NTNU
Kunnskap for en bedre verden

Department of Engineering Cybernetics

TTK4900 - Master project

Retrofitting a legacy robotic arm using open-source solutions

Author:
Kristian Blom

07.06.24

Table of Contents

List of Figures

List of Tables

1 Preface

2 Introduction

2.1 Purpose

This report describes the master thesis project conducted by Kristian Blom during the spring semester of 2024. The project builds directly on the specialisation project concluded the previous semester of autumn 2023. This report contains relevant excerpts from the specialisation project report.

2.2 Background and motivation

From the initial project description: *This project is defined and commissioned by a student in co-operation with Omega Verksted (OV). It builds on the specialisation project named "From hardware to control algorithms: Retrofitting a legacy robot using open source solutions", in which new control hardware was developed for an old robotic arm. The arm is a 6DOF industrial arm originally developed for chemical analysis and consists of a 5DOF arm with a pincer/manipulator installed on a rail. The developed hardware controls 6 brushed DC motors and processes information from 6 incremental encoders, 6 current sensors, 3 accelerometer/gyroscope units, 2 optocouplers and 1 mechanical switch.*

2.3 Project scope

The primary goal of this project has been to develop a robotic software system compatible with hardware developed during the specialisation project, such that the robotic arm may be operated by a non-expert third party. The primary use case is the mixing of beverages during informal meetings at Omega Verksted. The secondary goal of the project is that the software system should be readily expandable to support more sophisticated use cases and sensors, such as 3D printing and robotic vision, by an expert third party.

Project scope as itemised list:

1. Implement and verify required hardware improvements from the specialisation project.
2. Write hardware drivers for the relevant microcontroller peripherals.
3. Develop and implement a software architecture around the primary and secondary goals of the project.
4. Develop and implement tests to evaluate the system's performance.

3 Abbreviations

4 System specification

This section provides a high level description of the robotic system(4.1), a description of the hardware developed during the specialisation project(4.2), and a requirement specification for the software developed in this master project(4.3). The latter elaborates on scope items 2 and 3 from section 2.3.

4.1 Overview

The system consists of a robotic arm known from the original manufacturer, Beckman Coulter, as Optimized Robot for Chemical Analysis (ORCA); hardware developed as part of the specialisation project; and a general purpose desktop computer running the Linux/Ubuntu operating system. See figure 1.

The arm has several "hardpoints" on which PCBs or other equipment may be mounted, named after their physical location on the arm. The three control units developed in the specialisation project are mounted on the torso, shoulder and hand hardpoints, respectively. They are each responsible for the control of two joints, named after their function and/or physical location. For instance, the shoulder control unit is mounted on the shoulder hardpoint, and is responsible for controlling the elbow and wrist joints.

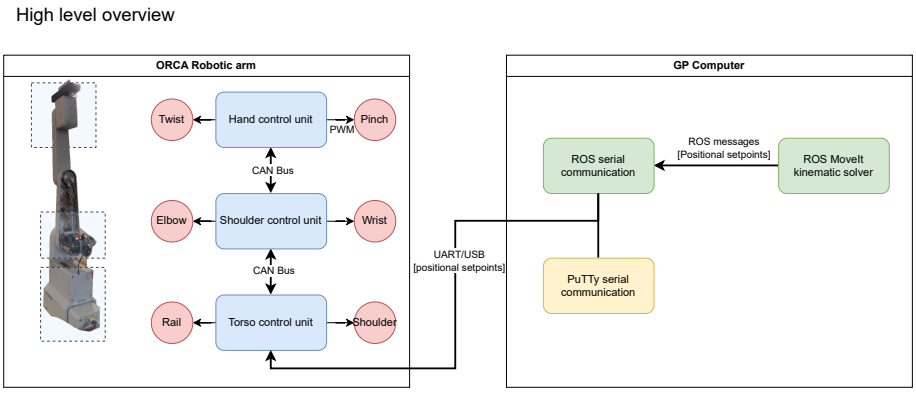


Figure 1: An overview of the robotic system

4.2 Hardware

As mentioned in section 2.2, all mechanical and some electromechanical parts were kept during the specialisation project in order to limit its scope. The parts are presented in table 1, originally presented in the specialisation project report. These parts were all found to be in working order, and replacing them would have required modification of the arm chassis.

