.IO.StreamReader file = new System.IO.StreamReader("initconnect.txt");
    connected = bool.Parse(file.ReadLine());
    leftandright = (int)Convert.ToDouble(file.ReadLine());
    file.Close();
    return connected;
}
private void FormStart(object sender, DoWorkEventArgs e)
{
    bool connected = false;
    do
    {
        try { connected = initConnected(); }
        catch { }
        Thread.Sleep(1);
    }
    while (!connected & !notpressing1and2);
    if (!notpressing1and2)
    {
        ISLEFT = true;
        ISRIGHT = true;
        diffM.Start();
        diffK.Start();
```

```
            objdata = Activator.CreateInstance(typeof(dataClass));
            if (leftandright == 1 | leftandright == 2 | leftandright == 4 | leftandright == 7)
            {
                objdataReadLeft = Activator.CreateInstance(typeof(dataReadClassLeft));
            }
            if (leftandright == 4 | leftandright == 5 | leftandright == 6 | leftandright == 7)
            {
                objdataReadRight = Activator.CreateInstance(typeof(dataReadClassRight));
            }
            if (leftandright == 4 | leftandright == 7)
            {
                objdataReadLeftRight = Activator.CreateInstance(typeof(dataReadClassLeftRight));
            }
            if (leftandright == 2 | leftandright == 3 | leftandright == 5 | leftandright == 7)
            {
                objdataReadWiimote = Activator.CreateInstance(typeof(dataReadClassWiimote));
                taskDReadWiimote = new Task(WiiJoy_thrDReadWiimote);
                taskDReadWiimote.Start();
            }
            try
            {
                Start();
                camenabled = true;
            }
            catch { }
            Thread.Sleep(2000);
            if (leftandright == 2 | leftandright == 3 | leftandright == 5 | leftandright == 7)
            {
                calibrationinit = -aBuffer[4] + 135f;
                stickviewxinit = -aBuffer[16] + 125f;
                stickviewyinit = -aBuffer[17] + 125f;
            }
            if (leftandright == 1 | leftandright == 2 | leftandright == 4 | leftandright == 7)
            {
                stick_rawLeft[0] = report_bufaLeft[6 + (ISLEFT ? 0 : 3)];
                stick_rawLeft[1] = report_bufaLeft[7 + (ISLEFT ? 0 : 3)];
                stick_rawLeft[2] = report_bufaLeft[8 + (ISLEFT ? 0 : 3)];
                stick_calibrationLeft[0] = (UInt16)(stick_rawLeft[0] | ((stick_rawLeft[1] & 0xf) << 8));
                stick_calibrationLeft[1] = (UInt16)((stick_rawLeft[1] >> 4) | (stick_rawLeft[2] << 4));
                acc_gcalibrationLeftX = (int)(avg((Int16)(report_bufLeft[13 + 0 * 12] | ((report_bufLeft[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[13 + 1 * 12] | ((report_bufLeft[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[13 + 2 *
12] | ((report_bufLeft[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationLeftY = (int)(avg((Int16)(report_bufLeft[15 + 0 * 12] | ((report_bufLeft[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[15 + 1 * 12] | ((report_bufLeft[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[15 + 2 *
12] | ((report_bufLeft[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                acc_gcalibrationLeftZ = (int)(avg((Int16)(report_bufLeft[17 + 0 * 12] | ((report_bufLeft[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufLeft[17 + 1 * 12] | ((report_bufLeft[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[17 + 2 *
12] | ((report_bufLeft[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
                gyr_gcalibrationLeftX = (int)(avg((int)((Int16)((report_bufLeft[19 + 0 * 12] | ((report_bufLeft[20 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[19 + 1 * 12] | ((report_bufLeft[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[19 + 2 * 12] | ((report_bufLeft[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationLeftY = (int)(avg((int)((Int16)((report_bufLeft[21 + 0 * 12] | ((report_bufLeft[22 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[21 + 1 * 12] | ((report_bufLeft[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[21 + 2 * 12] | ((report_bufLeft[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
                gyr_gcalibrationLeftZ = (int)(avg((int)((Int16)((report_bufLeft[23 + 0 * 12] | ((report_bufLeft[24 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[23 + 1 * 12] | ((report_bufLeft[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[23 + 2 * 12] | ((report_bufLeft[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
            }
            if (leftandright == 4 | leftandright == 5 | leftandright == 6 | leftandright == 7)
            {
                stick_rawRight[0] = report_bufaRight[6 + (!ISRIGHT ? 0 : 3)];
                stick_rawRight[1] = report_bufaRight[7 + (!ISRIGHT ? 0 : 3)];
                stick_rawRight[2] = report_bufaRight[8 + (!ISRIGHT ? 0 : 3)];
```

```
            stick_calibrationRight[0] = (UInt16)(stick_rawRight[0] | ((stick_rawRight[1] & 0xf) << 8));
            stick_calibrationRight[1] = (UInt16)((stick_rawRight[1] >> 4) | (stick_rawRight[2] << 4));
            acc_gcalibrationRightX = (int)(avg((Int16)(report_bufRight[13 + 0 * 12] | ((report_bufRight[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[13 + 1 * 12] | ((report_bufRight[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[13 +
2 * 12] | ((report_bufRight[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
            acc_gcalibrationRightY = -(int)(avg((Int16)(report_bufRight[15 + 0 * 12] | ((report_bufRight[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[15 + 1 * 12] | ((report_bufRight[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[15 +
2 * 12] | ((report_bufRight[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
            acc_gcalibrationRightZ = -(int)(avg((Int16)(report_bufRight[17 + 0 * 12] | ((report_bufRight[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[17 + 1 * 12] | ((report_bufRight[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[17 +
2 * 12] | ((report_bufRight[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f);
            gyr_gcalibrationRightX = (int)(avg((int)((Int16)((report_bufRight[19 + 0 * 12] | ((report_bufRight[20 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[19 + 1 * 12] | ((report_bufRight[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[19 + 2 * 12] | ((report_bufRight[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
            gyr_gcalibrationRightY = -(int)(avg((int)((Int16)((report_bufRight[21 + 0 * 12] | ((report_bufRight[22 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[21 + 1 * 12] | ((report_bufRight[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[21 + 2 * 12] | ((report_bufRight[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
            gyr_gcalibrationRightZ = -(int)(avg((int)((Int16)((report_bufRight[23 + 0 * 12] | ((report_bufRight[24 + 0 * 12] <<
8) & 0xff00)))), (int)((Int16)((report_bufRight[23 + 1 * 12] | ((report_bufRight[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[23 + 2 * 12] | ((report_bufRight[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f);
          }
          if (leftandright == 4 | leftandright == 7)
          {
            acc_gcalibrationX = (int)(avg((Int16)((report_bufferLeft[13 + 0 * 12] | ((report_bufferLeft[14 + 0 * 12] << 8) &
0xff00)) + (report_bufferRight[13 + 0 * 12] | ((report_bufferRight[14 + 0 * 12] << 8) & 0xff00))) / 2f,
(Int16)((report_bufferLeft[13 + 1 * 12] | ((report_bufferLeft[14 + 1 * 12] << 8) & 0xff00)) + (report_bufferRight[13 + 1 * 12]
| ((report_bufferRight[14 + 1 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[13 + 2 * 12] | ((report_bufferLeft[14 + 2
* 12] << 8) & 0xff00)) + (report_bufferRight[13 + 2 * 12] | ((report_bufferRight[14 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f /
16000f);
            acc_gcalibrationY = (int)(avg((Int16)((report_bufferLeft[15 + 0 * 12] | ((report_bufferLeft[16 + 0 * 12] << 8) &
0xff00)) - (report_bufferRight[15 + 0 * 12] | ((report_bufferRight[16 + 0 * 12] << 8) & 0xff00))) / 2f,
(Int16)((report_bufferLeft[15 + 1 * 12] | ((report_bufferLeft[16 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[15 + 1 * 12] |
((report_bufferRight[16 + 1 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[15 + 2 * 12] | ((report_bufferLeft[16 + 2 *
12] << 8) & 0xff00)) - (report_bufferRight[15 + 2 * 12] | ((report_bufferRight[16 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f /
16000f);
            acc_gcalibrationZ = (int)(avg((Int16)((report_bufferLeft[17 + 0 * 12] | ((report_bufferLeft[18 + 0 * 12] << 8) &
0xff00)) - (report_bufferRight[17 + 0 * 12] | ((report_bufferRight[18 + 0 * 12] << 8) & 0xff00))) / 2f,
(Int16)((report_bufferLeft[17 + 1 * 12] | ((report_bufferLeft[18 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[17 + 1 * 12] |
((report_bufferRight[18 + 1 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[17 + 2 * 12] | ((report_bufferLeft[18 + 2 *
12] << 8) & 0xff00)) - (report_bufferRight[17 + 2 * 12] | ((report_bufferRight[18 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f /
16000f);
            gyr_gcalibrationX = (int)(avg((int)((Int16)((report_bufferLeft[19 + 0 * 12] | ((report_bufferLeft[20 + 0 * 12] << 8)
& 0xff00)) + (report_bufferRight[19 + 0 * 12] | ((report_bufferRight[20 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[19 + 1 * 12] | ((report_bufferLeft[20 + 1 * 12] << 8) & 0xff00)) + (report_bufferRight[19 + 1 *
12] | ((report_bufferRight[20 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[19 + 2 * 12] |
((report_bufferLeft[20 + 2 * 12] << 8) & 0xff00)) + (report_bufferRight[19 + 2 * 12] | ((report_bufferRight[20 + 2 * 12] << 8)
& 0xff00))) / (2f)))) * (1.0f / 16000f);
            gyr_gcalibrationY = (int)(avg((int)((Int16)((report_bufferLeft[21 + 0 * 12] | ((report_bufferLeft[22 + 0 * 12] << 8)
& 0xff00)) - (report_bufferRight[21 + 0 * 12] | ((report_bufferRight[22 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[21 + 1 * 12] | ((report_bufferLeft[22 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[21 + 1 *
12] | ((report_bufferRight[22 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[21 + 2 * 12] |
((report_bufferLeft[22 + 2 * 12] << 8) & 0xff00)) - (report_bufferRight[21 + 2 * 12] | ((report_bufferRight[22 + 2 * 12] << 8)
& 0xff00))) / (2f)))) * (1.0f / 16000f);
            gyr_gcalibrationZ = (int)(avg((int)((Int16)((report_bufferLeft[23 + 0 * 12] | ((report_bufferLeft[24 + 0 * 12] << 8)
& 0xff00)) - (report_bufferRight[23 + 0 * 12] | ((report_bufferRight[24 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[23 + 1 * 12] | ((report_bufferLeft[24 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[23 + 1 *
12] | ((report_bufferRight[24 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[23 + 2 * 12] |
((report_bufferLeft[24 + 2 * 12] << 8) & 0xff00)) - (report_bufferRight[23 + 2 * 12] | ((report_bufferRight[24 + 2 * 12] << 8)
& 0xff00))) / (2f)))) * (1.0f / 16000f);
          }
          this.BackColor = System.Drawing.Color.WhiteSmoke;
        }
      }
```

```
    private double avg(double val1, double val2, double val3)
    {
        return (new double[] { val1, val2, val3 }).Average();
    }
    private async void WiiJoy_thrM()
    {
        try
        {
            mouseprogram.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod, null, mouseobj, new
object[] { 0f, 0f, 0f, 0f, 0f, false, false, 0f, 0f, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f, 0f, 0f,
0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, false, false, false, false, false, false, false, false, 0f,
0f, false, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f });
        }
        catch { }
        for (; ; )
        {
            if (endinvoke | !Getstate)
            {
                try
                {
                    mouseprogram.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod, null, mouseobj,
new object[] { 0f, 0f, 0f, 0f, 0f, false, false, 0f, 0f, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f,
0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, false, false, false, false, false, false, false, false,
false, 0f, 0f, false, false, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f });
                }
                catch { }
                return;
            }
            watchM2 = (double)diffM.ElapsedTicks / (Stopwatch.Frequency / (1000L * 1000L));
            watchM = (watchM2 - watchM1) / 1000f;
            watchM1 = watchM2;
            await asynctaskM();
        }
    }
    private void WiiJoy_thrK()
    {
        try
        {
            keyboardprogram.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod, null, keyboardobj,
new object[] { 0f, 0f, 0f, 0f, 0f, false, false, 0f, 0f, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f,
0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, false, false, false, false, false, false, false, false,
false, 0f, 0f, false, false, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f });
        }
        catch { }
        for (; ; )
        {
            if (endinvoke | !Getstate)
            {
                try
                {
                    keyboardprogram.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod, null,
keyboardobj, new object[] { 0f, 0f, 0f, 0f, 0f, false, false, 0f, 0f, false, false, false, false, false, false, false, false, false, false,
false, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, false, false, false, false , false, false, false,
false, false, false, false, 0f, 0f, false, false, false, false, false, false, false, false, false, false, false, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f, 0f,
0f, 0f, 0f });
                }
                catch { }
                return;
            }
            watchK2 = (double)diffK.ElapsedTicks / (Stopwatch.Frequency / (1000L * 1000L));
            watchK = (watchK2 - watchK1) / 1000f;
            watchK1 = watchK2;
            asynctaskK();
```

