INDINF_7.1

Verfasser: **Leonhard Stransky, 4AHIT**

Datum: **03.10.2023**

Voraussetzungen:

Im 3. Jahrgang hast du schon Kontakt mit einem Industrieroboter gehabt. Dabei sollten folgende Ziele erreicht worden sein:

- Du hast die Sicherheitseinweisung am spezifischen Roboter mitgemacht.
- Du kannst den Roboter grundlegend in Betrieb nehmen und einfache Meldungen des Systems bearbeiten.
- Du kannst alle Achsen des Roboters benennen und per Handbediengerät bewegen.
- Du kennst die Bewegungseinschränkungen und Limits aller Achsen und kannst den TCP gezielt an einen Punkt im Raum steue

Ziele:

- Du kennst die verschiedenen Koordinatensysteme, die bei Roboterbewegungen eingesetzt werden, und die Gründe dafür.
- Du kannst eigene Werkzeug- und Basis-Koordinatensysteme definieren und Roboterbewegungen sinnvoll einsetzen.
- Du kannst ein einfaches Programm zum Abfahren eines Pfades schreiben, ausführen und testen.

Projektteam:

- Julian Neuwirth
- Duchon Kevin
- Bakshi Pavel

Aufgaben:

Teil 1 (GK):

Aufgabenstellung:

Schüler 1: 1-->4, 2-->4 (Würfel 2 horizonal greifen) Schüler 2: 2-->5, 3-->5 (Würfel 3 horizonal greifen)
Schüler 3: 3-->4, 2-->5 (Würfel 2 horizonal greifen) Schüler 4: 1-->5, 3-->5 (Würfel 2 horizonal greifen)

Vermiss die durch das blaue Koordinatenkreuz gegebene Basis und speichere die Konfiguration unter Basis Nummer 10. Verwende dazu das bereits vermessene Werkzeug "LISE0 Greifer" (Nummer 2)

Wiederhole die Bewegungen aus 2. Welche Verbesserungen ergeben sich durch Benutzung des Basiskoordinatensystems beim Umsetzen der Würfel?

Ablauf:

Unser erster Teamversuch verlief nicht ganz nach Plan. Der Roboterarm bewegte sich zu schnell und stieß unerwartet gegen den Tisch. Das führte dazu, dass einige Teile des Lego Greifarms weggeschleudert wurden. Aber diese Teile ließen sich problemlos wieder montieren, sodass wir unsere Arbeit fortsetzen konnten.

Als Schüler 4 in unserer Gruppe hatte ich die Aufgabe, die Würfel 1 und 3 umzusetzen. Ich positionierte Würfel 1 auf Platz 5 und Würfel 3 auf Platz 5. Dabei hatte ich die Schwierigtkeit, dass ich aus der Perspektive vom Roboter steuern musste. Ansonsten lief alles andere einwandfrei.

Vermessung und Konfiguration des Koordinatensystems:

Die Nutzung eines kalibrierten Koordinatensystems führte zu einer deutlichen Verbesserung bei der Steuerung des Roboters. Wir navigierten den Roboter zu definierten Fixpunkten, um die Koordinaten einzugeben, was sich als effektiv erwies.

Das Vorhandensein eines Koordinatensystems auf der Arbeitsfläche erleichterte das Festlegen der Koordinatenpunkte und trug zu präziseren und effizienteren Roboterbewegungen bei.

Teil 2 (GK):

Aufgabenstellung:

1. Erstelle ein neues Modul mit den Name aller Schüler (ev. abgekürzt)
2. Programmiere die Bewegungsabläufe: "Teache" alle erforderlichen Bewegungspunkte. Überlege, welche Punkte sich für mehrere Bewegungen nutzen lassen und verwende sie wieder (nicht die selbe Koordinate in mehrere Variablen speichern). Benenne alle Punkte nachvollziehbar! Achte darauf, den Greifer nicht schon während der Bewegung zu öffnen bzw. zu schließen, sondern erst bei Stillstand des Roboters.
3. Verwende für Wege zwischen Würfelpositionen Point-to-Point-Bewegungen, für das Greifen der Würfel Linearbewegungen.
4. Teste das Programm ausführlich.