For an in-depth discussion and presentation of the hardware, see sections 6 and 7 of the specialisation report. A key point from those sections is that there are space limitations in the arm preventing IMUs from being placed such that they may interface with their respective joint controllers directly. As a consequence, a critical function of the CAN bus is to enable transmission of IMU data between control units.

The following subsections explain each PCB developed in the specialisation project, and are summarised in figure 2. Figure 3 provides a detailed description of the arm, and was originally presented

in the specialisation project report[2].

Table 1: Original parts kept in the project

Part name	**Description**	**Part no.**
Rail motor	Pittman 40mm 30.3V BDC motor	9233C59-R1[8]
Shoulder motor	Pittman 54mm 30.3VDC BDC motor	14205C389-R1[9]
Elbow motor	Pittman 40mm 24V BDC motor	9234C59-R1[8]
Wrist motor	Pittman 30mm 30.3V BDC motor	8224C143-R2[7]
Twist motor	Escap 23mm 23LT2R, 12V BDC motor	23LT2R[5]
Pinch motor	Escap 23mm 23LT2R, 12V BDC motor	23LT2R[5]
Wrist rotational sensor	TT Electronics Photologic Slotted Optical Switch	OPB971N51[4]
End switch	TT Electronics Photologic Slotted Optical Switch	OPB971N51
Motor encoder	Optical quadrature encoder 500CPR	HEDS-9100[10]

