Dokumentation Remote Car

Maxi Mahl, Felix Bindewald, Max Haag

Planungsphase/Ideen:

Das Remote Car unterlief von Anfang bis Ende unserer Durchführung auf Grund von nicht vorhersehbaren Problemen mehrere Phasen:
Anfangs schwebte uns ein einfaches, kleines Rennauto vor das mit einer Kamera ausgestattet mithilfe eines (Android) Handys über Bluetooth gesteuert werden konnte. Die Idee mit der Kamera verworfen wir jedoch schnell wieder als wir feststellten, dass alleine schon die Steuerung des Autos ein Problem für sich war. Nachdem wir mehrere Tests mit eher weniger geeigneten Funkmodulen durchführten, einigten wir uns schlussendlich darauf, die Original Funkmodule aus der Steuerung eines anderen Modellautos zu nehmen und diese einfach in unseres hineinzubauen. Nachdem diese aber auf unerklärlicher weise den Geist aufgaben, nutzen wir schlussendlich die USB Schnittstelle des Arduinos als Steuerung.

Problematik:

- Die Idee mit der Kamera wurde recht schnell verworfen als sich die Steuerung des Autos als größeres Problem herausstellte als wir dachten.
- Idee, Auto über Bluetooth mit Handy zu steuern wurde verworfen, da kein geeignetes Funkmodul zu finden war und sich die Programmierung als App als ein zu langwieriges Problem herausstellte.
- Gab Anfangs Probleme mit den Motoren, da wir erst herausfinden mussten mit welchen Impulsen wir diese Ansteuern und deren Geschwindigkeit regulieren konnten.
- Ein Teil des Funkmoduls, das wir bestellt hatten, erwies sich nach mehreren Tests als defekt und nicht verwendbar.
- Anfangs Probleme, mit Hilfe der Arduinosoftware die Motoren anzusteuern. Arduino erwies sich als zu ungenau, um die Motoren direkt anzusteuern.
- 2 Mikrocontroller mussten auf Grund von falscher Verkabelung des Arduinos dran glauben :)
- Mehrere unerklärliche Probleme, die dazu führten das Motoren sich nicht ansteuern liesen bzw. Kommunikation zwischen Motoren und Arduino nicht 100%ig funktionierte.
- Nachdem wir endlich mit Hilfe einer andere Steuerung für ein Modellbauauto funktionierende Funkmodule in die Hand bekamen erwiesen diese sich nach mehrmaliger Benutzung als defekt.

