

**POLITECHNIKA WROCŁAWSKA Instytut Informatyki, Automatyki i Robotyki Zakład Systemów Komputerowych**

**Urządzenia peryferyjne**

**Prowadzący zajęcia: Dr inż. Jan Nikodem**

**Ćwiczenie nr 10**

**Obsługa joysticka USB**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Wykonali:** |  | **Przemysław Wojcinowicz 225943,**  **Jakub Frąckiewicz 226033** |
|  |  |
|  |  |  |  |
|  | **Termin:** |  | **Pn/N 10.15-13.15** |
|  |  |  |  |
|  | **Data wykonania ćwiczenia:** |  | **15.01.18** |
|  |  |  |  |
|  | **Data oddania sprawozdania:** |  | **18.01.18** |
|  |  |  |  |
|  | **Ocena:** |  |  |
|  |  |  |  |
|  | |  |  |
| **Uwagi prowadzącego:** | |  |  |
|  |  |  |  |

Zadania:

Na zajęciach realizowaliśmy ćwiczenie nr 10, czyli obsługa joysticka USB. Zadanie polegało na napisaniu programu, który wyświetla dostępne urządzenia podłączone do komputera przez port usb, następnie umożliwia obsługę joysticka.

Wstęp:

Joystick (manipulator drążkowy) -[urządzenie wejścia](https://pl.wikipedia.org/wiki/Urz%C4%85dzenie_wej%C5%9Bcia-wyj%C5%9Bcia) [komputera](https://pl.wikipedia.org/wiki/Komputer), manipulator służący do sterowania ruchem obiektów na ekranie. W podstawowej wersji składa się z wychylnego drążka zamocowanego na podstawce, którego przechylenie w odpowiednim kierunku powoduje stosowną reakcję sterowanego obiektu, oraz z umieszczonych na drążku i podstawce przycisków uruchamiających przypisane im działania i dodatkowe funkcje sterujące. Pierwsze joysticki nie służyły do rozrywki, lecz do sterowania samolotami; nazywano je wtedy „drążkami sterowniczymi”.

Joysticki stosuje się również do sterowania [robotami](https://pl.wikipedia.org/wiki/Robot) i elektrycznymi [wózkami inwalidzkimi](https://pl.wikipedia.org/wiki/W%C3%B3zek_inwalidzki)

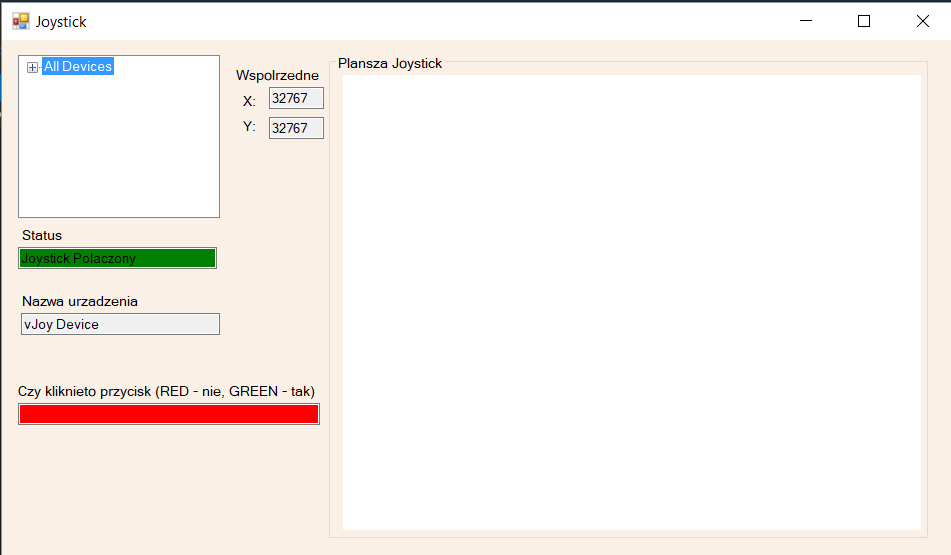
**HAL -** Warstwa abstrakcji sprzętowej ([ang.](https://pl.wikipedia.org/wiki/J%C4%99zyk_angielski) *hardware abstraction layer* – HAL) – [sterownik urządzenia](https://pl.wikipedia.org/wiki/Sterownik_urz%C4%85dzenia) dla [płyty głównej](https://pl.wikipedia.org/wiki/P%C5%82yta_g%C5%82%C3%B3wna). Stanowi ogniwo pośredniczące między sprzętem a [jądrem systemu operacyjnego](https://pl.wikipedia.org/wiki/J%C4%85dro_systemu_operacyjnego). Odseparowuje konkretną architekturę [systemu komputerowego](https://pl.wikipedia.org/wiki/System_komputerowy) od [oprogramowania użytkowego](https://pl.wikipedia.org/wiki/Oprogramowanie_u%C5%BCytkowe). Dla [programisty](https://pl.wikipedia.org/wiki/Programista) jest sposobem komunikacji ze sprzętem przez udostępniane funkcje biblioteczne i sterowniki.

