

Projekt Bipedal

Humanoider Roboter mit Ansatz von Robotis Dynamixel

Von: Manuel-Leonhard Rixen

Bearbeitungszeitraum: 11/15 - ?



Kurzfassung

Inhalt

1	Einleitung	1
2	Funktion und Aufbau	2
2.1	Vorbereitung	2
2.1.1	Testaufbau	2
2.1.2	Programmierung	3
2.1.2.1	OpenCM Code	3
2.1.2.2	Matlab Code	5
2.2	Nachrichtenaufbau	6
3	Ausblick	8
	Listings	I
	Abbildungsverzeichnis	II
	Tabellenverzeichnis	III

1 Einleitung

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 Betriebssystem 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 €		

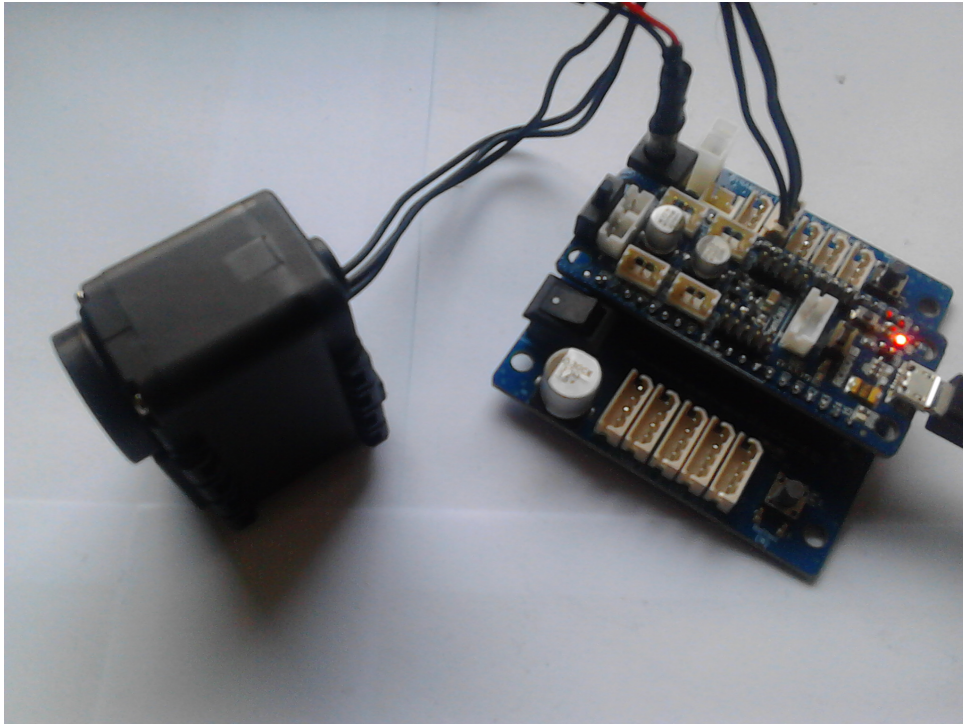


Abb. 2.1: Testaufbau

2.1.2 Programmierung

2.1.2.1 OpenCM Code

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 übergeordneter 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 Raspberry Pi gesendeten Befehle und leitet diese an die entsprechenden Aktoren weiter.

Das Programm, welches auf dem OpenCM-Controller läuft ist in 2.1.2.1 dargestellt. In Zeile 1-3 erfolgt das Definieren des seriellen Bus, welche verwendet werden können. Da das Extension Board zusammen mit dem OpenCM Controller Verwendung findet wird in Zeile 9 der Bus 3 übergeben.

Jeder Aktor erhält eine ID (zwischen 0 und 253) und ist aktuell (wie in Zeile 4 zu sehen) auf 1 gesetzt. Die Befehle, welche an den Servomotor geschickt werden

können, um Daten zu setzen oder abzurufen sind in ?? dargestellt. In Zeile 6 erfolgt die Definition des Befehls 30, die den Aktor zu einer Position (0-300 [deg], bzw. 0-1024).

In der setup-Routine (Zeile 11-15) erfolgt das Setzen der Baudrate (s. ??) und des Bewegungsmodus (Joint oder Wheel) für den entsprechenden Aktor. Sobald Daten auf der seriellen Schnittstelle vorliegen werden diese in Zeile 19-23 mit der Funktion `getValue()` abgerufen und an die Funktion `Dxl.writeWord()` (s. 2.1.2.1 übergeben.

Listing 2.1: Funktion `dxlWriteWord`

```
1 void dxl_write_word(  
2     int id,  
3     int address,  
4     int value  
5 );
```

Die Funktion `getValue()` liest die Anzahl der gesendeten Zeichen und splittet diese auf in die ID, den Befehl und den Wert für den Befehl (Zeile 17-XX).

