

Masterprojekt - Bericht

Sommersemester 2010

Praktikumsbericht

ERSTELLEN EINER **GUI** FÜR DEN ROBOCUP RESCUE ROBOTER

Autor / Matrikelnummer:

Bernhard KREIL, BSc / S0910564052

Gruppe:

INIF

Projektbetreuer:

DI (FH) Raimund EDLINGER

Projektzeitraum:

1. März 2010 – 7. Juli 2010

Bericht-Abgabetermin:

3. September 2010

Inhaltsverzeichnis

1. Ziel des Praktikums	3
2. Erreichte Ziele	4
3. Verwendete Entwicklungs-Software	4
4. Aufbau der GUI	5
5. Input-Devices.....	6
6. Software-Dokumentation	6
7. Erweiterungsmöglichkeiten	11
8. Zusätzliche Informationen	11

Abbildungsverzeichnis

Abbildung 1: Main-Window	5
Abbildung 2: Settings-Mask	5
Abbildung 3: Tastenbelegung	6

1. Ziel des Praktikums

Ziel dieses Projektes ist die Entwicklung einer Software zur Visualisierung von Sensordaten, wie z.B. CO2-Sensor, Live-Kamera, Thermokamera, Lage und Position des Roboters. Weiters soll die Möglichkeit bestehen, dass der Roboter per Joystick gesteuert wird und hier die wichtigsten Steuersignale implementiert werden.

Die Visualisierung des bereits vorhandenen 2D-Mappings steht ebenfalls im Vordergrund, wie die einwandfreie Kommunikation über WLAN (802.11.a). Deshalb sollten auch Überlegungen angestrebt werden, wie die Schnittstellen zwischen Roboter und Zentraleinheit definiert sind.

Als Fortsetzung dieser Aufgabenstellung ist im Rahmen weiterer Projekte im 3.Semester Master und mit anschließender Masterarbeit eine Erweiterung des entwickelten Konzeptes denkbar.

Arbeitsumfang:

- Literaturstudium über GUI
- Erstellung einer geeigneten Software mit C# oder C++
- Verwendung von OPEN CV
- Testfahrten mit Mark10 oder Schrotty
- Erstellen eines Praktikumsberichts

2. Erreichte Ziele

- Modular Aufgebaute und für zukünftige Änderungen am Roboter einfach anpassbare GUI für die Steuerung von Mark10 oder Schrotty sowie der Visualisierung von Sensordaten (IMU, CO2, Akkus, ...)
- 3D-Maus (3D Connexion SpacePilot Pro) oder wahlweise Joystick (Logitech Force 3D Pro) als Input Device für Steuerung implementiert
- Datenkommunikation via WLAN (802.11a) implementiert
- Erfolgreiche Testfahrt mit Mark10

Somit wurden die geforderten Ziele erreicht und einem Erfolg bei der kommenden Rescue Robot League sollte nichts Größeres mehr im Weg stehen.

3. Verwendete Entwicklungs-Software

- Für die Erstellung der GUI wurde Windows Forms gewählt und als Programmiersprache Microsoft Visual C# verwendet.
- Als IDE wurde Microsoft Visual Studio 2008 verwendet und die GUI für das .NET Framework 3.5 für x86 Rechner erstellt.
- Für die 3D-Visualisierung der IMU-Sensordaten wurde das Microsoft XNA Game Studio 3.1 benötigt.
- Um den Joystick als Input-Device zu implementieren (Funktion durch 3D-Maus abgelöst) wurde das Microsoft Direct X SDK (February 2010) verwendet.

4. Aufbau der GUI

Die GUI unterteilt sich in das Main-Window und der modalen Settings-Mask:

Main:

Das Main-Window dient zur Visualisierung der Sensordaten, 2D Maps, Live-Bild, ... sowie zum Starten/Beenden der WLAN Kommunikation.

