App-controlled LEGO robotic arm

Project of lecture "Mobile Computing" (winter term 2018/2019)

Christoph Ulrich
HTWG Konstanz
Constance, Germany
christoph.ulrich@htwg-konstanz.de

Benjamin Schaefer
HTWG Konstanz
Constance, Germany
benjamin.schaefer@htwg-konstanz.de

***Abstract*—In this document, a method for building and controlling a robot arm using an Android APP is proposed. The Android phone and the NXT board are connected via WLAN, through an additional computer. From the Android app commands can be generated via buttons or a graphical interface, which are received by the NXT and the robot arm controls. The control of the arm is written in Python. As in robotics, the ROS framework is used for communication.**

I. Introduction / Motivation

- control of a robotic arm is a fundamental task in robotics - easy hands-on experience for everybody to this fundamental robotic application with this paper and the created low cost LEGO 3-DoF (*Degrees of Freedom*) robot arm (incl. instructions)
- application and hardware could be used in basic lecture "Grundlagen der mobilen Robotik" to better understand robot kinematics, ROS and a bit of perception
- recycling of old and unused hardware of the robotics lab at the HTWG Konstanz
- typical industrial applications to control a robotic arm run on more powerful hardware and often offer a complicated and - for beginners - confusing GUI,

  Christoph: insert example(s)

  so we developed an easy to use mobile application for Android platforms
- ROS because widely used, very modular/extensible and basic framework which almost every student starting with robotics has to get in touch with

Christoph: new paragraph - description of background and main "problem"

- which platform to choose for controlling the arm and driving the motors (Arduino, Raspberry, etc.) - should consume as little energy as possible, should be flexible and portable

  Christoph: note that in III-B
- another aspect $\rightarrow$ how should the application on the mobile device communication with the controlling device $\rightarrow$ BT, WiFi, (lost of steering commands due to radio lacks etc.)
- arm construction $\rightarrow$ not too many components, not too heavy so that the motors are able to drive the arm even with more than one joint and with gripper load (to demonstrate gripping)
- app $\rightarrow$

Constructing a robot arm generally leads to some difficulties.

II. State of the Art

Although current approaches such as in [6] already control using an app arms, but here is usually an Arduino or Raspberry Pi used to control the arm. However, no hardware that can be reused from the lab is used here, and no known robotic frameworks are used here, which makes an extension or a rebuild more work Arms made from Lego Technic as in [7] have already been built, but these only offer 2DOF and can only lift very light objects due to the low gear ratio. However, in order to be able to show kinematics and the problems with arms in teaching, at least 3DOF is necessary. Also some private projects like in [3] show how to control a homemade arm via an app.

III. Proposed Approach

A. Requirements Engineering

The app should have the ability to control the arm via a graphical interface. To control the joints individually, there should also be a possibility to adjust the individual joints via buttons or sliders. Since the servomotors are likely to drift there should be a way to recalibrate the arm at any time.

The arm should be completely controllable by means of an APP via ROS. Furthermore, the goal is to build the arm completely from Lego Technic. In addition, the arm should be kept as light as possible in order to lift light 3D printed objects. It should also have a comfortable size so that the arm can be demonstrated in the teaching.

B. Platform Decision

Since the arm and the app should be used for teaching, there was a variety of hardware to choose from. In addition, one goal was to use existing and disused hardware. Thus, the decision quickly on the NXT bricks of Lego Mindstorm [1]. There are enough bricks as well as sensors and motors in the lab. In addition, from the existing Lego technology also the arm can be constructed. Since Lego Technic is made of lightweight plastic, the finished arm can also be taken to a lecture and

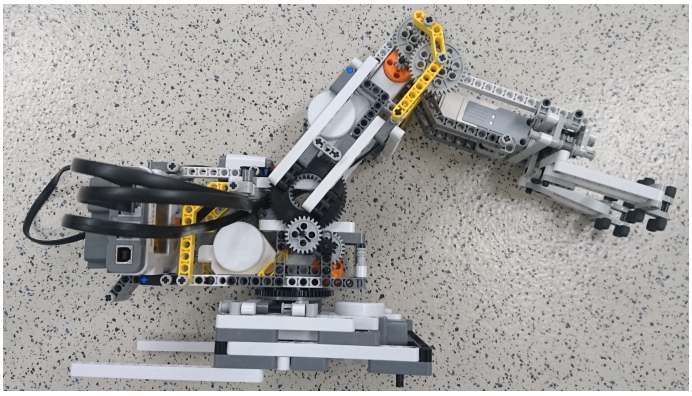

Fig. 1. Figure of the Lego Technic arm. The lower joint is marked in red and the upper joint in blue