```
        }
    }
    public async Task asynctaskM()
    {
        try
        {
            mouseprogram.InvokeMember("Main", BindingFlags.Default | BindingFlags.InvokeMethod, null, mouseobj, new
object[] { mWSNunchuckStateRawJoystickX, mWSNunchuckStateRawJoystickY, mWSNunchuckStateRawValuesX,
mWSNunchuckStateRawValuesY, mWSNunchuckStateRawValuesZ, mWSNunchuckStateC, mWSNunchuckStateZ,
mWSButtonStateIRX, mWSButtonStateIRY, mWSButtonStateA, mWSButtonStateB, mWSButtonStateMinus,
mWSButtonStateHome, mWSButtonStatePlus, mWSButtonStateOne, mWSButtonStateTwo, mWSButtonStateUp,
mWSButtonStateDown, mWSButtonStateLeft, mWSButtonStateRight, mWSRawValuesX, mWSRawValuesY,
mWSRawValuesZ, EulerAngles.X, EulerAngles.Y, EulerAngles.Z, DirectAngles.X, DirectAngles.Y, DirectAngles.Z, camx, camy,
EulerAnglesLeft.X, EulerAnglesLeft.Y, EulerAnglesLeft.Z, DirectAnglesLeft.X, DirectAnglesLeft.Y, DirectAnglesLeft.Z,
EulerAnglesRight.X, EulerAnglesRight.Y, EulerAnglesRight.Z, DirectAnglesRight.X, DirectAnglesRight.Y, DirectAnglesRight.Z,
LeftButtonSHOULDER_1, LeftButtonMINUS, LeftButtonCAPTURE, LeftButtonDPAD_UP, LeftButtonDPAD_LEFT,
LeftButtonDPAD_DOWN, LeftButtonDPAD_RIGHT, LeftButtonSTICK, RightButtonDPAD_DOWN, LeftButtonSL, LeftButtonSR,
GetStickLeft()[0], GetStickLeft()[1], RightButtonPLUS, RightButtonDPAD_RIGHT, RightButtonHOME,
RightButtonSHOULDER_1, RightButtonDPAD_LEFT, RightButtonDPAD_UP, RightButtonSTICK, RightButtonSL, RightButtonSR,
RightButtonSHOULDER_2, LeftButtonSHOULDER_2, GetStickRight()[0], GetStickRight()[1], GetAccel().X, GetAccel().Y,
GetAccel().Z, GetAccelRight().X, GetAccelRight().Y, GetAccelRight().Z, GetAccelLeft().X, GetAccelLeft().Y, GetAccelLeft().Z,
watchM });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public void asynctaskK()
    {
        try
        {
            keyboardprogram.InvokeMember("Main", BindingFlags.Default | BindingFlags.InvokeMethod, null, keyboardobj,
new object[] { mWSNunchuckStateRawJoystickX, mWSNunchuckStateRawJoystickY, mWSNunchuckStateRawValuesX,
mWSNunchuckStateRawValuesY, mWSNunchuckStateRawValuesZ, mWSNunchuckStateC, mWSNunchuckStateZ,
mWSButtonStateIRX, mWSButtonStateIRY, mWSButtonStateA, mWSButtonStateB, mWSButtonStateMinus,
mWSButtonStateHome, mWSButtonStatePlus, mWSButtonStateOne, mWSButtonStateTwo, mWSButtonStateUp,
mWSButtonStateDown, mWSButtonStateLeft, mWSButtonStateRight, mWSRawValuesX, mWSRawValuesY,
mWSRawValuesZ, EulerAngles.X, EulerAngles.Y, EulerAngles.Z, DirectAngles.X, DirectAngles.Y, DirectAngles.Z, camx, camy,
EulerAnglesLeft.X, EulerAnglesLeft.Y, EulerAnglesLeft.Z, DirectAnglesLeft.X, DirectAnglesLeft.Y, DirectAnglesLeft.Z,
EulerAnglesRight.X, EulerAnglesRight.Y, EulerAnglesRight.Z, DirectAnglesRight.X, DirectAnglesRight.Y, DirectAnglesRight.Z,
LeftButtonSHOULDER_1, LeftButtonMINUS, LeftButtonCAPTURE, LeftButtonDPAD_UP, LeftButtonDPAD_LEFT,
LeftButtonDPAD_DOWN, LeftButtonDPAD_RIGHT, LeftButtonSTICK, RightButtonDPAD_DOWN, LeftButtonSL, LeftButtonSR,
GetStickLeft()[0], GetStickLeft()[1], RightButtonPLUS, RightButtonDPAD_RIGHT, RightButtonHOME,
RightButtonSHOULDER_1, RightButtonDPAD_LEFT, RightButtonDPAD_UP, RightButtonSTICK, RightButtonSL, RightButtonSR,
RightButtonSHOULDER_2, LeftButtonSHOULDER_2, GetStickRight()[0], GetStickRight()[1], GetAccel().X, GetAccel().Y,
GetAccel().Z, GetAccelRight().X, GetAccelRight().Y, GetAccelRight().Z, GetAccelLeft().X, GetAccelLeft().Y, GetAccelLeft().Z,
watchK });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
    }
    public void Start()
    {
        CaptureDevice = new FilterInfoCollection(FilterCategory.VideoInputDevice);
        FinalFrame = new VideoCaptureDevice(CaptureDevice[0].MonikerString);
        videoCapabilities = FinalFrame.VideoCapabilities;
        FinalFrame.VideoResolution = videoCapabilities[videoCapabilities.Length - 1];
        FinalFrame.SetCameraProperty(CameraControlProperty.Zoom, 0, CameraControlFlags.Manual);
        FinalFrame.SetCameraProperty(CameraControlProperty.Focus, 0, CameraControlFlags.Manual);
        FinalFrame.SetCameraProperty(CameraControlProperty.Exposure, 0, CameraControlFlags.Manual);
        FinalFrame.SetCameraProperty(CameraControlProperty.Iris, 0, CameraControlFlags.Manual);
        FinalFrame.SetCameraProperty(CameraControlProperty.Pan, 0, CameraControlFlags.Manual);
        FinalFrame.NewFrame += FinalFrame_NewFrame;
```

```
        FinalFrame.Start();
    }
    void FinalFrame_NewFrame(object sender, NewFrameEventArgs eventArgs)
    {
        typeof(dataClass).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null, objdata, new
object[] { eventArgs });
    }
    public class dataClass
    {
        public void Data(NewFrameEventArgs eventArgs)
        {
            ReceiveRawRight(eventArgs);
        }
        private void ReceiveRawRight(NewFrameEventArgs eventArgs)
        {
            img = (Bitmap)eventArgs.Frame.Clone();
            brightnessfilter.ApplyInPlace(img);
            colorfilter.Red = new IntRange(0, 255);
            colorfilter.Green = new IntRange(205, 255);
            colorfilter.Blue = new IntRange(205, 255);
            colorfilter.ApplyInPlace(img);
            brightnessfilter.ApplyInPlace(img);
            euclideanfilter.CenterColor = new RGB(255, 255, 255);
            euclideanfilter.Radius = 175;
            euclideanfilter.ApplyInPlace(img);
            blobCounter.ProcessImage(img);
            blobs = blobCounter.GetObjectsInformation();
            for (int i = 0; i < blobs.Length; i++)
            {
                shapeChecker.RelativeDistortionLimit = 100f;
                shapeChecker.MinAcceptableDistortion = 20f;
                if (shapeChecker.IsCircle(blobCounter.GetBlobsEdgePoints(blobs[i])))
                {
                    backpointX = blobs[0].CenterOfGravity.X;
                    backpointY = blobs[0].CenterOfGravity.Y;
                }
            }
            EditableImg = new Bitmap(img);
            EditableImg.MakeTransparent();
            DrawLines(ref EditableImg, new System.Drawing.Point((int)backpointX, (int)backpointY));
            form.pictureBox1.Image = EditableImg;
            posRightX = backpointX - img.Width / 2;
            posRightY = backpointY - img.Height / 2;
            BackpointAnglesX = posRightX - InitBackpointAnglesX;
            BackpointAnglesY = posRightY - InitBackpointAnglesY;
            if (Setrecenter)
            {
                InitBackpointAnglesX = posRightX;
                InitBackpointAnglesY = posRightY;
            }
            camx = BackpointAnglesX;
            camy = BackpointAnglesY;
        }
        public void DrawLines(ref Bitmap image, System.Drawing.Point p)
        {
            Graphics g = Graphics.FromImage(image);
            Pen p1 = new Pen(Color.Red, 2);
            System.Drawing.Point ph = new System.Drawing.Point(image.Width, p.Y);
            System.Drawing.Point ph2 = new System.Drawing.Point(0, p.Y);
            g.DrawLine(p1, p, ph);
            g.DrawLine(p1, p, ph2);
            ph = new System.Drawing.Point(p.X, 0);
            ph2 = new System.Drawing.Point(p.X, image.Height);
```

```
            g.DrawLine(p1, p, ph);
            g.DrawLine(p1, p, ph2);
            g.Dispose();
        }
    }
    private async void timer1_Tick(object sender, EventArgs e)
    {
        await asynctaskSelection();
        if (leftandright == 1 | leftandright == 2 | leftandright == 4 | leftandright == 7)
            await asynctaskReadClassLeft();
        if (leftandright == 4 | leftandright == 5 | leftandright == 6 | leftandright == 7)
            await asynctaskReadClassRight();
        if (leftandright == 4 | leftandright == 7)
            await asynctaskReadClassLeftRight();
    }
    public async Task asynctaskSelection()
    {
        this[1] = GetAsyncKeyState(System.Windows.Forms.Keys.Decimal);
        if (_Valuechanged[1] & this[1] & !Getstate)
        {
            Getstate = true;
            scriptUpdate();
        }
        else
        {
            if (_Valuechanged[1] & this[1] & Getstate)
            {
                Getstate = false;
                this.BackColor = System.Drawing.Color.Black;
            }
        }
        if (camenabled)
            try
            {
                if (FinalFrame.IsRunning != true)
                    Start();
            }
            catch { }
        Setrecenter = !Getstate | GetAsyncKeyState(System.Windows.Forms.Keys.NumPad0);
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public async Task asynctaskReadClassLeft()
    {
        try
        {
            typeof(dataReadClassLeft).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataReadLeft, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public async Task asynctaskReadClassRight()
    {
        try
        {
            typeof(dataReadClassRight).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataReadRight, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
```

```
    }
    public async Task asynctaskReadClassLeftRight()
    {
        try
        {
            typeof(dataReadClassLeftRight).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataReadLeftRight, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    private void WiiJoy_thrDReadWiimote()
    {
        for (; ; )
        {
            if (endinvoke)
                return;
            typeof(dataReadClassWiimote).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataReadWiimote, new object[] { });
        }
    }
    public class dataReadClassLeft
    {
        public void Data()
        {
            ReceiveRawLeft();
        }
        public Quaternion GetVectoraLeft()
        {
            Vector3 v1 = new Vector3(j_aLeft.X, i_aLeft.X, k_aLeft.X);
            Vector3 v2 = -(new Vector3(j_aLeft.Z, i_aLeft.Z, k_aLeft.Z));
            return QuaternionLookRotationLeft(v1, v2);
        }
        public Quaternion GetVectorbLeft()
        {
            Vector3 v1 = new Vector3(j_bLeft.X, i_bLeft.X, k_bLeft.X);
            Vector3 v2 = -(new Vector3(j_bLeft.Z, i_bLeft.Z, k_bLeft.Z));
            return QuaternionLookRotationLeft(v1, v2);
        }
        public Quaternion GetVectorcLeft()
        {
            Vector3 v1 = new Vector3(j_cLeft.X, i_cLeft.X, k_cLeft.X);
            Vector3 v2 = -(new Vector3(j_cLeft.Z, i_cLeft.Z, k_cLeft.Z));
            return QuaternionLookRotationLeft(v1, v2);
        }
        private static Quaternion QuaternionLookRotationLeft(Vector3 forward, Vector3 up)
        {
            Vector3 vector = Vector3.Normalize(forward);
            Vector3 vector2 = Vector3.Normalize(Vector3.Cross(up, vector));
            Vector3 vector3 = Vector3.Cross(vector, vector2);
            var m00 = vector2.X;
            var m01 = vector2.Y;
            var m02 = vector2.Z;
            var m10 = vector3.X;
            var m11 = vector3.Y;
            var m12 = vector3.Z;
            var m20 = vector.X;
            var m21 = vector.Y;
            var m22 = vector.Z;
            double num8 = (m00 + m11) + m22;
            var quaternion = new Quaternion();
            if (num8 > 0f)
```

```
    {
        var num = (double)Math.Sqrt(num8 + 1f);
        quaternion.W = (float)num * 0.5f;
        num = 0.5f / num;
        quaternion.X = (float)(m12 - m21) * (float)num;
        quaternion.Y = (float)(m20 - m02) * (float)num;
        quaternion.Z = (float)(m01 - m10) * (float)num;
        return quaternion;
    }
    if ((m00 >= m11) && (m00 >= m22))
    {
        var num7 = (double)Math.Sqrt(((1f + m00) - m11) - m22);
        var num4 = 0.5f / num7;
        quaternion.X = 0.5f * (float)num7;
        quaternion.Y = (float)(m01 + m10) * (float)num4;
        quaternion.Z = (float)(m02 + m20) * (float)num4;
        quaternion.W = (float)(m12 - m21) * (float)num4;
        return quaternion;
    }
    if (m11 > m22)
    {
        var num6 = (double)Math.Sqrt(((1f + m11) - m00) - m22);
        var num3 = 0.5f / num6;
        quaternion.X = (float)(m10 + m01) * (float)num3;
        quaternion.Y = 0.5f * (float)num6;
        quaternion.Z = (float)(m21 + m12) * (float)num3;
        quaternion.W = (float)(m20 - m02) * (float)num3;
        return quaternion;
    }
    var num5 = (double)Math.Sqrt(((1f + m22) - m00) - m11);
    var num2 = 0.5f / num5;
    quaternion.X = (float)(m20 + m02) * (float)num2;
    quaternion.Y = (float)(m21 + m12) * (float)num2;
    quaternion.Z = 0.5f * (float)num5;
    quaternion.W = (float)(m01 - m10) * (float)num2;
    return quaternion;
}
public static Vector3 ToEulerAnglesLeft(Quaternion q)
{
    Vector3 pitchYawRoll = new Vector3();
    double sqw = q.W * q.W;
    double sqx = q.X * q.X;
    double sqy = q.Y * q.Y;
    double sqz = q.Z * q.Z;
    double unit = sqx + sqy + sqz + sqw;
    double test = q.X * q.Y + q.Z * q.W;
    if (test > 0.4999f * unit)                  // 0.4999f OR 0.5f - EPSILON
    {
        pitchYawRoll.Y = 2f * (float)Math.Atan2(q.X, q.W);  // Yaw
        pitchYawRoll.X = (float)Math.PI * 0.5f;             // Pitch
        pitchYawRoll.Z = 0f;                    // Roll
        return pitchYawRoll;
    }
    else if (test < -0.4999f * unit)            // -0.4999f OR -0.5f + EPSILON
    {
        pitchYawRoll.Y = -2f * (float)Math.Atan2(q.X, q.W); // Yaw
        pitchYawRoll.X = -(float)Math.PI * 0.5f;            // Pitch
        pitchYawRoll.Z = 0f;                    // Roll
        return pitchYawRoll;
    }
    else
    {
        pitchYawRoll.Y = (float)Math.Atan2(2f * q.Y * q.W - 2f * q.X * q.Z, sqx - sqy - sqz + sqw);      // Yaw
```

