SLOVAK UNIVERSITY OF TECHNOLOGY
IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-5414-82075

Design and construction of human-operated remote robotic manipulator

MASTER THESIS

2021 **Bc. Diana Dzurková**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reg. No.: FCHPT-5414-82075

Design and construction of human-operated remote robotic manipulator

MASTER THESIS

Study programme:	Process Control
Study field:	Cybernetics
Training workspace:	Institute of Information Engineering, Automation, and Mathematics
Thesis supervisor:	Ing. Martin Kalúz, PhD.

2021 **Bc. Diana Dzurková**

Slovak University of Technology in Bratislava
Department of Information Engineering and Process Control

Faculty of Chemical and Food Technology
Academic year: 2020/2021
Reg. No.: FCHPT-5414-82075

STU FCHPT

MASTER THESIS TOPIC

Student: **Bc. Diana Dzurková**
Student's ID: 82075
Study programme: Automation and Information Engineering in Chemistry and Food Industry
Study field: Cybernetics
Thesis supervisor: Ing. Martin Kalúz, PhD.
Workplace: Department of Information Engineering and Process Control

Topic: **Design and Development of a Human-operated Remote Robotic Manipulator**

Language of thesis: English

Specification of Assignment:

This work deals with the design, assembly, and programming of the mechatronic system of the remote manipulator. The developed system will be composed of two main parts. First will be a sensor module containing an electronic gyroscope and accelerometer that will allow sensing the movements of a human arm or other haptic inputs. The second part of the system will be a remote robotic manipulator that will translate these inputs into movement. Both, the manipulator and the sensor module will be equipped with electronic micro-controllers and will communicate via low-power radio modules.

The particular tasks of this project are:
- a complete design and assembly of the hand-held sensor module, including a selection of appropriate electronic components, communication interface, power delivery solution, and ergonomics
- design of the program for sensor module that will read the haptic inputs and will be sending them to the remote manipulator
- the selection of electronic platform for robotic manipulator
- derivation of a model for robotic arm based on kinematics
- design of algorithms for movement and control of the robotic arm

Length of thesis: 50

Selected bibliography:

1. Margolis, M. *Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects*. Sebastopol: O´Reilly Media, 2013. 699 p. ISBN 978-1-449-31387-6.
2. JANSCHEK, K. Mechatronic Systems Design: Methods, Models, Concepts (2012th Edition), Springer, 2012. ISBN: 978-3642175305

Acknowledgment

I would like to thank my thesis supervisor Ing. Martin Kalúz, PhD. for his patience and all the help and advice he has given me during the development of this project.

Abstract

The work is devoted to design, construction and programming of mechatronic system of remote manipulator. The control modul, containing an electronic gyroscope and accelerometer, allows to monitor motion of the human hand and to transmit this information wirelessly to a remote robotic manipulator, which will then copy this motion. Both the manipulator and the controller will be equipped with electronic microcontrollers and will communicate with each other by low-energy radio modules. The goal is to reproduce the hand movements so that the manipulator can grasp and carry objects using six degrees of freedom - three translational and three rotational.

Key words: robotic arm, IMU, forward kinematics, inverse kinematics

Contents

List of Figures

List of Acronyms

AHRS	Attitude and Heading Reference System
API	Application Programming Interface
D-H	Denavit-Hartenberg
DIY	Do It Yourself
GUI	Graphical User Interface
GPIO	General Purpose Input/Output
I^{2}C	Inter-Integrated Circuit
I^{2}S	Integrated Inter-IC Sound
IMU	Internal Measurement Unit
OLA	OpenLog Artemis
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
SCARA	Selective Compliant Articulated Robot for Assembly
FABRIK	The Forward And Backward Reaching Inverse Kinematics
IK	Inverse Kinematics
DOF	degrees of freedom
MEMS	Micro electro-mechanical systems
GPS	Global Positioning System
STL	STereoLithography
PLA	polylactic acid
LiPo	Lithium Polymer
BLE	Bluetooth Low Energy
IoT	Internet of things
ADCs	analog-to-digital converter
DACs	digital-to-analog converter
SPI	Serial Peripheral Interface
RMII	Reduced Media-Independent Interface

PWM	pulse width modulation
LCD	Liquid Crystal Display
WLAN	Wireless Local Area Network
CSV	Comma-separated values

Chapter 1

Introduction

According to the Robotics Institute of America (RIA): "A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks." A manipulator is a mechanical structure consisting of links connected with revolute(rotatory) or prismatic(sliding) joints forming a kinematic chain. A manipulator is considered a robot after the gripper and the wrist are attached and the control system is up and running. Manipulators are intended for moving and grasping objects, typically in several degrees of freedom. The last link that actively interacts with its environment is called the end effector. There are various types of end effectors depending on what is the manipulator used for (for example: grippers, spray paint guns, weld guns, etc.) Control is carried out by the operator, programmable controller, or logic system. The position and orientation of robotic manipulator is achieved by changing the links' configuration through actuators. Servo motors are used for rotational joints and linear actuators for prismatic ones. Position of arm and external control interventions are measured by sensors (internal and external).[5]

1.1 Applications

Remotely operated manipulators date back to the 1950s when the need for remote control became a necessity with the first industrial applications of radioactive materials. This types of manipulators is very useful for applications that call for highly complex and responsible performance while operating in extreme environments. These tasks belong to one of the most pressing matters modern robotics deals with [6]. Various applications are explained in detail in Jean Vertut, Philippe Coiffet: Teleoperation and Robotics: Applications and Technology[7], where are remote manipulator referred to as teleoperators. We're going briefly to mention a few of them. There are six main groups of teleoperator applications:

- nuclear
- underwater
- outer space
- medical field
- industry

Nuclear applications

In this field, mechanical master-slave manipulators have been widely used performing in the laboratory with the operator behind a protective shield, involved in tests that require delicate movements. These manipulators are also used to dismantle or assemble equipment in these laboratories as well as their maintenance. Manipulators are also widely used in various nuclear facilities. They perform maintenance on particle accelerators as well as all the necessary tasks involving experimental reactors. Manipulators are also used on a bigger scale such as industrial nuclear facility reactors anywhere from inspection and repair to decommissioning and emergency interventions.

Underwater applications

Teleoperation is used as a survey tool in underwater applications. Manned submarines and free-swimming underwater devices help in the exploration of sea beds before the sub-aquatic systems are deployed. It is also in the construction and maintenance of mentioned structures or accident scenarios of already existing underwater systems. Other applications involve underwater mining and cable and pipeline placement. To determine if a certain sea bed is suitable for underwater mining the sample extraction is necessary. These samples are collected from great depths where humans would have difficulty to access therefore the use of a remotely controlled device is necessary. The first of many important underwater embedding tasks was telephone line placement. Since then several different cables and pipelines have been installed at the sea beads. The embedding, as well as maintenance of these cables, is performed by specialized machines that are trawled and controller remotely.

Applications in outer space

With a growing interest in outer space exploration also grew the need for remotely operated system development. Various demands are put on teleoperators in outer space.

They are used in planetary exploration missions, as satellite maintenance devices, and in the construction and maintenance of space stations.

Applications in medical field

The teleoperators could be applied in various medical sectors. In direct hospital care, they are used for patient handling and transport. Another use can be found with improving the quality of life for people with motor disabilities. Arm and hand prostheses are basically equivalent to remote manipulators. While respecting the structural form of lost appendage they provide a controllable replacement. Cases, where is manipulator and operator in physical contact, are similar to mechanical master-slave systems. In cases where the limb is intact but paralyzed, they are used for movement training therapy and assisted walking. Teleoperators are also used as operation rooms, bilateral remote manipulation with scale reduction of one-tenth and one-hundredth for endoscopic surgeries.

Industrial applications

There is huge variety of tasks manipulators are needed for in industrial plants. They are mostly used in hostile environments or in applications where process calls for amplified force with high dexterity and manipulator is controlled manually. In metallurgy and forging are used for free forging, to create various rough shapes. Some machines used in public works can also be included into the teleoperator category, for example bulldozer with blade that is controlled by a laser beam. Application in hostile environments include high voltage power lines maintenance and mining works.

Security and civil protection

Hostile and dangerous situations are prominent in police work, firefighting, and general public safety emergencies. In these types of situations, teleoperators carry out life-threatening tasks that previously had to be done by people. The first remotely controlled vehicles equipped with fire hoses to fight fires. Simple remotely controlled vehicles with the mounted manipulator are mostly used in the USA to handle and dispose of unsafe munitions as well as defusing explosives of terrorist activities. Legged locomotion, remotely controlled vehicles intended for traversing uneven terrain are used in military and rescue missions.

1.2 Motivation

A previous project that I've worked on sparked in me a great interest in hardware and robotics therefore I wanted to work on something in these fields but slightly on a bigger scale. Thus further expanding my knowledge on these topics. Another great motivation was that a fully functioning platform like this can act as a learning platform for other students for robotic control testing, that have similar interests as me, since there isn't any like that right now available at our institute.

Chapter 2

Goals of the Thesis

The goal of this project is to create a functioning system of a manipulator with a remote controller. That includes the design and creation of the controller's casing and the selection of suitable electronics. Creating the kinematic model and deriving of the forward kinematics equation as well as testing of their functionality. Implementing inverse kinematics to the manipulator, assigning appropriate functionalities to sensor data, and implementing commercial controller to our system. Individual tasks can be summarized accordingly:

1. Study and control of the robotic arm.
    - control of arm's movement with microcontroller
    - finding physical constrains for every joint
2. Study and application of general kinematics
    - how to create a kinematic diagram
    - what are the main rules for kinematic calculations
    - forward kinematics and practical testing
    - inverse kinematics
3. Controller design
    - getting acquainted with Autodesk 360 and creating controller casing
    - selecting appropriate electronics and cabling management
    - installing magnets for controller closing
4. Establishing Xbee module communication
5. Implementing calculations for acquiring heading and position of controller

6. Creating serial data buffer and parser
7. Applying inverse kinematics and its testing
8. Getting acquainted with commercial controller and implementing it to our system
    - editing open-source software
    - applying functionalities to individual controller parts

Chapter 3

Kinematics of Robotic Arm

This chapter deals with the physical parts of a robotic arm and derivation of its mathematical model that allows us to plan and control robot motion. Further, we're going to define some basic kinematic terms used in these models and the most common kinematic model configurations. The theory that is necessary for achieving appropriate manipulator movement mainly consists of a kinematic model, derivation of forward kinematics equations as well as inverse kinematics ones.[9]

3.1 General symbolism of manipulator models

Configuration space

The term configuration of manipulator includes designating a specific location to every manipulator point. The whole set of these configurations is referred to as configuration space. Any point of the manipulator can be deduced if the joint variables are known.

Manipulator workspace

The set of points that the end effector reaches as every possible configuration of a manipulator is executed is called manipulator workspace. Mechanical joint constraints and manipulator geometry restrict the workspace. For instance, none of the servo motors in the robotic arm we are using is capable of 360° rotation. This workspace can be divided into dexterous and reachable. The reachable one, as the name suggests, consists of all the points that manipulator can reach. The dexterous workspace is made of points that are reachable by manipulator with the arbitrary orientation of the end effector. The dexterous space falls under the reachable one.[9]

Typical kinematic configurations

In theory, kinematic chains can be built from joints in various ways. However, most manipulator robots used in industry can be described by one of the following kinematic configurations: [9]:

- articulated manipulator (RRR)[1]- contains three revolute joints, usually referred to as waits, shoulder, and elbow
- spherical manipulator (RRP) - similar to the articulated one but the last joint is prismatic
- SCARA (Selective Compliant Articulated Robot for Assembly) manipulator(RRP) - designated for "pick-and-place" and montage tasks. Uses the same general configuration as spherical manipulator except for spherical one all joints are mutually perpendicular to each other and the SCARA has joints parallel to each other.
- cylindrical manipulator (RPP) - the first joint is revolute and handles rotation around the base, while the other two joints are prismatic
- cartesian manipulator (PPP) - manipulator with first three joints that are prismatic. The name derives from the fact that joint variables match the end effector's Cartesian coordinates.
- parallel manipulator - base and end effector are connected with at least two kinematic chains

[1] Acronym in brackets denotes the number and order of revolute(R) or prismatic (P) joints in a specific configuration

3.2 Robotic Arm

In this project, we are using TinkerKit Braccio robotic arm that is depicted in Fig. 3.1. This arm has a similar kinematic configuration as some of the industrial manipulators. The Arm is made out of plastic and has six servo motors as actuators and a wooden base with a circle that has marked servomotor angles of the base. This robotic kit contains three main parts:

- Robotic arm that consists of:
  1. **base** that enables rotation of the robot around vertical axis
  2. **shoulder** is the second link of manipulator, it controls vertical and forward/backward movements of manipulator
  3. **forearm** controls the movement in the same axes as shoulder
  4. **vertical wrist** controls arm's attack angle at which are objects picked up
  5. **rotatory wrist** that handles the rotation of the gripper around the X axis
  6. **gripper** is the part of arm that interacts with its surroundings this particular one has jaws for picking and moving objects
- Braccio shield
  - connects to the Arduino board and enables to control servos directly from Arduino board
  - compatible with Uno,Mega,Yun,Tian etc. Arduino boards
  - requires 5V power source
- microcontroller board
  - programmable circuit board where the control code runs

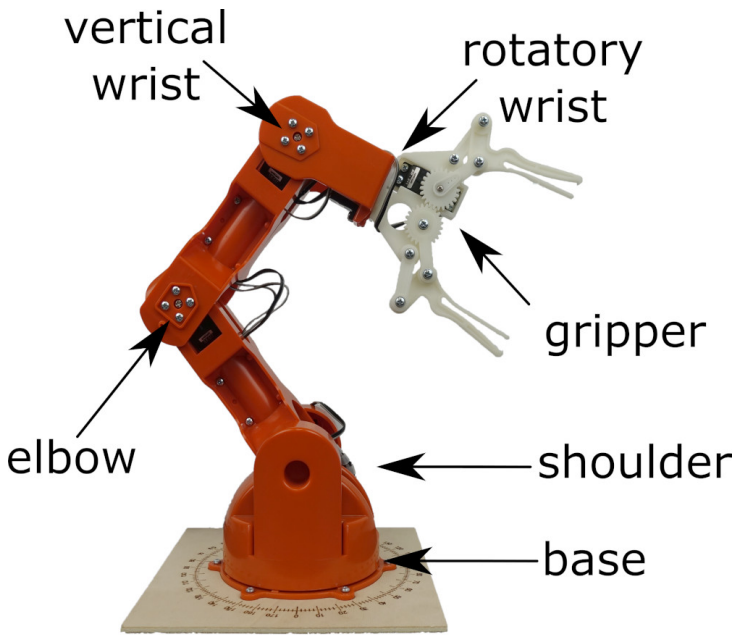


Figure 3.1: Robotic arm

3.3 Kinematic diagram

Kinematic diagram is necessary for derivation of kinematic equations and it is assembled according to the physical appearance of a specific robotic arm. In robotic models, links are rigid parts of a manipulator and are connected by joints, the moveable parts of a manipulator. These parts are connected into the kinematic chain. We are denoting joint variables as θ if the joint is rotational and d if the joint is prismatic. The joints themselves are denoted R for rotatory (revolute) and P for linear (prismatic). A revolute joint enables relative rotation among two links. The prismatic one handles relative linear motion in one specific axis between two links.[9]

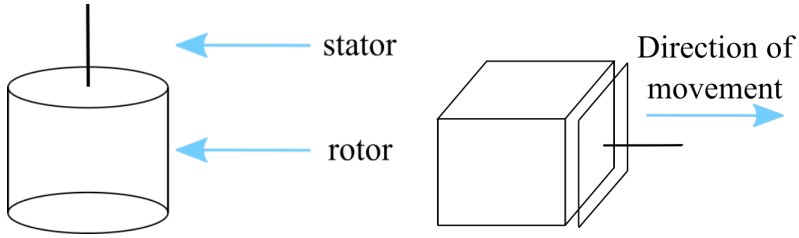


Figure 3.2: Example of schematic for joints: revolute joint(left) prismatic joint(right)

Four main rules need to be followed while creating kinematic diagrams which allow us to make shortcuts in deriving the mathematical equations for kinematic models. Those four rules are part of the *Denavit-Hartenberg method*[8]:

1. The Z-axis is always the axis of rotation for a revolute joint or direction of motion for a prismatic joint.
2. The X-axis must be perpendicular both to its own Z-axis as well as the Z-axis of the previous frame.
3. All frames must follow the right-hand rule.[2]
4. Each X-axis must intersect the Z-axis of the frame before it.

It's also important to label the direction of positive rotation on revolute joints that will match joints of a physical device. This direction is determined by another right-hand rule.[8][3]

[2]Right-hand rule is used to assign coordinate frames. While using the right hand, the fingers should point in the direction of the X-axis, the thumb goes in the direction of the Z-axis and the palm indicates the direction of the Y-axis.

[3]Thumb of the right-hand shows the direction out of servo horn, then the direction fingers curl shows the positive rotation direction.

The final product of this procedure for Braccio manipulator can be seen in Fig. 3.3

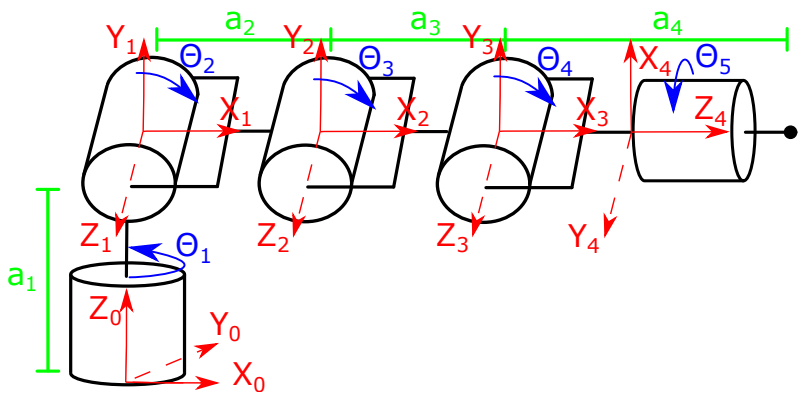


Figure 3.3: Kinematic diagram of Braccio manipulator

where:

- $a_n \quad n = 1, ..., 4$ are physical link lengths, a_1 is the link between base and shoulder with length 76 mm, a_2 connects shoulder and elbow and measures 126 mm, a_3 is the link that connects elbow and vertical wrist and has the same length as previous link 126 mm, a_4 connects elbow and vertical wrist and has a length of 62 mm, the last link is considered to be between vertical wrist and between the gripper's jaws at their middle length (we're combining the last two links between joints: vertical wrist $\leftrightarrow$ rotational wrist and rotational wrist $\leftrightarrow$ gripper into one since rotational wrist doesn't affect end-effector position)
- $theta_0, ..., theta_{n+1}$ represents the direction of a positive rotation of each joint
- axis $Z_{n-1}, Y_{n-1}, X_{n-1}$ represent coordinate frame assigned for each joint

Manipulator movement affects the position and rotation(orientation) of the end-effector. In robotic theory, the position is represented by displacement vectors and rotation by rotation matrices. Rotation matrix for each frame is created by comparing the first frame to the next frame, specifically, how is the second one rotated considering the position of the first one. The individual elements represent the projection of each axis to every axis of the previous frame[8]. For example let's take a look at two frames in Fig. 3.4, that differentiate in rotation around the Z-axis.

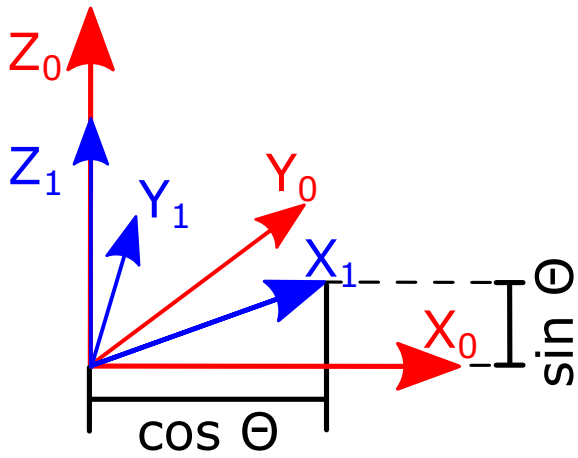


Figure 3.4: Two frames comparison with projection of X-axis

After projections for every axis (in the same way that the projection into X-axis is depicted in Fig. 3.4), we are going to get a matrix.[4] This type of matrix is called the Z-rotation matrix(in reference to frame rotation around Z-axis)[8]:

$$R_Z = R_0^1 = \begin{array}{c} \\ X_0 \\ Y_0 \\ Z_0 \end{array} \begin{array}{c} \begin{array}{ccc} X_1 & Y_1 & Z_1 \end{array} \\ \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \tag{3.1}$$

Each projection is represented by a goniometric function. In cases where axes are identical the projection will have values of 1, if axes are perpendicular to each other the value will be 0. We can similarly derive X and Y-rotation matrices[8]:

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.2}$$

$$R_Y = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.3}$$

Every possible frame rotation is created by various combinations of rotations around X,Y and Z axis. For example rotation in Fig. 3.5 was created by firstly rotating frame around Z axis and secondly around X_1 axis.

[4]Each axis is assumed to have a length of 1.

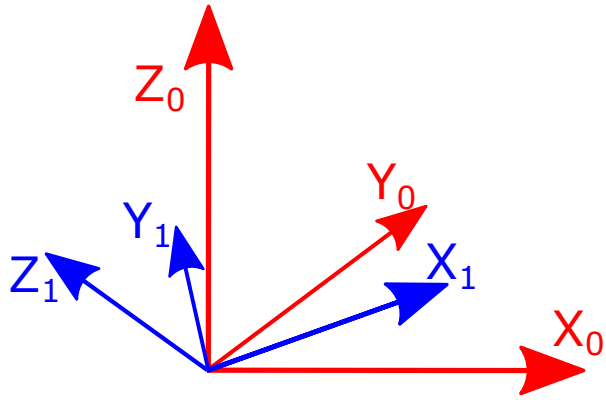


Figure 3.5: Two frames comparison with two rotation difference

We are able to determine rotation axis simply by multiplying Z and X rotation matrices together.[5]

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.4}$$

While creating the individual rotation matrices we usually start with one of the basic matrices(R_Z, R_X, R_Y) and left multiply it by another matrix that tells the rotational difference between two studied frames while joint angle is zero. If there is no rotation different we are going to multiply it by Identity matrix in equation 3.5.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Final goal is to derive the rotation matrix of the end-effector in reference to the base frame, throughout each frame in kinematic diagram in Fig. 3.1.

$$R_4^0 = R_1^0 R_2^1 R_3^2 R_4^3 \tag{3.6}$$

Rotation matrix in equation 3.6 a 3x3 matrix in which elements are a combination of goniometric functions for theta angles considering the kinematic diagram.

$$R_4^0 = [f(\theta_i)] \tag{3.7}$$

[5]Theta is unique for each individual rotation.

The next step is to derive displacement vectors (size 3x1) which general form is shown in equation 3.8.

$$d_n^m = \begin{bmatrix} f_x(a_{i-1}, \theta_i) \\ f_y(a_{i-1}, \theta_i) \\ f_z(a_{i-1}, \theta_i) \end{bmatrix} \tag{3.8}$$

Displacement vectors are created by redrawing the kinematic diagram from the top $\rightarrow$ down view and its elements are represented by lengths of the manipulator links a_i multiplied by the projection of the link to respective axes or have the value of 0 if there's no displacement between frames in the direction of the respective axis[8]. We are able to derive *displacement vectors* through each frame pair except for d_4^0 (between end-effector and base frame). In order to derive vector d_4^0 we need to combine rotation matrices and displacement vectors and therefore derive *homogeneous transformation matrix*. This matrix describes the position and rotation of one frame n relative to the another frame m. The rotation matrix is a 3x3 matrix and the displacement vector is a 3x1 vector. That would make created matrix not square which would prevent multiplication of matrices, so we have to append the last row like this [8]:

$$H_n^m = \begin{bmatrix} & R_n^m & & d_n^m \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

By multiplying homogenous transformation matrices for each frame pair we are able to derive a homogenous transformation matrix between end-effector and base frame, which contains $d0_4$ vector.

$$d_0^4 = \begin{bmatrix} f_x(a_i, \theta_i) \\ f_y(a_i, \theta_i) \\ f_z(a_i, \theta_i) \end{bmatrix} \tag{3.10}$$

Functions in equation 3.10 represent end-effector 3D position(X, Y and Z-axis) based on servo angle values.

3.4 Denavit-Hartenberg table method

Another faster approach to deriving forward kinematic equations is to create a Denavit-Hartenberg table for a given kinematic model. This method was used in our project and tested with a Robotic toolbox in Matlab. It consists of three main steps:

1. Creating kinematic diagram according to the four D-H (Denavit-Hartenberg) rules.
2. Finding the D-H (Denavit-Hartenberg) parameter table 3.1, where:
    - number of table's rows is one less than number of kinematic model's frames

- Table contains four columns. Two rotation columns and two displacement columns.

3. Obtaining the homogenous transformation matrix.

Parameters definition

micro table contains four variables: θ and α are used for rotation, a and d are used for displacement.

- Parameter θ represents the angle of rotation that is necessary to get x_{n-1} axis to orientation of x_n around $zn-1$ axis.
- Parameter α is the angle that $n-1$ frame needs to be rotated around x_n axis to get Z_{n-1} in the same orientation as Z_n.
- Parameter a represents the distance between origins of $n-1$ frame and n frame along the x_n axis.
- Parameter d is distance between x_{n-1} and x_n axes along the Z_{n-1}.

micro table for Arduino Braccio can be seen in table 3.1.