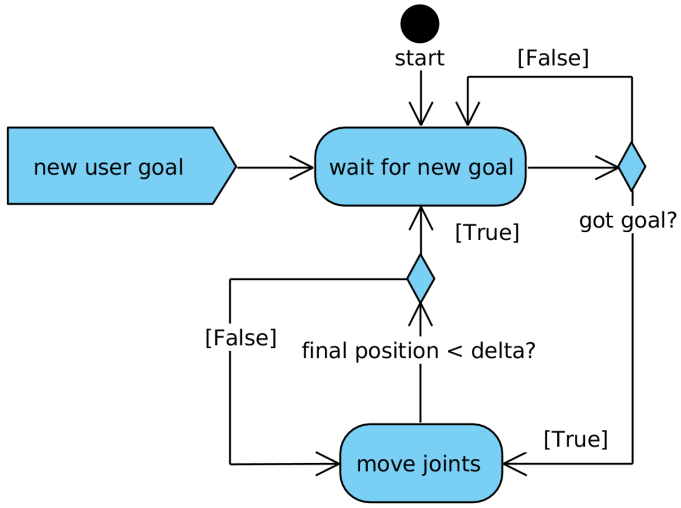

Fig. 2. Activity diagram of the Arm

presented.
There was also a choice of Raspberry Pi's or Arduinos, but here it would have been necessary to order hardware such as motors or parts for the arm.
Thus, it offered to use the Lego NXT's, since no hardware needs to be ordered here and disused hardware is used again.

C. Arm Construction

The arm, which can be seen in Figure 1, was designed entirely from Lego Technic. For this purpose, 2 NXT bricks were used. In addition, 4 motors and a touch sensor were used. An NXT was used to control the two arm joints, as well as the gripper. Since the NXTs only have three ports for motors, a second NXT was needed to rotate the base.
The servomotors have a built-in rotation sensor with an accuracy of 1 degree. Since these servomotors only have a torque of about 12 N.cm, a translation has been built into the wrist joint so that the motors can move the arm. For this purpose, a translation of 1/42 was used at the lower joint (red

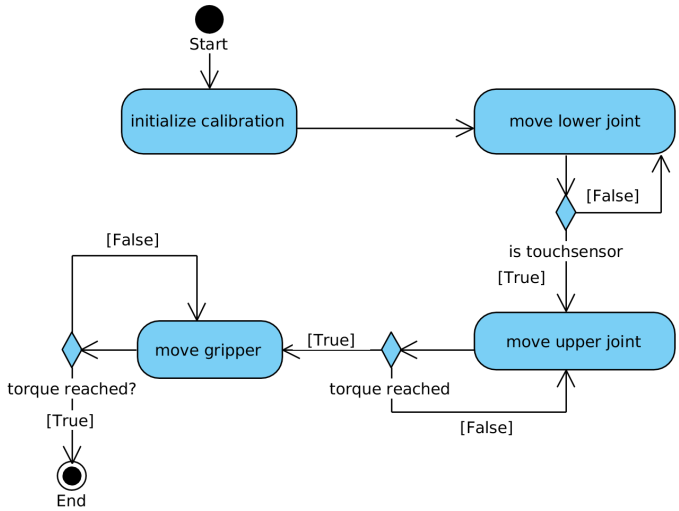

Fig. 3. Activity diagram of the calibration process

circle). The engine's ratio at the upper joint (blue circle) was 2/25.
In order to keep the arm as light as possible and to relieve the engines, sensors on the arm itself were completely dispensed with. However, since the engines have to be initialized, a touch sensor has been installed on the base to initialize the first arm articulated. The initialization of the second joint was solved by the torque of the motors, as there is less weight of the arm.

D. Algorithms

The following describes the calibration procedure, which is shown in Figure 2. Since the servomotors can only be controlled by the effort and have no encoder, the starting position of the motors is unknown. Therefore, these must be calibrated. A touch sensor is used for the lower joint. The lower motor rotates until the touch sensor responds to a touch. Then the upper arm is calibrated. Here, the engines rotate with the least force so long, with which the motors can just turn until the joint is in the final position and the engines can not turn, because the effort is too great. In order not to burden the motors unnecessarily, the calibration process is terminated as soon as a certain delta has been exceeded. Finally, the same procedure is repeated for the gripper.

Christoph: implementation of forward and backward kinematics

1) Kinematics: In order to control the arm it is essential to solve either it's forward or backward kinematics. For our application, we need both problems to be solved. Users of the application should be able to directly control each joint individually (forward kinematics) as well as to move the TCP into a desired position (backward kinematics). We will first take a look at the calculation of the forward kinematics and then introduce a short geometric solution of the backward kinematics for our 3-DoF arm.