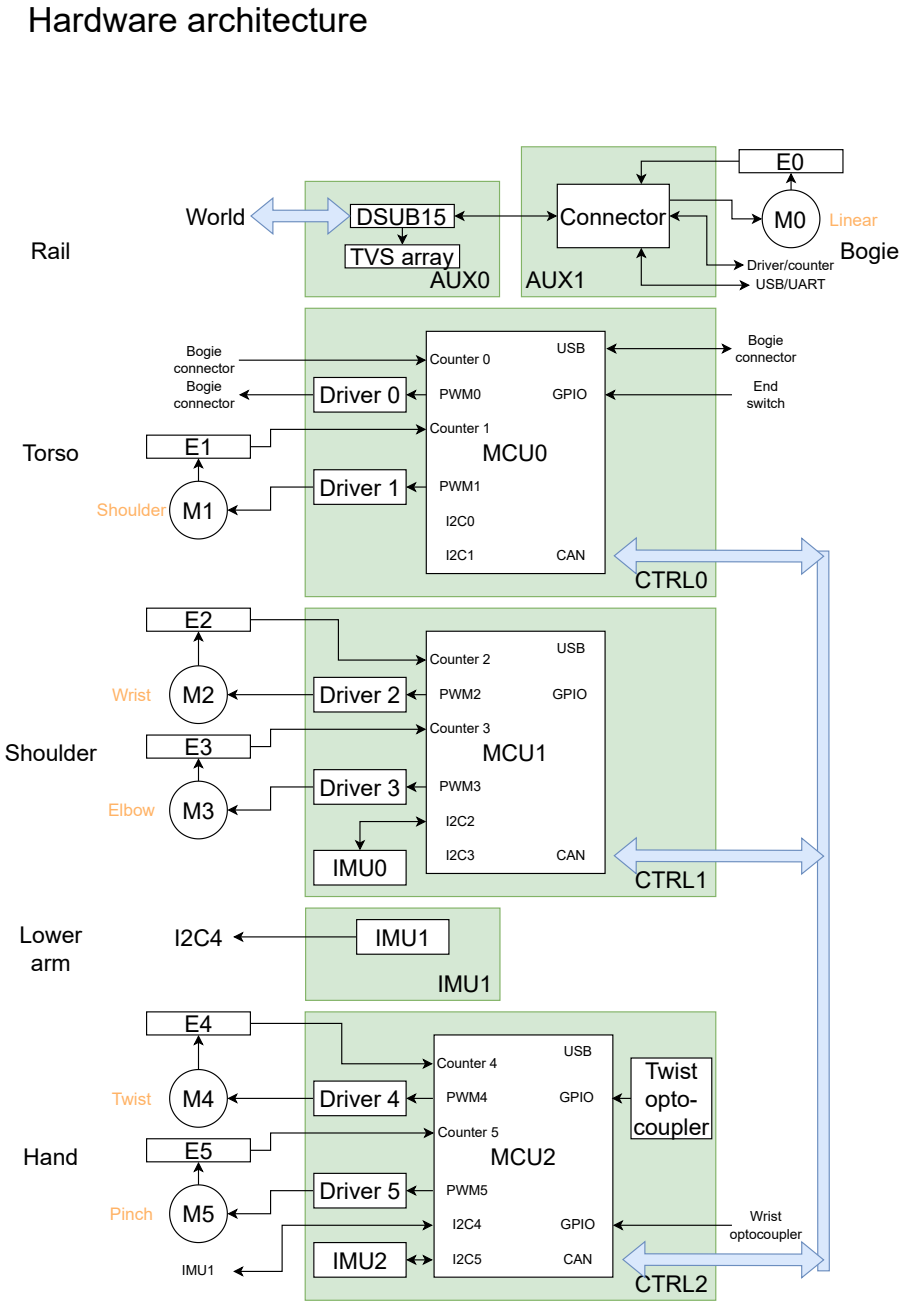
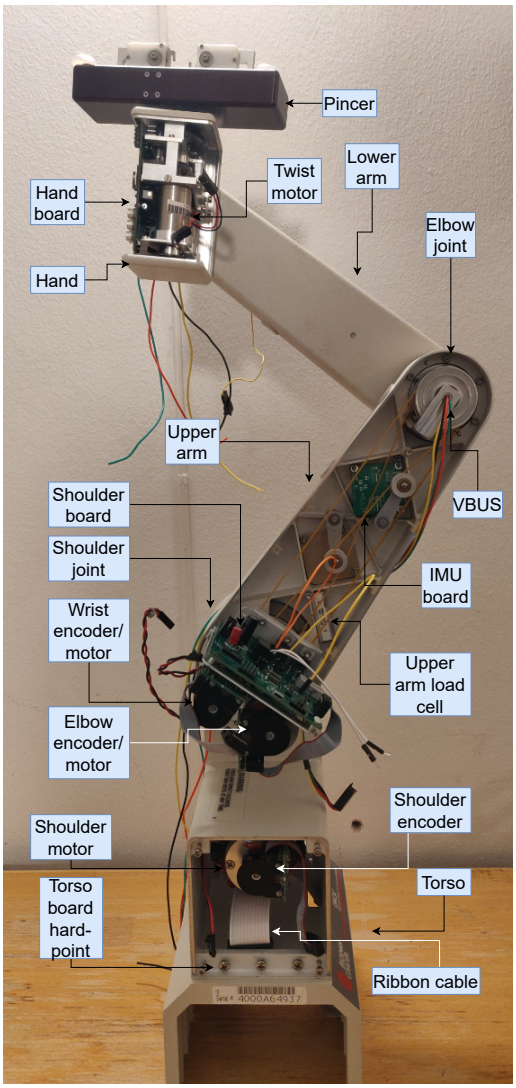
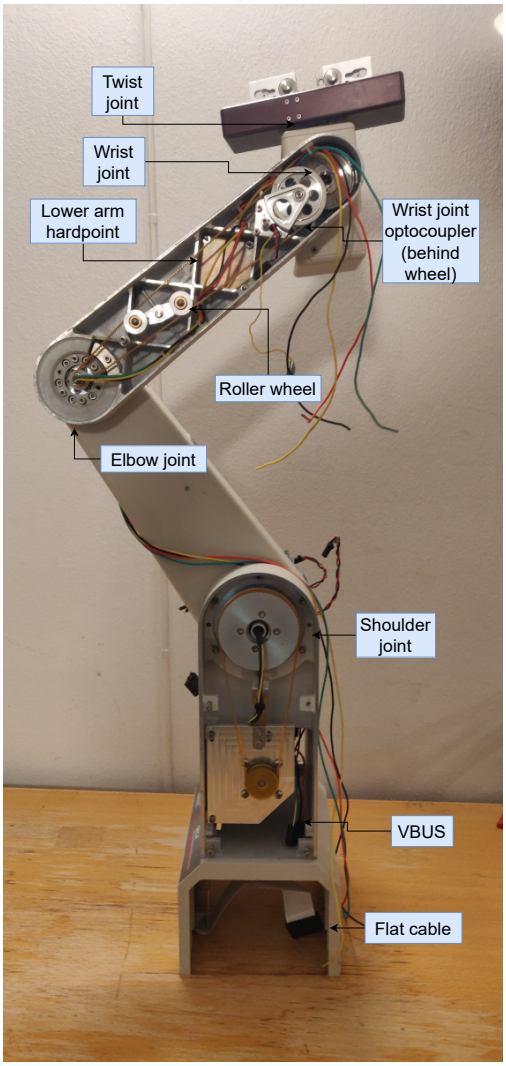