Joint i	θ_i**(deg)**	α_i**(deg)**	a_i**(cm)**	d_i**(cm)**
1	θ_1	a_1	0	$\frac{pi}{2}$
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	0	a_4+a_5	0

Table 3.1: D-H parameters table for Arduino Braccio robotic arm

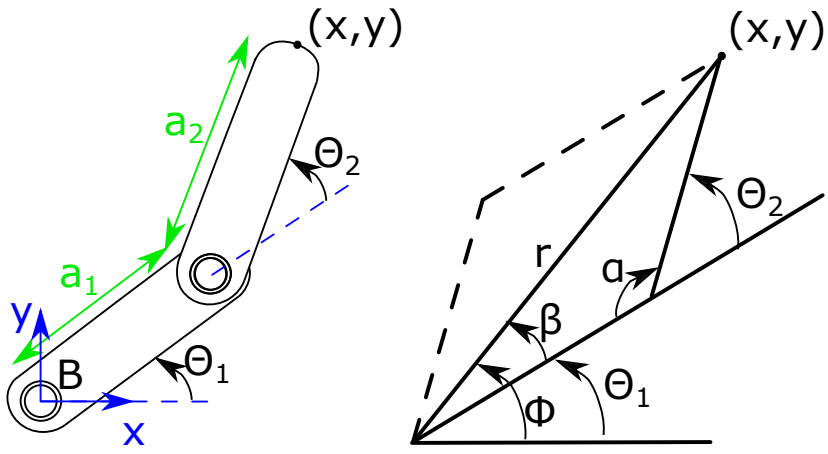
After obtaining the D-H (Denavit-Hartenberg) table parameters, we get partial homogenous transformation matrices between individual frames using formula 3.11:

$$H_{n-1}^{n} = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n)\cos(\alpha_n) & \sin(\theta_n)\sin(\alpha_n) & r_n\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n)\cos(\alpha_n) & -\cos(\theta_n)\sin(\alpha_n) & r_n\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

After containing all homogenous transformation matrices between all the adjacent frames, the matrices are multiplied and the homogenous transformation matrix between base and end-effector is obtained. If the procedure is correct the matrix from this method should be identical to the one from the previous approach.

3.5 Inverse Kinematics

In order to control the manipulator we need to find respective joint variables that would move end effector to desired position. The task of finding the joint variables is an inverse problem to the one that the forward kinematics deals with, therefore it is called Inverse Kinematics (IK). This procedure can be demonstrated on planar two-link manipulator shown in Fig. 3.6a.



(a) Two link manipulator (b) Kinematic model

Figure 3.6: Planar arm example model

Forward kinematics for the model in Fig. 3.6b can be determined by plane geometry in equations 3.12 and 3.13.

$$x = l_1 cos\theta_1 + l_2 \cos(\theta_2 + \theta_1) \tag{3.12}$$

$$y = l_1 cos\theta_1 + l_2 \cos(\theta_2 + \theta_1) \tag{3.13}$$

For inverse kinematics we need to solve for θ_1 and θ_2, while x and y are known variables. Standard method uses polar coordinates (r,ϕ) in Fig. 3.6b and θ_2 is determined using the law of cosines:

$$\theta_2 = \pi \pm \alpha \tag{3.14}$$

$$\alpha = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - r^2}{2l_1 l_2}\right) \tag{3.15}$$

θ_2 has two possible values if $\alpha \neq 0$. Second θ_2 solution is on dashed lines in 2.6 b, commonly called the "flip solution". Next step is to calculate ϕ (equation 3.16) and

solving for θ_1 (equation 3.18)

$$\phi = \arctan 2(x, y) \tag{3.16}$$

$$\beta = \arctan 2(x, y) \tag{3.17}$$

$$\alpha = \cos^{-1}(\frac{r^2 + l_1^2 - l_2^2}{2l_1 r}) \tag{3.18}$$

Inverse kinematics is getting more difficult with the number of non zero parameters of D-H (Denavit-Hartenberg) matrix. It is generally harder to solve than forward kinematics and can be solved through different approaches which are separated into two groups: closed-form and numerical solutions.

Closed-form Inverse Kinematics

Closed-form solutions, including the one in the example above, offers fast calculations of joint angles considering end effector position. They are efficient and relatively easy to implement, which is the reason why they are used in most industrial manipulators.[9]

Decoupling technique

Decoupling technique belongs to the one of the methods for obtaining inverse kinematic equations. Let's consider that homogenous transformation matrix H_0^n in equation 3.19 is given as a function of joint variables θ_1,θ_2,θ_3, ...,θ_n.[10]

$$H_n^0 = H_1^0(\theta_1)H_2^1(\theta_2)H_3^2(\theta_3)...H_n^{n-1}(\theta_n) \tag{3.19}$$

Manipulators are usually moved using joint variables, but manipulated object's position are expressed in global Cartesian coordinates, therefore kinematic transformation between these two spaces(joint↔Cartesian) is necessary.[10] In order to solve the inverse kinematics problem for 6 DOF (degrees of freedom) robot we're going to take 4x4 homogenous transformation matrix from forward kinematics:

$$H_0^6 = H_0^1 H_1^2 H_2^3 H_3^4 H_4^5 H_5^6 = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

where first three rows contain trigonometric functions with six joint variables. As mentioned before, upper left submatrix (3x3) is equation 3.20, is a rotation matrix,

there are only three independent elements because of orthogonality condition[6]. Only six equations are independent(out of twelve). Since the trigonometric functions give multiple solutions, solving these six equations will provide multiple robot configurations. The problem of inverse kinematics can be decoupled into two parts: inverse orientation and inverse position kinematics. This will allows us to break the initial problem into two independent ones. Each of these problems has only three unknowns. Applying this principle to a homogenous transformation matrix will decompose it into the translational and rotational parts.[10]

$$H_0^6 = \begin{bmatrix} R_0^6 & d_0^6 \\ 0 & 1 \end{bmatrix} = D_0^6 R_0^6 = \begin{bmatrix} I & d_0^6 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_0^6 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.21}$$

The translation matrix D_0^6 represents end-effector's position in reference to base frame and contains only three manipulator variables. If we solve for D_0^6 we're going to get manipulator's position variables. The rotation matrix D_0^6 represents end-effector's orientation in reference to base frame and involves three joint wrist variables. Solving for R_0^6 will get us manipulator's orientation variables.[10]

Inverse Transformation Technique

Another possible method for solving inverse kinematics is the inverse transformation technique. This method also starts from a homogenous transformation matrix in equation 3.20 for the end effector in reference to the base frame.[10] Considering that the individual transformation matrices in 3.22 are functions of joint variables.

$$H_0^1(\theta_1), H_1^2(\theta_2), H_2^3(\theta_3), H_3^4(\theta_4), H_4^5(\theta_5), H_5^6(\theta_6) \tag{3.22}$$

We can solve these equation and gain joint variables.[7]

$$H_1^6 = H_0^{1^{-1}} H_0^6 \tag{3.23}$$

$$H_2^6 = H_1^{2^{-1}} H_0^{1^{-1}} H_0^6 \tag{3.24}$$

$$H_3^6 = H_2^{3^{-1}} H_1^{2^{-1}} H_0^{1^{-1}} H_0^6 \tag{3.25}$$

$$H_4^6 = H_3^{4^{-1}} H_2^{3^{-1}} H_1^{2^{-1}} H_0^{1^{-1}} H_0^6 \tag{3.26}$$

$$H_5^6 = H_4^{5^{-1}} H_3^{4^{-1}} H_2^{3^{-1}} H_1^{2^{-1}} H_0^{1^{-1}} H_0^6 \tag{3.27}$$

$$I = H_5^{6^{-1}} H_4^{5^{-1}} H_3^{4^{-1}} H_2^{3^{-1}} H_1^{2^{-1}} H_0^{1^{-1}} H_0^6 \tag{3.28}$$

This technique is sometimes also called the Pieper technique.[10]

[6] Orthogonality of matrix comes from the fact that it represents projection of an orthogonal coordinate frame to another orthogonal coordinate frame.

[7] More details for this method as well as example can be found in [10].

3.6 Numerical Inverse Kinematics

The previous approaches derive a closed-form solution. Another method is to use numerical, iterative algorithms. Numerical methods are mostly used when closed-form solution doesn't exist or in cases of redundant manipulators.

3.6.1 Jacobian Inverse Method

One of the most commonly used numerical methods for deriving inverse kinematics is Jacobian Inverse method. Considering that x^d is desired robot position and $f(\theta)$ is forward kinematics function, the $f(\theta)$ gets expanded in a Taylor series.

$$f(\theta) = f(\theta^d) + J(q^d)(q - q^d) + h.o.t. \tag{3.29}$$

In the next step the analytic Jacobian is taken and neglecting the higher-order terms (h.o.t), the equation is transformed:

$$\theta^d - \theta = J^{-1}(\theta)(x^d - f(\theta)) \tag{3.30}$$

Note that Jacobian is square and invertible. Solution for θ^d involves initial guess θ_0 and formation of a successive estimates θ_0, θ_1, etc as:

$$\theta_k = \theta_{k-1} + \alpha_k J^{-1}(\theta_{k-1})(x^d - f(\theta_{k-1})) \quad k = 1, 2... \tag{3.31}$$

where $\alpha_k > 0$ is step size, which is used to aid convergence. It can be constant or a function of k, a scalar or diagonal matrix.[9]

Jacobian Transpose Method

Another numerical method that can be used for solving inverse kinematics is Jacobian Transpose method. This method begins with defining of the optimization problem.

$$\min_{\theta} F(\theta) = \min_{\theta} \frac{1}{2}(f(\theta) - x^d)^T(f(\theta) - x^d) \tag{3.32}$$

where x^d is again desired configuration and $f(\theta)$ stands for forward kinematics equations. The cost function $F(\theta)$ has gradient.

$$\nabla F(\theta) = J^T(\theta)(f(\theta) - x^d) \tag{3.33}$$

After that the gradient decent algorithm is applied.

$$\theta_k = \theta_{k-1} - \alpha_k \nabla F(\theta_{k-1}) = \theta_{k-1} - \alpha_k J^T(\theta_{k-1})(f(\theta_{k-1} - x^d))\alpha_k > 0 \tag{3.34}$$

This method is more convenient since the Jacobian transpose is easier to calculate than the Jacobian inverse and configuration singularities don't occur. However the convergence might be slower for this method.[9]

3.7 Micro electro-mechanical devices

Micro electro-mechanical systems (MEMS) are one of the primary technologies that involve sensor applications and belong to one of the most advanced sensor technology created by humans. MEMS technology involves the miniaturization of electromechanical and electrical systems. These devices are often contained inside smartphone sensors, autopilot units, and automotive systems. The most frequently used are gyroscope, accelerometer, magnetometer, Hall-effect sensor, barometric pressure sensors, etc. The majority of MEMS have some kind of inertial feature. Sensors convert and transfer the physical motion of the sensor's body into the quantifiable signal quantity. Very often the forces exerted to the sensor get converted to the scaled linear voltage output with specific sensitivity. These sensors are usually tiny, cheap, and provide precise performance. They are fabricated on a mass scale and polycrystalline silicon is used to achieve a thickness of a micron. Low cost however means a trade-off between cost and sensitivity.[12]

MEMS accelerometer

The structure of the accelerometer in Figure 3.7 contains core proof mass stretched between an anchor and a dashpot. When the acceleration is applied to the sensor, the spring stretches under the inertial mass position shift.[12]

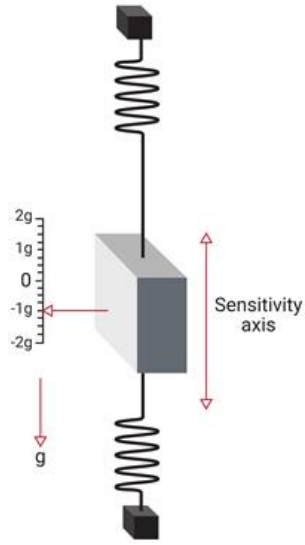


Figure 3.7: Internal model of simple accelerometer[1]

The most common types contain mass suspended between two plates of a capacitor. Applied acceleration results in charge density change and is later processed and amplified by the electronic circuit. Another possible method utilizes a piezo-resistor to monitor any stress applied to the string of inertial mass.[12]

MEMS gyroscope

The angular rate gyroscope in Figure 3.8 contains a moving frame(gimbal) that has a vibrating mass in its core. The mass oscillates at a certain velocity inside of a fixed reference plane. While this frame rotates, the Coriolis force is formed. This results in the Coriolis acceleration that causes the mass' displacement and spring's tension is measured and further processed into the final measurement.[12]

