Projekt Bipedal

Humanoider Roboter mit Ansatz von Robotis Dynamixel

Von: Manuel-Leonhard Rixen

Bearbeitungszeitraum: 11/15 - ?

Kurzfassung

Um in möglichst kurzer Zeit wichtige Informationen über ein System mit integrierten Industrieroboter zu erhalten dient die Application (App) *Fast Diagnose*. Die auszugebenden Informationen beinhalten die Taktzeit (Aktual- und Durchschnittswert) mit grafischer Darstellung, sowie Konsolenausgabe und Anlagenparameter. Zudem wird der Benutzer anhand von Events über einen Zyklusstopp informiert. Aktuell werden ausschließlich Roboter des Herstellers ABB unterstützt.

Inhalt

1 Einleitung

Mit der Application (App) *Fast Diagnose* ist es möglich über ein laufendes System mit einer ABB-Robotersteuerung Informationen abzurufen. Diese Daten enthalten zunächst die tabellarisch zusammengefassten Projektdaten (SOP, FAT, Betriebsstunden, Projekt-Nr., etc.) zudem wird die Zykluszeit als aktueller, sowie als Mittelwert ausgegeben. Neben den Logs wird der Meldungstext von System-Events (Information, Warnung, Fehler) ebenfalls dargestellt.
Bisher konnten Systeminformationen ausschließlich über das FlexPendant der ABB-Steuerung ausgegeben werden. Die App Fast Diagnose erleichtert die Inbetriebnahme der Anlage beim Arbeitgeber und Arbeitnehmer, denn wichtige Systeminformationen werden direkt auf dem Smartphone dargestellt. Wenn im Testbetrieb die Anlage stehen bleibt, so wird der Inbetriebnehmer duch das Vibrieren des Smartphones und dem Darstellen eines Dialogs darauf aufmerkam gemacht. Es ist somit nicht notwendig an der Anlage anwesend zu sein. Zudem kann bei FAT und SOP schnell und bequem die Takzeit kontrolliert werden.
Das Kapitel **??** befasst sich mit dem Aufbau der Nachricht und **??** zeigt den Programmaufbau, d.h. die Tasks, Module und Klassen. Während **??** die grafischen Elemente erläutert, zeigt **??** das Einbinden der Programmdaten. Kapitel 3 beinhaltet eine offene Punkte Liste.

2 Funktion und Aufbau

2.1 Vorbereitung

2.1.1 Testaufbau

Im Folgenden wird die Arbeitsumgebung, d.h. Entwicklungsplattform, sowie Hard- und Software-Komponenten erläutert.

Als Betriebsystem wird Windows 7 32 und 64 Bit verwendet.

Es liegen folgende Hardware-Komponenten vor:

- AX-12A
- OpenCM 485 Expansion Board
- OpenCM9.04-C

Es liegen folgende Software-Komponenten vor:

- ROBOTIS_OpenCM
- Visual Studio 2013 Community

Tabelle 2.1: Investitionskosten

Hardware	**Kosten**	**Software**	**Kosten**
AX-12A	46,80 €	ROBOTIS_OpenCM	
OpenCM 485 Expansion Board	27,50 €	Visual Studio 2013 Community	
OpenCM9.04-C	21,50 €		
Versandkosten	11,29 €		
Summe	**107,09 €**		