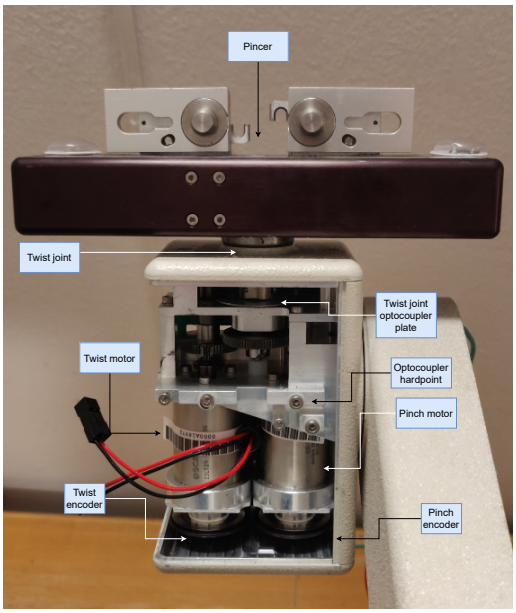
Figure 2: Hardware architecture. Note that the upper arm IMU has been moved from a separate PCB onto the shoulder control unit PCB, compared to the presentation in the specialisation report



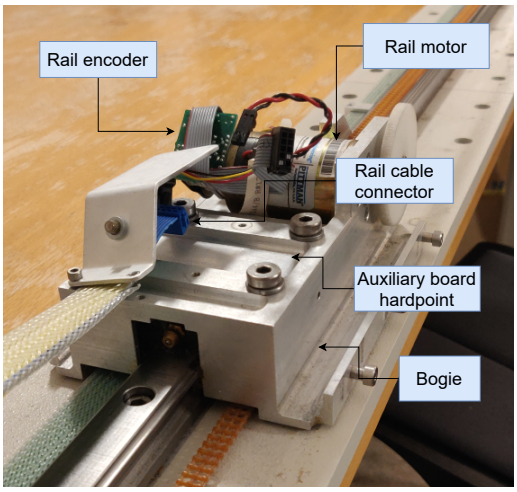
Arm viewed from the front.



Arm from the back.



A more detailed image of the hand and pincer.



The bogie on which the torso is mounted.

Figure 3: The arm with annotations.

4.2.1 Auxiliary 0: Rail

The rail PCB is responsible for the arm's interface, a DSUB15 connector with signals for USB, UART and power. It is mounted on the rail hardpoint, and provides a socket for the 16 wire ribbon cable through which it interfaces with the bogie PCB(4.2.2). Additionally, it protects the USB and UART data lines from voltage spikes via its TVS array CDSC706[3]. The TVS array is clamped at 5V sourced from the torso control unit(4.2.3).