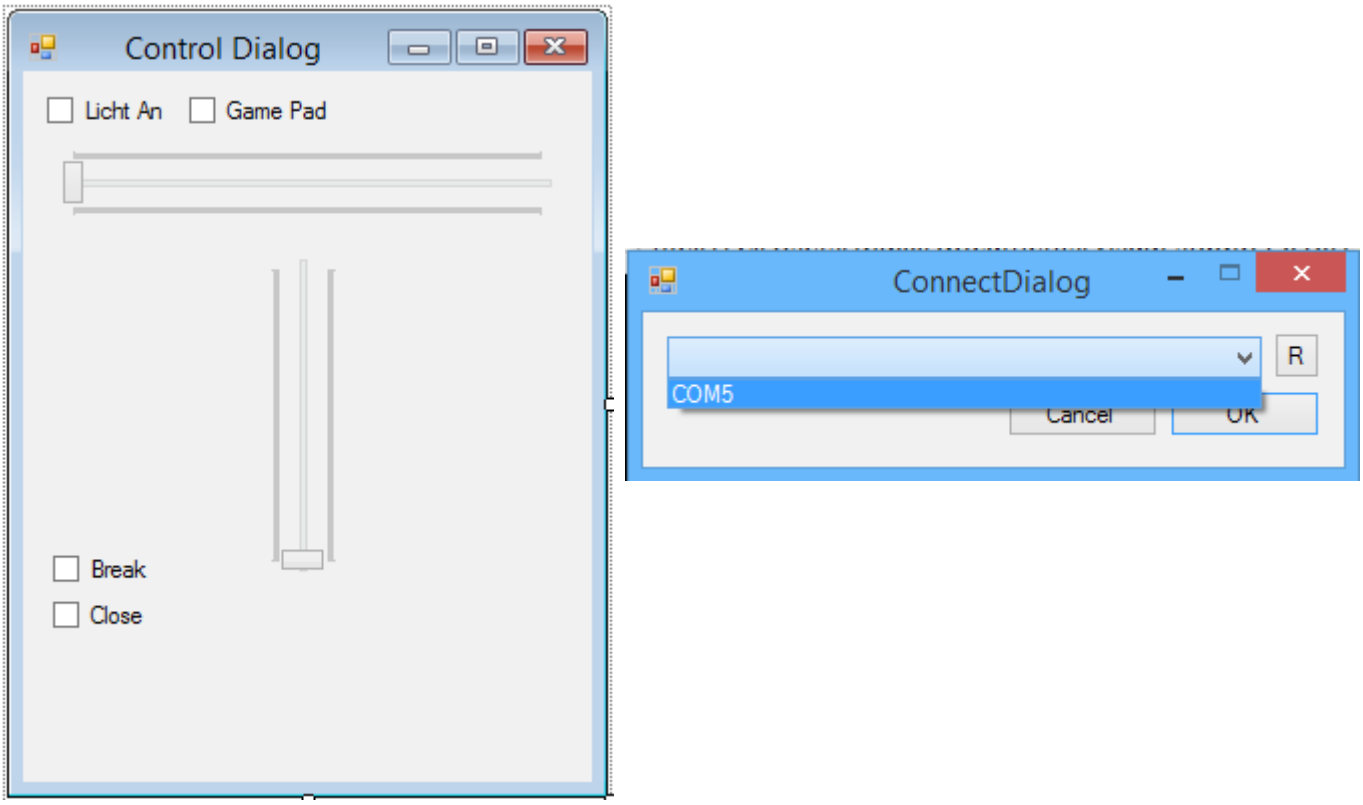
Programmierung

Momentan empfängt der Arduino über USB die Daten, welche er dann an die Motoren weitergibt. Die Motoren M1 und M2 sind für den Antrieb da, an ihnen hängt ein Steuergerät, mit welchem man diese wie Servomotoren ansteuern kann. Nur dass man an stelle des Winkels, die Geschwindigkeit übergibt. Der Servo ist zum Lenken da.
Der folgende Code ist der Code für den Arduino:

```
#include <Servo.h>
// Ports für die Motoren definieren
#define SPIN 2
#define M1PIN 3
#define M2PIN 4

int i = 0;
Servo servo;
Servo m1;
Servo m2;

void setup(){
  Serial.begin(9600);         // Seriellen Übertragung starten;
  pinMode(13,OUTPUT);         // Pin 13 auf Ausgang für LED setzten
  servo.attach(SPIN);         // Servos Initialisieren
  m1.attach(M1PIN);
  m2.attach(M2PIN);
}

void loop(){
  int bytes=Serial.available();// Überprüfen, wie viele Bytes verfügbar sind
  byte buffer[] = {0,0,255};// Den Buffer für die Serielle Übertragung
erstellen
  for(int i=0;i<0+bytes;i++){// Bytes der reihe nach aus Seriellem Port lesen
    buffer[i]=Serial.read();
  }
  if(sizeof(buffer) > 0 && buffer[0] != 0)// Nach Übertragungsfehler Überprüfen
  {
    servo.write(buffer[0]);// Lenkeinschlag setzten
  }
  if(sizeof(buffer) > 1 && buffer[0] != 0)// Nach Übertragungsfehler Überprüfen
  {
    m1.write(buffer[1]);// Geschwindigkeiten der Motoren setzten
    m2.write(buffer[0]);
  }
  if(sizeof(buffer) > 2 && buffer[2] < 255)// Nach Übertragungsfehler
Überprüfen
  {
    digitalWrite(13,buffer[2]);// LED setzten
  }

  Serial.print(buffer[0]);// Debug Infos an PC Senden
  Serial.print("-");
  Serial.print(buffer[1]);
  Serial.print("\n");

  delay(100);
}
```

Maxi Mahl, Felix Bindewald, Max Haag

Die Steuerung, die nebenher auf dem PC läuft ist in C# geschrieben. Bei ihr kann man die Geschwindigkeit und die Lenkung Wahlweise über Regler oder einem Gamepad steuern. IM folgenden sind nur die für die Übertragung und die Gamepad Abfrage wichtigen Codezeilen aufgeschrieben. Code zum erstellen der Fenster und Anzeigen der Fenster ist nicht dabei.