Ablauf:

Beim Programmieren des Roboters mussten wir oft experimentieren, besonders um die richtigen Befehle herauszufinden. Anfangs war es nicht immer einfach, aber mit der Zeit kamen wir besser zurecht. Die Herausforderung bestand darin, die Punkte zu finden, an denen der Roboter stoppen sollte.

Leider reichte die Zeit nicht aus, um das Programm komplett fertigzustellen. Wir konnten nur einen Teil der geplanten Funktionen implementieren.

Trotzdem haben wir viel über den Umgang mit dem Roboter gelernt. Für jeden Punkt, den der Roboter ansteuern sollte, haben wir zuerst manuell mit dem Steuerungsstick die Position eingestellt. So konnte der Roboterarm später genau diese Punkte anfahren.

Abschlussworte:

Im Rückblick war dieses Robotik-Projekt eine interessante Herausforderung. Wir haben uns intensiv mit der Programmierung und Steuerung des Roboters auseinandergesetzt. Trotz der Zeitbeschränkungen konnten wir einige wichtige Aspekte in der Praxis umsetzen. Es war eine gute Gelegenheit, mit dem Team zu arbeiten und gemeinsam an Lösungen zu tüfteln.

Verbesserung1 der Abgabe:

- Entfernen bzw. Kürzen der Angaben
- Ausformulierung bzw. Erweiterung des Textes
- Abschlussworte hinzugefügt
- Alle Quellen aufgelistet

INDINF_7.1

Umsetzung:

Wir haben uns für die Variante mit dem Bussystem zum Roboter entschieden. Hierbei wird ein Arduino Uno mit einem EasyCat Shield verbunden um Daten vom Mikrocontroller über ein Ethernetkabel an den Roboter zu senden.

Ein teil des Teams befasste sich mit der Implementierung von einem Distance Sensor.

Währenddessen habe ich die Online Dokumentationen von Bausano für das EasyCat Shield durchgelesen. In cer Arduino IDE habe ich dann gleich die von der Website heruntergeladene Library eingebunden und ein Beispielprogramm am Arduino Uno ausprobiert.

Das EasyCat Shield verwendet ein Protokoll namens EtherCat um Datenpackete zu senden. Diese können im Code erstellt werden oder indem man mit dem Easy Configurator eigene Entries erstellt. Die standard firmware hat 32 + 32 bytes an Input und Output Entries.

Dort habe ich dann ein Input Entry erstellt, dass ein aknowledgement signal vom Roboter sein soll. (ok uint8_t) und ein Output Entry, dass die Daten vom Distance Sensor beinhaltet. (distance float)

Diese Einstellungen musste man dann zum EEPROM des EasyCat Shields flashen. Dann wurde auch ein .h file erstellt was man in den Ordner des Arduino Projekts kopieren musste. Dieses File beinhaltet die Adressen der Entries.

Julian Neuwirth und ich haben dann ein Programm geschrieben, dass die Daten mittels dem EasyCat Shield an den Roboter sendet.

indinf_7.2.ino:

```
#include "INDINF.h"
#include "EasyCAT.h"
#include <SPI.h>


EasyCAT EASYCAT;
```

```
const int sharpPin = A0; // Der Sensor ist am analogen Pin A0 angeschlossen
float voltage;
float distance;



unsigned long PreviousMillis = 0;

void setup()
{
  Serial.begin(9600);

  Serial.print("\nEasyCAT - Generic EtherCAT slave\n");


  if (EASYCAT.Init() == true)
  {
    Serial.print("initialized");
  }
  else
  {
    Serial.print("initialization failed");
    pinMode(13, OUTPUT);
    while (1)
    {
      digitalWrite(13, LOW);
      delay(500);
      digitalWrite(13, HIGH);
      delay(500);
    }
  }

  PreviousMillis = millis();
}

void loop()
{
  EASYCAT.MainTask();

  int sensorValue = analogRead(sharpPin);
  voltage = sensorValue * (5.0 / 1023.0);

  distance = 27.86 * pow(voltage, -1.15);

  Serial.print("Entfernung: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Fügen Sie die folgende Zeile hinzu, um die Variable distance in den EtherCAT-
Datenpuffer zu schreiben.
  EASYCAT.BufferIn.Float[0] = distance;

  delay(100);
}
```