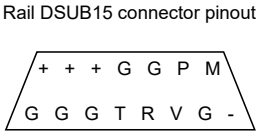


Figure 4: DSUB15 pinout

Table 2: Legend for figure 4, DSUB15 connector

Legend	**Meaning**
+	Input voltage
G	Ground
P	USB_DP
M	USB_DM
T	UART Tx
R	UART Rx
V	USB_VBUS
-	Not connected

4.2.2 Auxiliary 1: Bogie

The bogie PCB provides sockets for the 16 and 20 wire ribbon cables, as well as the rail motor and encoder connectors. Its purpose is to bundle the 16 wire ribbon cable with the rail motor and encoder wires into the 20 wire ribbon cable, through which it interfaces with the torso control unit(4.2.3).

4.2.3 Control unit 0: Torso

The torso control unit is responsible for the control of the rail and shoulder joints via motor 0 and motor 1 respectively, as well as external communication via the USB/UART lines. It interfaces with the upper arm IMU, IMU0 (4.2.4), via the shoulder control unit and the CAN bus. IMU0 is used in conjunction with E1 for the estimation of shoulder joint position. See table 3 for a presentation of the symbols in figure 2.

4.2.4 Control unit 1: Shoulder

The shoulder control unit is responsible for the control of the elbow and wrist joints via motors 2 and 3 respectively. It interfaces with the lower arm and hand IMUs, IMU1 and IMU2, via the hand control unit (4.2.6) and the CAN bus, for control of the elbow and wrist joints in conjunction with encoders 2 and 3, respectively. Additionally, it is responsible for direct communication with IMU0 via I2C. For remaining items, see table 3.

Table 3: Control unit 0

Symbol	Description	Unit name
Encoder 0 (E0)	Registers movement in the rail motor, used in estimation of the rail joint position. Relative, quadrature	HEDS-9100
Encoder 1 (E1)	Shoulder motor, see E0	HEDS-9100
End switch	Optical sensor, detects linear rail end position	See table 1
Driver 0	Motor driver, supplies power to the rail motor. PWM control	DRV8251A
Driver 1	Shoulder motor, see driver 0	DRV8251A
IMU0	Upper arm IMU	LSM6DSM
Microcontroller 0 (MCU 0)	Processing unit for control unit 0	STM32F303
Motor 0 (M0)	Linear rail motor, brushed DC	See table 1
Motor 1 (M1)	Shoulder joint motor	See table 1

4.2.5 IMU board 1: Lower arm

The IMU board serves as a mount point for the lower arm IMU, IMU1, which is used in the estimation of elbow joint position, and interfaces with the hand control unit via I2C. The IMU board also interfaces with the wrist optical sensor, relaying 5V from the hand control unit and sensor output to the hand control unit. IMU board 1 is mounted on the lower arm hardpoint.

4.2.6 Control unit 2: Hand

The hand control unit is responsible for the control of the twist and pinch joints via motors 4 and 5 respectively. It interfaces with the twist optical sensor and the wrist optical sensor via GPIO interrupts, and the CAN bus. Additionally, it is responsible for direct communication with IMUs 1 and 2 via I2C.

The twist optical sensor activates when the twist joint is in one specific position. As the twist joint has no hardpoints on which an IMU may be mounted, this sensor is essential to the estimation of twist joint position.

The wrist optical sensor activates for a set of joint positions, and may be used in the estimation of wrist joint position.

For remaining items, see table 3.

4.3 Software

4.3.1 Functional analysis

5 Theory

5.1

6 Tools and workflow

This section describes the tools used in the project, for the purpose of reproduction and/or future development.

6.1 Software

6.1.1 Tools

KiCad 7, CERN distro[SOURCE] was used in the development of PCBs in the project's initial phase, without non-standard plugins. KiCad is an open source, freely available CAD software for the design of PCBs. When necessary, component footprints were downloaded from the UltraLibrarian[SOURCE] website when available, or otherwise drawn in KiCad based on component datasheets. These footprints may be found in FOLDER.

