

Mechatronic Project - Electronic Lead Screw

January 2022 - Lukas Schwörer

Preface

This project is part of the masters degree program "System engineering" of Aalen University and is scheduled to be performed during the first two semesters. This report covers the work realized from April 2021 to February 2022.

The practical work and the writing for this project was performed from home.

Dieses Projekt ist Teil des Masterstudiengangs "System Engineering" der Hochschule Aalen und muss während der ersten beiden Semester absolviert werden. Die in diesem Bericht beschriebene praktische Arbeit wurde von April 2021 bis zum Februar 2022 realisiert.

Die praktische Arbeit, wie auch das Schreiben des Berichts wurde Zuhause ausgeführt.

Abstract

This Project is about the development, setup, testing and qualification of an Electronic Leadscrew (ELS). This project was proposed to the university by myself. Its aim is to develop a system to replace the gearbox inside a conventional lathe which will synchronize the rotation of the Leadscrew to the rotation of the spindle. The ELS needs to be able to keep up with the spindle rotation during conventional turning with different feeds and speeds. In addition to this it should be possible to cut precise metric and imperial threads.

The electro-mechanical system of the ELS is build around an encoder to read rotational position of the main spindle and a servo motor to control the position of the Leadscrew. A micro-controller computes the information gathered by the encoder and commands the servo-motor to the correct positions.

To be able to easily change and add Features as well as to predict the behavior of the system the development of the ELS needs to be model based. This model needs to incorporate all aspects of the system including the spindle, the encoder, the micro-controller and the servo motor. As comparison the conventional gearbox should also be modeled.

Kurzfassung

Dieses Projekt beschäftigt sich mit der Entwicklung, dem Aufbau, dem Testen und Qualifizieren einer Elektronischen Leitspindel (ELS). Dieses Projekt wurde der Universität von mir vorgeschlagen. Sein Ziel ist es ein System zu entwickeln, dass das Getriebe in einer konventionellen Drehbank ersetzt und die Rotation der Leitspindel zu der Rotation der Hauptspindel synchronisiert. Die ELS muss fähig sein mit der Rotation der Hauptspindel mitzuhalten während einer konventionellen Drehbearbeitung mit unterschiedlichen Drehzahlen und Vorschüben. Zusätzlich muss es möglich sein präzise metrische und Imperische Gewinde zu drehen.

Das elektro-mechanische System der ELS besteht aus einem Encoder der die Position der Hauptspindel ausliest und einem Servo-Motor, der die Position der Leitspindel kontrolliert. Ein Microcontroller verarbeitet die vom Encoder gesammelten Informationen und bestimmt die korrekte Position des Servo-Motors.

Um ein einfaches Ändern und Entfernen von Features zu ermöglichen und das Verhalten des Systems vorherzusagen, muss die Entwicklung der ELS Modellbasiert durchgeführt werden. Dieses Modell muss alle Komponenten des realen Systems beinhalten, eingeschlossen Spindel, Encoder, Mikrocontroller und Servo-Motor. Zum Vergleich sollte auch ein System mit einem konventionellen Getriebe modelliert werden.

Acknowledgement

At this point I would like to thank the following people who made this project possible:

- **Prof. Dr. Markus Glaser** For supervising and supporting my project.
- **James Clough (Clough42)** For helping me to find hardware matching my specifications and supplying me with the great documentation about his own ELS project.