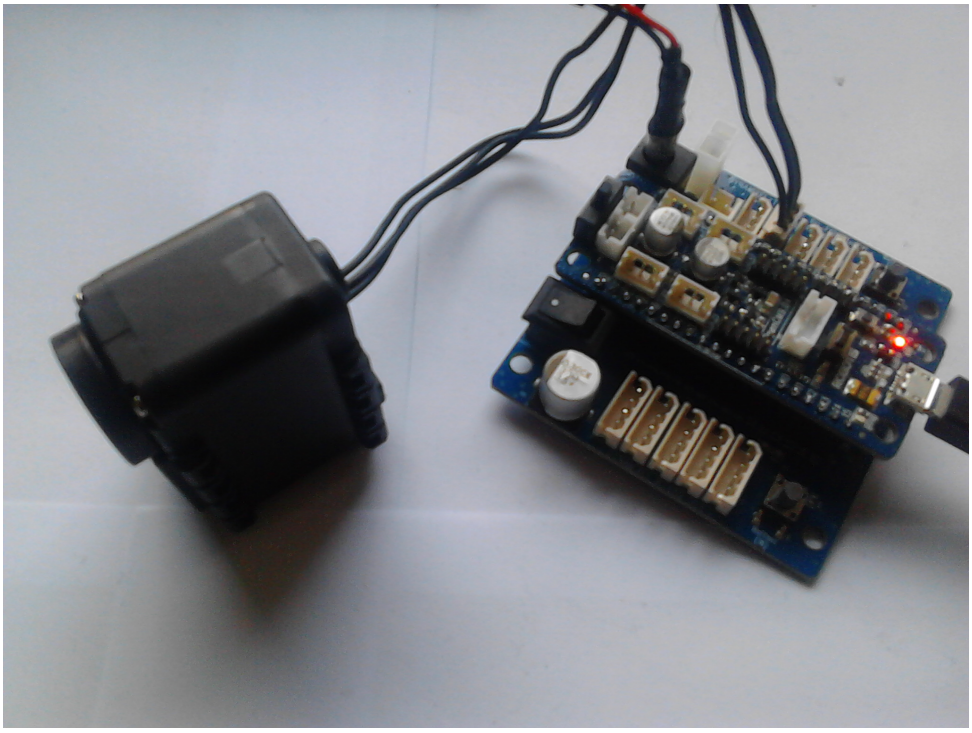

Abb. 2.1: Testaufbau

2.1.2 Programmierung

Das OpenCM Board wird mit der glside Robotis OpenCM (modifizierte Arduino-Umgebung) programmiert, da das Programm sehr schlicht gehalten wird. Der Speicherplatz von OpenCM ist stark begrenzt, sodass dieses nur als Befehlsverteiler fungiert.
Das bedeutet, dass ein übergeorndneter Rechner (geplant ist Raspberry Pi) eine in C# geschriebene Human Machine Interface (HMI) enthält mit der die komplette Regelung aktiviert und (in gewissen Mae) modifiziert werden kann.
Das Board OpenCM empfängt die vom Rasbperry Pi gesendeten Befehle und leitet diese an die entsprechenden Aktoren weiter.

Das Programm, welches auf dem OpenCM-Controller läuft ist in **??** dargestellt.

Listing 2.1: Programm OpenCM

```
1 #define DXL_BUS_SERIAL1 1  //Dynamixel on Serial1(USART1)  <-OpenCM9.04
2 #define DXL_BUS_SERIAL2 2  //Dynamixel on Serial2(USART2)  <-LN101,BT210
3 #define DXL_BUS_SERIAL3 3  //Dynamixel on Serial3(USART3)  <-OpenCM 485EXP

5 #define ID_NUM 1

7 #define GOAL_POSITION 30
```

```
9  Dynamixel Dxl(DXL_BUS_SERIAL3);
10 int movementData;


13 void setup() {
14   // Dynamixel 2.0 Baudrate -> 0: 9600, 1: 57600, 2: 115200, 3: 1Mbps
15   Dxl.begin(3);
16   Dxl.jointMode(ID_NUM); //jointMode() is to use position mod
17 }

19 void loop() {
20    movementData = getValue();
21    if(movementData != -1) Dxl.writeWord(ID_NUM, GOAL_POSITION, movementData);
22    delay(100);
23 }

25 int getValue(){
26   int inValue;
27   int multiplicator;
28   int sollPosition;
29   int cntr;

31   multiplicator = 1;
32   sollPosition = 0;

34   cntr = SerialUSB.available();
35   if(cntr>0){
36    for(unsigned int i=0; i < cntr-1;i++){
37       multiplicator = multiplicator*10;
38    }
39   for(unsigned int i=0; i < cntr;i++){
40     inValue = (SerialUSB.read())-48;
41     sollPosition = sollPosition+(inValue*multiplicator);
42     multiplicator = multiplicator/10;
43   }
44   return sollPosition;
45 }
46 else return -1;

48 }
```

3 Ausblick

Die App ist in der aktuellen Version noch ausbaufähig. Der wichtigste Aspekt betrifft das Hinzufügen der Kompatibilität für eine Kuka-Steuerung und eventuell weitere Robotersteuerungen.
Eine vollständige Lister der Verbesserungen sind folgend dargestellt.

Offene Punkte ABB

- Vollständige Meldeparameter (str1, str2, etc.) übertragen
- Verbindung von mehreren Clients ermöglichen
- Machinendaten aus Textdatei einlesen

Offene Punkte Android

- Speichern und Auswählen bisheriger Verbindungen
- Diagramme für Zykluszeit hinzufügen
- GUI überarbeiten
- Handshake beim Senden von Daten implementieren, um Datenverlust zu vermeiden
- Manuelle Eingabe der Adresse ermöglichen

Listings

Abbildungsverzeichnis

Tabellenverzeichnis