Christoph: TCP - list of abbreviations may be necessary

Beide: measurement of arm dimensions

The goal of forward kinematics is to determine the pose (position and orientation) of the TCP for a given set of joint angles. The pose of the TCP regarding the origin of the robot arm can be described as a concatenation of n transformations, where n is the number of joints, of which every one has its own coordinate system. For our case n is 3:

$$T_0^3 = T_0^1 * T_1^2 * T_2^3 \tag{1}$$

where T_i^{i-1} is a transformation matrix according to the *Denavit-Hartenberg* convention:

$$T_i^{i-1} = Tl(0,0,d_i) * R(z,\theta_i) * Tl(a_i,0,0) * R(x,\alpha_i) \tag{2}$$

with Tl being a translation matrix, R a rotation matrix, d_i a translation in z, a_i a translation (for prismatic joints) in x, θ_i a rotation around z and α_i a constant tilt angle between both joints. In our case we don't use prismatic joints and there are no constant tilt angles between the joints (they are all variable). So equation 2 simplifies to

$$T_i^{i-1} = Tl(0,0,d_i) * R(z,\theta_i) \tag{3}$$

By solving equation 3 we get the following transformation matrix T_i^{i-1} for the forward kinematics:

$$\begin{matrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & l_i \cdot \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) & 0 & l_i \cdot \sin(\theta_i) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Finally, one generally can get the pose of the TCP in the world coordinate system (usually that means with reference to the coordinate system of the robot's base) by multiplying the concatenated transformation matrix 1 built from the set of given joint angles θ_i and the known lengths of the arm parts l_i with the origin of the base:

$$p_{tcp} = T_0^3 * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4}$$

Note that one have to use homogeneous coordinates here. Also one should remember that p_{tcp} is the pose of the TCP-origin. Usually users want to get the gripping point of their tool - in this case just add the half of the gripper length to the x-direction to the vector in equation 4.

Vice versa, the goal of backward kinematics is to calculate a set of joint angles from a desired TCP-pose. χ

- generally for actual 3 dof
- visualization and control over 2 dof, separate control of base and gripper

E. App Development

Christoph: communication/process description, ROS, navigation strategy, ...

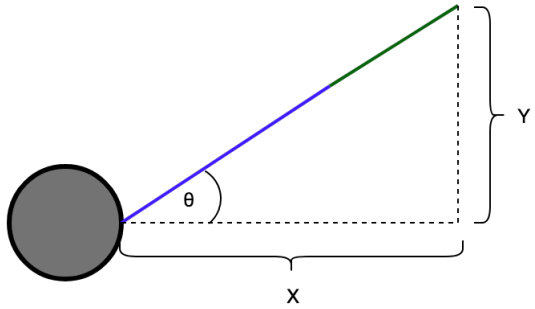

Fig. 4. Top view of the robot. Rotation angle of base joint can be easily calculated with the arctan.

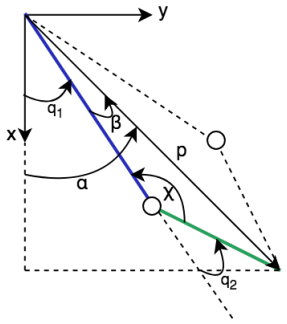

Fig. 5. Top view of the robot. Rotation angle of base joint can be easily calculated with the arctan.

F. Expected Results

Christoph: speed, accuracy, ...

IV. Results

V. Conclusion

The NXT can be used to control a robot arm with a smartphone from a remote location. The present scenario has several disadvantages, e.g. Restrictions for cables and the additional computer. By using ROS this scenario can be extended quickly, eg. by multiple users or a second arm. Despite the rather coarse Lego Technic parts, the arm has a relatively high accuracy and is ideal for demonstrations or teaching.

VI. Further Work

- 3D graphics in App
- EV3
- algorithms
- external better sensors to optimize

In further work, the NXT could be exchanged for an EV3. On the one hand, this one more port, which can be 1 NXT saved. On the other hand, the EV3 offers a WiFi interface, which eliminates the need for a laptop for the connection between smartphone and arm. Currently the app offers only a 2D view of the arm. For a better operation a 3D view could be developed. Another important point would be to install more and better external sensors to achieve a higher positional accuracy of the arm.

QUELLENVERZEICHNIS

[1] Lego Mindstorm - NXT,
https://www.lego.com/de-de/mindstorms/downloads/nxt-software-download

[2] ROS - Kinetic,
http://wiki.ros.org/kinetic/Installation

[3] Arduino-Arm with android-app,
https://www.hackster.io/slantconcepts/control-arduino-robot-arm-with-android-app-1c0d96

[4] Robot-Arm-Arduino,
https://www.instructables.com/id/Robot-Arm-Arduino-App/

[5] Kuka-hrc-guide,
https://www.kuka.com/en-us/products/robotics-systems/software/application-software/kuka-hrc-guide-app

[6] K. Premkumar, K. Nigel "Smart Phone Based Robotic Arm Control Using Raspberry Pi, Android and Wi-Fi" IEEE, March 2015

[7] Lego Technic 2DOF arm construction,
http://www.nxtprograms.com/robot$_arm/steps.html$