Table of Contents

| | |
|---|------------|
| Preface | i |
| Abstract | ii |
| Kurzfassung | iii |
| Acknowledgement | iv |
| 1. Introduction | 1 |
| 1.1. Working environment | 1 |
| 1.1.1. University of Aalen | 1 |
| 1.1.2. Home office | 1 |
| 1.2. Project background | 1 |
| 1.2.1. Aim of study | 1 |
| 1.2.2. Limitations | 1 |
| 2. Theoretical Background | 2 |
| 2.1. Numerical Mathematics | 2 |
| 2.1.1. Floating-Point Arithmetic | 2 |
| 2.1.2. Fixed-Point Arithmetic | 2 |
| 2.2. Manufacturing Methods | 2 |
| 2.2.1. Additive Manufacturing | 3 |
| 2.2.2. Subtractive Manufacturing | 3 |
| 3. Hardware and Software | 4 |
| 3.1. Hardware | 4 |
| 3.1.1. Rotary Encoder | 4 |
| 3.1.2. Stepper Motors | 4 |
| 3.1.3. Closed Loop Servos | 4 |
| 3.1.4. Microcontroller | 5 |
| 3.1.5. Raspberry Pi | 5 |
| 3.2. Software | 6 |
| 3.2.1. Matlab and Matlab-Simulink | 6 |
| 3.2.2. Programing Language C | 6 |
| 3.2.3. Programing Language Python | 6 |
| 3.2.4. Code Composer Studio | 7 |
| 3.2.5. Git | 7 |
| 4. Experimental | 8 |
| 4.1. Model in the Loop | 8 |
| 4.1.1. Initial Considerations | 9 |
| 4.1.2. Parameter Identification | 9 |

| | | |
|-----------------|-------------------------------------|-----------|
| 4.1.3. | Simulink Model | 9 |
| 4.1.4. | Code Generation | 9 |
| 4.1.5. | Mechanical Implementation | 9 |
| 4.2. | Software in the Loop | 9 |
| 4.2.1. | Logic | 10 |
| 4.2.2. | Arithmetic | 10 |
| 4.3. | Hardware in the Loop | 10 |
| 4.3.1. | Component Test | 10 |
| 4.3.2. | System Test | 10 |
| 4.3.3. | Integration Test | 11 |
| 5. | Results and Discussion | 12 |
| 5.0.1. | Arythmatic | 12 |
| 5.0.2. | Threading | 12 |
| 5.0.3. | Turning | 12 |
| 6. | System Validation | 13 |
| 7. | Outlook | 14 |
| 7.1. | Features | 14 |
| 8. | List of Figures | 15 |
| 9. | List of Tables | 16 |
| 10. | References | 17 |
| Appendix | | I |

1. Introduction

In this chapter the working environment, project background and all involved companies will be introduced. Furthermore the aim and limitations of the project will be described.

1.1. Working environment

1.1.1. University of Aalen

1.1.2. Home office

Metall workshop

1.2. Project background

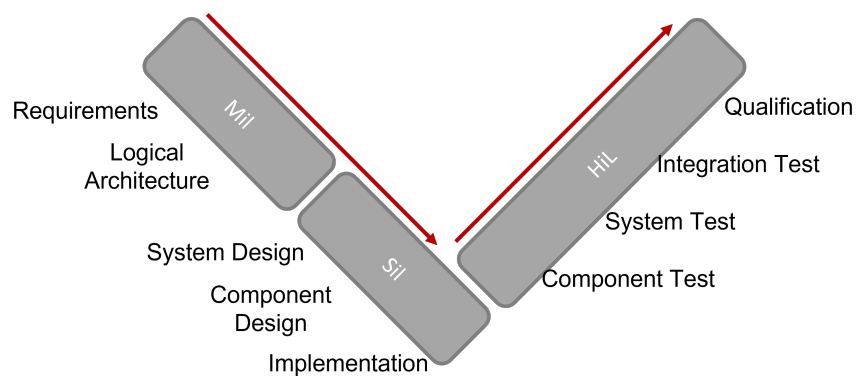


Figure 1.1.: V-Model

1.2.1. Aim of study

1.2.2. Limitations

2. Theoretical Background

This chapter will discuss the needed theoretical background knowledge that was part of the decision making during the development process.

2.1. Numerical Mathematics

The science of the numerical mathematics is a branch of mathematics that concentrates on the research of solving continuous problems with discrete systems such as a computer. One of the main problems in numerical mathematics is the representation of fractional numbers. This is because computers can only represent a finite subset decimal, due to memory and processing limitations. This can lead to rounding errors that can have a significant impact on the precision of any calculation with fractional numbers. Further, calculations with numbers that have a large amount of decimal places, require more calculation steps by the processor and therefore slow down the computation.[1]

2.1.1. Floating-Point Arithmetic

One way to represent fractional number is the floating point number system. With this system are stored in memory in two parts: the significant and its base. The significant represents the number as an integer, without the decimal point. The base (10^x) describes the position of the decimal point in the original number.[1]

2.1.2. Fixed-Point Arithmetic

The fixed point representation of numbers on a numerical system, defines the fractional number with a fixed amount of rational numbers. Further, the amount of significant numbers and decimal places is fixed. The intention behind this number representation is to simplify and therefore speed up numerical calculations.[1]

2.2. Manufacturing Methods

Nowadays engineers have a broad spectrum of different manufacturing methods available when it comes to producing a part. The decision which to choose should incorporate component specific properties such as the desired amount to be produced, the general shape and material of the part as well as the environment the part will be used in.