INDINF.h:

```
#ifndef CUSTOM_PDO_NAME_H
#define CUSTOM_PDO_NAME_H

//-----------------------------------------------------------------------//
//                                                                       //
//     This file has been created by the Easy Configurator tool          //
//                                                                       //
//     Easy Configurator project INDINF.prj
//                                                                       //
//-----------------------------------------------------------------------//


#define CUST_BYTE_NUM_OUT   4
#define CUST_BYTE_NUM_IN    1
#define TOT_BYTE_NUM_ROUND_OUT  4
#define TOT_BYTE_NUM_ROUND_IN   4


typedef union                                           //---- output buffer -
---
{
    uint8_t  Byte [TOT_BYTE_NUM_ROUND_OUT];
    struct
    {
        float       distance;
    }Cust;
} PROCBUFFER_OUT;


typedef union                                           //---- input buffer --
--
{
    uint8_t  Byte [TOT_BYTE_NUM_ROUND_IN];
    struct
    {
        uint8_t     ok;
    }Cust;
} PROCBUFFER_IN;

#endif
```

Jedoch leider hat der Code weder in der alten IDE noch in der neuen funktioniert.

Es war dann auch nicht mehr genug Zeit um das Problem zu lösen. Wir haben dann beim nächsten mal das Projekt an das andere Team übergeben. Dieses fand später raus, dass der code in einen custom mode gesetzt werden muss. Das stand in einem Example file der Library.

```
#define CUSTOM                      // Custom mode
```

Verbesserung2 der Abgabe:

- Code und Protokoll von INDINF_7.2 eingefügt
- Added INDINF.h file

Code:

Beim ersten Modul konnte ich den Code vom Roboter leider nicht kopieren, da ich damals nicht wusste, wie das geht. Deshalb war dieser im Bericht nicht enthalten. Ich habe auch schon meine Schulkollegen nach dem Skript gefragt, aber niemand hatte es. Soweit ich mich erinnern kann, haben wir im Projekt den Roboterarm dazu verwendet, fixe Punkte in unserem vorab kalibrierten Koordinatensystem zu definieren. Dazu steuerten wir den Arm manuell mit einem Joystick an die gewünschten Positionen. Diese Positionen wurden im System vom Roboter gespeichert und so so programmiert/angeordnet, dass dieser Würfel stapelen und verschieben konnte. Die präzise Festlegung dieser Fixpunkte erforderte viel Testen und Optimieren der Armpositionen. Falls Sie aber den Code aus dem Modul 7.2 für den ESP32 mit dem EasyCat Shield meinen, habe ich diesen hier auch noch beigefügt. Ursprünglich war dieser Teil eines separaten Protokolls für 7.2, das ich ebenfalls hochgeladen hatte.

Quellen:

[1] www.elearning.tgm.ac.at 2023. [online] Available at: Basisvermessung.pdf [Accessed 06 Oktober 2023].

[2] www.elearning.tgm.ac.at 2023. [online] Available at: Dokumentation Kuka.pdf [Accessed 06 Oktober 2023].

[3] www.elearning.tgm.ac.at 2023. [online] Available at: KRL_Cheatsheet.pdf [Accessed 06 Oktober 2023].

[4] www.elearning.tgm.ac.at 2023. [online] Available at: KUKA KRL-Quickguide.pdf [Accessed 06 Oktober 2023].

[5] www.elearning.tgm.ac.at 2023. [online] Available at: SYT-Lab Layout 4.pdf [Accessed 06 Oktober 2023].