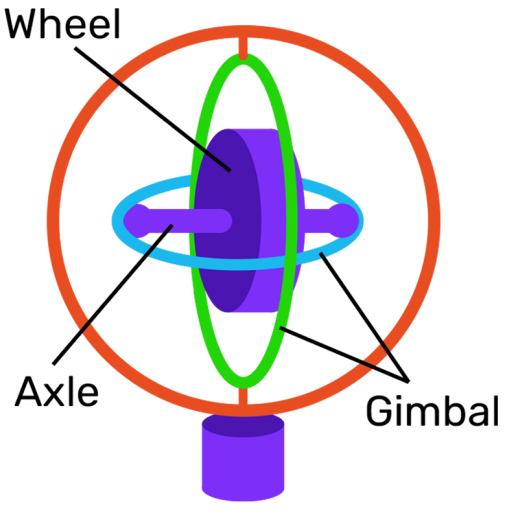


Figure 3.8: Internal model of gyroscope[2]

MEMS magnetometer

A magnetometer measures the direction and strength of a magnetic field. Most magnetometers measure magnetic fields using magnetoresistance and contain permalloys that have variable resistance based on magnetic field changes. The magnetic field is typically a combination of Earth's magnetic field and magnetic fields generated by objects in the sensor's surroundings.[1]

3.8 Internal measurement units

Individual sensors are only able to measure along a single axis. In order to achieve a three-dimensional measurement, an orthogonal cluster of three inertial sensors needs to be created. This cluster is also known as the triad. This sensor set is usually referred to as a 3-axis sensor, since the cluster provides measurement along three axes. Similarly, a set of 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer create a 9-axis IMU system. An IMU (Internal Measurement Unit) is a device capable of monitoring and reporting specific angular rate and acceleration of an object it is mounted on. IMU usually contains a gyroscope that measures angular rate, an accelerometer measuring specific force or acceleration, and optionally a magnetometer that measures the magnetic field that surrounds the system. There are four IMUs performance-based categories: consumer/automotive, industrial, tactical, navigational. These grades differ in price ranges and in-run bias stability of sensors. The incorporation of a magnetometer and filter algorithms for establishing the orientation of the device is called the Attitude and Heading Reference System (AHRS). In AHRS the sensor fusion from accelerometer, gyroscope, and magnetometer is performed creating the system's orientation estimate, usually using a Kalman filter. Kalman filter calculates the gyroscope's drift on top of the attitude values. Gyroscope's drift is then used to determine raw gyroscope values. Applying the Kalman filter to each sensor, the unbiased high-rate attitude values can be obtained. There are still some challenges that create inaccurate orientation values. The most common ones are sustained dynamic acceleration, external and internal magnetic field disturbances, and transient and AC disturbances within accelerometer and magnetometer.[1]

IMU applications

IMUs are frequently utilized in navigational devices or as navigational equipment components such as manned and unmanned aircrafts, GPS (Global Positioning System)(when a vehicle loses connection to the satellite), smartphones and tablets, sport training applications, segways, and hoverboards.

Chapter 4

Handheld Controllers

In order to monitor hand movement in 3D space, we need some kind of device that would measure and record hand position movements. This chapter deals with an overview of the designed handheld controller and commercial controller. The individual part of controllers is described in detail with their functionalities described.

4.1 DIY controller

The first approach to manipulator arm control was designed ergonomic handheld wireless controller that continuously senses the heading of the user's hand. According to the user's hand orientation, the manipulator's base and rotational wrist servo motors angles are controlled. The controller also has a built-in simple button that functions as a gripper trigger. The controller's casing was designed in Autodesk Fusion 360, a program used for 3D printing and other technical applications. The main goal was to create an ergonomic professional-looking controller that would fit any hand. The final 3D design in Fig. 4.1 was then exported in STL (STereoLithography) mesh format into the slicer, where it was converted to G-code. The final format, which can be seen in Fig. 4.2 was printed on a 3D printer from PLA (polylactic acid) material.

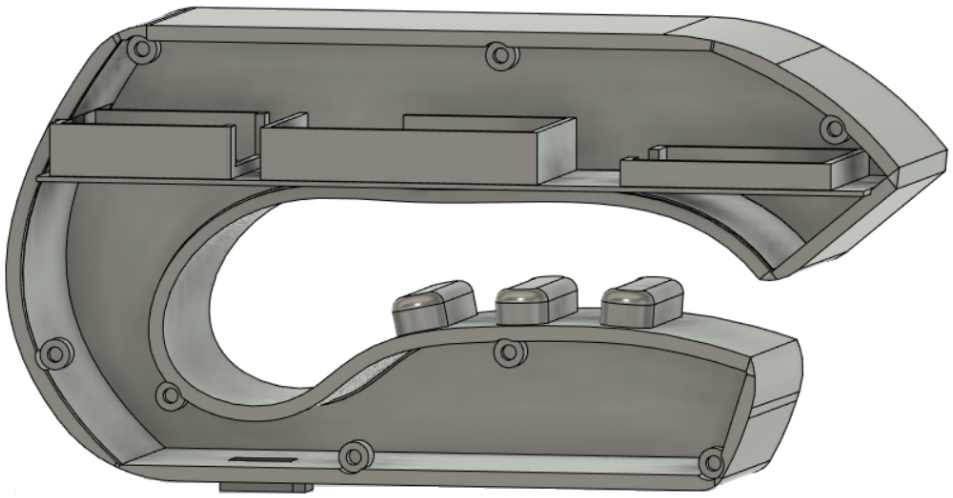


Figure 4.1: Cross section of designed controller

Figure 4.2: 3D printed controller cross section

All the electronics are placed into the slidable slot(designed version in Fig. 4.3) for easy access for maintenance. As we can see in Fig. 4.4 the electronics are placed in

order from left to right as follows: the OpenLog Artemis (OLA) board that monitors the controller's movement in 3D space and has two pins connected to the button for hardware interrupt monitoring when the button is pressed; the lithium battery that that powers the OLA board, therefore, securing the wireless nature of controller; and the Xbee module that receives data from OLA board and sends it wirelessly to the other paired Xbee.

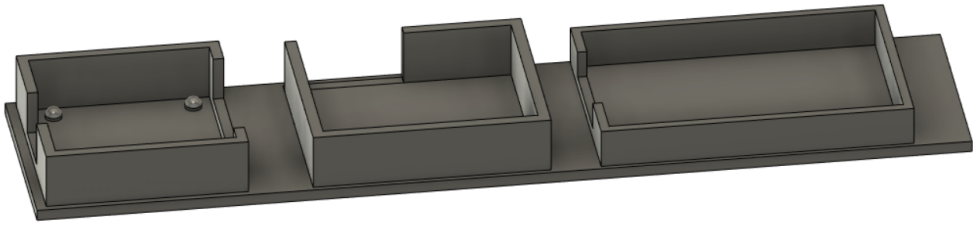


Figure 4.3: Desing of the slot containing all the electronics

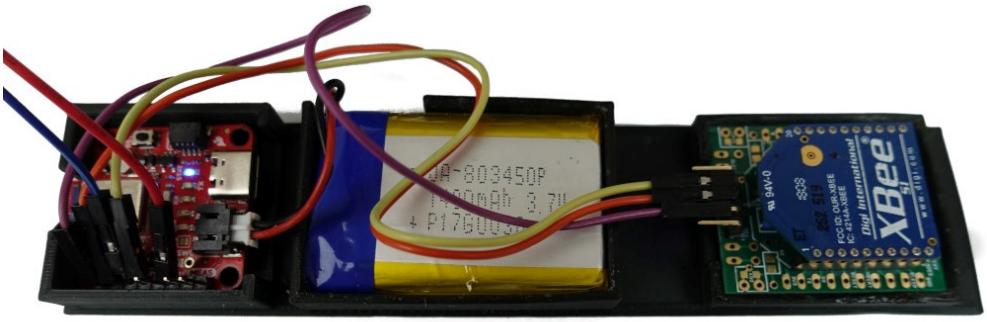


Figure 4.4: Physical 3D-printed slot

The closing lid in (Fig. 4.5) of the controller is attached to the controller's body via set of small neodymium magnets. This mechanism allows an easy access to controller's internal parts.

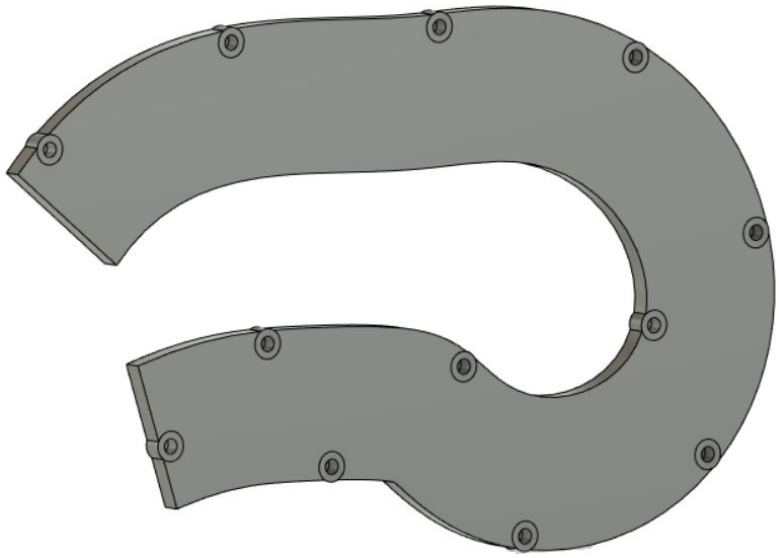


Figure 4.5: Controller cover side

Lithium Ion Battery

Since the DYI controller is designed as a stand-alone device that contains active electronics(microcontroller and Xbee module), it requires an independent power delivery solution. For this purpose, we have chosen a compact Lithium Polymer (LiPo) battery (Fig. 4.6). This battery has nominal voltage of 3.7V and 400 mAh capacity. The battery is connected directly to microcontroller board that also contains a charging circuit.

Figure 4.6: Lithium battery

Battery includes built-in protection against over voltage, over current, and minimum voltage[13].

SparkFun OpenLog Artemis

The main electronic part of the handheld controller is SparkFun OLA (OpenLog Artemis) shown in Fig. 4.7 (top view) and Fig. 4.8 (bottom view). This device provides several important functions that makes the controller operational. The OLA

board contains programmable microcontroller that can run a user defined program. It also has integrated an IMU unit that is essential for sensing hand movements. Additionally, the board has several exposed General Purpose Input/Output (GPIO) that can be used to connect additional peripherals. In the case of DIY (Do It Yourself) controller, this signal interface was used to connect an Xbee module and button to the board.
The SparkFun OpenLog Artemis (OLA) is open source preprogrammed data logger that automatically logs various data from a large number of sensors. OLA contains Apollo 3ARM Cortex-M4F that runs at 48MHz which despite being powerful has low power consumption. It comes with built-in Bluetooth Low Energy (BLE) wireless communication that transmits data up to 70 m. Every OLA board has built-in an Internal Measurement Unit(IMU) that measures and logs data from a triple-axis accelerometer, gyroscope and magnetometer. OLA uses ICM-20948, the world's lowest power 9-axis motion tracking device, that can log data at nearly 250Hz frequency.[1]

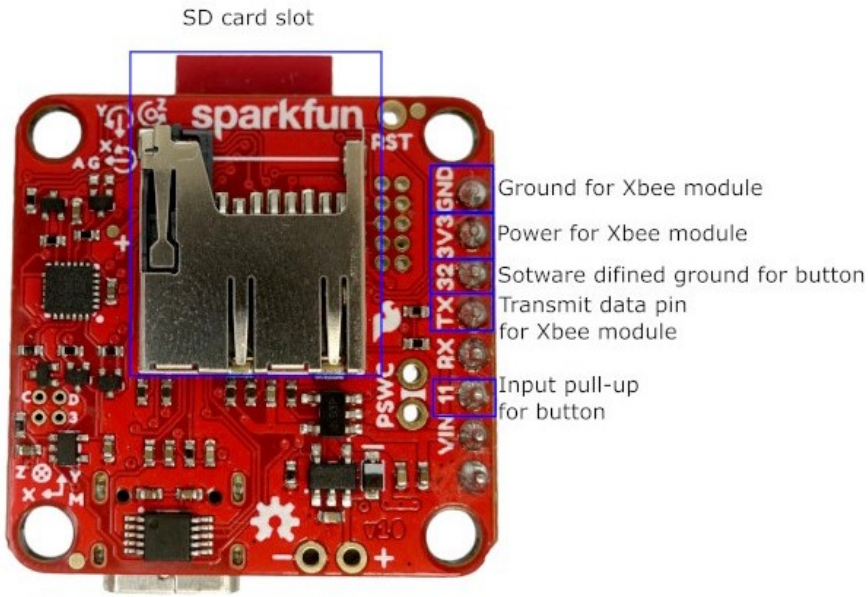


Figure 4.7: IMU board parts top overview

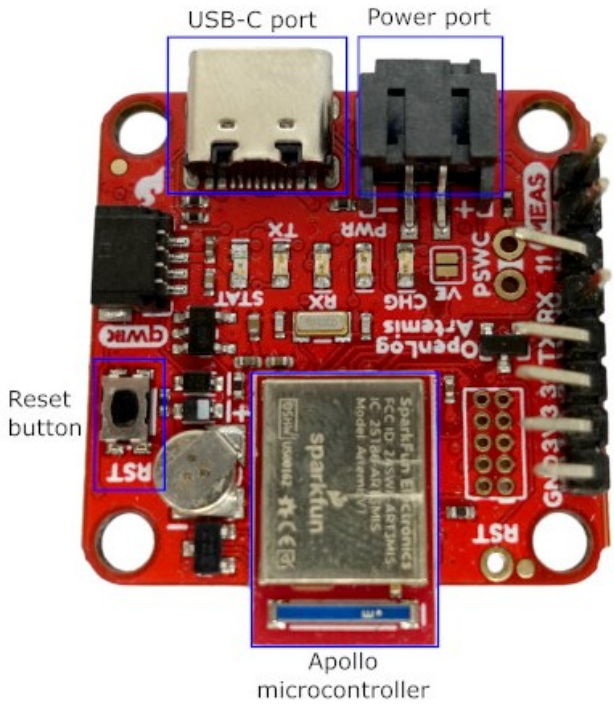


Figure 4.8: IMU board parts bottom overview

Device can output various values, such as:

1. linear acceleration
2. angular rotation
3. velocity
4. magnetic field vectors

This data can be used to determine position and orientation of device. First step is to determine euler angles, also know as roll, pitch and yaw, using attitude heading reference systems (AHRS). There are various power-related connectors and pins that all have regulated voltage down to 3.3V:

- USB-C
- Lipo Battery connector
- 3V3 Pin
- VIN

USB-C port can be used as a serial communication port for connection to the computer and board programming. This can also be used for connecting the power supply to the board or charging the Lipo battery. The battery connector enables "on the move" data logging and has a built-in charger. OLA comes with a built-in temperature sensor. In addition to a pair of ICs, the board contains a microSD card socket, LiPo battery charging port, power-control switch, and a host of I/O breakouts for project expansion. It comes preprogrammed with example firmware and an Arduino-compatible bootloader, which enables the firmware customization and code upload over a USB connection[1]. The controller has additionally attached one push button that was reused from an old computer mouse and works as an on/off switch as well as a robotic claw close trigger. It is connected to the OLA board that has designated pins, one as input with software pull-up resistor the other as software ground. We have encountered several implementation issues which are described in detail in the discussion chapter.

4.2 JoyC Omni-directional Controller

The second approach has been proposed and tested with JoyC controller(Fig. 4.9) which would provide not only wireless but also truly remote control of manipulator through Wi-Fi. JoyC is a module for the M5StickC IoT (Internet of things) board. JoyC controller supports the two-handed operation and is intuitive to use. It comes with an STM32F030F4 control chip with I^2C (Inter-Integrated Circuit) communication and host M5StickC for data transmission. The control module comes with two joysticks and it's powered by a rechargeable lithium battery with a 700mAh capacity.

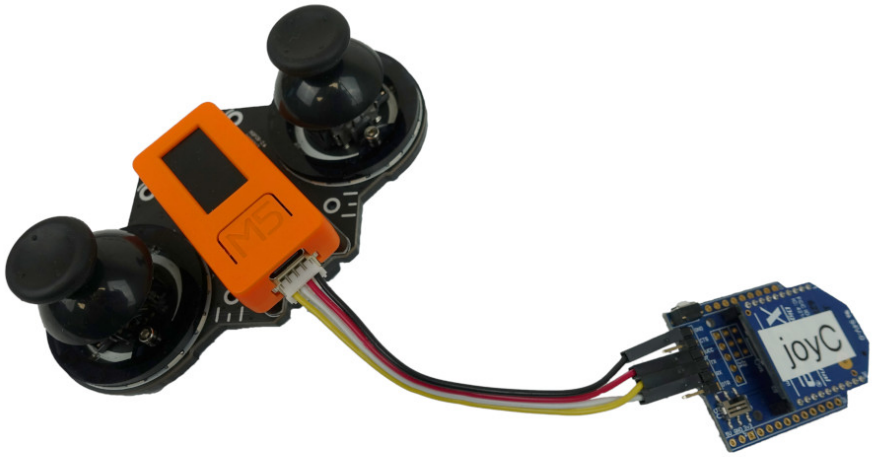


Figure 4.9: JoyC controller

Each joystick moves in two axes to both sides and uses potentiometers to provide analog values of displacement. These values are mapped to an analog range of 0-200 with the 100 value in the middle and therefore providing omnidirectional movement. Additionally, both joysticks double up as buttons.[14] To ensure the communication of JoyC controller with microcontroller of manipulator and Xbee module is used. This module is attached to the JoyC via Grove connector port(Fig. 4.9).

M5StickC ESP32-PICO Mini belongs to the M5Stack microcontroller series. It is portable, user-friendly, and comes with open-source software. M5StickC has integrated ESP32 which is a low-powered chip containing 32-bit dual-core microprocessor with a 240MHz clock rate and 520KB SRAM. This chip supports Wi-Fi and dual-mode Bluetooth. There are three hardware serial ports on the ESP32, one for programming the other two are free. Chip also has manny peripherals namely: capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I^2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), I^2S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation) and many other. M5StickC also contains a built-in 6-axis IMU, infrared transmitter, microphone, button, LCD (Liquid Crystal Display), and built-in Lipo battery.[15]

Functionality

The detailed functionality of specific JoyC controller parts is shown in Fig. 4.10 The left-hand side joystick controls vertical position (Z-axis coordinate) of imaginary point in 2D space (Z and X-axis), which is represented by vertical left-hand side joystick movement and gripper opening/closing , which is controlled by horizontal left-hand side joystick movement. The right-hand side joystick controls position of imaginary point in X for horizontal right-hand side joystick movement. The base rotates imaginary point around Z-axis, base rotation is represented by vertical right-hand side joystick movement. The controllers roll values control rotation of robotic arm wrist and pitch values control grippers attack angles. The right-hand joystick button is used to return arm to it's home position, the left-hand one resets the gripper and the last button starts the controller → arm communications.

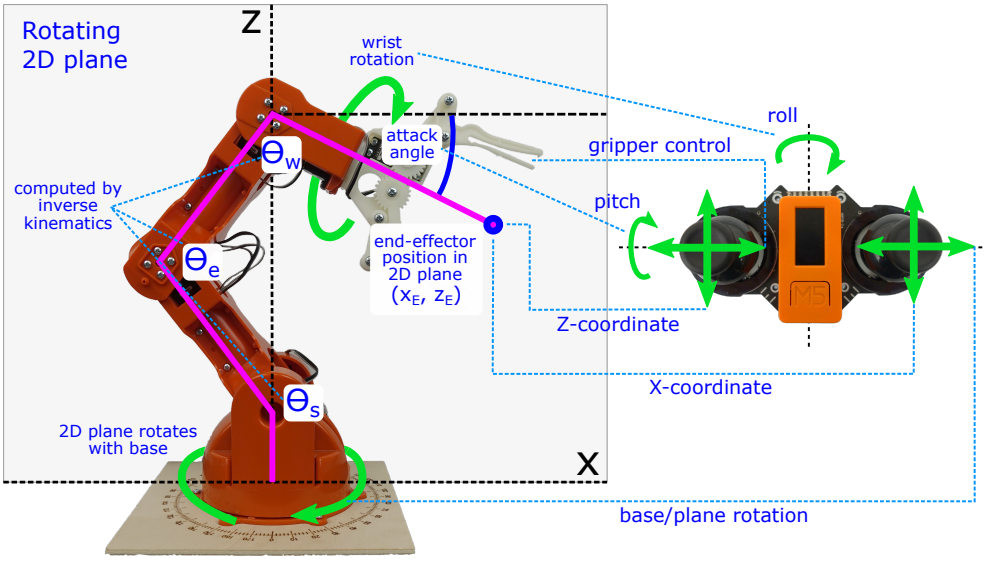


Figure 4.10: JoyC functionality visualisation in reference to the robotic arm control

The servo angles θ_s (shoulder), θ_e (elbow) and θ_w (wrist-vertical) move end effector to the desired X and Z coordinates, these servo angles are calculated using inverse kinematics solver FABRIK (The Forward And Backward Reaching Inverse Kinematics). For code simplification and more interactive control, the simplified kinematic model has been used. The model contains only three links, the base joint and the vertical wrist joint are neglected since they are controlled directly by user.

4.3 Wireless communication module

Various wireless communication technologies are used in projects that require wireless data transfer between its components, namely: WLAN (Wireless Local Area Network), Bluetooth, Zigbee, etc. In this project, we are using Xbee modules that use Zigbee communication protocol. Zigbee operates on low data rates (10-115.2kps) and has low power consumption. The protocol is very reliable and secure and offers a suitable network topology for control applications and multisensor monitoring.[16]

4.3.1 Xbee

XBee modules manufactured by Digi International that support peer-to-peer as well as multi-point network communication with data rates up to 250 kbits/s.[17] Modules cannot edit sent data, only transfer it. However, serial interface communication with other devices is possible.
The communication between two, such as computers or microcontrollers, can be described as follows. The first module takes data sent through the serial input,transmits it through the air to the other side where receiving Xbee module listens for any data transmission on the network. After successful reception, the module writes the data into its serial output. Whether connected to PC or microcontroller, Xbee can be configured and used for data transfer using serial interface[3]. Configuration is a vital step for enabling communication between Xbee modules. Modules are configured using XCTU software by setting DH and DL parameters (destination high and low addresses) of each module that matches the SH and SL parameters (serial low and high numbers) of the other module[18]. There are two types of data transmission as can be seen in Figure 4.11 that take place in a module[3]:

1. Wireless communication: It takes place between Xbee modules. In order to have working communication we need to have modules part of the same network that operate on the same radio frequency. Multiple modules can communicate in such network as long as they meet such requirements.
2. Serial communication: Communication between the Xbee module and the intelligent device connected to it through the serial interface.

4.3.2 Xbee operating modes

The operating mode determines how the host device communicates with an Xbee module through the serial interface. There are two operating modes supported by Xbee:

Figure 4.11: Types of data transmission in an XBee communication process[3]

- transparent - the module passes information in the same way as it receives it. The serial data from an intelligent device, received by the radio module, is sent wirelessly to a remote Xbee module. The other module receives the data and sends it out through the serial port exactly as it was received.
- API (Application Programming Interface) mode is an alternative to transparent mode. When using API mode, a protocol determines the way information is transferred. Data is sent in packets (called API frames). API mode allows us to form larger networks and is more suitable for creating sensor networks to collect data from multiple locations[19].

Chapter 5

Robotic Arm Hardware in Detail

This chapter deals with hardware parts of the robotic arm, their technical description, and how they operate. The robotic arm contains servo motors that act as actuators and carry out all of the arms movement (these are the joints from a kinematic point of view), a Braccio shield that acts as an interface for microcontroller and contains six outputs for servo control, and microcontroller Romeo V2 on which the control code runs.

5.1 Servo motors

The arm itself contains 6 servo motors that handle rotations or movement at each given place. Since the last servo controls the opening and closing of the gripper we're considering the Braccio manipulator to have five degrees of freedom in terms of end effector position.

Term servo can be used to describe various different machines. Servo is basically any system containing a motor with built-in position feedback control. Servos can be found in:

- heavy machinery
- power steering in vehicles
- robotics
- wide variety of electronics

5.1.1 Servo operation principle

Standard servo motor (Fig. 5.1) contains three main components:

- DC/AC motor
- controller circuit
- potentiometer(or other feedback mechanism)

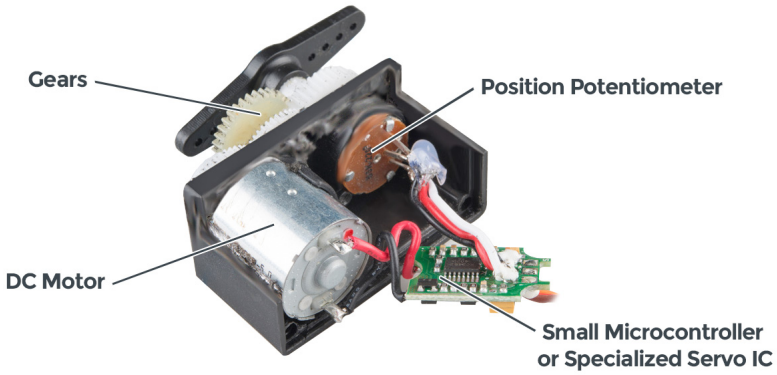


Figure 5.1: Cross section of standard servo motor[4]

The motor is attached to the gearbox and drives a shaft. The control circuit interprets position command sent from the controller, and the potentiometer monitors the output shaft's position which is feedback for the controller. Most of the standard servo motors are powered and controlled through a standard three-pin connector. The color coding varies between brands but the pins' position is mostly universal. Through these three wires, the servo can be powered and the output shaft's direction, speed, and position can be controlled.[4]

Servo control

Servo is controlled by signals that are precisely timed. Standard servo motors are controller by pulses in 20ms time periods where position is determined by pulse width. A control signal shaped like this is often called PWM. Controller inside of servo takes inputs from other hardware components such as potentiometer, joystick or sensor feedback and sets position command for servo. Other control options are PWM-capable pins on the microcontroller that send signals straight to the servo.[4]

Powering servo

Input voltage for servos varies depending on it's size and torque output but servos,that are designed for low-torque and DIY applications, run on 5V. The current draw when

servo is moving with attached load is very important. The majority of servos when unloaded pull around 10mA, however larger servos with load are able to pull more than Ampere.[4]

Servo types

A standard (closed-loop) servo (Fig. 5.2) have range of movement from 90 to 180 degrees (or greater/lesser range). Standard servo sends feedback over the control signal wire to the controller for position monitoring. This allows for precise servo positioning with the correct length pulse send from controller.