HEL - (ang. [*Hardware*](http://www.i-slownik.pl/629,hardware/) Emulation Layer – [warstwa](http://www.i-slownik.pl/2079,warstwa/) emulacji sprzętu) – umożliwia programową emulację funkcji, które nie są wsparte sprzętowo.

**USB** (od [ang.](https://pl.wikipedia.org/wiki/J%C4%99zyk_angielski) *Universal Serial Bus*, czyli uniwersalna magistrala szeregowa) – komputerowe [złącze](https://pl.wikipedia.org/wiki/Z%C5%82%C4%85cze_(elektronika)) komunikacyjne (tak zwany [port](https://pl.wikipedia.org/wiki/Port_(sprz%C4%99t_komputerowy)) lub [interfejs](https://pl.wikipedia.org/wiki/Interfejs_(urz%C4%85dzenie))) zastępujące stare [porty szeregowe](https://pl.wikipedia.org/wiki/Port_szeregowy) i [porty równoległe](https://pl.wikipedia.org/wiki/Port_r%C3%B3wnoleg%C5%82y). Zostało opracowane przez firmy [Microsoft](https://pl.wikipedia.org/wiki/Microsoft), [Intel](https://pl.wikipedia.org/wiki/Intel), [Compaq](https://pl.wikipedia.org/wiki/Compaq" \o "Compaq), [IBM](https://pl.wikipedia.org/wiki/IBM) i [DEC](https://pl.wikipedia.org/wiki/Digital_Equipment_Corporation).

Port USB jest uniwersalny w tym sensie, że można go wykorzystać do podłączenia do [komputera](https://pl.wikipedia.org/wiki/Komputer) wielu różnych urządzeń np. joysticka.

**HID** ([ang.](https://pl.wikipedia.org/wiki/J%C4%99zyk_angielski) *Human Input Devices* lub *Human Interface Device* – urządzenia do wprowadzania danych przez człowieka) nazwa kodowa dla [urządzeń peryferyjnych](https://pl.wikipedia.org/w/index.php?title=Urz%C4%85dzenie_preryferyjne&action=edit&redlink=1) służących do wprowadzania informacji do komputera, takich jak [dżojstik](https://pl.wikipedia.org/wiki/D%C5%BCojstik), [mysz](https://pl.wikipedia.org/wiki/Mysz_komputerowa), [trackball](https://pl.wikipedia.org/wiki/Trackball) czy [klawiatura](https://pl.wikipedia.org/wiki/Klawiatura_komputerowa). Urządzenia HID są produkowane z myślą o podłączeniu ich do portu [USB](https://pl.wikipedia.org/wiki/USB), dzięki czemu są łatwo rozpoznawane i odpowiednio przez komputer numerowane.

Wygląd programu.



Urządzenie używane na zajęciach.

Kod programu:

using System;

using System.Runtime.InteropServices;

using System.Windows;

using System.Threading;

using System.Windows.Controls;

using Microsoft.DirectX.DirectInput;

using System.Windows.Forms;

using System.Windows.Media;

using System.Windows.Shapes;

using System.Windows.Threading;

using Brushes = System.Drawing.Brushes;

using Color = System.Drawing.Color;

using TreeView = System.Windows.Forms.TreeView;

using Point = System.Drawing.Point;

using Rectangle = System.Drawing.Rectangle;

namespace UPJoystickv2

{

public partial class Form1 : Form

{

[DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention = CallingConvention.StdCall)]

public static extern void mouse\_event(uint dwFlags, uint dx, uint dy, uint cButtons, uint dwExtraInfo);

Device joystick;

private int xValue;

private int yValue;

private bool buttonValue;

public Form1()

{

InitializeComponent();

PopulateAllDevices(treeView1);

InitJoystick();

JoystickService();

WpfHost();

WpfPaintHost();

}

private void WpfHost()

{

System.Windows.Controls.Canvas wpfCanvas = new System.Windows.Controls.Canvas();

//wpfCanvas.EditingMode = InkCanvasEditingMode.InkAndGesture;

Ellipse myEllipse = new Ellipse();

// Create a SolidColorBrush with a red color to fill the

// Ellipse with.