```
            pitchYawRoll.X = (float)Math.Asin(2f * test / unit);                                    // Pitch
            pitchYawRoll.Z = (float)Math.Atan2(2f * q.X * q.W - 2f * q.Y * q.Z, -sqx + sqy - sqz + sqw);     // Roll
        }
        return pitchYawRoll;
    }
    private void ReceiveRawLeft()
    {
        if (!endinvoke)
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataLeft", System.IO.File Mode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.ReadAsync(report_bufaLeft, 0, 49);
            };
            report_bufLeft = report_bufaLeft;
            ProcessButtonsAndStickLeft();
            ExtractIMUValuesLeft();
            System.Threading.Thread.Sleep(10);
        }
    }
    private void ProcessButtonsAndStickLeft()
    {
        LeftButtonSHOULDER_1 = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x40) != 0;
        LeftButtonSHOULDER_2 = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x80) != 0;
        LeftButtonSR = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x10) != 0;
        LeftButtonSL = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & 0x20) != 0;
        LeftButtonDPAD_DOWN = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x01 : 0x04)) != 0;
        LeftButtonDPAD_RIGHT = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x04 : 0x08)) != 0;
        LeftButtonDPAD_UP = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x02 : 0x02)) != 0;
        LeftButtonDPAD_LEFT = (report_bufLeft[3 + (ISLEFT ? 2 : 0)] & (ISLEFT ? 0x08 : 0x01)) != 0;
        LeftButtonMINUS = ((report_bufLeft[4] & 0x01) != 0);
        LeftButtonCAPTURE = ((report_bufLeft[4] & 0x20) != 0);
        LeftButtonSTICK = ((report_bufLeft[4] & (ISLEFT ? 0x08 : 0x04)) != 0);
        stick_rawLeft[0] = report_bufLeft[6 + (ISLEFT ? 0 : 3)];
        stick_rawLeft[1] = report_bufLeft[7 + (ISLEFT ? 0 : 3)];
        stick_rawLeft[2] = report_bufLeft[8 + (ISLEFT ? 0 : 3)];
        stick_precalLeft[0] = (UInt16)(stick_rawLeft[0] | ((stick_rawLeft[1] & 0xf) << 8));
        stick_precalLeft[1] = (UInt16)((stick_rawLeft[1] >> 4) | (stick_rawLeft[2] << 4));
        stickLeft = CenterSticksLeft(stick_precalLeft);
    }
    private void ExtractIMUValuesLeft()
    {
        acc_gLeft.X = (int)(averageLeft((Int16)(report_bufLeft[13 + 0 * 12] | ((report_bufLeft[14 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[13 + 1 * 12] | ((report_bufLeft[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[13 + 2 * 12] |
((report_bufLeft[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftX;
        acc_gLeft.Y = (int)(averageLeft((Int16)(report_bufLeft[15 + 0 * 12] | ((report_bufLeft[16 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[15 + 1 * 12] | ((report_bufLeft[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[15 + 2 * 12] |
((report_bufLeft[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftY;
        acc_gLeft.Z = (int)(averageLeft((Int16)(report_bufLeft[17 + 0 * 12] | ((report_bufLeft[18 + 0 * 12] << 8) & 0xff00)),
(Int16)(report_bufLeft[17 + 1 * 12] | ((report_bufLeft[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufLeft[17 + 2 * 12] |
((report_bufLeft[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationLeftZ;
        gyr_gLeft.X = (int)(averageLeft((int)((Int16)((report_bufLeft[19 + 0 * 12] | ((report_bufLeft[20 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[19 + 1 * 12] | ((report_bufLeft[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[19 + 2 * 12] | ((report_bufLeft[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftX;
        gyr_gLeft.Y = (int)(averageLeft((int)((Int16)((report_bufLeft[21 + 0 * 12] | ((report_bufLeft[22 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[21 + 1 * 12] | ((report_bufLeft[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[21 + 2 * 12] | ((report_bufLeft[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftY;
        gyr_gLeft.Z = (int)(averageLeft((int)((Int16)((report_bufLeft[23 + 0 * 12] | ((report_bufLeft[24 + 0 * 12] << 8) &
0xff00)))), (int)((Int16)((report_bufLeft[23 + 1 * 12] | ((report_bufLeft[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufLeft[23 + 2 * 12] | ((report_bufLeft[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationLeftZ;
```

```
            acc_rLeft.X = ((int)(acc_gLeft.X * roundLeft)) / roundLeft;
            acc_rLeft.Y = ((int)(acc_gLeft.Y * roundLeft)) / roundLeft;
            acc_rLeft.Z = ((int)(acc_gLeft.Z * roundLeft)) / roundLeft;
            gyr_rLeft.X = ((int)(gyr_gLeft.X * roundLeft)) / roundLeft;
            gyr_rLeft.Y = ((int)(gyr_gLeft.Y * roundLeft)) / roundLeft;
            gyr_rLeft.Z = ((int)(gyr_gLeft.Z * roundLeft)) / roundLeft;
            if (!Getstate)
               InitDirectAnglesLeft =acc_rLeft;
            DirectAnglesLeft =acc_rLeft - InitDirectAnglesLeft;
            if (((int)(i_cLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_cLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_cLeft.Z *
roundLeft)) / roundLeft == 0f))
               i_cLeft = new Vector3(1, 0, 0);
            if (((int)(j_cLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_cLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_cLeft.Z
* roundLeft)) / roundLeft == 0f))
               j_cLeft = new Vector3(0, 1, 0);
            if (((int)(k_cLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_cLeft.X * roundLeft)) / roundLeft == 0f & ((int)(k_cLeft.Y
* roundLeft)) / roundLeft == 0f))
               k_cLeft = new Vector3(0, 0, 1);
            if (((int)(i_bLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_bLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_bLeft.Z
* roundLeft)) / roundLeft == 0f))
               i_bLeft = new Vector3(1, 0, 0);
            if (((int)(j_bLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_bLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_bLeft.Z
* roundLeft)) / roundLeft == 0f))
               j_bLeft = new Vector3(0, 1, 0);
            if (((int)(k_bLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_bLeft.X * roundLeft)) / roundLeft == 0f &
((int)(k_bLeft.Y * roundLeft)) / roundLeft == 0f))
               k_bLeft = new Vector3(0, 0, 1);
            if (((int)(i_aLeft.X * roundLeft)) / roundLeft == 1f | (((int)(i_aLeft.Y * roundLeft)) / roundLeft == 0f & ((int)(i_aLeft.Z
* roundLeft)) / roundLeft == 0f))
               i_aLeft = new Vector3(1, 0, 0);
            if (((int)(j_aLeft.Y * roundLeft)) / roundLeft == 1f | (((int)(j_aLeft.X * roundLeft)) / roundLeft == 0f & ((int)(j_aLeft.Z
* roundLeft)) / roundLeft == 0f))
               j_aLeft = new Vector3(0, 1, 0);
            if (((int)(k_aLeft.Z * roundLeft)) / roundLeft == 1f | (((int)(k_aLeft.X * roundLeft)) / roundLeft == 0f &
((int)(k_aLeft.Y * roundLeft)) / roundLeft == 0f))
               k_aLeft = new Vector3(0, 0, 1);
            if (((int)(EulerAnglescLeft.Y * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.Y * roundLeft)) / roundLeft ==
0f)
            {
               i_cLeft = new Vector3(1, 0, 0);
               j_cLeft = new Vector3(0, 1, 0);
               k_cLeft = new Vector3(0, 0, 1);
               InitEulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft());
            }
            if (((int)(EulerAnglesbLeft.X * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.X * roundLeft)) / roundLeft ==
0f)
            {
               i_bLeft = new Vector3(1, 0, 0);
               j_bLeft = new Vector3(0, 1, 0);
               k_bLeft = new Vector3(0, 0, 1);
               InitEulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft());
            }
            if (((int)(EulerAnglesaLeft.Z * roundLeft)) / roundLeft == 0f | ((int)(EulerAnglesLeft.Z * roundLeft)) / roundLeft ==
0f)
            {
               i_aLeft = new Vector3(1, 0, 0);
               j_aLeft = new Vector3(0, 1, 0);
               k_aLeft = new Vector3(0, 0, 1);
               InitEulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft());
            }
            i_cLeft = new Vector3(1, 0, 0);
            j_cLeft.X = 0f;
            k_cLeft.X = 0f;
```

```
        k_cLeft.Y = 0f;
        k_cLeft.Z = 1f;
        k_bLeft = new Vector3(0, 0, 1);
        i_bLeft.Z = 0f;
        j_bLeft.Z = 0f;
        j_aLeft = new Vector3(0, 1, 0);
        i_aLeft.Y = 0f;
        k_aLeft.X = 0f;
        k_aLeft.Y = 0f;
        if (!Getstate)
        {
            i_cLeft = new Vector3(1, 0, 0);
            j_cLeft = new Vector3(0, 1, 0);
            k_cLeft = new Vector3(0, 0, 1);
            InitEulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft());
            i_bLeft = new Vector3(1, 0, 0);
            j_bLeft = new Vector3(0, 1, 0);
            k_bLeft = new Vector3(0, 0, 1);
            InitEulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft());
            i_aLeft = new Vector3(1, 0, 0);
            j_aLeft = new Vector3(0, 1, 0);
            k_aLeft = new Vector3(0, 0, 1);
            InitEulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft());
        }
        i_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_cLeft);
        j_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_cLeft);
        k_cLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_cLeft);
        i_cLeft = Vector3.Normalize(i_cLeft - Vector3.Dot(i_cLeft, j_cLeft) * 0.5f * j_cLeft);
        j_cLeft = Vector3.Normalize(j_cLeft - Vector3.Dot(i_cLeft, j_cLeft) * 0.5f * i_cLeft);
        k_cLeft = Vector3.Cross(i_cLeft, j_cLeft);
        EulerAnglescLeft = ToEulerAnglesLeft(GetVectorcLeft()) - InitEulerAnglescLeft;
        i_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_bLeft);
        j_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_bLeft);
        k_bLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_bLeft);
        i_bLeft = Vector3.Normalize(i_bLeft - Vector3.Dot(i_bLeft, j_bLeft) * 0.5f * j_bLeft);
        j_bLeft = Vector3.Normalize(j_bLeft - Vector3.Dot(i_bLeft, j_bLeft) * 0.5f * i_bLeft);
        k_bLeft = Vector3.Cross(i_bLeft, j_bLeft);
        EulerAnglesbLeft = ToEulerAnglesLeft(GetVectorbLeft()) - InitEulerAnglesbLeft;
        i_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, i_aLeft);
        j_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, j_aLeft);
        k_aLeft += Vector3.Cross(Vector3.Negate(gyr_rLeft) * 0.04f, k_aLeft);
        i_aLeft = Vector3.Normalize(i_aLeft - Vector3.Dot(i_aLeft, j_aLeft) * 0.5f * j_aLeft);
        j_aLeft = Vector3.Normalize(j_aLeft - Vector3.Dot(i_aLeft, j_aLeft) * 0.5f * i_aLeft);
        k_aLeft = Vector3.Cross(i_aLeft, j_aLeft);
        EulerAnglesaLeft = ToEulerAnglesLeft(GetVectoraLeft()) - InitEulerAnglesaLeft;
        EulerAnglesLeft = new Vector3(EulerAnglesbLeft.X, EulerAnglescLeft.Y, EulerAnglesaLeft.Z);
    }
    private double averageLeft(double val1, double val2, double val3)
    {
        arrayLeft = new double[] { val1, val2, val3 };
        return arrayLeft.Average();
    }
    private double[] CenterSticksLeft(UInt16[] vals)
    {
        double[] s = { 0, 0 };
        s[0] = ((int)((vals[0] - stick_calibrationLeft[0]) / 100f)) / 13f;
        s[1] = ((int)((vals[1] - stick_calibrationLeft[1]) / 100f)) / 13f;
        return s;
    }
}
public class dataReadClassRight
{
    public void Data()
```

```
{
    ReceiveRawRight();
}
public Quaternion GetVectoraRight()
{
    Vector3 v1 = new Vector3(j_aRight.X, i_aRight.X, k_aRight.X);
    Vector3 v2 = -(new Vector3(j_aRight.Z, i_aRight.Z, k_aRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
public Quaternion GetVectorbRight()
{
    Vector3 v1 = new Vector3(j_bRight.X, i_bRight.X, k_bRight.X);
    Vector3 v2 = -(new Vector3(j_bRight.Z, i_bRight.Z, k_bRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
public Quaternion GetVectorcRight()
{
    Vector3 v1 = new Vector3(j_cRight.X, i_cRight.X, k_cRight.X);
    Vector3 v2 = -(new Vector3(j_cRight.Z, i_cRight.Z, k_cRight.Z));
    return QuaternionLookRotationRight(v1, v2);
}
private static Quaternion QuaternionLookRotationRight(Vector3 forward, Vector3 up)
{
    Vector3 vector = Vector3.Normalize(forward);
    Vector3 vector2 = Vector3.Normalize(Vector3.Cross(up, vector));
    Vector3 vector3 = Vector3.Cross(vector, vector2);
    var m00 = vector2.X;
    var m01 = vector2.Y;
    var m02 = vector2.Z;
    var m10 = vector3.X;
    var m11 = vector3.Y;
    var m12 = vector3.Z;
    var m20 = vector.X;
    var m21 = vector.Y;
    var m22 = vector.Z;
    double num8 = (m00 + m11) + m22;
    var quaternion = new Quaternion();
    if (num8 > 0f)
    {
        var num = (double)Math.Sqrt(num8 + 1f);
        quaternion.W = (float)num * 0.5f;
        num = 0.5f / num;
        quaternion.X = (float)(m12 - m21) * (float)num;
        quaternion.Y = (float)(m20 - m02) * (float)num;
        quaternion.Z = (float)(m01 - m10) * (float)num;
        return quaternion;
    }
    if ((m00 >= m11) && (m00 >= m22))
    {
        var num7 = (double)Math.Sqrt(((1f + m00) - m11) - m22);
        var num4 = 0.5f / num7;
        quaternion.X = 0.5f * (float)num7;
        quaternion.Y = (float)(m01 + m10) * (float)num4;
        quaternion.Z = (float)(m02 + m20) * (float)num4;
        quaternion.W = (float)(m12 - m21) * (float)num4;
        return quaternion;
    }
    if (m11 > m22)
    {
        var num6 = (double)Math.Sqrt(((1f + m11) - m00) - m22);
        var num3 = 0.5f / num6;
        quaternion.X = (float)(m10 + m01) * (float)num3;
        quaternion.Y = 0.5f * (float)num6;
```

```
            quaternion.Z = (float)(m21 + m12) * (float)num3;
            quaternion.W = (float)(m20 - m02) * (float)num3;
            return quaternion;
        }
        var num5 = (double)Math.Sqrt(((1f + m22) - m00) - m11);
        var num2 = 0.5f / num5;
        quaternion.X = (float)(m20 + m02) * (float)num2;
        quaternion.Y = (float)(m21 + m12) * (float)num2;
        quaternion.Z = 0.5f * (float)num5;
        quaternion.W = (float)(m01 - m10) * (float)num2;
        return quaternion;
    }
    public static Vector3 ToEulerAnglesRight(Quaternion q)
    {
        Vector3 pitchYawRoll = new Vector3();
        double sqw = q.W * q.W;
        double sqx = q.X * q.X;
        double sqy = q.Y * q.Y;
        double sqz = q.Z * q.Z;
        double unit = sqx + sqy + sqz + sqw;
        double test = q.X * q.Y + q.Z * q.W;
        if (test > 0.4999f * unit)                // 0.4999f OR 0.5f - EPSILON
        {
            pitchYawRoll.Y = 2f * (float)Math.Atan2(q.X, q.W);  // Yaw
            pitchYawRoll.X = (float)Math.PI * 0.5f;             // Pitch
            pitchYawRoll.Z = 0f;                  // Roll
            return pitchYawRoll;
        }
        else if (test < -0.4999f * unit)          // -0.4999f OR -0.5f + EPSILON
        {
            pitchYawRoll.Y = -2f * (float)Math.Atan2(q.X, q.W); // Yaw
            pitchYawRoll.X = -(float)Math.PI * 0.5f;            // Pitch
            pitchYawRoll.Z = 0f;                  // Roll
            return pitchYawRoll;
        }
        else
        {
            pitchYawRoll.Y = (float)Math.Atan2(2f * q.Y * q.W - 2f * q.X * q.Z, sqx - sqy - sqz + sqw);      // Yaw
            pitchYawRoll.X = (float)Math.Asin(2f * test / unit);                                             // Pitch
            pitchYawRoll.Z = (float)Math.Atan2(2f * q.X * q.W - 2f * q.Y * q.Z, -sqx + sqy - sqz + sqw);     // Roll
        }
        return pitchYawRoll;
    }
    private void ReceiveRawRight()
    {
        if (!endinvoke)
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataRight", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.ReadAsync(report_bufaRight, 0, 49);
            };
            report_bufRight = report_bufaRight;
            ProcessButtonsAndStickRight();
            ExtractIMUValuesRight();
            System.Threading.Thread.Sleep(10);
        }
    }
    public void ProcessButtonsAndStickRight()
    {
        RightButtonSHOULDER_1 = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x40) != 0;
        RightButtonSHOULDER_2 = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x80) != 0;
        RightButtonSR = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x10) != 0;
```