2.2.1. Additive Manufacturing

Additive manufacturing (AM) is a process, where material is built up in order to create the desired shape. AM has first appeared in 1981, with the first commercial available machine in 1988. Additive manufacturing incorporates a wide variety of different manufacturing methods. The most well known Additive manufacturing method today probably is Fused Deposition Modeling (FDM), also known as 3D-Printing. With FDM, a thin thermoplastic filament is molten and then squeezed through a nozzle. The molten plastic is then laid down in layers, in order to create a 3D shape. The core advantages of FDM are its speed and flexibility. This makes prototyping and small production batches very cost effective and easy. Further, because 3D-Printing is a layer by layer process, almost any shape can be created, this is especially important for shapes with undercuts and included geometry's.[2]

2.2.2. Subtractive Manufacturing

Subtractive manufacturing is a process, where material is removed from a stock material in order to create a 2D or 3D shape. Two of the most common subtractive manufacturing processes are turning and milling.

Turning is a process where the stock material is rotated and is cut with a fixed tool.

3. Hardware and Software

This chapter will give an in-depth look into the hardware and software used to complete the project. This knowledge is necessary to understand the design decisions made.

3.1. Hardware

3.1.1. Rotary Encoder

A rotary encoder is an electromechanical transducer to convert a rotary movement into a electrical signal. One of the most commonly used rotary encoders is the optical rotary encoder. Its working principle is based on a Light emitting diode and a photo sensitive device, such as a photo-transistor or photo-resistor. The diode and the photo-sensitive device are located on opposite sides of a light blocking disk. The light blocking disk is perforated with small slits in a rotations symmetric and regular pattern. As soon as the disc is rotated between the diode and the photo-sensitive device the light is transmitted and blocked in an alternating pattern. This creates a squarewave on the output of the photo-sensitive device which frequency is dependant on the rotationary speed of the disk.

3.1.2. Stepper Motors

A stepper motor is an electrical, synchronous motor. Its rotor follows the magnetic field created in the stator exactly. In contrast to commonly known DC or AC motors, a stepper motor has a minimum angle that it needs to be rotated. This has the big advantage, that the position is always a multiple of the step angle. Therefor a sensor is not necessary to rotate the motor by a very precise amount. A stepper motor is a cheap alternative to closed loop controlled motors, where an additional sensor has to determine the positional error of the motor in order to move the motor precisely.

The disadvantage of stepper motors usually is their low rpm limit. At a certain speed, the rotor of the motor is no longer able to follow the magnetic field created by the stator. This can also happen when the motor is under high load and is very hard to detect. Therefore, stepper motors are often combined with rotational sensors in order to compensate the positional error, when the motor comes out of synchronization.[3]

3.1.3. Closed Loop Servos

Closed Loop Servos is an electromechanical actuator with an continuous feedback loop between its output (speed, torque, position, etc..) and input. Servos require additional

electronics to compute the difference between its desired output of the motor and its current output. These electronics are usually referred to as "controller" or "driver". Servos are used in a wide variety of mechatronic applications when high precision, reliability or dynamics are required.[4]

3.1.4. Microcontroller

A Mikrocontroller is an integrated circuit designed to serve a specific task in a system. Mikrocontroller consist of a processing unit, memory and In- and Outputs. Within the system Mikrocontrollers can be used for a wide variety of tasks such as communication, data acquisition or motor control. For mechatronic tasks, microcontrollers are commonly used in an so called embedded system. An embedded system is usually designed around a microcontroller and a specific task within a project.[5]