Git was used during all phases of the project, and the project's repository may be found at Github[SOURCE].

STMCubeMX[SOURCE] was used for the initialisation of microprocessor peripherals, i.e. the generation of peripheral drivers, and the generation of the project's Makefile. STMCubeMX is a configuration tool published and maintained by ST, the manufacturer of the STM microprocessor family, providing a GUI through which the user may create a pinout, enable interrupts et cetera for their microprocessor. It is freely available, but requires the registration of a user account with ST[SOURCE]. The tool generates .h and .c files for the relevant peripherals based on choices made in the GUI, as well as either a Makefile for use with development tools outside the ST software ecosystem, or the equivalent for use with ST's own STMCubeIDE IDE. This project utilised the Makefile option as it seeks to minimise the use of non-open-source solutions.

VSCode[SOURCE] with ST extension[SOURCE] was the primary IDE for this project. The extension *stm32-for-vscode* was used to build and flash binaries for the MCUs.

PuTTy was used for serial communication with and debugging of the MCUs.

ROS2 Iron

MoveIt

6.1.2 Workflow

MCU binaries:

- Set peripheral configuration parameters in STMCubeMX
- Ensuring that the "overwrite user code" option is unchecked, generate code.
- Edit generated files as necessary, ensuring code is added between USER CODE labels to avoid being overwritten the next time STMCubeMX is used to generate/update peripheral settings.
- Add and edit source files in the TTK4900_drivers folder under the root folder created by STMCubeMX.
- Ensure source files are registered in the Makefile generated by STMCubeMX.
- Build binary file using the ST extension in VSCode.
- Flash binary to MCU using the ST extension and ST-LINK unit embedded in the development kit6.2.

ROS nodes

- Do stuff

6.2 Electronic/Hardware

6.3 Mechanical

- KiCad
- Git
- ST CubeMX
- VSCode (extensions: ST, C/C++)
- Development kit
- Angular measurement tool
- Power source
- Oscilloscope
- MoveIt
- Putty
- Categories: Electr(on)ic, Software, Mechanical
- Fusion 360
- Prusa Slicer

7 Hardware improvements

7.1 Improvement matrix

Table 4 summarises the "Further work" section of the specialisation report[2] and was originally presented there.

Table 4: A summary of further work

Item/affects unit	Hand	Shoulder	Torso	Bogie	Rail	IMU (board)	MCU
USB VBUS voltage divider		X	X				
USB VBUS pin		X	X				X
USB pullup on DM		X	X				
USB pullup transistor		X	X				
IMU 5V and 3.3V lines	X	X				X	
IMU/OPT header design	X	X				X	X
IMU header placement		X					
CAN verification	X	X	X				
CAN/48V connector placement			X				
CAN transciever placement	X						
Nylon bolts	X						
Mount point placement		X	X		X		
Mount point size		X	X				
20 pin connector			X	X			
Motor driver to PWM output	X	X	X				X
Motor driver ADC/PWM							X
Motor driver header silk	X						
Optocoupler 5V	X					X	X
Wrist optocoupler function	X					X	X
Bulk cap/pin header swap	X						
1.27mm jumpers	X						
Switching regulators	X	X	X				
Horisontal voltage regulators			X				
Voltage regulator LED resistor	X	X	X				
Encoder circuits							
Encoder header rotation				X			
TVS array					X		
Motor driver relay control		X	X				
Motor characterisation							
PCB production	X			X	X		

7.2 MCU pinout

The MCU pinout lays the foundation for layout of the PCBs, and was therefore changed first.

- USB VBUS detection was set to pin PA9 in accordance with AN4879 chapter 2.6[1].
-

TODO: get illustration from CubeMX, compare with specialisation report

7.3 Installation

7.4 Verification

8 Software architecture

8.1 Arm onboard

Input: UART/USB Structure: Peer-to-peer vs master/slave? MCUs are equal, but torso deals with external comms. Backbone: CAN bus Module inclusion: Preprocessor statements

8.2 ROS nodes

9 Peripheral configuration

10 Hardware drivers

Point out that these are the TTK4900 drivers, not CubeMX generated files. Not including joint controllers, can executives; these are control system domain. Defining positive and negative direction for the joints, make figures

11 Control system

Main loop, joint controllers, can executives, state machine

12 Calibration

Method of calibration, usage in the state machine.