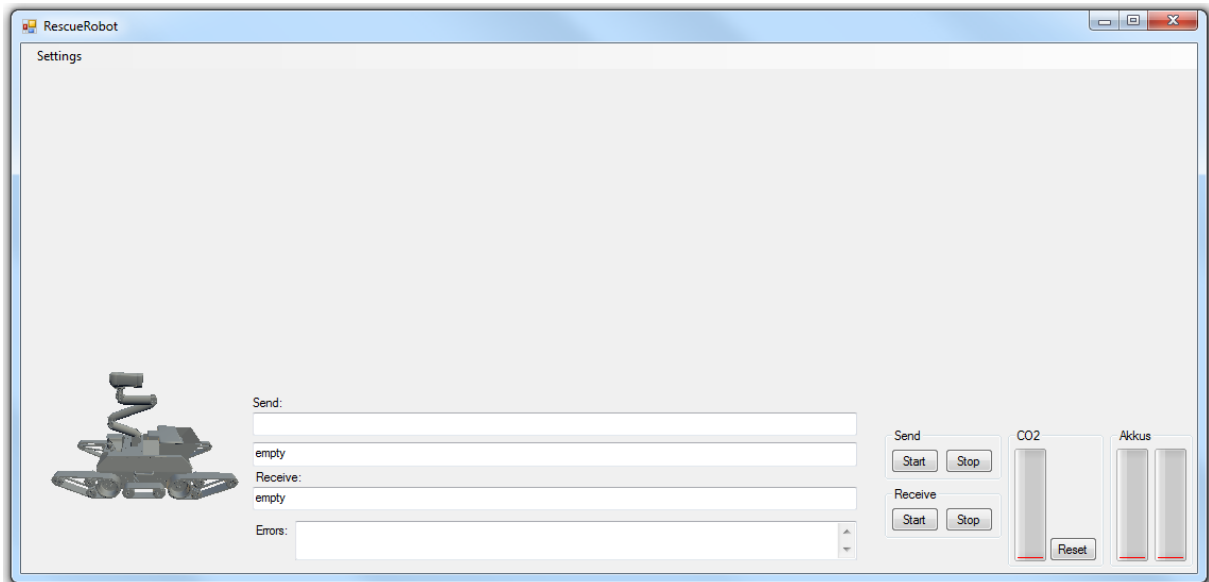


Abbildung 1: Main-Window

Settings:

Der modale Settings-Dialog erlaubt die Auswahl des zu ladenden Roboter-CAD-Modells (*.x Format) für die IMU Visualisierung. Wird die 3D-Maus nicht als Steuergerät verwendet kann der Joystick als Alternative gewählt werden. In den Feldern Control Communication werden IP sowie Ports für die Kommunikation zwischen Operator GUI und dem EMX Board auf dem jeweiligen Roboter eingestellt.

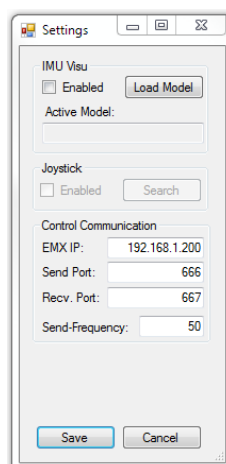


Abbildung 2: Settings-Mask

5. Input-Devices

Nachfolgend ist die Tastenbelegung der 3D-Maus gegeben. Die jeweilige Aktion eines Buttons wird in einem String als 1/0 via WLAN an das EMX Board am Roboter übertragen und dort die jeweiligen Aktoren geschaltet.