SolidColorBrush mySolidColorBrush = new SolidColorBrush();

// Describes the brush's color using RGB values.

// Each value has a range of 0-255.

//mySolidColorBrush.Color = Color.Red;

myEllipse.Fill = mySolidColorBrush;

myEllipse.StrokeThickness = 2;

//myEllipse.Stroke = Color.Black;

// Set the width and height of the Ellipse.

myEllipse.Width = 200;

myEllipse.Height = 100;

// Add the Ellipse to the StackPanel.

wpfCanvas.Children.Add(myEllipse);

elementHost1.Dock = DockStyle.Fill;

elementHost1.Child = wpfCanvas;

}

private void WpfPaintHost()

{

System.Windows.Controls.InkCanvas wpfCanvas = new System.Windows.Controls.InkCanvas();

wpfCanvas.EditingMode = InkCanvasEditingMode.InkAndGesture;

elementHost1.Dock = DockStyle.Fill;

elementHost1.Child = wpfCanvas;

dataGridView2.Controls.Add(elementHost1);}

private void JoystickService()

{

//Jeżeli brak joysticka, przerywamy metodę

if (joystick == null)

{

return;

}

//Uruchamiamy wątek, który będzie monitorował stan joysticka (odnośnie jego współrzędnych oraz stanu przycisków)

Thread joystickUpdateThread = new Thread(new ThreadStart(GetJoystickState));

//ustawienie pola IsBackground na wartość true powoduje to, że gdy zakończy się główny wątek, ten wątek również się zakończy

joystickUpdateThread.IsBackground = true;

joystickUpdateThread.Start();

//Uruchamiamy wątek do obsługi kursora myszy za pomocą joysticka

Thread moveMouseThread = new Thread(new ThreadStart(MoveCursor));

//pole IsBackground analogicznie jak przed uruchomieniem poprzedniego wątku

moveMouseThread.IsBackground = true;

moveMouseThread.Start();

}

private void MoveCursor()

{

while (true)

{

Thread.Sleep(20);

//Przekazujemy do wątku głównego akutalną współrzędną kursora (aktualizowana co 20ms)

this.Invoke((MethodInvoker) delegate

{

//Porównujemy aktualną pozycję joysticka z pozycją kursora

//Jeżeli akutalna współrzędna X jest mniejsza od współrzędnej X kursora

if (currentXPos > xValue)

{

//Tworzymy nową pozycję kursora, do której przypisujemy punkt, odpowiednio pomniejszony o przesunięcie współrzędnej X

this.Cursor = new Cursor(Cursor.Current.Handle);

Cursor.Position = new Point(Cursor.Position.X - (currentXPos / 1000), Cursor.Position.Y);

Cursor.Clip = new Rectangle(this.Location, this.Size);

//Zmniejszamy aktualną pozycję X o wartość przesunięcia

currentXPos -= (currentXPos - xValue);

}

//Analogicznie, gdy współrzędna X joysticka jest większa od aktualnej pozycji X

else if (currentXPos < xValue)

{

this.Cursor = new Cursor(Cursor.Current.Handle);

Cursor.Position = new Point(Cursor.Position.X + (currentXPos / 1000), Cursor.Position.Y);

Cursor.Clip = new Rectangle(this.Location, this.Size);

currentXPos += (xValue - currentXPos);

}

//Analogicznie, gdy współrzędna Y joysticka jest mniejsza od aktualnej pozycji Y

else if (currentYPos > yValue)

{

this.Cursor = new Cursor(Cursor.Current.Handle);

Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y - currentYPos / 1000);

Cursor.Clip = new Rectangle(this.Location, this.Size);

currentYPos -= (currentYPos - yValue);

}

//Analogicznie, gdy współrzędna Y joysticka jest większa od aktualnej pozycji Y

else if (currentYPos < yValue)

{

this.Cursor = new Cursor(Cursor.Current.Handle);

Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y + (currentYPos / 1000));

Cursor.Clip = new Rectangle(this.Location, this.Size);

currentYPos += (yValue - currentYPos);

}

});

}

}

private int currentXPos;

private int currentYPos;

private JoystickState state;