TI LaunchXL F280049C

The Texas Instruments Launchpad F280049C is a prototyping board designed around their Picolo F280049C Real-Time microcontroller as an embedded system. Its design specificalay targets highly dynamic controll tasks while maintaining a low unit cost. The Picolo F280049C is a 32bit floating point microcontroller with 256KB Flash memory, 100KB RAM and a operation frequency of 100MHz. Further, the microcontroller has hardware support for up to two rotary encoder, a wide variety of communication protocols and general purpose I/O. In addition, the LaunchPad developement board build around the microcontroller offers a debugging probe. This probe can be used to flash the microcontroller with maschinecode or observe and direct the operation of the microcontroller. This way it is for excample possible to stop the execution of the program on the microcontroller and access the value of every bit in every register of the microcontroller.[6]

Logic Level Shifter

A logic level shifter is an electronic circuit designed to convert one logic voltage level into another one, while maintaining the signal integrity. In modern mechatronic systems it is often necessary that different electronic systems can communicate with each other. In some cases however, it is not possible that all components work with the same voltage level. This makes it inevitable to use a logic level shifting circuit.

3.1.5. Raspberry Pi

A Raspberry Pi is a Single board computer that features a full graphic operating system based on linux. Further it offers wireless conectivety as well as general purpose I/O's. The raspberry pi is mostly open source and very cost effective. Because it has a very wide customer base, the Raspberry Pi has wide variety of hardware and software add-ons available.[7]

Touchscreen

A touchscreen is a Human Maschine Interface (HMI) that can display information and react to touch inputs. The commonly used touchscreen combines a regular LED screen with touch sensitive layer. Highly sensitive touchscreens that can detect multiple touch inputs at once, usually work on a capacitive principle.

3.2. Software

3.2.1. Matlab and Matlab-Simulink

MATrix LABoratory (MATLAB) is an Integrated Development Environment (IDE) and programming language. MATLAB was developed in the need of a numerical algebraic system at the university of New Mexico. On this base, the company MathWorks (see B) was created. Since 1984 MathWorks is further developing MATLAB and additional software for numerical algebraic computing. To support different hardware packages MathWorks offers so called toolboxes. These toolboxes contain functions and scripts for communication, data acquisition, motion control etc. MATLAB is mainly used in technical development and research.[8]

3.2.2. Programing Language C

C is a high level programing language. It is most commonly used to write applications for embedded systems. It is chosen over low level programing languages such as Assembly because of the increasing complexity of programs for embedded systems. C is a compiled programming language. This means that a program needs to be translated (compiled) to machine code before it can be executed. This translation is done with a program called "compiler". Compared to lower level languages such as Assembly, the code reusability of C has proven to be a step forward in the development of embedded systems.[9]

3.2.3. Programing Language Python

Python is a high level object oriented programming language. In contrast to C, Python is an interpreted language. This means, an Interpreter translates source code into machine commands in real time, while the program is executed. Python features a good combination of capability and ease of uses. This makes the language easy to learn for beginners while remaining very powerful. Python further features a huge variety of proprietary and community build support packages and is based on the one of the most used programming languages used today.[10]

Kivy

Kivy is an open source Python library for the rapid development of grafical user interfaces and apps. It is script based and its scripting language can bei directly implemented in Python. Further, kivy supports use with touch screens.[11]

3.2.4. Code Composer Studio

Code Composer Studio is an Integrated Development Environment produced by Texas Instruments. It presents an interface to the Hardware produced by Texas Instruments and offers optimizing compilers for C and C++. Further, Code Composer Studio features support packages for the development and deployment of software to Texas Instruments Hardware such as the TI LaunchXL F280049C. These support packages in combination with a hardware debugging probe, enable the programmer To interact with the processor in real time.[12]

3.2.5. Git

Git is an software tool for source code management and versioning, as well as for parallel development. Git was developed by Linus Torvald in 2005. Git was developed in the need of source code management software for the development of the operating system Linux. Git allows development in so-called branches. These are independent copies of an already existing and probably used software. This ensures that modifications or enhancements are not affecting already used software. If a feature is finished, it can be merged back into the higher-level branch. Merging is the process of bringing two files, directories or branches together. Different developers, working on different features of the same project is called parallel development. Further Git is a tool for software versioning. Software versioning is a tracking of changes in a project. In case of an mistake the project can be recovered to every tracked point. Git was very heavily used for software development after and during our testing.