13 ROS nodes

14 Tests

15 Results

State machine is whack, inconsistent calibration for twist for whatever reason Accelerometers are whack System works well overall when accelerometers are disabled ROS is really powerful, and can probably be expanded readily CAN bus craps out for voltages above 25V

16 Discussion

17 Conclusion

18 Operations

Bibliography

[1] ST AN4879. *AN4879: Introduction to USB hardware and PCB guidelines using STM32 MCUs*. 2023.

[2] Kristian Blom. *From hardware to control algorithms: Retrofitting a legacy robot using open source solutions*. 2023.

[3] Bourns. *CDSC706-0504C - Surface Mount TVS Diode Array*. 2015.

[4] TT Electronics. *Photologic® Slotted Optical Switch OPB960, OPB970, OPB980, OPB990 Series*. 2019.

[5] Escap. *escap 23LT12 graphite/copper commutation systems*. 2024.

[6] U. Ghia, K. N. Ghia and C. T. Shin. 'High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method'. In: *Journal of Computational Physics* 48 (1982), pp. 387–411.

[7] Pittman Metek. *Brush commutated DC motors DC030B Series*. 2024.

[8] Pittman Metek. *Brush commutated DC motors DC040B Series*. 2024.

[9] Pittman Metek. *Brush commutated DC motors DC054B Series*. 2024.

[10] Avago Technologies. *HEDS-9000/9100 Two Channel Optical Incremental Encoder Modules*. 2016.

Appendix

A Hello World Example

```cpp
int main {
    // This is a comment
    std::cout << "Hello World from C++!" << std::endl;
    std::cout << "I am using the default style to print this code in beautiful
    ↪  colors. Since the text is so long I have to include the 'breaklines'
    ↪  option as well" << std::endl;
    return 0;
}
```

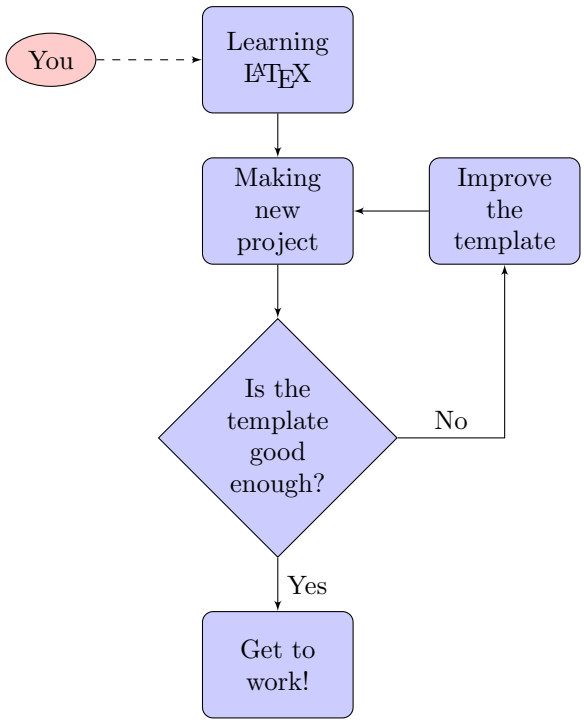
⋮

```python
# This is a comment
print('Hello world from Python!')
print('I am using the "rrt" style to print this code in beautiful colors')
```