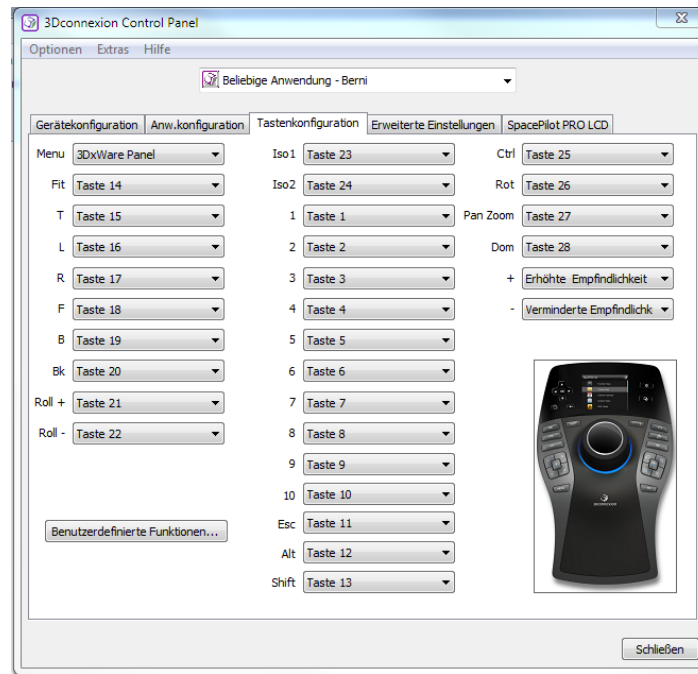


Abbildung 3: Tastenbelegung

6. Software-Dokumentation

Der Quellcode der erstellten Software ist durchgehend, verständlich und nachvollziehbar. Dokumentiert weswegen in diesem Bericht nicht auf den gesamten Code eingegangen werden muss.

FrmSettings.cs:

Die Klasse FrmSettings.cs dient um Roboterspezifische Einstellungen vorzunehmen. Die Settings-Mask wird vom Main-Window mit frmSettings.ShowDialog() als modales Dialogfeld gestartet und bei einem DialogResult.OK werden die Einstellungen im Main-Window übernommen:

```
// Settings Form öffnen und eingestellte Werte übernehmen
private void settingsToolStripMenuItem_Click(
    object sender, EventArgs e)
{
    try
    {
        if (this.frmSettings.ShowDialog() == DialogResult.OK)
        {
            this.ip_EMX = frmSettings.IP_EMX;
        }
    }
}
```

```

        this.portS_EMX = frmSettings.PortS_EMX;
        this.portR_EMX = frmSettings.PortR_EMX;
        this.sendeZyklus = frmSettings.SendeZyklus;

        this.imuEnabled = frmSettings.ImuEnabled;
        if (imuEnabled)
        {
            if (this.imuModelPath != frmSettings.ImuModelPath
                && File.Exists(frmSettings.ImuModelPath))
            {
                this.imuModelPath = frmSettings.ImuModelPath;
                LoadModel(this.imuModelPath);
            }
        }
        else
        {
            // Unload any existing model.
            modelViewerControll.Model = null;
            contentManager.Unload();
        }
    }
}
catch (Exception ex)
{
    tbxErrorLog.AppendText(ex.Message);
}
}

```

FrmMain.cs:

Die Klasse FrmMain.cs erbt von Windows.Forms.Form und enthält das eigentliche Hauptprogramm.

- Für die 3D Maus als Input Device wird ein Verweis auf die Bibliothek TDx.TDxInput.dll benötigt. Diese ist die .Net Schnittstelle von 3D Connexion SpacePilot Pro:

```

private TDx.TDxInput.Sensor sensor;
private TDx.TDxInput.Keyboard keyboard;
private TDx.TDxInput.Device device;
. . .
private void SearchSpacePilot()
{
    try
    {
        this.device = new TDx.TDxInput.Device();
        sensor = this.device.Sensor;

        this.keyboard = this.device.Keyboard;

        this.lastUpdate = System.DateTime.Now;

        SetDeviceText();
        SetMotionTexts();
    }
}

```

```

        // Add the event handlers
        this.device.DeviceChange += (this.device_DeviceChange);
        this.sensor.SensorInput += (this.sensor_SensorInput);
        this.keyboard.KeyDown += (this.keyboard_KeyDown);
        this.keyboard.KeyUp +=(this.keyboard_KeyUp);

        //Connect everything up
        this.device.Connect();
    }
    catch (COMException e)
    {
        Console.WriteLine("{0} Caught exception #1.", e);
    }
}

```

In der Funktion SetDeviceText() Funktion wird der String welcher die aktuellen Status der Buttons darstellt in eine Hexadezimalzahl konvertiert und die Funktion SetMotionTexts() erstellt den gesamten Steuerungsstring der vom Operator Computer an den jeweiligen Roboter gesendet wird.