```
            RightButtonSL = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & 0x20) != 0;
            RightButtonDPAD_DOWN = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x01 : 0x04)) != 0;
            RightButtonDPAD_RIGHT = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x04 : 0x08)) != 0;
            RightButtonDPAD_UP = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x02 : 0x02)) != 0;
            RightButtonDPAD_LEFT = (report_bufRight[3 + (!ISRIGHT ? 2 : 0)] & (!ISRIGHT ? 0x08 : 0x01)) != 0;
            RightButtonPLUS = ((report_bufRight[4] & 0x02) != 0);
            RightButtonHOME = ((report_bufRight[4] & 0x10) != 0);
            RightButtonSTICK = ((report_bufRight[4] & (!ISRIGHT ? 0x08 : 0x04)) != 0);
            stick_rawRight[0] = report_bufRight[6 + (!ISRIGHT ? 0 : 3)];
            stick_rawRight[1] = report_bufRight[7 + (!ISRIGHT ? 0 : 3)];
            stick_rawRight[2] = report_bufRight[8 + (!ISRIGHT ? 0 : 3)];
            stick_precalRight[0] = (UInt16)(stick_rawRight[0] | ((stick_rawRight[1] & 0xf) << 8));
            stick_precalRight[1] = (UInt16)((stick_rawRight[1] >> 4) | (stick_rawRight[2] << 4));
            stickRight = CenterSticksRight(stick_precalRight);
        }
        private void ExtractIMUValuesRight()
        {
            acc_gRight.X = (int)(averageRight((Int16)(report_bufRight[13 + 0 * 12] | ((report_bufRight[14 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[13 + 1 * 12] | ((report_bufRight[14 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[13 +
2 * 12] | ((report_bufRight[14 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightX;
            acc_gRight.Y = -(int)(averageRight((Int16)(report_bufRight[15 + 0 * 12] | ((report_bufRight[16 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[15 + 1 * 12] | ((report_bufRight[16 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[15 +
2 * 12] | ((report_bufRight[16 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightY;
            acc_gRight.Z = -(int)(averageRight((Int16)(report_bufRight[17 + 0 * 12] | ((report_bufRight[18 + 0 * 12] << 8) &
0xff00)), (Int16)(report_bufRight[17 + 1 * 12] | ((report_bufRight[18 + 1 * 12] << 8) & 0xff00)), (Int16)(report_bufRight[17 +
2 * 12] | ((report_bufRight[18 + 2 * 12] << 8) & 0xff00)))) * (1.0f / 16000f) - acc_gcalibrationRightZ;
            gyr_gRight.X = (int)(averageRight((int)((Int16)((report_bufRight[19 + 0 * 12] | ((report_bufRight[20 + 0 * 12] << 8)
& 0xff00)))), (int)((Int16)((report_bufRight[19 + 1 * 12] | ((report_bufRight[20 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[19 + 2 * 12] | ((report_bufRight[20 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightX;
            gyr_gRight.Y = -(int)(averageRight((int)((Int16)((report_bufRight[21 + 0 * 12] | ((report_bufRight[22 + 0 * 12] << 8)
& 0xff00)))), (int)((Int16)((report_bufRight[21 + 1 * 12] | ((report_bufRight[22 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[21 + 2 * 12] | ((report_bufRight[22 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightY;
            gyr_gRight.Z = -(int)(averageRight((int)((Int16)((report_bufRight[23 + 0 * 12] | ((report_bufRight[24 + 0 * 12] << 8)
& 0xff00)))), (int)((Int16)((report_bufRight[23 + 1 * 12] | ((report_bufRight[24 + 1 * 12] << 8) & 0xff00)))),
(int)((Int16)((report_bufRight[23 + 2 * 12] | ((report_bufRight[24 + 2 * 12] << 8) & 0xff00)))))) * (1.0f / 16000f) -
gyr_gcalibrationRightZ;
            acc_rRight.X = ((int)(acc_gRight.X * roundRight)) / roundRight;
            acc_rRight.Y = ((int)(acc_gRight.Y * roundRight)) / roundRight;
            acc_rRight.Z = ((int)(acc_gRight.Z * roundRight)) / roundRight;
            gyr_rRight.X = ((int)(gyr_gRight.X * roundRight)) / roundRight;
            gyr_rRight.Y = ((int)(gyr_gRight.Y * roundRight)) / roundRight;
            gyr_rRight.Z = ((int)(gyr_gRight.Z * roundRight)) / roundRight;
            if (!Getstate)
                InitDirectAnglesRight = acc_rRight;
            DirectAnglesRight = acc_rRight - InitDirectAnglesRight;
            if (((int)(i_cRight.X * roundRight)) / roundRight == 1f | (((int)(i_cRight.Y * roundRight)) / roundRight == 0f &
((int)(i_cRight.Z * roundRight)) / roundRight == 0f))
                i_cRight = new Vector3(1, 0, 0);
            if (((int)(j_cRight.Y * roundRight)) / roundRight == 1f | (((int)(j_cRight.X * roundRight)) / roundRight == 0f &
((int)(j_cRight.Z * roundRight)) / roundRight == 0f))
                j_cRight = new Vector3(0, 1, 0);
            if (((int)(k_cRight.Z * roundRight)) / roundRight == 1f | (((int)(k_cRight.X * roundRight)) / roundRight == 0f &
((int)(k_cRight.Y * roundRight)) / roundRight == 0f))
                k_cRight = new Vector3(0, 0, 1);
            if (((int)(i_bRight.X * roundRight)) / roundRight == 1f | (((int)(i_bRight.Y * roundRight)) / roundRight == 0f &
((int)(i_bRight.Z * roundRight)) / roundRight == 0f))
                i_bRight = new Vector3(1, 0, 0);
            if (((int)(j_bRight.Y * roundRight)) / roundRight == 1f | (((int)(j_bRight.X * roundRight)) / roundRight == 0f &
((int)(j_bRight.Z * roundRight)) / roundRight == 0f))
                j_bRight = new Vector3(0, 1, 0);
            if (((int)(k_bRight.Z * roundRight)) / roundRight == 1f | (((int)(k_bRight.X * roundRight)) / roundRight == 0f &
```

```
((int)(k_bRight.Y * roundRight)) / roundRight == 0f))
            k_bRight = new Vector3(0, 0, 1);
        if (((int)(i_aRight.X * roundRight)) / roundRight == 1f | (((int)(i_aRight.Y * roundRight)) / roundRight == 0f &
((int)(i_aRight.Z * roundRight)) / roundRight == 0f))
            i_aRight = new Vector3(1, 0, 0);
        if (((int)(j_aRight.Y * roundRight)) / roundRight == 1f | (((int)(j_aRight.X * roundRight)) / roundRight == 0f &
((int)(j_aRight.Z * roundRight)) / roundRight == 0f))
            j_aRight = new Vector3(0, 1, 0);
        if (((int)(k_aRight.Z * roundRight)) / roundRight == 1f | (((int)(k_aRight.X * roundRight)) / roundRight == 0f &
((int)(k_aRight.Y * roundRight)) / roundRight == 0f))
            k_aRight = new Vector3(0, 0, 1);
        if (((int)(EulerAnglescRight.Y * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.Y * roundRight)) /
roundRight == 0f)
        {
            i_cRight = new Vector3(1, 0, 0);
            j_cRight = new Vector3(0, 1, 0);
            k_cRight = new Vector3(0, 0, 1);
            InitEulerAnglescRight = ToEulerAnglesRight(GetVectorcRight());
        }
        if (((int)(EulerAnglesbRight.X * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.X * roundRight)) /
roundRight == 0f)
        {
            i_bRight = new Vector3(1, 0, 0);
            j_bRight = new Vector3(0, 1, 0);
            k_bRight = new Vector3(0, 0, 1);
            InitEulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight());
        }
        if (((int)(EulerAnglesaRight.Z * roundRight)) / roundRight == 0f | ((int)(EulerAnglesRight.Z * roundRight)) /
roundRight == 0f)
        {
            i_aRight = new Vector3(1, 0, 0);
            j_aRight = new Vector3(0, 1, 0);
            k_aRight = new Vector3(0, 0, 1);
            InitEulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight());
        }
        i_cRight = new Vector3(1, 0, 0);
        j_cRight.X = 0f;
        k_cRight.X = 0f;
        k_cRight.Y = 0f;
        k_cRight.Z = 1f;
        k_bRight = new Vector3(0, 0, 1);
        i_bRight.Z = 0f;
        j_bRight.Z = 0f;
        j_aRight = new Vector3(0, 1, 0);
        i_aRight.Y = 0f;
        k_aRight.X = 0f;
        k_aRight.Y = 0f;
        if (!Getstate)
        {
            i_cRight = new Vector3(1, 0, 0);
            j_cRight = new Vector3(0, 1, 0);
            k_cRight = new Vector3(0, 0, 1);
            InitEulerAnglescRight = ToEulerAnglesRight(GetVectorcRight());
            i_bRight = new Vector3(1, 0, 0);
            j_bRight = new Vector3(0, 1, 0);
            k_bRight = new Vector3(0, 0, 1);
            InitEulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight());
            i_aRight = new Vector3(1, 0, 0);
            j_aRight = new Vector3(0, 1, 0);
            k_aRight = new Vector3(0, 0, 1);
            InitEulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight());
        }
        i_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_cRight);
```

```
        j_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_cRight);
        k_cRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_cRight);
        i_cRight = Vector3.Normalize(i_cRight - Vector3.Dot(i_cRight, j_cRight) * 0.5f * j_cRight);
        j_cRight = Vector3.Normalize(j_cRight - Vector3.Dot(i_cRight, j_cRight) * 0.5f * i_cRight);
        k_cRight = Vector3.Cross(i_cRight, j_cRight);
        EulerAnglescRight = ToEulerAnglesRight(GetVectorcRight()) - InitEulerAnglescRight;
        i_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_bRight);
        j_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_bRight);
        k_bRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_bRight);
        i_bRight = Vector3.Normalize(i_bRight - Vector3.Dot(i_bRight, j_bRight) * 0.5f * j_bRight);
        j_bRight = Vector3.Normalize(j_bRight - Vector3.Dot(i_bRight, j_bRight) * 0.5f * i_bRight);
        k_bRight = Vector3.Cross(i_bRight, j_bRight);
        EulerAnglesbRight = ToEulerAnglesRight(GetVectorbRight()) - InitEulerAnglesbRight;
        i_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, i_aRight);
        j_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, j_aRight);
        k_aRight += Vector3.Cross(Vector3.Negate(gyr_rRight) * 0.04f, k_aRight);
        i_aRight = Vector3.Normalize(i_aRight - Vector3.Dot(i_aRight, j_aRight) * 0.5f * j_aRight);
        j_aRight = Vector3.Normalize(j_aRight - Vector3.Dot(i_aRight, j_aRight) * 0.5f * i_aRight);
        k_aRight = Vector3.Cross(i_aRight, j_aRight);
        EulerAnglesaRight = ToEulerAnglesRight(GetVectoraRight()) - InitEulerAnglesaRight;
        EulerAnglesRight = new Vector3(EulerAnglesbRight.X, EulerAnglescRight.Y, EulerAnglesaRight.Z);
    }
    private double averageRight(double val1, double val2, double val3)
    {
        arrayRight = new double[] { val1, val2, val3 };
        return arrayRight.Average();
    }
    private double[] CenterSticksRight(UInt16[] vals)
    {
        double[] s = { 0, 0 };
        s[0] = ((int)((vals[0] - stick_calibrationRight[0]) / 100f)) / 13f;
        s[1] = ((int)((vals[1] - stick_calibrationRight[1]) / 100f)) / 13f;
        return s;
    }
}
public class dataReadClassLeftRight
{
    public void Data()
    {
        ReceiveRaw();
    }
    public Quaternion GetVectora()
    {
        Vector3 v1 = new Vector3(j_a.X, i_a.X, k_a.X);
        Vector3 v2 = -(new Vector3(j_a.Z, i_a.Z, k_a.Z));
        return QuaternionLookRotation(v1, v2);
    }
    public Quaternion GetVectorb()
    {
        Vector3 v1 = new Vector3(j_b.X, i_b.X, k_b.X);
        Vector3 v2 = -(new Vector3(j_b.Z, i_b.Z, k_b.Z));
        return QuaternionLookRotation(v1, v2);
    }
    public Quaternion GetVectorc()
    {
        Vector3 v1 = new Vector3(j_c.X, i_c.X, k_c.X);
        Vector3 v2 = -(new Vector3(j_c.Z, i_c.Z, k_c.Z));
        return QuaternionLookRotation(v1, v2);
    }
    private static Quaternion QuaternionLookRotation(Vector3 forward, Vector3 up)
    {
        Vector3 vector = Vector3.Normalize(forward);
        Vector3 vector2 = Vector3.Normalize(Vector3.Cross(up, vector));
```

```
    Vector3 vector3 = Vector3.Cross(vector, vector2);
    var m00 = vector2.X;
    var m01 = vector2.Y;
    var m02 = vector2.Z;
    var m10 = vector3.X;
    var m11 = vector3.Y;
    var m12 = vector3.Z;
    var m20 = vector.X;
    var m21 = vector.Y;
    var m22 = vector.Z;
    double num8 = (m00 + m11) + m22;
    var quaternion = new Quaternion();
    if (num8 > 0f)
    {
        var num = (double)Math.Sqrt(num8 + 1f);
        quaternion.W = (float)num * 0.5f;
        num = 0.5f / num;
        quaternion.X = (float)(m12 - m21) * (float)num;
        quaternion.Y = (float)(m20 - m02) * (float)num;
        quaternion.Z = (float)(m01 - m10) * (float)num;
        return quaternion;
    }
    if ((m00 >= m11) && (m00 >= m22))
    {
        var num7 = (double)Math.Sqrt(((1f + m00) - m11) - m22);
        var num4 = 0.5f / num7;
        quaternion.X = 0.5f * (float)num7;
        quaternion.Y = (float)(m01 + m10) * (float)num4;
        quaternion.Z = (float)(m02 + m20) * (float)num4;
        quaternion.W = (float)(m12 - m21) * (float)num4;
        return quaternion;
    }
    if (m11 > m22)
    {
        var num6 = (double)Math.Sqrt(((1f + m11) - m00) - m22);
        var num3 = 0.5f / num6;
        quaternion.X = (float)(m10 + m01) * (float)num3;
        quaternion.Y = 0.5f * (float)num6;
        quaternion.Z = (float)(m21 + m12) * (float)num3;
        quaternion.W = (float)(m20 - m02) * (float)num3;
        return quaternion;
    }
    var num5 = (double)Math.Sqrt(((1f + m22) - m00) - m11);
    var num2 = 0.5f / num5;
    quaternion.X = (float)(m20 + m02) * (float)num2;
    quaternion.Y = (float)(m21 + m12) * (float)num2;
    quaternion.Z = 0.5f * (float)num5;
    quaternion.W = (float)(m01 - m10) * (float)num2;
    return quaternion;
}
public static Vector3 ToEulerAngles(Quaternion q)
{
    Vector3 pitchYawRoll = new Vector3();
    double sqw = q.W * q.W;
    double sqx = q.X * q.X;
    double sqy = q.Y * q.Y;
    double sqz = q.Z * q.Z;
    double unit = sqx + sqy + sqz + sqw;
    double test = q.X * q.Y + q.Z * q.W;
    if (test > 0.4999f * unit)                // 0.4999f OR 0.5f - EPSILON
    {
        pitchYawRoll.Y = 2f * (float)Math.Atan2(q.X, q.W);  // Yaw
        pitchYawRoll.X = (float)Math.PI * 0.5f;                // Pitch
```