Figure 5.2: Standard servo movement range

Continuous rotation (open-loop) servos (Fig. 5.3) have full 360 degrees movement range, but position is not controlled through generated control signals, only direction and speed. There's no position feedback, therefore they aren't suitable for applications that call for movement between exact rotation arc's points.[4]

Figure 5.3: Servo with continuous rotation movement range

Driving a servo

There are multiple ways to control servo. Two main ones are using a direct servo controller or PWM pin on the microcontroller. If we want to control the servo through a microcontroller we need to use a circuit board that has at least one PWM-capable pin. Using controller, of course, requires intermediate programming knowledge in Arduino, Python, or other programming languages suitable for microcontroller boards. Using a microcontroller is suitable when we want to bind servo movement to a specific event such as a change in sensor data. One drawback is that without a proper code there's no direct way to control the servo unless we convert the microcontroller to the servo driver. Using a servo driver: A dedicated driver for servo usually comes pre-programmed with a microcontroller for interpreting external inputs such as a button, potentiometer, or received serial data. Servo driver enables direct user interaction with direction, position, and speed of servo motor and is suitable for projects that require direct position control.[4]

5.2 Braccio shield

Shield enables direct servo control from microcontroller and is powered by 5V. Shield sit directly above the microcontroller and acts as direct electrical control interface. It extends all the pins from microcontroller and has similar form as standard Arduino shields, therefore it can be extended by various other shields (Fig. 5.4)[1].

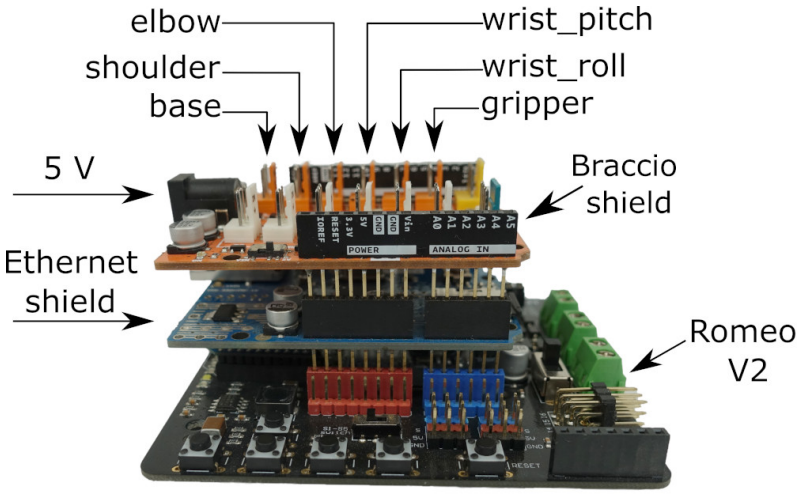


Figure 5.4: Braccio shield demonstration of stack-ability with marked outputs for servo control

5.3 RoMeo V2 board

Romeo V2 board (Fig. 5.5) is alternative to Arduino Uno that comes with the Braccio shield mainly because it has two hardware serial communication lines. Arduino Uno has only one hardware serial line. It's possible to use software serial communication and assign specific pins for it but it's not very reliable. RoMeo V2 is Arduino compatible microcontroller suitable for robotic applications. Board is supported by numerous open source codes and can be expanded by various extension shields of Arduino form factor. Board comes with integrated two-way DC motor driver and Xbee socket. RoMeo V2 is equipped with Atmel ATmega32u4 chip and an Arduino Leonardo bootloader. It can be powered through USB (Universal Serial Bus) port or via power connector (6-23 Vdc). The board provides 20 GPIO pins, of which 12 have analog input capability and

[1]Ethernet shield is used only for stack-ability demonstration. It isn't used in project.

7 can be configured as 8-bit PWM outputs. There are various interfaces supported on the board, namely: SPI, I2C/TWI and UART.

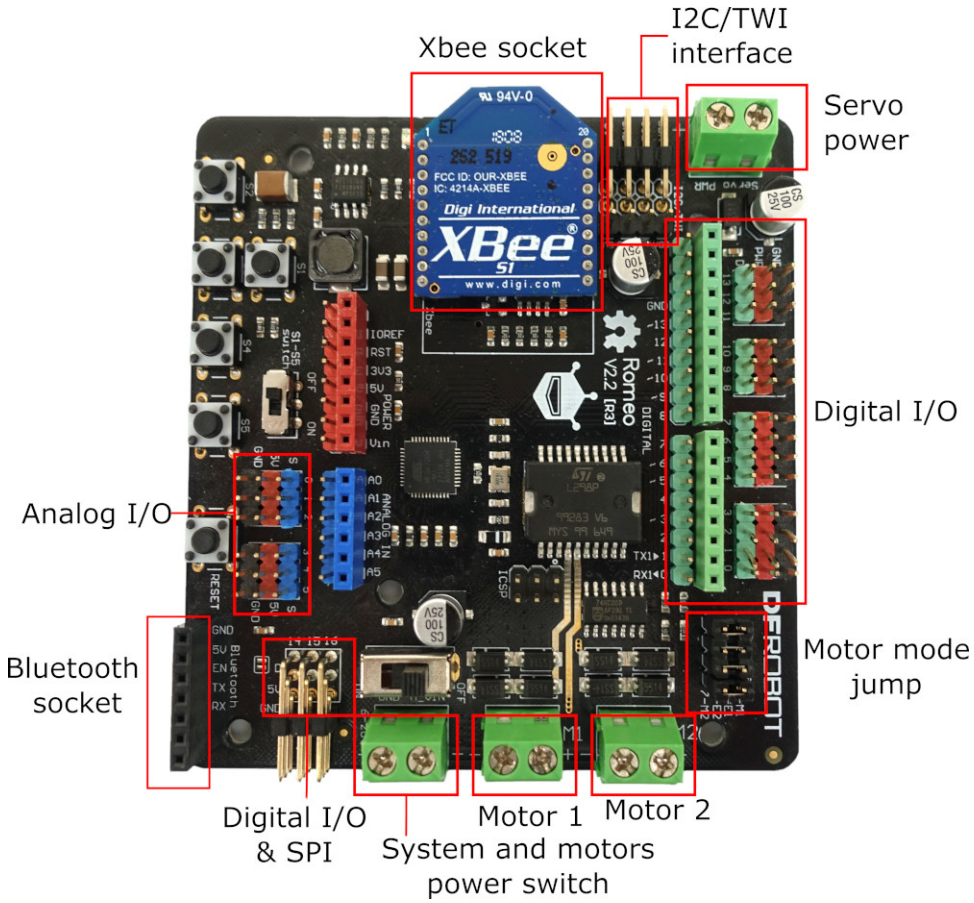


Figure 5.5: Romeo V2 pinout

5.4 Description of Operation

DIY controller

Implementation of arm's control for DIY controller required the 9-DOF AHRS library, accelerometer and gyroscope calibration, and gravity compensation for roll, pitch and yaw values. The user holds the on/off the button and starts the roll, pitch, yaw, and button state data reading and serial communication. OLA board reads accelerometer and gyroscope data, these values are scaled and roll, pitch and yaw are calculated. Further, the gravity compensation is calculated. Heading and button state values are forwarded to the Xbee module. Xbee sends data to another paired Xbee module. On the manipulator side, Romeo V2 listens for any data from serial communication. The servo motors are controlled according to heading values and the gripper is opened/closed according to the current button state. Control command is sent through the Braccio shield to the manipulator. Data flow can is visualised for better understanding in Fig.5.6 and Fig. 5.7.

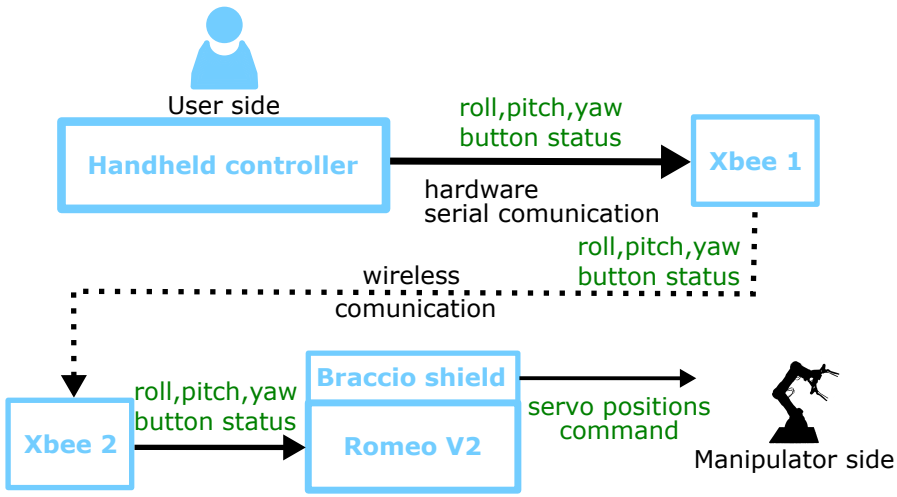


Figure 5.6: General data flow for DYI controller and manipulator system

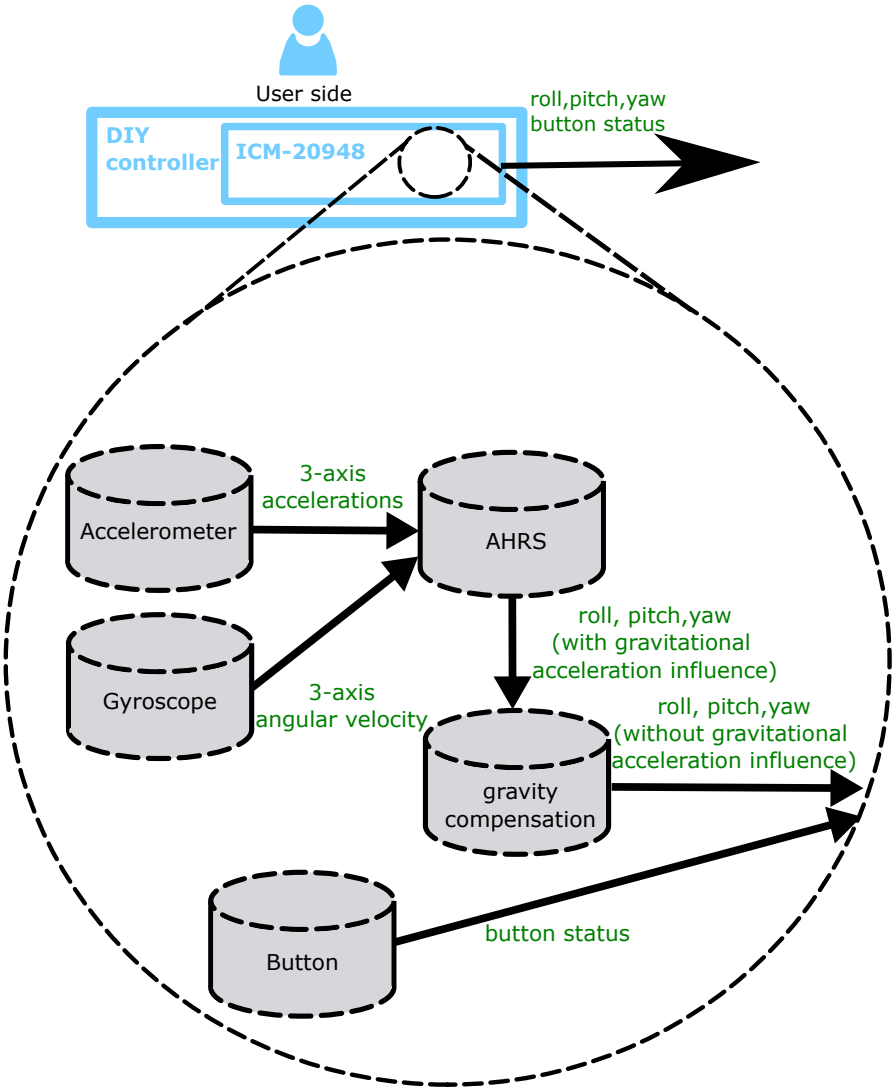


Figure 5.7: Diagram of some of the OLA boards internal parts and data flow within them

JoyC controller

In the case of JoyC controller, all the necessary calculations are carried out at the controller side and only the servo control message is sent to the manipulator. Every movement of vertical joystick movement (right or left side) results in a relative increment in the X or Z axis and it is added to desired position of the point that represents the end effector of the arm. The absolute end effector position at the beginning matches the arm's home position. The horizontal movement of the right-hand side joystick controls the rotation of the arm's base. The values in the X and Z axes are used to calculate servo angles using Fabrik solver[20] which we have implemented in the code. These servo motor values are constrained so they wouldn't exceed the physical limits of servos. The horizontal left-hand side joystick servers as gripper opening/closing control. Integrated IMU has ahrs included and transforms raw sensor data to roll and pitch values, we are using roll for rotational movement of robotic wrist and pitch for "attack angle". The left button serves as manipulator position reset and sends the manipulator back to the home position, the right one resets the gripper to the initial position.