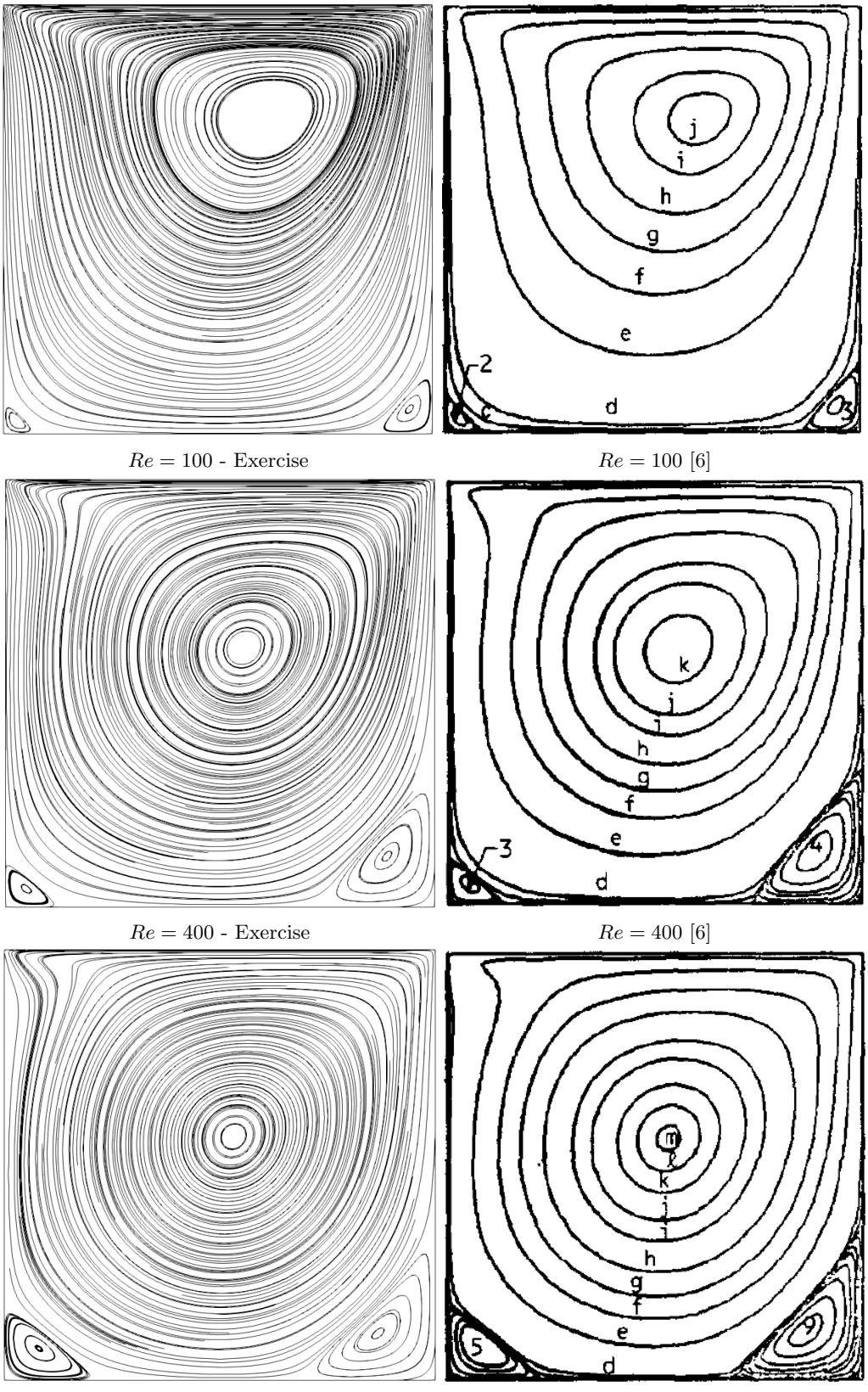
⋮

```matlab
% This is a comment
disp("Hello World from MATLAB!");
disp("I am using the "tango" style to print this code in beautiful colors");
```

B Flow Chart Example



C Sub-figures Example



$Re = 100$ - Exercise $Re = 100$ [6]

$Re = 400$ - Exercise $Re = 400$ [6]

$Re = 1000$ - Exercise $Re = 1000$ [6]

Figure 5: Streamlines for the problem of a lid-driven cavity.