```
            pitchYawRoll.Z = 0f;                        // Roll
            return pitchYawRoll;
        }
        else if (test < -0.4999f * unit)                // -0.4999f OR -0.5f + EPSILON
        {
            pitchYawRoll.Y = -2f * (float)Math.Atan2(q.X, q.W); // Yaw
            pitchYawRoll.X = -(float)Math.PI * 0.5f;                    // Pitch
            pitchYawRoll.Z = 0f;                        // Roll
            return pitchYawRoll;
        }
        else
        {
            pitchYawRoll.Y = (float)Math.Atan2(2f * q.Y * q.W - 2f * q.X * q.Z, sqx - sqy - sqz + sqw);      // Yaw
            pitchYawRoll.X = (float)Math.Asin(2f * test / unit);                                            // Pitch
            pitchYawRoll.Z = (float)Math.Atan2(2f * q.X * q.W - 2f * q.Y * q.Z, -sqx + sqy - sqz + sqw);     // Roll
        }
        return pitchYawRoll;
    }
    private void ReceiveRaw()
    {
        if (!endinvoke)
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataLeft", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.ReadAsync(report_bufferbLeft, 0, 49);
            };
            report_bufferLeft = report_bufferbLeft;
            using (System.IO.FileStream fs = new System.IO.FileStream("dataRight", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.ReadAsync(report_bufferbRight, 0, 49);
            };
            report_bufferRight = report_bufferbRight;
            ExtractIMUValues();
            System.Threading.Thread.Sleep(10);
        }
    }
    private void ExtractIMUValues()
    {
        acc_g.X = (int)(average((Int16)((report_bufferLeft[13 + 0 * 12] | ((report_bufferLeft[14 + 0 * 12] << 8) & 0xff00)) +
(report_bufferRight[13 + 0 * 12] | ((report_bufferRight[14 + 0 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[13 + 1 *
12] | ((report_bufferLeft[14 + 1 * 12] << 8) & 0xff00)) + (report_bufferRight[13 + 1 * 12] | ((report_bufferRight[14 + 1 * 12]
<< 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[13 + 2 * 12] | ((report_bufferLeft[14 + 2 * 12] << 8) & 0xff00)) +
(report_bufferRight[13 + 2 * 12] | ((report_bufferRight[14 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f / 16000f) -
acc_gcalibrationX;
        acc_g.Y = (int)(average((Int16)((report_bufferLeft[15 + 0 * 12] | ((report_bufferLeft[16 + 0 * 12] << 8) & 0xff00)) -
(report_bufferRight[15 + 0 * 12] | ((report_bufferRight[16 + 0 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[15 + 1 *
12] | ((report_bufferLeft[16 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[15 + 1 * 12] | ((report_bufferRight[16 + 1 * 12]
<< 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[15 + 2 * 12] | ((report_bufferLeft[16 + 2 * 12] << 8) & 0xff00)) -
(report_bufferRight[15 + 2 * 12] | ((report_bufferRight[16 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f / 16000f) -
acc_gcalibrationY;
        acc_g.Z = (int)(average((Int16)((report_bufferLeft[17 + 0 * 12] | ((report_bufferLeft[18 + 0 * 12] << 8) & 0xff00)) -
(report_bufferRight[17 + 0 * 12] | ((report_bufferRight[18 + 0 * 12] << 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[17 + 1 *
12] | ((report_bufferLeft[18 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[17 + 1 * 12] | ((report_bufferRight[18 + 1 * 12]
<< 8) & 0xff00))) / 2f, (Int16)((report_bufferLeft[17 + 2 * 12] | ((report_bufferLeft[18 + 2 * 12] << 8) & 0xff00)) -
(report_bufferRight[17 + 2 * 12] | ((report_bufferRight[18 + 2 * 12] << 8) & 0xff00))) / 2f)) * (1.0f / 16000f) -
acc_gcalibrationZ;
        gyr_g.X = (int)(average((int)((Int16)((report_bufferLeft[19 + 0 * 12] | ((report_bufferLeft[20 + 0 * 12] << 8) &
0xff00)) + (report_bufferRight[19 + 0 * 12] | ((report_bufferRight[20 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[19 + 1 * 12] | ((report_bufferLeft[20 + 1 * 12] << 8) & 0xff00)) + (report_bufferRight[19 + 1 *
12] | ((report_bufferRight[20 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[19 + 2 * 12] |
((report_bufferLeft[20 + 2 * 12] << 8) & 0xff00)) + (report_bufferRight[19 + 2 * 12] | ((report_bufferRight[20 + 2 * 12] << 8)
```

```
& 0xff00))) / (2f)))) * (1.0f / 16000f) - gyr_gcalibrationX;
        gyr_g.Y = (int)(average((int)((Int16)((report_bufferLeft[21 + 0 * 12] | ((report_bufferLeft[22 + 0 * 12] << 8) &
0xff00)) - (report_bufferRight[21 + 0 * 12] | ((report_bufferRight[22 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[21 + 1 * 12] | ((report_bufferLeft[22 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[21 + 1 *
12] | ((report_bufferRight[22 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[21 + 2 * 12] |
((report_bufferLeft[22 + 2 * 12] << 8) & 0xff00)) - (report_bufferRight[21 + 2 * 12] | ((report_bufferRight[22 + 2 * 12] << 8)
& 0xff00))) / (2f)))) * (1.0f / 16000f) - gyr_gcalibrationY;
        gyr_g.Z = (int)(average((int)((Int16)((report_bufferLeft[23 + 0 * 12] | ((report_bufferLeft[24 + 0 * 12] << 8) &
0xff00)) - (report_bufferRight[23 + 0 * 12] | ((report_bufferRight[24 + 0 * 12] << 8) & 0xff00))) / (2f)),
(int)((Int16)((report_bufferLeft[23 + 1 * 12] | ((report_bufferLeft[24 + 1 * 12] << 8) & 0xff00)) - (report_bufferRight[23 + 1 *
12] | ((report_bufferRight[24 + 1 * 12] << 8) & 0xff00))) / (2f)), (int)((Int16)((report_bufferLeft[23 + 2 * 12] |
((report_bufferLeft[24 + 2 * 12] << 8) & 0xff00)) - (report_bufferRight[23 + 2 * 12] | ((report_bufferRight[24 + 2 * 12] << 8)
& 0xff00))) / (2f)))) * (1.0f / 16000f) - gyr_gcalibrationZ;
        acc_r.X = ((int)(acc_g.X * round)) / round;
        acc_r.Y = ((int)(acc_g.Y * round)) / round;
        acc_r.Z = ((int)(acc_g.Z * round)) / round;
        gyr_r.X = ((int)(gyr_g.X * round)) / round;
        gyr_r.Y = ((int)(gyr_g.Y * round)) / round;
        gyr_r.Z = ((int)(gyr_g.Z * round)) / round;
        if (!Getstate)
          InitDirectAngles = acc_r;
        DirectAngles = acc_r - InitDirectAngles;
        if (((int)(i_c.X * round)) / round == 1f | (((int)(i_c.Y * round)) / round == 0f & ((int)(i_c.Z * round)) / round == 0f))
          i_c = new Vector3(1, 0, 0);
        if (((int)(j_c.Y * round)) / round == 1f | (((int)(j_c.X * round)) / round == 0f & ((int)(j_c.Z * round)) / round == 0f))
          j_c = new Vector3(0, 1, 0);
        if (((int)(k_c.Z * round)) / round == 1f | (((int)(k_c.X * round)) / round == 0f & ((int)(k_c.Y * round)) / round == 0f))
          k_c = new Vector3(0, 0, 1);
        if (((int)(i_b.X * round)) / round == 1f | (((int)(i_b.Y * round)) / round == 0f & ((int)(i_b.Z * round)) / round == 0f))
          i_b = new Vector3(1, 0, 0);
        if (((int)(j_b.Y * round)) / round == 1f | (((int)(j_b.X * round)) / round == 0f & ((int)(j_b.Z * round)) / round == 0f))
          j_b = new Vector3(0, 1, 0);
        if (((int)(k_b.Z * round)) / round == 1f | (((int)(k_b.X * round)) / round == 0f & ((int)(k_b.Y * round)) / round == 0f))
          k_b = new Vector3(0, 0, 1);
        if (((int)(i_a.X * round)) / round == 1f | (((int)(i_a.Y * round)) / round == 0f & ((int)(i_a.Z * round)) / round == 0f))
          i_a = new Vector3(1, 0, 0);
        if (((int)(j_a.Y * round)) / round == 1f | (((int)(j_a.X * round)) / round == 0f & ((int)(j_a.Z * round)) / round == 0f))
          j_a = new Vector3(0, 1, 0);
        if (((int)(k_a.Z * round)) / round == 1f | (((int)(k_a.X * round)) / round == 0f & ((int)(k_a.Y * round)) / round == 0f))
          k_a = new Vector3(0, 0, 1);
        if (((int)(EulerAnglesc.Y * round)) / round == 0f | ((int)(EulerAngles.Y * round)) / round == 0f)
        {
          i_c = new Vector3(1, 0, 0);
          j_c = new Vector3(0, 1, 0);
          k_c = new Vector3(0, 0, 1);
          InitEulerAnglesc = ToEulerAngles(GetVectorc());
        }
        if (((int)(EulerAnglesb.X * round)) / round == 0f | ((int)(EulerAngles.X * round)) / round == 0f)
        {
          i_b = new Vector3(1, 0, 0);
          j_b = new Vector3(0, 1, 0);
          k_b = new Vector3(0, 0, 1);
          InitEulerAnglesb = ToEulerAngles(GetVectorb());
        }
        if (((int)(EulerAnglesa.Z * round)) / round == 0f | ((int)(EulerAngles.Z * round)) / round == 0f)
        {
          i_a = new Vector3(1, 0, 0);
          j_a = new Vector3(0, 1, 0);
          k_a = new Vector3(0, 0, 1);
          InitEulerAnglesa = ToEulerAngles(GetVectora());
        }
        i_c = new Vector3(1, 0, 0);
        j_c.X = 0f;
```

```
            k_c.X = 0f;
            k_c.Y = 0f;
            k_c.Z = 1f;
            k_b = new Vector3(0, 0, 1);
            i_b.Z = 0f;
            j_b.Z = 0f;
            j_a = new Vector3(0, 1, 0);
            i_a.Y = 0f;
            k_a.X = 0f;
            k_a.Y = 0f;
            if (!Getstate)
            {
                i_c = new Vector3(1, 0, 0);
                j_c = new Vector3(0, 1, 0);
                k_c = new Vector3(0, 0, 1);
                InitEulerAnglesc = ToEulerAngles(GetVectorc());
                i_b = new Vector3(1, 0, 0);
                j_b = new Vector3(0, 1, 0);
                k_b = new Vector3(0, 0, 1);
                InitEulerAnglesb = ToEulerAngles(GetVectorb());
                i_a = new Vector3(1, 0, 0);
                j_a = new Vector3(0, 1, 0);
                k_a = new Vector3(0, 0, 1);
                InitEulerAnglesa = ToEulerAngles(GetVectora());
            }
            i_c += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, i_c);
            j_c += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, j_c);
            k_c += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, k_c);
            i_c = Vector3.Normalize(i_c - Vector3.Dot(i_c, j_c) * 0.5f * j_c);
            j_c = Vector3.Normalize(j_c - Vector3.Dot(i_c, j_c) * 0.5f * i_c);
            k_c = Vector3.Cross(i_c, j_c);
            EulerAnglesc = ToEulerAngles(GetVectorc()) - InitEulerAnglesc;
            i_b += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, i_b);
            j_b += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, j_b);
            k_b += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, k_b);
            i_b = Vector3.Normalize(i_b - Vector3.Dot(i_b, j_b) * 0.5f * j_b);
            j_b = Vector3.Normalize(j_b - Vector3.Dot(i_b, j_b) * 0.5f * i_b);
            k_b = Vector3.Cross(i_b, j_b);
            EulerAnglesb = ToEulerAngles(GetVectorb()) - InitEulerAnglesb;
            i_a += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, i_a);
            j_a += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, j_a);
            k_a += Vector3.Cross(Vector3.Negate(gyr_r) * 0.04f, k_a);
            i_a = Vector3.Normalize(i_a - Vector3.Dot(i_a, j_a) * 0.5f * j_a);
            j_a = Vector3.Normalize(j_a - Vector3.Dot(i_a, j_a) * 0.5f * i_a);
            k_a = Vector3.Cross(i_a, j_a);
            EulerAnglesa = ToEulerAngles(GetVectora()) - InitEulerAnglesa;
            EulerAngles = new Vector3(EulerAnglesb.X, EulerAnglesc.Y, EulerAnglesa.Z);
        }
        private double average(double val1, double val2, double val3)
        {
            array = new double[] { val1, val2, val3 };
            return array.Average();
        }
    }
    public class dataReadClassWiimote
    {
        public void Data()
        {
            ReceiveRawWiimote();
        }
        private void ReceiveRawWiimote()
        {
            if (!endinvoke)
```

```
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataWiimote", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.ReadAsync(aBuffer, 0, 22);
            };
            bBuffer = aBuffer;
            mWSIR0found = (bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8) > 1 & (bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8)
< 1023;
            mWSIR1found = (bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8) > 1 & (bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8)
< 1023;
            if (mWSIR0notfound == 0 & mWSIR1found)
                mWSIR0notfound = 1;
            if (mWSIR0notfound == 1 & !mWSIR0found & !mWSIR1found)
                mWSIR0notfound = 2;
            if (mWSIR0notfound == 2 & mWSIR0found)
            {
                mWSIR0notfound = 0;
                if (!mWSIRswitch)
                    mWSIRswitch = true;
                else
                    mWSIRswitch = false;
            }
            if (mWSIR0notfound == 0 & mWSIR0found)
                mWSIR0notfound = 0;
            if (mWSIR0notfound == 0 & !mWSIR0found & !mWSIR1found)
                mWSIR0notfound = 0;
            if (mWSIR0notfound == 1 & mWSIR0found)
                mWSIR0notfound = 0;
            if (mWSIR0found)
            {
                mWSIRSensors0X = (bBuffer[6] | ((bBuffer[8] >> 4) & 0x03) << 8);
                mWSIRSensors0Y = (bBuffer[7] | ((bBuffer[8] >> 6) & 0x03) << 8);
            }
            if (mWSIR1found)
            {
                mWSIRSensors1X = (bBuffer[9] | ((bBuffer[8] >> 0) & 0x03) << 8);
                mWSIRSensors1Y = (bBuffer[10] | ((bBuffer[8] >> 2) & 0x03) << 8);
            }
            if (mWSIRswitch)
            {
                mWSIR0foundcam = mWSIR0found;
                mWSIR1foundcam = mWSIR1found;
                mWSIRSensors0Xcam = mWSIRSensors0X - 512f;
                mWSIRSensors0Ycam = mWSIRSensors0Y - 384f;
                mWSIRSensors1Xcam = mWSIRSensors1X - 512f;
                mWSIRSensors1Ycam = mWSIRSensors1Y - 384f;
            }
            else
            {
                mWSIR1foundcam = mWSIR0found;
                mWSIR0foundcam = mWSIR1found;
                mWSIRSensors1Xcam = mWSIRSensors0X - 512f;
                mWSIRSensors1Ycam = mWSIRSensors0Y - 384f;
                mWSIRSensors0Xcam = mWSIRSensors1X - 512f;
                mWSIRSensors0Ycam = mWSIRSensors1Y - 384f;
            }
            if (mWSIR0foundcam & mWSIR1foundcam)
            {
                irx2e = mWSIRSensors0Xcam;
                iry2e = mWSIRSensors0Ycam;
                irx3e = mWSIRSensors1Xcam;
                iry3e = mWSIRSensors1Ycam;
```