Inverse kinematics implementation

The Forward And Backward Reaching Inverse Kinematics (FABRIK) solver[11] provides a fast solution based on iterative calculations. Each joint position is found by locating a point on a line. Solver supports all chain classes, has an option for fixed end-effector orientation, and can be applied to highly complex systems. The designed algorithm is efficient whether applied to simple or complex problems while requiring fewer iteration steps and shorter processing time compared to other sophisticated methods. The FABRIK algorithm can be found in[20].

Solver has been implemented directly into the JoyC controller's microcontroller with consideration of a simplified kinematic model with only three links.

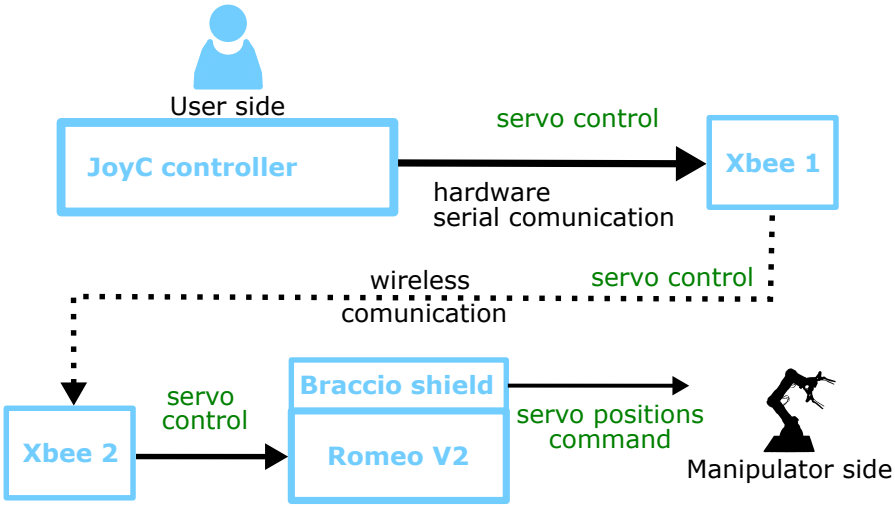


Figure 5.8: General data flow for JoyC controller and manipulator system

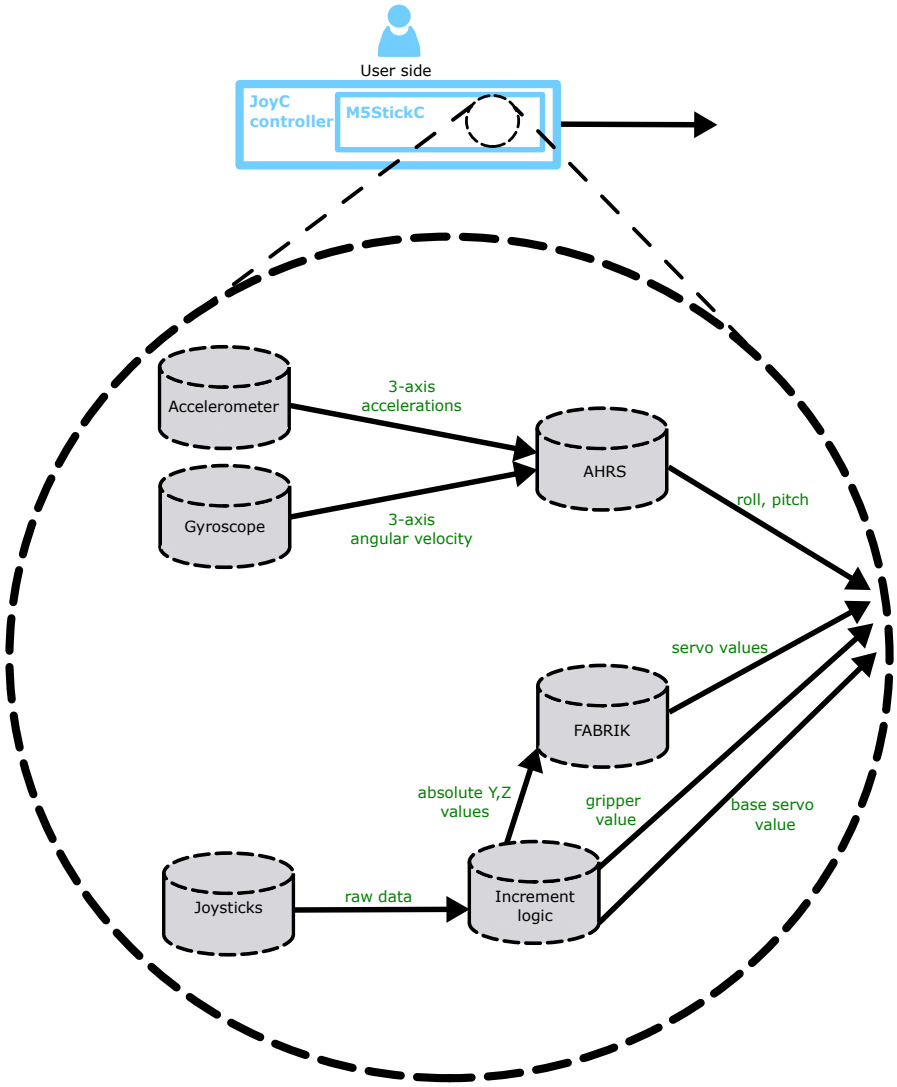


Figure 5.9: Internal parts and data flow diagram of JoyC controller

Chapter 6

Demonstration of functionality on a pick and place task

We can verify whether our kinematic equations are correct by demonstrating some simple movements on the physical robotic arm. We are going to calculate these equations with selected series of different *theta* values which will give us X, Y, and Z-axis values for the position of the end-effector. Next, we are going to set servo motor positions of the physical arm to the same *theta* values. If the position of the end-effector in the physical arm matches the values from kinematic equations the equation was assembled correctly. The X, Y, and Z coordinate values need to be measured by ruler or some other measuring tool, or the arm needs to be placed into the environment with graph paper surfaces and values need to be measured manually. The more practical approach is to use a robotic toolbox[11] in Matlab. That provides interactive kinematic model visualization that changes θ values according to user's inputs and outputs the X, Y, and Z-axis end effector positions values. This model needs to be first also tested by comparing it with the physical arm. After testing, the model replaces the physical arm for all the other testing such as inverse kinematic equations testing. This virtual model is more practical since it immediately outputs end effector position values and doesn't require the physical arm.[1] The process itself needs to be tested for different end-effector positions. In figures below is depicted the "pick and place" task for inverse kinematic and overall functionality test of JoyC $\leftrightarrow$ Braccio robotic arm system. The process reqiured this steps:

- picking up and placing of an object with arm using JoyC controller (Fig. 6.3)
- outputting data during this tasks to serial line, namely: raw joystick data, roll and pitch values from the gyroscope, X and Z-axis position values of end effector, and joint angle values calculated using inverse kinematics
- saving the data in CSV (Comma-separated values) format into the file

[1] Considering the restricted access to university due to the world pandemic.

- loading the data into Matlab and plotting the graphs (Fig. 6.2)
- movement recreation with the model in the Robotic toolbox at specific selected times (Fig. 6.1)
- comparing the results (It's important to note that the real-life arm and its model are depicted from different angles and the model has Y-axis converted. Therefore they might not look the same even though they are performing the same movement.)