- Die Datenübertragung wurde mit Sockets realisiert:

```

//Daten Senden an EMX - Steuerung
private IPAddress ip_EMX = IPAddress.Parse("192.168.1.200");
private int portS_EMX = 666;
private int sendeZyklus = 50;
private Socket outSocket;
private bool senden = false;
private bool co2Reset = false;

//Daten Empfangen von EMX - Steuerung
private bool empfangen = false;
private int portR_EMX = 667;
string empfangenString = "empty";
byte[] inBuffer = new byte[1024];
Socket listeningSocket;

```

Das senden/empfangen wird in je einem eigenen Thread welcher vom ThreadPool verwaltet wird verarbeitet:

```

private void btnStartSend_Click(object sender, EventArgs e)
{
    //starte den Sendethread
    ThreadPool.QueueUserWorkItem(new WaitCallback(sendeThread));
    btnStartSend.Enabled = false;
}

. . .

void sendeThread(Object stateInfo)
{
    senden = true;
    try
    {
        while (senden)
        {
            //Wird gesendet wenn sich Steuerungsstring geändert hat
            if (steuerungString != steuerungStringAlt)
            {

```

```

//Using wird verwendet um den Socket wieder sicher
zu schließen
using (outSocket = new Socket(
    AddressFamily.InterNetwork,
    SocketType.Stream,
    ProtocolType.Tcp))
{
    //Verbinden zum Embedded Masters Modul
    outSocket.Connect(new IPEndPoint(ip_EMX,
                                    portS_EMX));

    lock (steuerungString)
    {
        //Erstellen der Nachricht
        string outText = steuerungString;

        byte[] messageBytes =
            Encoding.UTF8.GetBytes(outText);
        //Senden der Nachricht

        outSocket.Send(messageBytes);
    }

    if (co2Reset)
    {
        co2Reset = false;
    }
    steuerungStringAlt = steuerungString;
}
}

...

//While-Schleife bremsen
Thread.Sleep(sendeZyklus);
}
}

catch (Exception ex)
{
    Debug.WriteLine(ex.ToString());
}
}

```

- Das EMX Board sendet laufend Daten von Sensoren an die Operator-Station welche im Thread empfangsThread verarbeitet werden:

```

void empfangsThread(Object stateInfo)
{
    empfangen = true;
    while (empfangen)
    {
        try
        {
            using (listeningSocket = new Socket(
                AddressFamily.InterNetwork,
                SocketType.Stream,
                ProtocolType.Tcp))
            {
                listeningSocket.Bind(new IPEndPoint(IPAddress.Any,
                                                    portR_EMX));
                listeningSocket.Listen(10);
            }
        }
    }
}

```


7. Erweiterungsmöglichkeiten

- Übertragen und Darstellen des Live-Bildes der Kamera(s) in der erstellten GUI auf dem bereits vorhandenen WLAN Kommunikationsweg.
- Übertragen und Darstellen der 2D Map auf dem bereits vorhanden WLAN Kommunikationsweg.
- Steuerung für Roboterarm (7 axis+ grabber - noch nicht vorhanden) implementieren.

8. Zusätzliche Informationen

- GUI wurde auf Windows 7 64bit entwickelt, für .Net 3.5 x86 kompiliert und auf XP, Vista sowie 7 32bit getestet.
- Für korrekte Darstellung der GUI-Elemente dürfen Visuelle-Stile nicht deaktiviert sein.