```
                mWSIRSensorsXcam = mWSIRSensors0Xcam - mWSIRSensors1Xcam;
                mWSIRSensorsYcam = mWSIRSensors0Ycam - mWSIRSensors1Ycam;
            }
            if (mWSIR0foundcam & !mWSIR1foundcam)
            {
                irx2e = mWSIRSensors0Xcam;
                iry2e = mWSIRSensors0Ycam;
                irx3e = mWSIRSensors0Xcam - mWSIRSensorsXcam;
                iry3e = mWSIRSensors0Ycam - mWSIRSensorsYcam;
            }
            if (mWSIR1foundcam & !mWSIR0foundcam)
            {
                irx3e = mWSIRSensors1Xcam;
                iry3e = mWSIRSensors1Ycam;
                irx2e = mWSIRSensors1Xcam + mWSIRSensorsXcam;
                iry2e = mWSIRSensors1Ycam + mWSIRSensorsYcam;
            }
            MyAngle = (irx2e - irx3e > 0 ? 1f : -1f) * (iry2e - iry3e) / 6000f;
            irxc = irx2e + irx3e;
            iryc = iry2e + iry3e;
            mWSButtonStateIRX = irxc * (1f - (MyAngle > 0 ? MyAngle : -MyAngle)) + MyAngle * iryc;
            mWSButtonStateIRY = iryc * (1f - (MyAngle > 0 ? MyAngle : -MyAngle)) - MyAngle * irxc;
            mWSButtonStateA = (bBuffer[2] & 0x08) != 0;
            mWSButtonStateB = (bBuffer[2] & 0x04) != 0;
            mWSButtonStateMinus = (bBuffer[2] & 0x10) != 0;
            mWSButtonStateHome = (bBuffer[2] & 0x80) != 0;
            mWSButtonStatePlus = (bBuffer[1] & 0x10) != 0;
            mWSButtonStateOne = (bBuffer[2] & 0x02) != 0;
            mWSButtonStateTwo = (bBuffer[2] & 0x01) != 0;
            mWSButtonStateUp = (bBuffer[1] & 0x08) != 0;
            mWSButtonStateDown = (bBuffer[1] & 0x04) != 0;
            mWSButtonStateLeft = (bBuffer[1] & 0x01) != 0;
            mWSButtonStateRight = (bBuffer[1] & 0x02) != 0;
            mWSRawValuesX = bBuffer[3] - 135f + calibrationinit;
            mWSRawValuesY = bBuffer[4] - 135f + calibrationinit;
            mWSRawValuesZ = bBuffer[5] - 135f + calibrationinit;
            mWSNunchuckStateRawJoystickX = bBuffer[16] - 125f + stickviewxinit;
            mWSNunchuckStateRawJoystickY = bBuffer[17] - 125f + stickviewyinit;
            mWSNunchuckStateRawValuesX = bBuffer[18] - 125f;
            mWSNunchuckStateRawValuesY = bBuffer[19] - 125f;
            mWSNunchuckStateRawValuesZ = bBuffer[20] - 125f;
            mWSNunchuckStateC = (bBuffer[21] & 0x02) == 0;
            mWSNunchuckStateZ = (bBuffer[21] & 0x01) == 0;
            System.Threading.Thread.Sleep(1);
        }
    }
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    endinvoke = true;
    notpressing1and2 = true;
    runningoff = true;
    Thread.Sleep(100);
    TimeEndPeriod(1);
    try
    {
        if (FinalFrame.IsRunning == true)
            FinalFrame.Stop();
    }
    catch { }
    IntPtr handle;
    System.Diagnostics.Process[] workers = System.Diagnostics.Process.GetProcessesByName("WiiJoy4FPS");
    foreach (System.Diagnostics.Process worker in workers)
```

```
        {
            try
            {
                handle = worker.MainWindowHandle;
                SwitchToThisWindow(handle, true);
                System.Threading.Thread.Sleep(500);
                SendKeys.Send("%{F4}");
            }
            catch { }
        }
    }
    private const string vendor_id = "57e", vendor_id_ = "057e", product_r1 = "0330", product_r2 = "0306";
    public enum EFileAttributes : uint
    {
        Overlapped = 0x40000000,
        Normal = 0x80
    };
    public struct SP_DEVICE_INTERFACE_DATA
    {
        public int cbSize;
        public Guid InterfaceClassGuid;
        public int Flags;
        public IntPtr RESERVED;
    }
    public struct SP_DEVICE_INTERFACE_DETAIL_DATA
    {
        public UInt32 cbSize;
        [System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.ByValTStr, SizeConst
= 256)]
        public string DevicePath;
    }
    private static double[] stickLeft = { 0, 0 };
    private static byte[] stick_rawLeft = { 0, 0, 0 };
    private static UInt16[] stick_calibrationLeft = { 0, 0 };
    private static UInt16[] stick_precalLeft = { 0, 0 };
    private static Vector3 gyr_gLeft = new Vector3();
    private static Vector3 acc_gLeft = new Vector3();
    private static Vector3 gyr_rLeft = new Vector3();
    private static Vector3 acc_rLeft = new Vector3();
    private const uint report_lenLeft = 49;
    public static Vector3 i_aLeft = new Vector3(1, 0, 0);
    public static Vector3 j_aLeft = new Vector3(0, 1, 0);
    public static Vector3 k_aLeft = new Vector3(0, 0, 1);
    public static Vector3 i_bLeft = new Vector3(1, 0, 0);
    public static Vector3 j_bLeft = new Vector3(0, 1, 0);
    public static Vector3 k_bLeft = new Vector3(0, 0, 1);
    public static Vector3 i_cLeft = new Vector3(1, 0, 0);
    public static Vector3 j_cLeft = new Vector3(0, 1, 0);
    public static Vector3 k_cLeft = new Vector3(0, 0, 1);
    private static Vector3 InitDirectAnglesLeft, DirectAnglesLeft;
    private static Vector3 InitEulerAnglesaLeft, EulerAnglesaLeft, InitEulerAnglesbLeft, EulerAnglesbLeft,
InitEulerAnglescLeft, EulerAnglescLeft, EulerAnglesLeft;
    private static bool LeftButtonSHOULDER_1, LeftButtonSHOULDER_2, LeftButtonSR, LeftButtonSL,
LeftButtonDPAD_DOWN, LeftButtonDPAD_RIGHT, LeftButtonDPAD_UP, LeftButtonDPAD_LEFT, LeftButtonMINUS,
LeftButtonSTICK, LeftButtonCAPTURE;
    private static byte[] report_bufLeft, report_bufaLeft = new byte[report_lenLeft], report_bufferbLeft = new
byte[report_lenLeft];
    private static double[] arrayLeft;
    public static uint hDevInfoLeft;
    public static double indexLeft = 0;
    private static float roundLeft = 16000f;
    public static float acc_gcalibrationLeftX, acc_gcalibrationLeftY, acc_gcalibrationLeftZ, gyr_gcalibrationLeftX,
gyr_gcalibrationLeftY, gyr_gcalibrationLeftZ;
```

```
    public double[] GetStickLeft()
    {
        return stickLeft;
    }
    public Vector3 GetAccelLeft()
    {
        return acc_gLeft;
    }
    private static double[] stickRight = { 0, 0 };
    private static byte[] stick_rawRight = { 0, 0, 0 };
    private static UInt16[] stick_calibrationRight = { 0, 0 };
    private static UInt16[] stick_precalRight = { 0, 0 };
    private static Vector3 acc_gRight = new Vector3();
    private static Vector3 gyr_gRight = new Vector3();
    private static Vector3 acc_rRight = new Vector3();
    private static Vector3 gyr_rRight = new Vector3();
    private const uint report_lenRight = 49;
    public static Vector3 i_cRight = new Vector3(1, 0, 0);
    public static Vector3 j_cRight = new Vector3(0, 1, 0);
    public static Vector3 k_cRight = new Vector3(0, 0, 1);
    public static Vector3 i_bRight = new Vector3(1, 0, 0);
    public static Vector3 j_bRight = new Vector3(0, 1, 0);
    public static Vector3 k_bRight = new Vector3(0, 0, 1);
    public static Vector3 i_aRight = new Vector3(1, 0, 0);
    public static Vector3 j_aRight = new Vector3(0, 1, 0);
    public static Vector3 k_aRight = new Vector3(0, 0, 1);
    private static Vector3 InitDirectAnglesRight, DirectAnglesRight;
    private static Vector3 InitEulerAnglesaRight, EulerAnglesaRight, InitEulerAnglesbRight, EulerAnglesbRight,
InitEulerAnglescRight, EulerAnglescRight, EulerAnglesRight;
    private static bool RightButtonSHOULDER_1, RightButtonSHOULDER_2, RightButtonSR, RightButtonSL,
RightButtonDPAD_DOWN, RightButtonDPAD_RIGHT, RightButtonDPAD_UP, RightButtonDPAD_LEFT, RightButtonPLUS,
RightButtonSTICK, RightButtonHOME;
    private static byte[] report_bufRight, report_bufaRight = new byte[report_lenRight], report_bufferbRight = new
byte[report_lenRight];
    private static double[] arrayRight;
    public static uint hDevInfoRight;
    public static double indexRight = 0;
    private static float roundRight = 16000f;
    public static float acc_gcalibrationRightX, acc_gcalibrationRightY, acc_gcalibrationRightZ, gyr_gcalibrationRightX,
gyr_gcalibrationRightY, gyr_gcalibrationRightZ;
    public double[] GetStickRight()
    {
        return stickRight;
    }
    public Vector3 GetAccelRight()
    {
        return acc_gRight;
    }
    private static Vector3 gyr_g = new Vector3();
    private static Vector3 acc_g = new Vector3();
    private static Vector3 gyr_r = new Vector3();
    private static Vector3 acc_r = new Vector3();
    public static Vector3 i_a = new Vector3(1, 0, 0);
    public static Vector3 j_a = new Vector3(0, 1, 0);
    public static Vector3 k_a = new Vector3(0, 0, 1);
    public static Vector3 i_b = new Vector3(1, 0, 0);
    public static Vector3 j_b = new Vector3(0, 1, 0);
    public static Vector3 k_b = new Vector3(0, 0, 1);
    public static Vector3 i_c = new Vector3(1, 0, 0);
    public static Vector3 j_c = new Vector3(0, 1, 0);
    public static Vector3 k_c = new Vector3(0, 0, 1);
    private static Vector3 InitDirectAngles, DirectAngles;
    private static Vector3 InitEulerAnglesa, EulerAnglesa, InitEulerAnglesb, EulerAnglesb, InitEulerAnglesc, EulerAnglesc,
```

```
EulerAngles;
        private static byte[] report_bufferLeft, report_bufferRight;
        private static double[] array;
        private static float round = 16000f;
        public static float acc_gcalibrationX, acc_gcalibrationY, acc_gcalibrationZ, gyr_gcalibrationX, gyr_gcalibrationY,
gyr_gcalibrationZ;
        public Vector3 GetAccel()
        {
            return acc_g;
        }
    }
}
```

3. *C# Windows Form Codes for FPS (WiiJoy4FPS)*

```
using Microsoft.Win32.SafeHandles;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Diagnostics;
using System.Numerics;
using System.Text.RegularExpressions;
using Microsoft.CSharp;
using System.CodeDom;
using System.Reflection;
namespace WiiJoy4FPS
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        [DllImport("WiiJoyPairing.dll", EntryPoint = "connect")]
        public static unsafe extern int connect();
        [DllImport("WiiJoyPairing.dll", EntryPoint = "disconnect")]
        public static unsafe extern bool disconnect();
        [DllImport("hid.dll")]
        public static unsafe extern void HidD_GetHidGuid(out Guid gHid);
        [DllImport("hid.dll")]
        public extern unsafe static bool HidD_SetOutputReport(IntPtr HidDeviceObject, byte[] lpReportBuffer, uint
ReportBufferLength);
        [DllImport("setupapi.dll")]
        public static unsafe extern IntPtr SetupDiGetClassDevs(ref Guid ClassGuid, string Enumerator, IntPtr hwndParent,
UInt32 Flags);
        [DllImport("setupapi.dll")]
        public static unsafe extern Boolean SetupDiEnumDeviceInterfaces(IntPtr hDevInfo, IntPtr devInvo, ref Guid
interfaceClassGuid, Int32 memberIndex, ref SP_DEVICE_INTERFACE_DATA deviceInterfaceData);
        [DllImport("setupapi.dll")]
        public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr hDevInfo, ref SP_DEVICE_INTERFACE_DATA
deviceInterfaceData, IntPtr deviceInterfaceDetailData, UInt32 deviceInterfaceDetailDataSize, out UInt32 requiredSize, IntPtr
deviceInfoData);
        [DllImport("setupapi.dll")]
        public static unsafe extern Boolean SetupDiGetDeviceInterfaceDetail(IntPtr hDevInfo, ref SP_DEVICE_INTERFACE_DATA
```