Links: Steuerfenster, Rechts: Auswahlfenster für Ports

Code 1 Auflisten der Seriellen Ports:

```
// Alle bereits vorhandenen Items auf der Liste löschen.
portList.Items.Clear();
// Alle verfügbaren Ports in Array schreiben
string[] ports = SerialPort.GetPortNames();
foreach (string port in ports)
portList.Items.Add(port); // Portname zur Liste hinzufügen
```

Code 2 Senden der Daten

Globale Variablen:

```
public const float MaxAxis = 65535;

public static SerialPort port = null;
Stopwatch stopwatch = new Stopwatch(); // Stoppuhr um Zeiten zu messen
DirectInput dInput;                    // Für die Joystick abfragen
Joystick joyDevice = null;             // benötigte Objekte
JoystickState joyState;
```

Initialisieren des Gamepads:

```
// Alle am PC angeschlossenen Gamecontroller auflisten
int deviceCount = dInput.GetDevices(DeviceClass.GameController,
DeviceEnumerationFlags.AttachedOnly).Count;
// Überprüfen ob ein Gamecontroller angeschlossen ist
if (deviceCount > 0)
{
    // Ersten Gamecontroller auswählen und Aktivieren
    DeviceInstance dev = dInput.GetDevices(DeviceClass.GameController,
DeviceEnumerationFlags.AttachedOnly)[0];
    joyDevice = new Joystick(dInput, dev.InstanceGuid);
    joyDevice.Acquire();

    // Gamepad Option verfügbar schalten
    cbGamePad.Enabled = true;
}
```

Initialisieren des Seriellen Ports:

```
// Port öffnen
port = new SerialPort(ConnectDialog.port, 9600);
connected = true;
```

Dauerschleife zum Übertragen der Daten: (wird dauerhaft im Hintergrund ausgeführt)

```
// Stoppuhr zum Messen der verstrichenen Zeit starten
stopwatch.Restart();
byte[] buffer = new byte[3];
if (!cbGamePad.Checked) // Überprüfen, ob auf Gamepadsteuerung aktiviert ist
{
  buffer[0] = (byte)tbLenkung.Value;            // Lenkeinschlag auf erstes Byte setzten
  buffer[1] = (byte)tbMotor.Value;              // Geschwindigkeit auf zweites Byte setzten
  buffer[2] = (byte)(checkBox1.Checked ? 1 : 0);// Licht auf drittes Byte setzten
}
else
{
  joyState = joyDevice.GetCurrentState();
  // Joystickeinschlag als erstes Byte schreiben
  buffer[0] = (byte)(1 + joyState.X / MaxAxis * 254f);
  // Auf zweites Byte schreiben, ob Knopf gedrückt ist
  buffer[1] = (byte)(joyState.GetButtons()[0] ? 255 : 1);
  buffer[2] = (byte)(checkBox1.Checked ? 1 : 0);
}
port.Write(buffer, 0, 3);                        // Bytes an Arduino übertragen

// Warten bis 100 Millisekunden verstrichen sind
while (stopwatch.ElapsedMilliseconds < 100) { Thread.Sleep(1); }
```

Komplett

Hier ist der Code noch einmal fast komplett. Ich habe den Teil, für die Fenster weg gelassen, da C# diesen automatisch erstellt.

Program.cs:

```
using System;
using System.Windows.Forms;
using System.Threading;

namespace Serial
{
    static class Program
    {
        static bool running = true;

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            // Neune Thread erstellen und Starten
            Thread thread = new Thread(delegate() { background(); });
            thread.Start();

            // Das Fenster zum Einrichten der Verbindung mit dem Arduino anzeigen
            Application.Run(new ConnectDialog());
            // Falls nicht 'Abbrechen' gedrückt wurde, das Hauptfenster anzeigen
            if(ConnectDialog.port != null)
                Application.Run(new Form1());

            // 'running' auf false setzten, damit die Hintergrundschleife stoppt.
            running = false;
        }

        static void background()
        {
            while (running)
            {
                // Abfragen ob ein Port geöffnet wurde
                if (Form1.port != null && Form1.port.IsOpen)
                {
                    // Auf Antwort des Arduinos warten und diese in die Konsole schreiben
                    try
                    {
                        byte[] buffer = new byte[1];
                        Form1.port.Read(buffer, 0, 1);
                        Console.Write((char)buffer[0]);
                    }
                    catch (Exception ex)
                    {
                        // Bei einem Fehler, die Fehlermeldung in die Konsole schreiben
                        Console.ForegroundColor = ConsoleColor.Red;
                        Console.WriteLine(ex.Message);
                        Console.ResetColor();
                    }
                }
            }
        }
    }
}
```