//Metoda służąca do pobierania w określonych odstępach czasu stanu joysticka

private void GetJoystickState()

{

//Pobieramy startową pozycję osi X i Y

joystick.Poll();

// Przypisujemy wartości osi do zmiennych

currentXPos = joystick.CurrentJoystickState.X;

currentYPos = joystick.CurrentJoystickState.Y;

while (true)

{

Thread.Sleep(20);

try

{

// pobieramy dane z joysticka

joystick.Poll();

// aktualizujemy pole aktualnym stanem joysticka

state = joystick.CurrentJoystickState;

//Przypisujemy wartości zmiennym, za pomocą których ustawiane są TextBoxy w interfejsie graficznym

xValue = state.X;

yValue = state.Y;

//Wywołujemy metodę, która przekazuje dane (współrzędne X i Y) do wątku głównego, w którym pracuje interfejs graficzny

SetValues(state.X, state.Y);

}

catch (Exception err)

{

}

}

}

//Metoda odpowiedzialna za przekazanie danych do wątku głównego (interfejsu graficznego)

private void SetValues(int x, int y)

{

//Za pomocą delegaty przekazujemy dane z naszego wątku pobierającego stan joysticka do wątku głównego

this.Invoke((MethodInvoker)delegate

{

//Ustawiamy współrzędne textboxów

xTextBox.Text = x.ToString();

yTextBox.Text = y.ToString();

//Przekazujemy również stan przycisku

if (state.GetButtons()[0] >= 128)

{

buttonClickedTextBox.BackColor = Color.Green;

buttonValue = true;

}

else

{

buttonClickedTextBox.BackColor = Color.Red;

buttonValue = false;

}

});

}

private void InitJoystick()

{

//W pętli foreach znajdowane są wszystkie podłączone urządzenia, które są należą do klasy urządzeń GameControl

foreach (

DeviceInstance di in

Manager.GetDevices(

DeviceClass.GameControl,

EnumDevicesFlags.AttachedOnly))

{

//Tworzona jest nowa instancja urządzenia

joystick = new Device(di.InstanceGuid);

//Ustawiamy TextBox, oraz podświetlenie informujące o tym, że joystick został znaleziony i podłączony

connectionStatusTextBox.Text = "Joystick Polaczony";

connectionStatusTextBox.BackColor = Color.Green;

//Pobieramy i ustawiamy nazwę joysticka

joystickNameTextBox.Text = di.InstanceName;

//Ustawiamy parametry połączenia (między innymi aby joystick działał, gdy okno będzie nieaktywne)

joystick.SetCooperativeLevel(this, CooperativeLevelFlags.Background | CooperativeLevelFlags.NonExclusive);

//Ustawiamy typ urządzenia jako joystick

joystick.SetDataFormat(DeviceDataFormat.Joystick);

//Potwierdzamy i zapisujemy dane o urządzeniu

joystick.Acquire();

break;

}

//Jeżeli nie znaleziono joysticka

if (joystick == null)

{

//Ustawiamy odpowiedni komunikat

connectionStatusTextBox.Text = "Joystick Nie Polaczony";

connectionStatusTextBox.BackColor = Color.Red;

joystickNameTextBox.Text = "";

return;

}

}

private void PopulateAllDevices(TreeView tvDevices)

{

//Add "All Devices" node to TreeView

TreeNode allNode = new TreeNode("All Devices");

tvDevices.Nodes.Add(allNode);

//Populate All devices

foreach (DeviceInstance di in Manager.GetDevices(DeviceClass.All, EnumDevicesFlags.AllDevices))

{

//Get Device name

TreeNode nameNode = new TreeNode(di.InstanceName);

//Is device attached?

TreeNode attachedNode = new TreeNode(

"Attached = " +

Manager.GetDeviceAttached(di.ProductGuid));

//Get device Guid

TreeNode guidNode = new TreeNode(

"Guid = " + di.InstanceGuid);

//Add nodes

nameNode.Nodes.Add(attachedNode);

nameNode.Nodes.Add(guidNode);

allNode.Nodes.Add(nameNode);

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void timer1\_Tick(object sender, EventArgs e)

{

}

private void backgroundWorker1\_DoWork(object sender, System.ComponentModel.DoWorkEventArgs e)

{

}

private void elementHost1\_ChildChanged(object sender, System.Windows.Forms.Integration.ChildChangedEventArgs e)

{

}

private void buttonClickedTextBox\_TextChanged(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

Environment.Exit(Environment.ExitCode);

}

}

}