4. Experimental

4.1. Model in the Loop

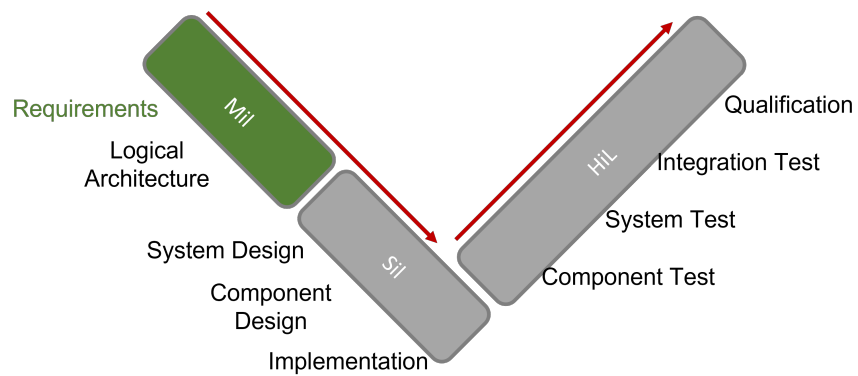


Figure 4.1.: V Model Requirements Stage

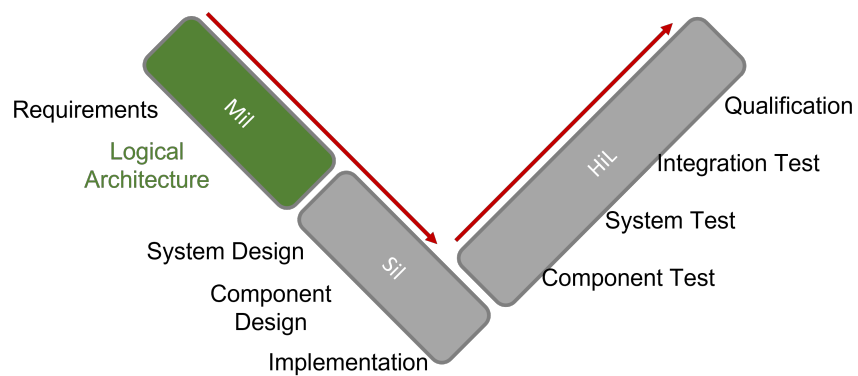


Figure 4.2.: V Model Logical Architecture Stage

4.1.1. Initial Considerations

4.1.2. Parameter Identification

4.1.3. Simulink Model

Encoder and Spindel

Microcontroler

Stepper Motor

Gearbox

4.1.4. Code Generation

4.1.5. Mechanical Implementation

4.2. Software in the Loop

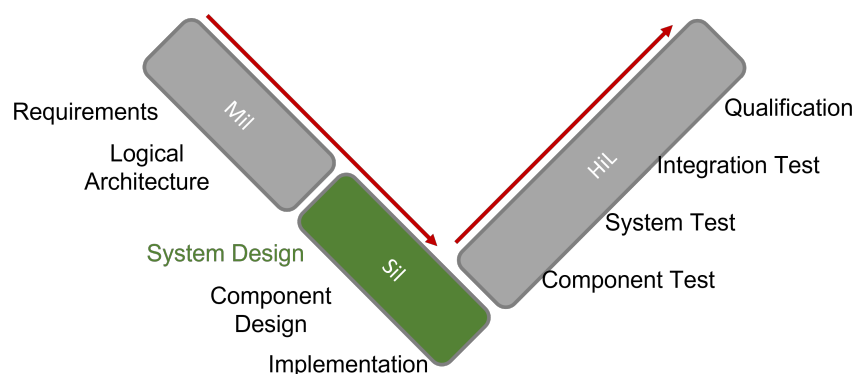


Figure 4.3.: V Model System Design Stage

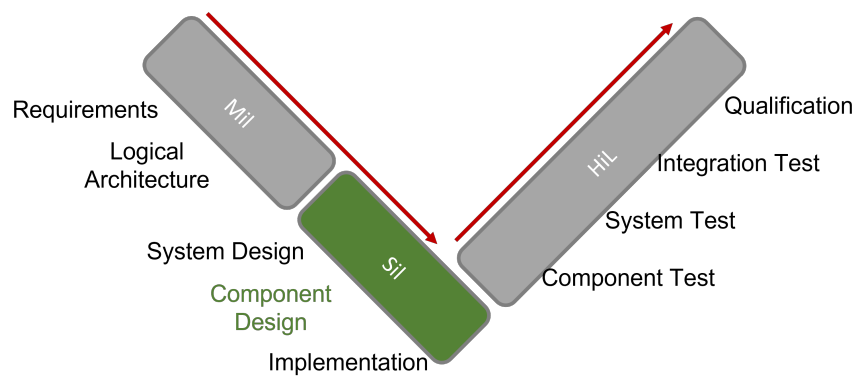


Figure 4.4.: V Model Component Design Stage

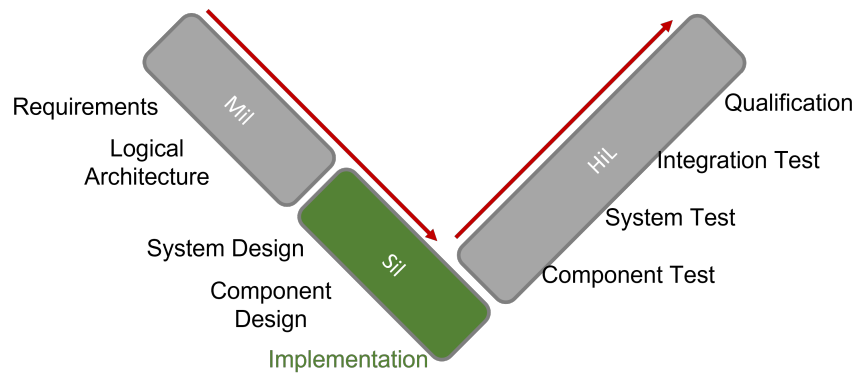


Figure 4.5.: V Model Implementation Stage

4.2.1. Logic

4.2.2. Arithmetic