ConnectDialog.cs:

```
using System;
using System.Text;
using System.Windows.Forms;

using System.IO.Ports;

namespace Serial
{
    public partial class ConnectDialog : Form
    {
        public static string port;

        public ConnectDialog()
        {
            // Fenster und Steuerelemente initialisieren
            InitializeComponent();

            // Serielle Ports auflisten und in eine "Combo Box" schreiben
            foreach (string port in SerialPort.GetPortNames())
                portList.Items.Add(port);
        }

        // Wenn der Button 'button1' gedrückt wird, wird die Aktuelle Port-Auswahl
        // in die öffentliche Variable 'port' geschrieben
        private void button1_Click(object sender, EventArgs e)
        {
            port = portList.Text;
            this.Close();
        }

        // Wird ausgeführt, wenn der Button 'btnRefresh' gedrückt wird
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            // Alle bereits vorhandenen Items auf der Liste löschen.
            portList.Items.Clear();
            // Alle verfügbaren Ports in Array schreiben
            string[] ports = SerialPort.GetPortNames();
            foreach (string port in ports)
                portList.Items.Add(port); // Portname zur Liste hinzufügen
        }

        // Wird beim Drücken des Buttons 'btnCancel' ausgefürt
        private void btnCancel_Click(object sender, EventArgs e)
        {
            port = null;
            this.Close();
        }
    }
}
```

Form1.cs (Control Dialog):

```
using System;
using System.Windows.Forms;

using System.IO.Ports;
using System.Diagnostics;
using System.Threading;

// Namespace in dem Klassen zum Verwenden von Direct X enthalten sind
using SlimDX.DirectInput;
```

```
namespace Serial
{
    public partial class Form1 : Form
    {
        public const float MaxAxis = 65535;

        public static SerialPort port = null;
        Stopwatch stopwatch = new Stopwatch(); // Stoppuhr um Zeiten zu messen
        DirectInput dInput;                    // Für die Joystick abfragen
        Joystick joyDevice = null;             // benötigte Objekte
        JoystickState joyState;

        public Form1()
        {
            InitializeComponent();
            dInput = new DirectInput();
            // Alle am PC angeschlossenen Gamecontroller auflisten
            int deviceCount = dInput.GetDevices(DeviceClass.GameController,
DeviceEnumerationFlags.AttachedOnly).Count;
            // Überprüfen ob ein Gamecontroller angeschlossen ist
            if (deviceCount > 0)
            {
                // Ersten Gamecontroller auswählen und Aktivieren
                DeviceInstance dev = dInput.GetDevices(DeviceClass.GameController,
DeviceEnumerationFlags.AttachedOnly)[0];
                joyDevice = new Joystick(dInput, dev.InstanceGuid);
                joyDevice.Acquire();

                // Gamepad Option verfügbar schalten
                cbGamePad.Enabled = true;
            }
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
            Control.CheckForIllegalCrossThreadCalls = false;

            bool connected = false;
            while (!connected)
            {
                try
                {
                    // Port öffnen
                    port = new SerialPort(ConnectDialog.port, 9600);
                    connected = true;
                }
                catch (Exception ex)
                {
                    // Bei Verbindungsfehler eine neue Porteingabe verlangen
                    port = null;
                    MessageBox.Show(ex.Message, "Error while connecting...",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    new ConnectDialog().ShowDialog();
                    if (ConnectDialog.port == null)
                        Process.GetCurrentProcess().Kill();
                }
            }

            // Hintergrundschleife Starten
            Thread thread = new Thread(delegate() { backgroundLoop(); });
            thread.Start();
        }
```