Listing 2.2: OpenCM-Programm

```
1 #define DXL_BUS_SERIAL3 3  
2 #define ID_NUM 1  
3 #define DIM_ID_ARRAY 3  
4 #define DIM_CMD_ARRAY 2  
5 #define DIM_VAL_ARRAY 4  
  
7 Dynamixel Dxl(DXL_BUS_SERIAL3);  
8     int id;  
9     int command;  
10    int sollPosition;  
  
12 void setup() {  
13     Dxl.begin(3);  
14     Dxl.jointMode(ID_NUM);  
15 }  
  
17 void loop() {  
18     id = getValue();  
19     command = getValue();  
20     sollPosition = getValue();  
21     if((id != -1) & (command != -1) & (sollPosition != -1)) Dxl.writeWord(id,  
22         command, sollPosition);  
23     delay(100);  
24 }  
  
26 int getValue() {  
27     int inValue;  
28     int cntr;  
29     int motorId[DIM_ID_ARRAY];  
30     int command[DIM_CMD_ARRAY];  
31     int value[DIM_VAL_ARRAY];
```

```
33 // Check if there is something to receive
34 cntnr = SerialUSB.available();
35 if(cntnr>0){
36     // Read the first parameter to the first semicolon
37     for(unsigned int i=0; i < cntnr;i++){
38         inValue = (SerialUSB.read())-48;
39         if(inValue != 11) {
40             motorId[i] = inValue;
41         }
42         else return calcValue(motorId, i,calcMult(i));;
43     }
44 }
45 else return -1;
46 }

48 // Caclulate the multiplier
49 int calcMult(int steps){
50     int multiplicator = 1;

52     for(unsigned int i=0; i < steps-1;i++){
53         multiplicator = multiplicator*10;
54     }
55     return multiplicator;
56 }

58 // Calculate the value
59 int calcValue(int data[], int dim, int factor){
60     int retVal = 0;

62     for(unsigned int i=0; i <= dim-1;i++){
63         retVal = retVal+(data[i]*factor);
64         factor = factor/10;
65     }
66     return retVal;
67 }
```

2.1.2.2 Matlab Code

Die entsprechenden Befehle können über ein Matlab-Skript (s. 2.1.2.2) an das Board OpenCM gesendet werden.

Listing 2.3: Matlab-Skript

```
1 function ComDeltaRobot()
2     pTime = 0.8;

4     PortID = serial( 'COM3' );
5     set( PortID,'BaudRate',9600, 'StopBits', 2);

7     fopen( PortID );

9     fprintf(PortID,'1;30;55;');
10    pause(pTime);
```



```
12     fprintf(PortID,'1;30;0;');
13     pause(pTime);

15     fprintf(PortID,'1;30;55;');
16     pause(pTime);

18     fprintf(PortID,'1;30;0;');
19     pause(pTime);

21     OutPort = instrfind('Port','COM3');
22     NewObjs = instrfind;
23     fclose(NewObjs);
24 end
```

2.2 Nachrichtenaufbau

Jede Nachricht, die von dem Master-Computer (Raspberry Pi) gesendet und von dem OpenCM-Board korrekt empfangen werden soll, muss folgendes Format aufweisen:

Nachricht = {ID};{COMMAND};{VALUE};

Die Werte für {ID} sind für den entsprechenden Servomotor bestimmt und ist auf den Wertebereich zwischen 0 - 253 beschränkt.

Die Werte für {COMMAND} sind der Tabelle zu entnehmen, die auf folgender Website zu finden ist. Die wichtigsten Befehle sind zur Übersicht in 2.2 dargestellt.

2 Funktion und Aufbau

R A M	24 (0x18)	Torque Enable	Torque On/Off	RW	0 (0x00)
	25 (0x19)	LED	LED On/Off	RW	0 (0x00)
	26 (0x1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0x01)
	27 (0x1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0x01)
	28 (0x1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0x20)
	29 (0x1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0x20)
	30 (0x1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0x1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0x20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0x21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0x22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0x23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0x24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0x25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0x26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0x27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0x28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0x29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0x2A)	Present Voltage	Current Voltage	R	-
	43 (0x2B)	Present Temperature	Current Temperature	R	-
	44 (0x2C)	Registered	Means if Instruction is registered	R	0 (0x00)
	46 (0x2E)	Moving	Means if there is any movement	R	0 (0x00)
	47 (0x2F)	Lock	Locking EEPROM	RW	0 (0x00)
	48 (0x30)	Punch(L)	Lowest byte of Punch	RW	32 (0x20)
	49 (0x31)	Punch(H)	Highest byte of Punch	RW	0 (0x00)

Abb. 2.2: Wichtige Kommandos

Der Wert {VALUE} ist zunächst dafür vorgesehen, um den Drehwinkel für den Aktor angeben zu können. Um einen Servomotor mit der ID = 1 auf 55 [deg] zu positionieren muss folgender String übertragen werden: **1;30;55;**

Auf dem übergeordneten Rechner kann dann eine Regelstruktur implementiert werden, die entsprechende Kommandos an die jeweils beteiligten Aktoren sendet.

3 Ausblick

Listings

2.1	Funktion dxlWriteWord	4
2.2	OpenCM-Programm	4
2.3	Matlab-Skript	5

Abbildungsverzeichnis

2.1	Testaufbau	3
2.2	Wichtige Kommandos	7

Tabellenverzeichnis

2.1	Investitionskosten	2
-----	------------------------------	---