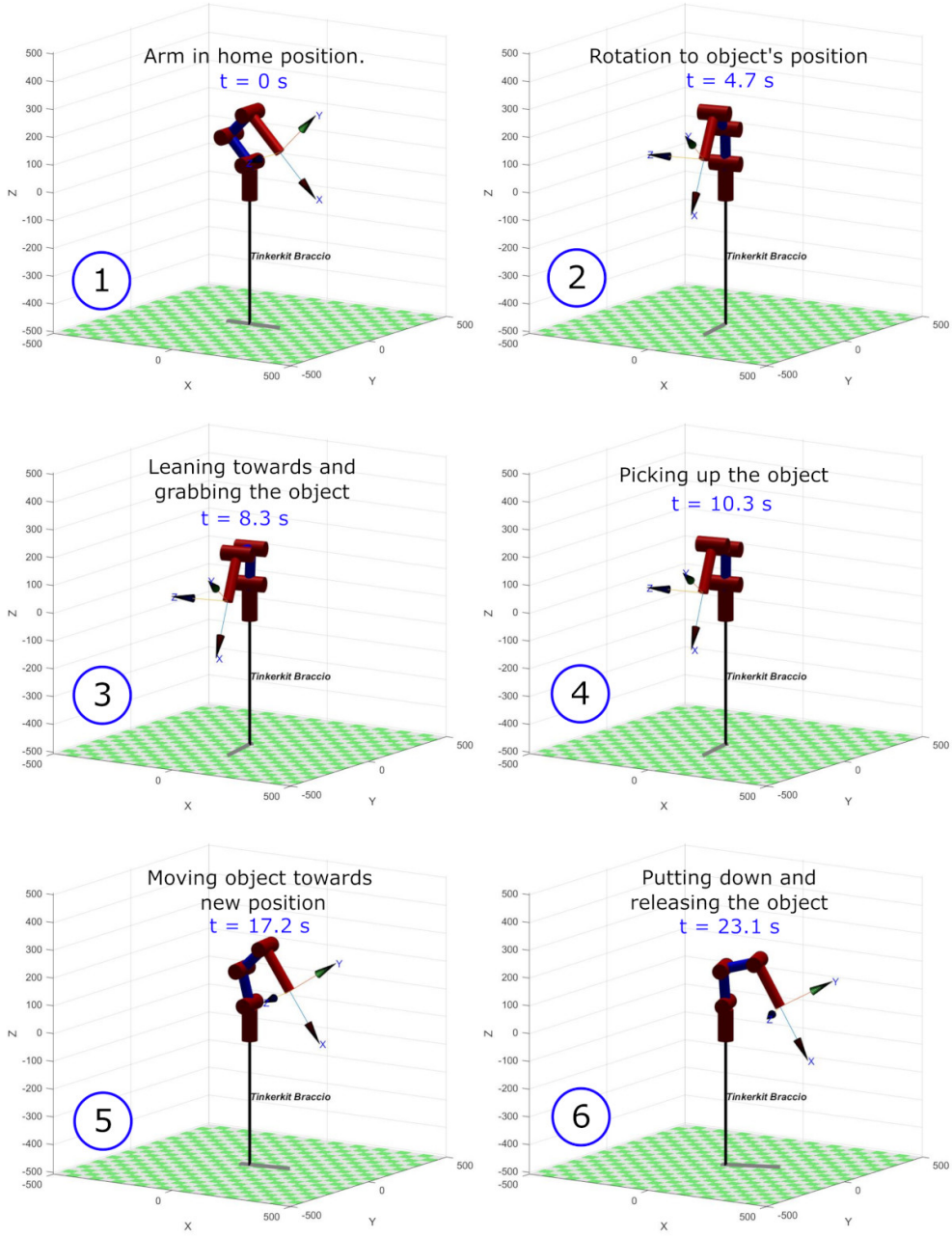


Figure 6.1: Robotic toolbox visualisation for simple pick and place task

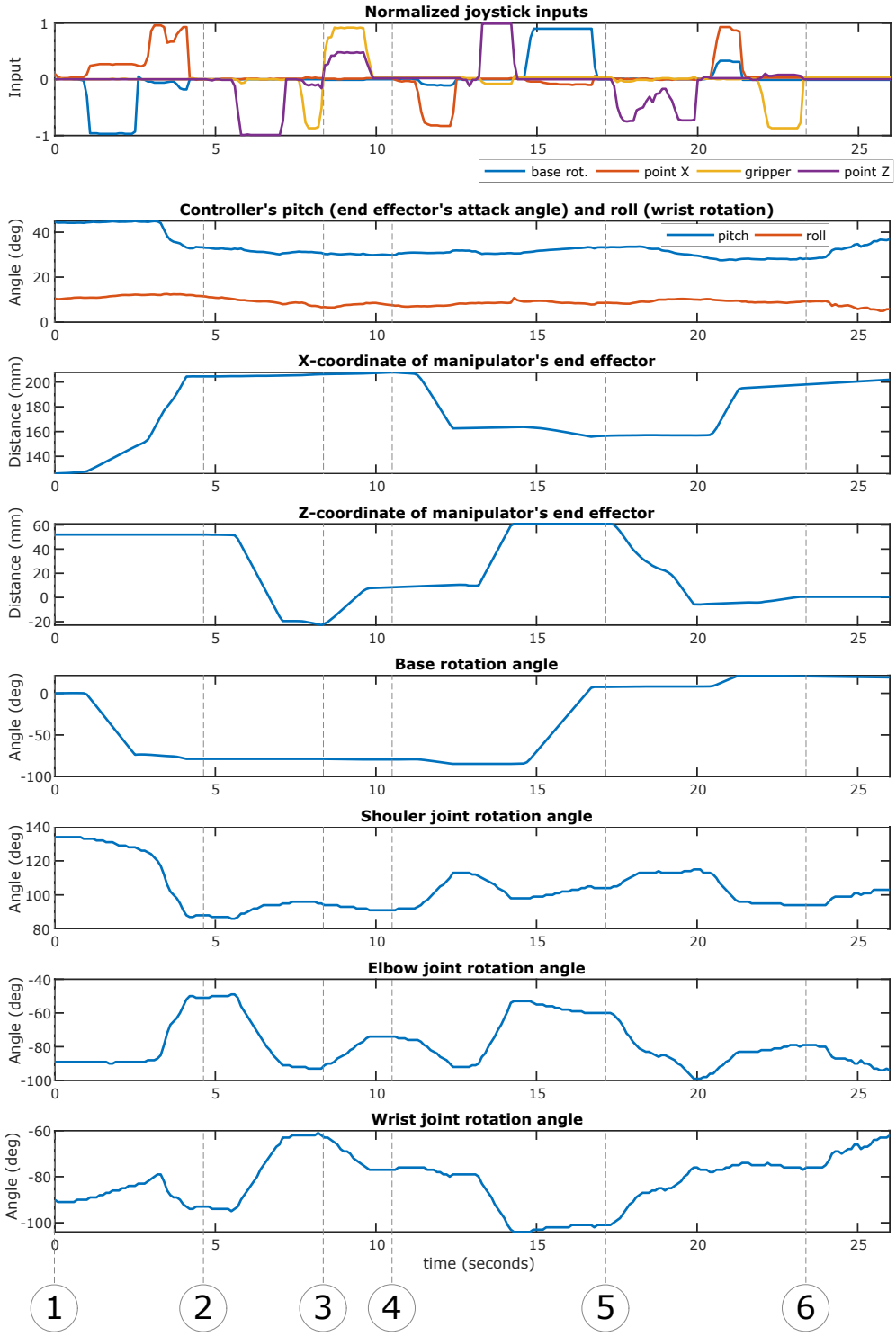


Figure 6.2: Graph of parameter values for pick and place task over time

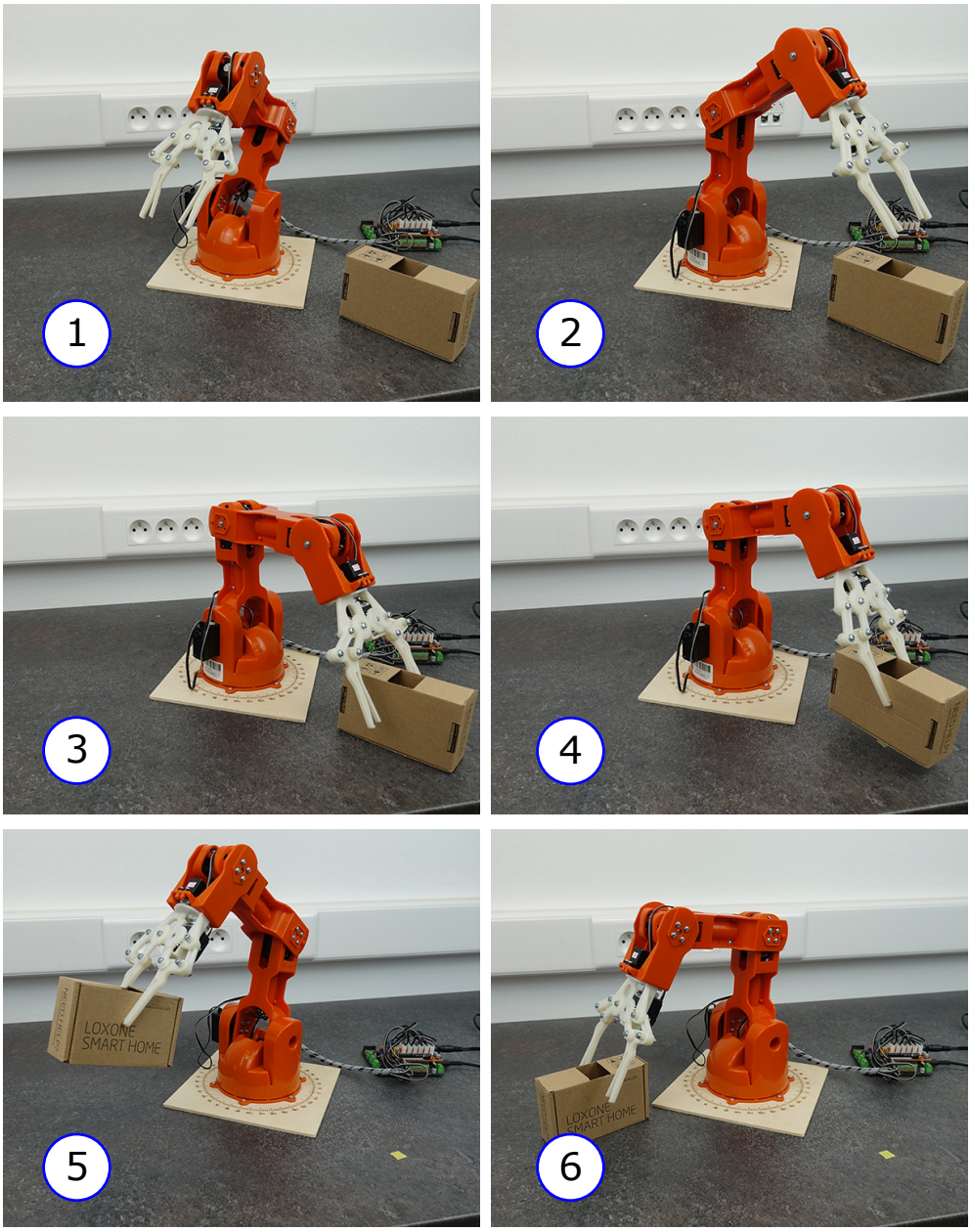


Figure 6.3: The arm position for pick and place task in selected times

Chapter 7

Conclusion and Discussion

Conclusion

The kinematic diagram for Arduino Braccio has been assembled and the forward and inverse kinematics for Braccio robotic arm has been derived. The DIY controller was designed, 3D printed, and assembled. Its internal electronics have been configured and calibrated, and AHRS code has been implemented. In the next step, the inverse kinematics has been implemented to the commercial controller and to the Romeo V2 board. Both controllers have implemented the mechatronic system. For both controllers, the wireless communication has been configured. In the last step, the system functionality and kinematic equations have been tested with a demonstration of "pick and place" tasks which has proven to be successful.

Discussion

We have encountered several implementation issues. The first issue was with heading calculations within the handheld controller. Roll and pitch are easily calculated but yawn can be accurately determined only by using magnetometer data. In order to get reliable magnetometer data, two kinds of calibration need to be performed. The first, standard one is to compensate Earth's magnetic field from sensor data also known as hard iron errors and can be affected by nearby electronics and metal objects.

The second, advanced magnetometer calibration compensates hard and soft iron errors from sensor data output but currently available GUI (Graphical User Interface) interfaces aren't compatible with the board we are using. Since we are using strong magnets as a closing mechanism the magnetometer data is rather unreliable and therefore cannot be used.

The second one was with position calculation. Accelerometer also requires calibration and gravitational force compensation which requires accurate sensor orientation esti-

mate. Since the orientation can never be estimated perfectly some percentage of normal forces won't be removed and will be mistaken for physical acceleration. In order to calculate position, we need to integrate acceleration twice which results in even bigger errors and these positions drift rapidly over time. Apparently, all low-cost sensors (below a hundred dollars) will have an error of such magnitude that accelerometer-based estimates of velocity and position will be useless in most applications. An angle error of sensor orientation even as small as one degree causes errors of $1.7m/s$ in velocity and $17.1meters$ after only $10seconds$ and builds up over time to $62kilometres$ after $10minutes$.[21] This issue can be solved by applying advanced compensation algorithms to the sensor values.

Another problem arose with inverse kinematics implementing not only is gets gradually difficult to adjust calculations to match frames from control device through a kinematic model of given inverse kinematics method to the manipulator, but another issue is to find and implement library or solver that would:

- be compatible with used microcontroller board
- provide solutions that are reliable and feasible in our work frame

This problem can be overcomed by either simplifying the kinematic model or by using more complex inverse kinematic methods.

The last issues was connected to the fact that the arm, controller and solver had each differently oriented frames. In order to achieve correct results, the frame unification during implementing had to be performed.

Bibliography

[1] "Openlog artemis hookup guide." https://learn.sparkfun.com/tutorials/openlog-artemis-hookup-guide. Accessed: 12-01-2021.

[2] "What is a gyroscope and how does it work?." https://www.youngwonks.com/blog/What-is-a-Gyroscope-and-How-Does-It-Work.

[3] "How xbee devices communicate." https://www.digi.com/resources/documentation/Digidocs/90001942-13/concepts/c_how_xbees_communicate.htm?tocpath=How%20XBee%20devices%20work%7C_____1. Accessed: 12-01-2021.

[4] "Servos explained." https://www.sparkfun.com/servos. Accessed: 03-05-2021.

[5] R. N. Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Control*. Springer Science and Business Media, 2010.

[6] L. I. Slutski, *Remote manipulation systems: quality evaluation and improvement*, vol. 17. Kluwer Academic, 1998.

[7] P. C. Jean Vertut, *Teleoperation and Robotics: Evolution and development Robot technology – Volume 3, pages =*.

[8] A. Sodemann, "Robotics 1 2017." https://www.youtube.com/watch?v=pLXoDRctwRg&list=PLT_0lwItn0sDBE98BsbaZezflB96ws12b, note = Accessed: 12-01-2020.

[9] M. V. Mark W. Spong, Seth Hutchinson, *Robot Modeling and Control, year =*.

[10] R. M. Murray, *A Mathematical Introduction to Robotic Manipulation, year =*.

[11] "Matlab robotic toolbox." https://www.sciencedirect.com/science/article/pii/S152407031100017

[12] D. Bensky, *Short-range Wireless Communication: Fundamentals of RF System Design and Application [Communications Engineering Series]*. Communications Engineering, Independently Published, 2020.

[13] "Polymer lithium ion battery - 400mah (sparkfun prt-10718)." https://rlx.sk/sk/battery-lipo-li-po-polymer-lithium-ion/3305-polymer-lithium-ion-battery-400mah-sparkfun-prt-10718.html. Accessed: 12-01-2021.

[14] "Joyc." https://docs.m5stack.com/en/hat/hat-joyc?id=easyloader.

[15] "M5stickc esp32-pico mini iot development kit." https://shop.m5stack.com/products/stick-c.

[16] N. D. Amartya Mukherjee, Ayan Kumar Panja, *A Beginner's Guide to Data Agglomeration and Intelligent Sensing*. Elsevier, 2020.

[17] "Xbee datasheet." https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf. Accessed: 12-01-2021.

[18] "Configure the xbee modules." https://www.digi.com/resources/documentation/Digidocs/90001456-13/tasks/t_wk_configure_xbees.htm. Accessed: 12-01-2021.

[19] "Operating modes." https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_transparent_and_api_mode.htm?TocPath=How%20XBee%20devices%20work%7CWireless%20communication%7CSerial%20communication%7C_____1. Accessed: 12-01-2021.

[20] A. Aristidou and J. Lasenby, "Fabrik: A fast, iterative solver for the inverse kinematics problem," *Graphical Models*, vol. 73, no. 5, pp. 243–260, 2011.

[21] "Using accelerometers to estimate position and velocity." http://www.chrobotics.com/library/accel-position-velocity. Accessed: 25-04-2021.