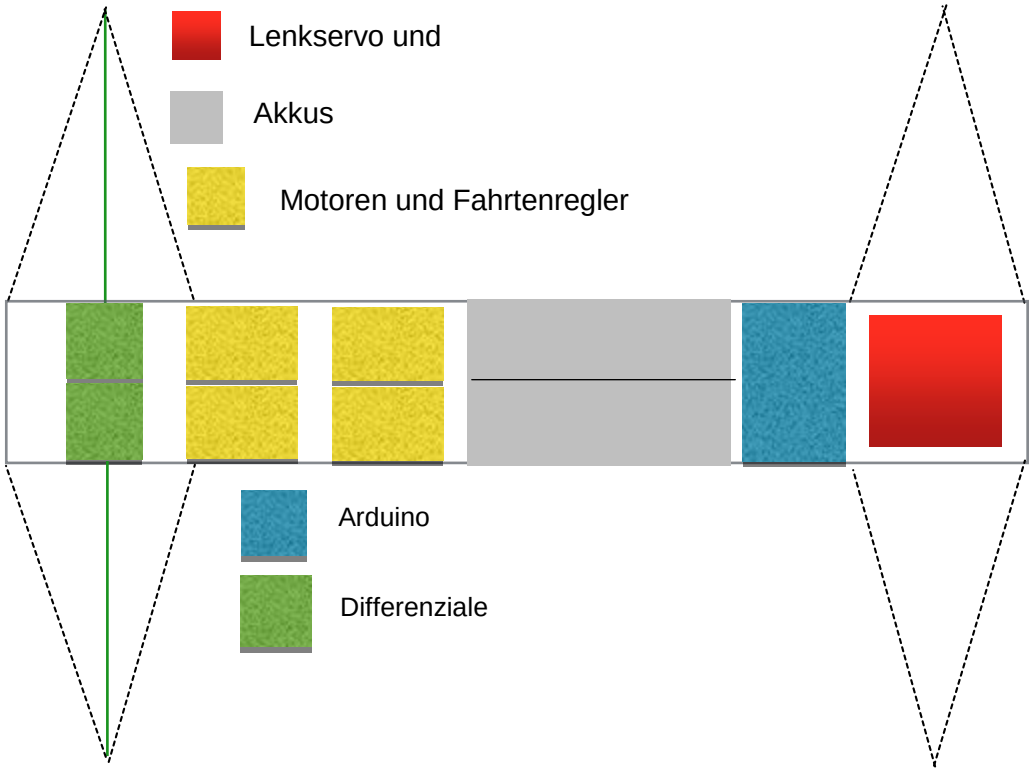
```
        void backgroundLoop()
        {
            port.Open();
            while (this.Visible)
            {
                // Testen ob Port geöffnet ist
                if (port.IsOpen && !cbBreak.Checked)
                {
                    try
                    {
                        // Stoppuhr zum Messen der verstrichenen Zeit starten
                        stopwatch.Restart();
                        byte[] buffer = new byte[3];
                        if (!cbGamePad.Checked) // Überprüfen, ob auf Gamepadsteuerung
gesetzt ist
                        {
                            buffer[0] = (byte)tbLenkung.Value;// Lenkeinschlag auf erstes
Byte setzten
                            buffer[1] = (byte)tbMotor.Value; // Geschwindigkeit auf zweites
Byte setzten
                            buffer[2] = (byte)(checkBox1.Checked ? 1 : 0);// Licht auf
drittes Byte setzten
                        }
                        else
                        {
                            joyState = joyDevice.GetCurrentState();
                            // Joystickeinschlag als erstes Byte schreiben
                            buffer[0] = (byte)(1 + joyState.X / MaxAxis * 254f);
                            // Auf zweites Byte schreiben, ob Knopf gedrückt ist
                            buffer[1] = (byte)(joyState.GetButtons()[0] ? 255 : 1);
                            buffer[2] = (byte)(checkBox1.Checked ? 1 : 0);
                        }
                        port.Write(buffer, 0, 3); // Bytes an Arduino übertragen

                        // Warten bis 100 Millisekunden verstrichen sind
                        while (stopwatch.ElapsedMilliseconds < 100) { Thread.Sleep(1); }
                    }
                    catch (Exception ex)
                    {
                        // Bei Fehlern, Nachricht in die Konsole schreiben
                        Console.ForegroundColor = ConsoleColor.Yellow;
                        Console.WriteLine(ex.StackTrace);
                        Console.ResetColor();
                    }
                }
            }
            port.Close();
        }

        // Beim Ändern eines Wertes für die Lenkung, neuen Wert in einem Label ausgeben
        private void trackBar1_Scroll_1(object sender, EventArgs e)
        {
            lblLenkung.Text = tbLenkung.Value.ToString();
        }

        // Beim Ändern eines Wertes für die Geschwindigkeit, neuen Wert in einem Label
ausgeben
        private void tbMotor_Scroll(object sender, EventArgs e)
        {
            lblMotor.Text = tbMotor.Value.ToString();
        }

        // Verbindung vorübergehend Trennen, wenn 'cbClose' markiert wird
        // Die Verbindung zu Trennen ist wichtig, wenn der Arduino neu beschrieben wird
```

```
        private void cbClose_CheckedChanged(object sender, EventArgs e)
        {
            if (cbClose.Checked)
                port.Close();
            else
                port.Open();
        }
    }
}
```

Maxi Mahl, Felix Bindewald, Max Haag

Bau des Autos

Anfänglich war mein Plan das Auto so zu Bauen wie unten dargestellt. Diese Bauweise erwies sich dann als zu klein und ich habe angefangen die Bodenplatte größer zu machen.



Anfangs hat alles gut geklappt. Als dann jedoch die vordere Aufhängung fertig war, war das erste Problem, dass die Stoßdämpfer zu weich waren. Jedoch habe ich dann schnell härtere Ersatzstoßdämpfer bestellt und diese Problem war gelöst. Das nächste Problem war die hintere Aufhängung. Das Problem ist leider noch nicht gelöst.