```
deviceInterfaceData, ref SP_DEVICE_INTERFACE_DETAIL_DATA deviceInterfaceDetailData, UInt32
deviceInterfaceDetailDataSize, out UInt32 requiredSize, IntPtr deviceInfoData);
    [DllImport("Kernel32.dll")]
    public static unsafe extern IntPtr CreateFile(string fileName, System.IO.FileAccess fileAccess, System.IO.FileShare
fileShare, IntPtr securityAttributes, System.IO.FileMode creationDisposition, EFileAttributes flags, IntPtr template);
    [DllImport("Kernel32.dll")]
    public static unsafe extern SafeFileHandle CreateFile(string fileName, [MarshalAs(UnmanagedType.U4)] FileAccess
fileAccess, [MarshalAs(UnmanagedType.U4)] FileShare fileShare, IntPtr securityAttributes,
[MarshalAs(UnmanagedType.U4)] FileMode creationDisposition, [MarshalAs(UnmanagedType.U4)] uint flags, IntPtr
template);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "Lhid_read_timeout")]
    public static unsafe extern int Lhid_read_timeout(SafeFileHandle dev, byte [] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "Rhid_read_timeout")]
    public static unsafe extern int Rhid_read_timeout(SafeFileHandle dev, byte[] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_write")]
    public static unsafe extern int hid_write(SafeFileHandle device, byte[] data, UIntPtr length);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_open_path")]
    public static unsafe extern SafeFileHandle hid_open_path(IntPtr handle);
    [DllImport("hidread.dll", CallingConvention = CallingConvention.Cdecl, EntryPoint = "hid_close")]
    public static unsafe extern void hid_close(SafeFileHandle device);
    [DllImport("winmm.dll", EntryPoint = "timeBeginPeriod")]
    public static extern uint TimeBeginPeriod(uint ms);
    [DllImport("winmm.dll", EntryPoint = "timeEndPeriod")]
    public static extern uint TimeEndPeriod(uint ms);
    [DllImport("ntdll.dll", EntryPoint = "NtSetTimerResolution")]
    public static extern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint CurrentResolution);
    [DllImport("user32.dll")]
    public static extern bool GetAsyncKeyState(System.Windows.Forms.Keys vKey);
    private int leftandright;
    public static double REGISTER_IR = 0x04b00030, REGISTER_EXTENSION_INIT_1 = 0x04a400f0,
REGISTER_EXTENSION_INIT_2 = 0x04a400fb, REGISTER_EXTENSION_TYPE = 0x04a400fa,
REGISTER_EXTENSION_CALIBRATION = 0x04a40020, REGISTER_MOTIONPLUS_INIT = 0x04a600fe, irx2e, iry2e, irx3e, iry3e,
irx, iry, irxc, iryc, mWSIRSensorsXcam, mWSIRSensorsYcam, mWSIRSensors0X, mWSIRSensors0Y, mWSIRSensors1X,
mWSIRSensors1Y, mWSButtonStateIRX, mWSButtonStateIRY, mWSIRSensors0Xcam, mWSIRSensors1Xcam,
mWSIRSensors0Ycam, mWSIRSensors1Ycam, MyAngle, mWSIR0notfound = 0, mWSRawValuesX, mWSRawValuesY,
mWSRawValuesZ, calibrationinit, camx, camy, watchM = 50, watchM1 = 2, watchM2 = 0, stickviewxinit, stickviewyinit,
mWSNunchuckStateRawValuesX, mWSNunchuckStateRawValuesY, mWSNunchuckStateRawValuesZ,
mWSNunchuckStateRawJoystickX, mWSNunchuckStateRawJoystickY;
    public static bool mWSIR1foundcam, mWSIR0foundcam, mWSIR1found, mWSIR0found, mWSIRswitch,
mWSButtonStateA, mWSButtonStateB, mWSButtonStateMinus, mWSButtonStateHome, mWSButtonStatePlus,
mWSButtonStateOne, mWSButtonStateTwo, mWSButtonStateUp, mWSButtonStateDown, mWSButtonStateLeft,
mWSButtonStateRight, runningoff, Getstate, Setrecenter, notpressing1and2, readingfile, mWSNunchuckStateC,
mWSNunchuckStateZ;
    public static int readingfilecount;
    public static string processname, path;
    public static byte[] buff = new byte[] { 0x55 }, mBuff = new byte[22], aBuffer = new byte[22], bBuffer = new byte[22];
    public static byte Type = 0x12, IR = 0x13, WriteMemory = 0x16, ReadMemory = 0x16, IRExtensionAccel = 0x37;
    public static Guid guid = new System.Guid();
    public static uint hDevInfo, CurrentResolution = 0;
    public static BackgroundWorker backgroundWorkerS = new BackgroundWorker();
    public static ThreadStart threadstart;
    public static Thread thread;
    public static Task taskDHidReadWiimote, taskDWriteWiimote, taskDHidReadLeft, taskDWriteLeft, taskDHidReadRight,
taskDWriteRight;
    private static bool endinvoke;
    private static Stopwatch diffM = new Stopwatch();
    private Type programwiimote, programjoyconleft, programjoyconright;
    private object objwiimote, objjoyconleft, objjoyconright, objdataWriteWiimote, objdataWriteLeft, objdataWriteRight;
    private Assembly assemblywiimote, assemblyjoyconleft, assemblyjoyconright;
    private System.CodeDom.Compiler.CompilerResults resultswiimote, resultsjoyconleft, resultsjoyconright;
    private Microsoft.CSharp.CSharpCodeProvider providerwiimote, providerjoyconleft, providerjoyconright;
    private System.CodeDom.Compiler.CompilerParameters parameterswiimote, parametersjoyconleft,
parametersjoyconright;
```

```
    private string joyconleftcode, joyconrightcode, wiimotecode;
    private static System.IO.FileStream mStream;
    private void Form1_Shown(object sender, EventArgs e)
    {
      TimeBeginPeriod(1);
      NtSetTimerResolution(1, true, ref CurrentResolution);
      System.Diagnostics.Process process = Process.GetCurrentProcess();
      process.PriorityClass = System.Diagnostics.ProcessPriorityClass.RealTime;
      backgroundWorkerS.DoWork += new DoWorkEventHandler(FormStart);
      backgroundWorkerS.RunWorkerAsync();
    }
    private void initConnect()
    {
      System.IO.StreamWriter file = new System.IO.StreamWriter("initconnect.txt");
      file.WriteLine("True");
      file.WriteLine(leftandright.ToString());
      file.Close();
    }
    private void FormStart(object sender, DoWorkEventArgs e)
    {
      do
      {
        Thread.Sleep(1);
        leftandright = connect();
      }
      while (leftandright != 1 & leftandright != 2 & leftandright != 3 &leftandright != 4 & leftandright != 5 & leftandright !=
6 & leftandright != 7 & !notpressing1and2);
      if (!notpressing1and2)
      {
        if (leftandright == 4 | leftandright == 5 | leftandright == 6 | leftandright == 7)
          do
            Thread.Sleep(1);
          while (!ScanRight());
        if (leftandright == 1 | leftandright == 2 | leftandright == 4 | leftandright == 7)
          do
            Thread.Sleep(1);
          while (!ScanLeft());
        if (leftandright == 2 | leftandright == 3 | leftandright == 5 | leftandright == 7)
          do
            Thread.Sleep(1);
          while (!ScanWiimote());
        if (leftandright == 1 | leftandright == 2 | leftandright == 4 | leftandright == 7)
        {
          joyconleftcode = @"using System;
        using System.Runtime.InteropServices;
        using Microsoft.Win32.SafeHandles;
        namespace StringToCode
        {
          public class dataHidReadClassLeft
          {
            [DllImport(""winmm.dll"", EntryPoint = ""timeBeginPeriod"")]
            public static extern uint TimeBeginPeriod(uint ms);
            [DllImport(""winmm.dll"", EntryPoint = ""timeEndPeriod"")]
            public static extern uint TimeEndPeriod(uint ms);
            [DllImport(""ntdll.dll"", EntryPoint = ""NtSetTimerResolution"")]
            public static extern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint
CurrentResolution);
            public static uint CurrentResolution = 0;
            public void Open()
            {
              TimeBeginPeriod(1);
              NtSetTimerResolution(1, true, ref CurrentResolution);
            }
```

```
            public void Close()
            {
                TimeEndPeriod(1);
            }
            [DllImport(""hidread.dll"", CallingConvention = CallingConvention.Cdecl, EntryPoint = ""Lhid_read_timeout"")]
            public static extern int Lhid_read_timeout(SafeFileHandle dev, byte[] data, UIntPtr length);
            private static byte[] report_bufaLeft = new byte[49];
            public object[] Data(SafeFileHandle handleLeft)
            {
                Lhid_read_timeout(handleLeft, report_bufaLeft, (UIntPtr)49);
                return new object[] { report_bufaLeft };
            }
        }
    }";
            parametersjoyconleft = new System.CodeDom.Compiler.CompilerParameters();
            parametersjoyconleft.GenerateExecutable = false;
            parametersjoyconleft.GenerateInMemory = true;
            parametersjoyconleft.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            parametersjoyconleft.ReferencedAssemblies.Add("System.Drawing.dll");
            providerjoyconleft = new Microsoft.CSharp.CSharpCodeProvider();
            resultsjoyconleft = providerjoyconleft.CompileAssemblyFromSource(parametersjoyconleft, joyconleftcode);
            assemblyjoyconleft = resultsjoyconleft.CompiledAssembly;
            programjoyconleft =assemblyjoyconleft.GetType("StringToCode.dataHidReadClassLeft");
            objjoyconleft = Activator.CreateInstance(programjoyconleft);
            objdataWriteLeft = Activator.CreateInstance(typeof(dataWriteClassLeft));
            taskDHidReadLeft = new Task(WiiJoy_thrDHidReadLeft);
            taskDHidReadLeft.Start();
            taskDWriteLeft = new Task(WiiJoy_thrDWriteLeft);
            taskDWriteLeft.Start();
        }
        if (leftandright == 4 | leftandright == 5 | leftandright == 6 | leftandright == 7)
        {
            joyconrightcode = @"using System;
        using System.Runtime.InteropServices;
        using Microsoft.Win32.SafeHandles;
        namespace StringToCode
        {
            public class dataHidReadClassRight
            {
                [DllImport(""winmm.dll"", EntryPoint = ""timeBeginPeriod"")]
                public static extern uint TimeBeginPeriod(uint ms);
                [DllImport(""winmm.dll"", EntryPoint = ""timeEndPeriod"")]
                public static extern uint TimeEndPeriod(uint ms);
                [DllImport(""ntdll.dll"", EntryPoint = ""NtSetTimerResolution"")]
                public static extern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint
CurrentResolution);
                public static uint CurrentResolution = 0;
                public void Open()
                {
                    TimeBeginPeriod(1);
                    NtSetTimerResolution(1, true, ref CurrentResolution);
                }
                public void Close()
                {
                    TimeEndPeriod(1);
                }
                [DllImport(""hidread.dll"", CallingConvention = CallingConvention.Cdecl, EntryPoint = ""Rhid_read_timeout"")]
                public static extern int Rhid_read_timeout(SafeFileHandle dev, byte[] data, UIntPtr length);
                private static byte[] report_bufaRight = new byte[49];
                public object[] Data(SafeFileHandle handleRight)
                {
                    Rhid_read_timeout(handleRight, report_bufaRight, (UIntPtr)49);
                    return new object[] { report_bufaRight };
```

```
            }
        }
    }";
        parametersjoyconright = new System.CodeDom.Compiler.CompilerParameters();
        parametersjoyconright.GenerateExecutable = false;
        parametersjoyconright.GenerateInMemory = true;
        parametersjoyconright.ReferencedAssemblies.Add("System.Windows.Forms.dll");
        parametersjoyconright.ReferencedAssemblies.Add("System.Drawing.dll");
        providerjoyconright = new Microsoft.CSharp.CSharpCodeProvider();
        resultsjoyconright = providerjoyconright.CompileAssemblyFromSource(parametersjoyconright,
joyconrightcode);
        assemblyjoyconright = resultsjoyconright.CompiledAssembly;
        programjoyconright = assemblyjoyconright.GetType("StringToCode.dataHidReadClassRight");
        objjoyconright = Activator.CreateInstance(programjoyconright);
        objdataWriteRight = Activator.CreateInstance(typeof(dataWriteClassRight));
        taskDHidReadRight = new Task(WiiJoy_thrDHidReadRight);
        taskDHidReadRight.Start();
        taskDWriteRight = new Task(WiiJoy_thrDWriteRight);
        taskDWriteRight.Start();
    }
    if (leftandright == 2 | leftandright == 3 | leftandright == 5 | leftandright == 7)
    {
        wiimotecode = @"using System;
    using System.Runtime.InteropServices;
    namespace StringToCode
    {
        public class dataHidReadClassWiimote
        {
            [DllImport(""winmm.dll"", EntryPoint = ""timeBeginPeriod"")]
            public static extern uint TimeBeginPeriod(uint ms);
            [DllImport(""winmm.dll"", EntryPoint = ""timeEndPeriod"")]
            public static extern uint TimeEndPeriod(uint ms);
            [DllImport(""ntdll.dll"", EntryPoint = ""NtSetTimerResolution"")]
            public static extern void NtSetTimerResolution(uint DesiredResolution, bool SetResolution, ref uint
CurrentResolution);
            public static uint CurrentResolution = 0;
            public void Open()
            {
                TimeBeginPeriod(1);
                NtSetTimerResolution(1, true, ref CurrentResolution);
            }
            public void Close()
            {
                TimeEndPeriod(1);
            }
            public static byte[] aBuffer = new byte[22];
            public static bool readingfile;
            public object[] Data(System.IO.FileStream mStream)
            {
                try
                {
                    mStream.Read(aBuffer, 0, 22);
                    readingfile = true;
                }
                catch
                {
                    readingfile = false;
                }
                return new object[] { aBuffer, readingfile };
            }
        }
    }";
        parameterswiimote = new System.CodeDom.Compiler.CompilerParameters();
```

```
            parameterswiimote.GenerateExecutable = false;
            parameterswiimote.GenerateInMemory = true;
            parameterswiimote.ReferencedAssemblies.Add("System.Windows.Forms.dll");
            parameterswiimote.ReferencedAssemblies.Add("System.Drawing.dll");
            providerwiimote = new Microsoft.CSharp.CSharpCodeProvider();
            resultswiimote = providerwiimote.CompileAssemblyFromSource(parameterswiimote, wiimotecode);
            assemblywiimote = resultswiimote.CompiledAssembly;
            programwiimote = assemblywiimote.GetType("StringToCode.dataHidReadClassWiimote");
            objwiimote = Activator.CreateInstance(programwiimote);
            objdataWriteWiimote = Activator.CreateInstance(typeof(dataWriteClassWiimote));
            taskDHidReadWiimote = new Task(WiiJoy_thrDHidReadWiimote);
            taskDHidReadWiimote.Start();
            taskDWriteWiimote = new Task(WiiJoy_thrDWriteWiimote);
            taskDWriteWiimote.Start();
          }
          try { initConnect(); }
          catch
          {
            using (System.IO.StreamWriter createdfile = System.IO.File.AppendText("initconnect.txt"))
            {
              createdfile.WriteLine("True");
              createdfile.WriteLine(leftandright.ToString());
            }
          }
          assemblywiimote = null;
          assemblyjoyconleft = null;
          assemblyjoyconright = null;
          resultswiimote = null;
          resultsjoyconleft = null;
          resultsjoyconright = null;
          providerwiimote = new Microsoft.CSharp.CSharpCodeProvider();
          providerjoyconleft = new Microsoft.CSharp.CSharpCodeProvider();
          providerjoyconright = new Microsoft.CSharp.CSharpCodeProvider();
          parameterswiimote = new System.CodeDom.Compiler.CompilerParameters();
          parametersjoyconleft = new System.CodeDom.Compiler.CompilerParameters();
          parametersjoyconright = new System.CodeDom.Compiler.CompilerParameters();
          joyconleftcode = "";
          joyconrightcode = "";
          wiimotecode = "";
          this.BackColor = System.Drawing.Color.WhiteSmoke;
        }
      }
      private async void WiiJoy_thrDHidReadLeft()
      {
        programjoyconleft.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod, null, objjoyconleft,
new object[] { });
        for (; ; )
        {
          if (endinvoke)
          {
            programjoyconleft.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objjoyconleft, new object[] { });
            return;
          }
          await asynctaskDHidReadLeft();
        }
      }
      public async Task asynctaskDHidReadLeft()
      {
        object[] val = (object[])programjoyconleft.InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod,
null, objjoyconleft, new object[] { handleLeft });
        report_bufaLeft = (byte[])val[0];
        await Task.Delay(TimeSpan.FromMilliseconds(0));
```

```
    }
    public async void WiiJoy_thrDWriteLeft()
    {
        for (; ; )
        {
            if (endinvoke)
                return;
            await asynctaskDWriteLeft();
        }
    }
    public async Task asynctaskDWriteLeft()
    {
        try
        {
            typeof(dataWriteClassLeft).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataWriteLeft, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public class dataWriteClassLeft
    {
        public void Data()
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataLeft", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.WriteAsync(report_bufaLeft, 0, 49);
            };
        }
    }
    private async void WiiJoy_thrDHidReadRight()
    {
        programjoyconright.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod, null, objjoyconright,
new object[] { });
        for (; ; )
        {
            if (endinvoke)
            {
                programjoyconright.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objjoyconright, new object[] { });
                return;
            }
            await asynctaskDHidReadRight();
        }
    }
    public async Task asynctaskDHidReadRight()
    {
        object[] val = (object[])programjoyconright.InvokeMember("Data", BindingFlags.Default |
BindingFlags.InvokeMethod, null, objjoyconright, new object[] { handleRight });
        report_bufaRight = (byte[])val[0];
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public async void WiiJoy_thrDWriteRight()
    {
        for (; ; )
        {
            if (endinvoke)
                return;
            await asynctaskDWriteRight();
        }
    }
```

```
    public async Task asynctaskDWriteRight()
    {
        try
        {
            typeof(dataWriteClassRight).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataWriteRight, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public class dataWriteClassRight
    {
        public void Data()
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataRight", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.WriteAsync(report_bufaRight, 0, 49);
            };
        }
    }
    private async void WiiJoy_thrDHidReadWiimote()
    {
        programwiimote.InvokeMember("Open", BindingFlags.Default | BindingFlags.InvokeMethod, null, objwiimote, new
object[] { });
        for (; ; )
        {
            if (runningoff)
            {
                programwiimote.InvokeMember("Close", BindingFlags.Default | BindingFlags.InvokeMethod, null, objwiimote,
new object[] { });
                return;
            }
            await asynctaskDHidReadWiimote();
        }
    }
    public async Task asynctaskDHidReadWiimote()
    {
        object[] val = (object[])programwiimote.InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod,
null, objwiimote, new object[] { mStream });
        aBuffer = (byte[])val[0];
        readingfile = (bool)val[1];
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public async void WiiJoy_thrDWriteWiimote()
    {
        for (; ; )
        {
            if (endinvoke)
                return;
            await asynctaskDWriteWiimote();
        }
    }
    public async Task asynctaskDWriteWiimote()
    {
        try
        {
            typeof(dataWriteClassWiimote).InvokeMember("Data", BindingFlags.Default | BindingFlags.InvokeMethod, null,
objdataWriteWiimote, new object[] { });
        }
        catch { }
        System.Threading.Thread.Sleep(TimeSpan.FromMilliseconds(1));
```

```
        await Task.Delay(TimeSpan.FromMilliseconds(0));
    }
    public class dataWriteClassWiimote
    {
        public void Data()
        {
            using (System.IO.FileStream fs = new System.IO.FileStream("dataWiimote", System.IO.FileMode.OpenOrCreate,
System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, bufferSize: 4096, useAsync: true))
            {
                fs.WriteAsync(aBuffer, 0, 22);
            };
            if (readingfilecount == 0)
                readingfile = false;
            readingfilecount++;
            if (readingfilecount > 100)
            {
                if (!readingfile & !runningoff)
                    WiimoteFound(path);
                readingfilecount = 0;
            }
        }
    }
    private void FormClose()
    {
        disconnect();
    }
    private void Form1_FormClosed(object sender, FormClosedEventArgs e)
    {
        endinvoke = true;
        notpressing1and2 = true;
        runningoff = true;
        Thread.Sleep(100);
        TimeEndPeriod(1);
        try
        {
            hid_close(handleLeft);
        }
        catch { }
        try
        {
            hid_close(handleRight);
        }
        catch { }
        System.IO.StreamWriter file = new System.IO.StreamWriter("initconnect.txt");
        file.WriteLine("False");
        file.WriteLine(leftandright.ToString());
        file.Close();
        threadstart = new ThreadStart(FormClose);
        thread = new Thread(threadstart);
        thread.Start();
    }
    private const string vendor_id = "57e", vendor_id_ = "057e", product_l = "2006", product_r = "2007", product_r1 =
"0330", product_r2 = "0306";
    public enum EFileAttributes : uint
    {
        Overlapped = 0x40000000,
        Normal = 0x80
    };
    public struct SP_DEVICE_INTERFACE_DATA
    {
        public int cbSize;
        public Guid InterfaceClassGuid;
        public int Flags;
```

```
        public IntPtr RESERVED;
    }
    public struct SP_DEVICE_INTERFACE_DETAIL_DATA
    {
        public UInt32 cbSize;
        [System.Runtime.InteropServices.MarshalAs(System.Runtime.InteropServices.UnmanagedType.ByValTStr, SizeConst
= 256)]
        public string DevicePath;
    }
    private static SafeFileHandle handleLeft;
    private const uint report_lenLeft = 49;
    private static byte[] report_bufaLeft = new byte[report_lenLeft];
    private static byte[] buf_Left = new byte[report_lenLeft];
    private bool ScanLeft()
    {
        int index = 0;
        System.Guid guid;
        HidD_GetHidGuid(out guid);
        System.IntPtr hDevInfo = SetupDiGetClassDevs(ref guid, null, new System.IntPtr(), 0x00000010);
        SP_DEVICE_INTERFACE_DATA diData = new SP_DEVICE_INTERFACE_DATA();
        diData.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(diData);
        while (SetupDiEnumDeviceInterfaces(hDevInfo, new System.IntPtr(), ref guid, index, ref diData))
        {
            System.UInt32 size;
            SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, new System.IntPtr(), 0, out size, new System.IntPtr());
            SP_DEVICE_INTERFACE_DETAIL_DATA diDetail = new SP_DEVICE_INTERFACE_DETAIL_DATA();
            diDetail.cbSize = 5;
            if (SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, ref diDetail, size, out size, new System.IntPtr()))
            {
                if ((diDetail.DevicePath.Contains(vendor_id) | diDetail.DevicePath.Contains(vendor_id_)) &
diDetail.DevicePath.Contains(product_l))
                {
                    AttachJoyLeft(diDetail.DevicePath);
                    AttachJoyLeft(diDetail.DevicePath);
                    AttachJoyLeft(diDetail.DevicePath);
                    return true;
                }
            }
            index++;
        }
        return false;
    }
    public void AttachJoyLeft(string path)
    {
        do
        {
            IntPtr handle = CreateFile(path, System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, new
System.IntPtr(), System.IO.FileMode.Open, EFileAttributes.Normal, new System.IntPtr());
            handleLeft = hid_open_path(handle);
            SubcommandLeft(0x3, new byte[] { 0x30 }, 1);
            SubcommandLeft(0x40, new byte[] { 0x1 }, 1);
        }
        while (handleLeft.IsInvalid);
    }
    private void SubcommandLeft(byte sc, byte[] buf, uint len)
    {
        Array.Copy(buf, 0, buf_Left, 11, len);
        buf_Left[0] = 0x1;
        buf_Left[1] = 0;
        buf_Left[10] = sc;
        hid_write(handleLeft, buf_Left, (UIntPtr)(len + 11));
        Uhid_read_timeout(handleLeft, buf_Left, (UIntPtr)49);
    }
```

```
    private static SafeFileHandle handleRight;
    private const uint report_lenRight = 49;
    private static byte[] report_bufaRight = new byte[report_lenRight];
    private static byte[] buf_Right = new byte[report_lenRight];
    private bool ScanRight()
    {
        int index = 0;
        System.Guid guid;
        HidD_GetHidGuid(out guid);
        System.IntPtr hDevInfo = SetupDiGetClassDevs(ref guid, null, new System.IntPtr(), 0x00000010);
        SP_DEVICE_INTERFACE_DATA diData = new SP_DEVICE_INTERFACE_DATA();
        diData.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(diData);
        while (SetupDiEnumDeviceInterfaces(hDevInfo, new System.IntPtr(), ref guid, index, ref diData))
        {
            System.UInt32 size;
            SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, new System.IntPtr(), 0, out size, new System.IntPtr());
            SP_DEVICE_INTERFACE_DETAIL_DATA diDetail = new SP_DEVICE_INTERFACE_DETAIL_DATA();
            diDetail.cbSize = 5;
            if (SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, ref diDetail, size, out size, new System.IntPtr()))
            {
                if ((diDetail.DevicePath.Contains(vendor_id) | diDetail.DevicePath.Contains(vendor_id_)) &
diDetail.DevicePath.Contains(product_r))
                {
                    AttachJoyRight(diDetail.DevicePath);
                    AttachJoyRight(diDetail.DevicePath);
                    AttachJoyRight(diDetail.DevicePath);
                    return true;
                }
            }
            index++;
        }
        return false;
    }
    public void AttachJoyRight(string path)
    {
        do
        {
            IntPtr handle = CreateFile(path, System.IO.FileAccess.ReadWrite, System.IO.FileShare.ReadWrite, new
System.IntPtr(), System.IO.FileMode.Open, EFileAttributes.Normal, new System.IntPtr());
            handleRight = hid_open_path(handle);
            SubcommandRight(0x3, new byte[] { 0x30 }, 1);
            SubcommandRight(0x40, new byte[] { 0x1 }, 1);
        }
        while (handleRight.IsInvalid);
    }
    private void SubcommandRight(byte sc, byte[] buf, uint len)
    {
        Array.Copy(buf, 0, buf_Right, 11, len);
        buf_Right[0] = 0x1;
        buf_Right[1] = 0;
        buf_Right[10] = sc;
        hid_write(handleRight, buf_Right, (UIntPtr)(len + 11));
        Rhid_read_timeout(handleRight, buf_Right, (UIntPtr)49);
    }
    private bool ScanWiimote()
    {
        int index = 0;
        System.Guid guid;
        HidD_GetHidGuid(out guid);
        System.IntPtr hDevInfo = SetupDiGetClassDevs(ref guid, null, new System.IntPtr(), 0x00000010);
        SP_DEVICE_INTERFACE_DATA diData = new SP_DEVICE_INTERFACE_DATA();
        diData.cbSize = System.Runtime.InteropServices.Marshal.SizeOf(diData);
        while (SetupDiEnumDeviceInterfaces(hDevInfo, new System.IntPtr(), ref guid, index, ref diData))
```

```
        {
            System.UInt32 size;
            SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, new System.IntPtr(), 0, out size, new System.IntPtr());
            SP_DEVICE_INTERFACE_DETAIL_DATA diDetail = new SP_DEVICE_INTERFACE_DETAIL_DATA();
            diDetail.cbSize = 5;
            if (SetupDiGetDeviceInterfaceDetail(hDevInfo, ref diData, ref diDetail, size, out size, new System.IntPtr()))
            {
                if ((diDetail.DevicePath.Contains(vendor_id) | diDetail.DevicePath.Contains(vendor_id_)) &
(diDetail.DevicePath.Contains(product_r1) | diDetail.DevicePath.Contains(product_r2)))
                {
                    path = diDetail.DevicePath;
                    WiimoteFound(diDetail.DevicePath);
                    WiimoteFound(diDetail.DevicePath);
                    WiimoteFound(diDetail.DevicePath);
                    return true;
                }
            }
            index++;
        }
        return false;
    }
    public static void WiimoteFound(string path)
    {
        SafeFileHandle handle = null;
        do
        {
            handle = CreateFile(path, FileAccess.ReadWrite, FileShare.ReadWrite, IntPtr.Zero, FileMode.Open,
(uint)EFileAttributes.Overlapped, IntPtr.Zero);
            WriteData(handle, IR, (int)REGISTER_IR, new byte[] { 0x08 }, 1);
            WriteData(handle, Type, (int)REGISTER_EXTENSION_INIT_1, new byte[] { 0x55 }, 1);
            WriteData(handle, Type, (int)REGISTER_EXTENSION_INIT_2, new byte[] { 0x00 }, 1);
            WriteData(handle, Type, (int)REGISTER_MOTIONPLUS_INIT, new byte[] { 0x04 }, 1);
            ReadData(handle, 0x0016, 7);
            ReadData(handle, (int)REGISTER_EXTENSION_TYPE, 6);
            ReadData(handle, (int)REGISTER_EXTENSION_CALIBRATION, 16);
            ReadData(handle, (int)REGISTER_EXTENSION_CALIBRATION, 32);
        }
        while (handle.IsInvalid);
        mStream = new System.IO.FileStream(handle, System.IO.FileAccess.ReadWrite, 22, true);
    }
    public static void ReadData(SafeFileHandle _hFile, int address, short size)
    {
        mBuff[0] = (byte)ReadMemory;
        mBuff[1] = (byte)((address & 0xff000000) >> 24);
        mBuff[2] = (byte)((address & 0x00ff0000) >> 16);
        mBuff[3] = (byte)((address & 0x0000ff00) >> 8);
        mBuff[4] = (byte)(address & 0x000000ff);
        mBuff[5] = (byte)((size & 0xff00) >> 8);
        mBuff[6] = (byte)(size & 0xff);
        HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
    }
    public static void WriteData(SafeFileHandle _hFile, byte mbuff, int address, byte[] buff, short size)
    {
        mBuff[0] = (byte)mbuff;
        mBuff[1] = (byte)(0x04);
        mBuff[2] = (byte)IRExtensionAccel;
        Array.Copy(buff, 0, mBuff, 3, 1);
        HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
        mBuff[0] = (byte)WriteMemory;
        mBuff[1] = (byte)(((address & 0xff000000) >> 24));
        mBuff[2] = (byte)((address & 0x00ff0000) >> 16);
        mBuff[3] = (byte)((address & 0x0000ff00) >> 8);
        mBuff[4] = (byte)((address & 0x000000ff) >> 0);
```

```
            mBuff[5] = (byte)size;
            Array.Copy(buff, 0, mBuff, 6, 1);
            HidD_SetOutputReport(_hFile.DangerousGetHandle(), mBuff, 22);
        }
    }
}
```

4. *Use and Agreement Contract*

Owner: Michael Andre Franiatte.

Contact: michael.franiatte@gmail.com.

Owning: All works from scratch of the owner.

Proof of Owning: Works published, and writings/speakings all over.

Requirements of Use: Pay the owner, quote the owner, agreement of the owner.

Availability of Works: Only under the shapes of the owner built, only for personal use.

Subjects of Claims: Works published by the owner on Google Play and Google Books.

Concerning Author Rights: Equations and codes from scratch of the owner, softwares built from it, all things of people arising from it.

End User License Agreement: A commercial license is required to use in personal manner. Do not redistributing in any manner, including by computer media, a file server, an email attachment, etc. Do not embedding in or linking it to another programs, source codes and assistances including internal applications, scripts, batch files, etc. Do not use for any kind of technical support including on customer or retailer computer, hardware or software development, research, discovery, teachery, talk, speech, write, etc. Do not use for win money or for commercialisation of any products arising from my programs, source codes and assistances. Do not use and do not copy the way it run in other programs, source codes and assistances. Do not use without pay me, quote me and my agreement. Do not steal or copy or reproduce or modify or peer or share. Do not use in other manner than personal. It stand for my programs, source codes and assistances or programs, source codes and assistances stealing or copying or reproducing or modifying or peering or sharing my programs, source codes, and assistances. If you aren't agree you shall not use.

Terms of License and Price: The present contract acceptance is required to use works of the owner and built from it in all kind of manner. The price for each user shall be defined with the owner by contacting him and this for each subject of works the owner claims. Each user shall contact the owner for asking his agreement. It can be refused by the owner depending who asking and the price defined. People don’t respecting the present contract shall not use the works of the owner.