4.3. Hardware in the Loop

4.3.1. Component Test

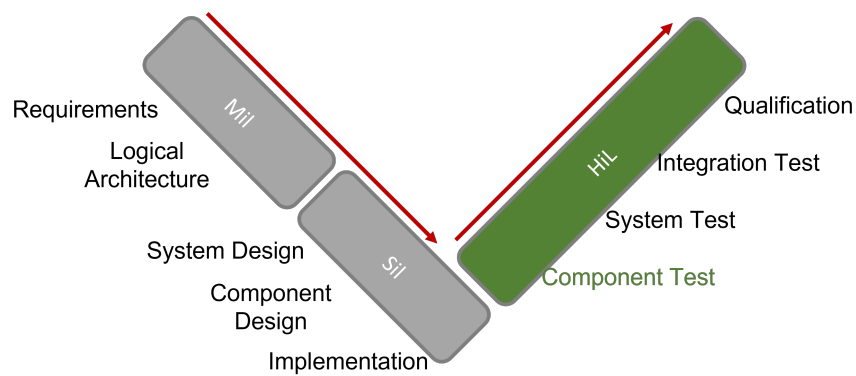


Figure 4.6.: V Model Component Test Stage

4.3.2. System Test

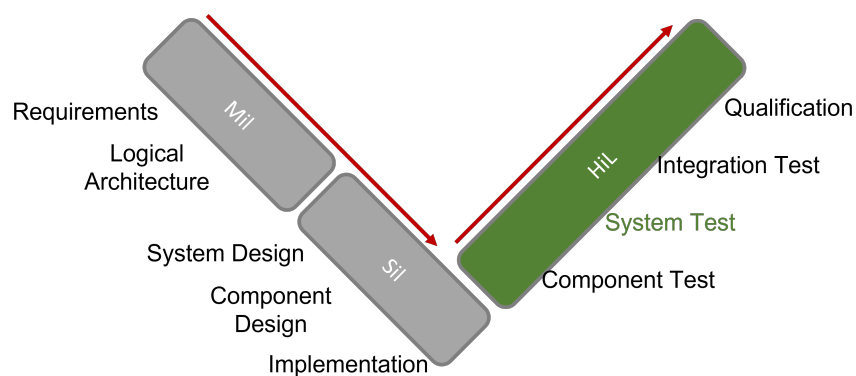


Figure 4.7.: V Model System Test Stage

4.3.3. Integration Test

Turning

Threading

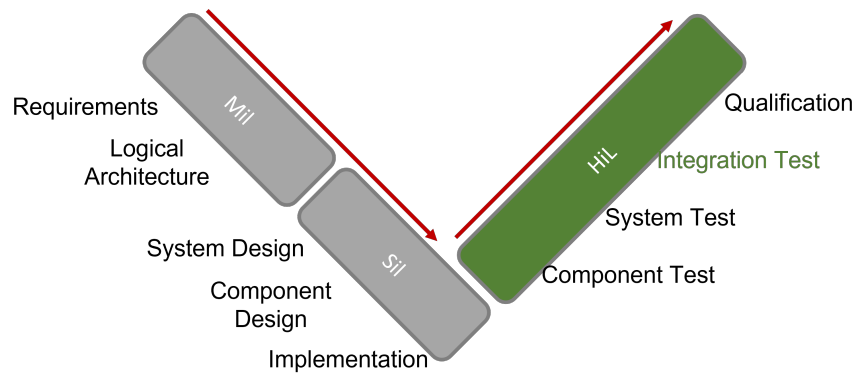


Figure 4.8.: V Model Integration Test Stage

5. Results and Discussion

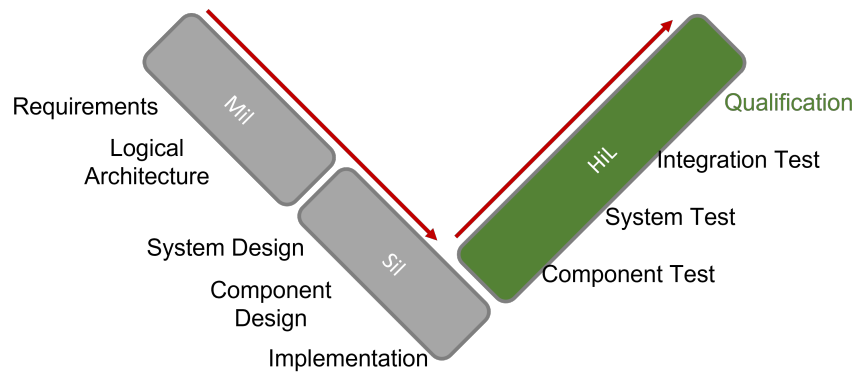


Figure 5.1.: V Model Qualification Stage

5.0.1. Arythmatic

5.0.2. Threading

5.0.3. Turning

6. System Validation

7. Outlook

7.1. Features

8. List of Figures

| | |
|---|----|
| 1.1. V-Model Complete | 1 |
| 4.1. V Model Requirements | 8 |
| 4.2. V Model Logical Architecture | 8 |
| 4.3. V Model System Design | 9 |
| 4.4. V Model Component Design | 9 |
| 4.5. V Model Implementation | 10 |
| 4.6. V Model Component Test | 10 |
| 4.7. V Model System Test | 10 |
| 4.8. V Model Integration Test | 11 |
| 5.1. V Model Qualification | 12 |

9. List of Tables

10. References

- [1] Alfio Quarteroni. *Numerical Mathematics*. New York, NY: Springer, 2007. ISBN: 9781475773941.
- [2] Fraunhofer-Institut IGCV. *Additive Manufacturing*. https://www.igcv.fraunhofer.de/de/forschung/kompetenzen/additive_fertigung_am.html.
- [3] Faulhaber Group. *Schrittmotoren für anspruchsvolle Positionieraufgaben*. <https://www.faulhaber.com/de/produkte/schrittmotoren/>. Jan. 2022.
- [4] Suk-Hwan Suh. *Theory and Design of CNC Systems*. Springer-Verlag GmbH, Aug. 2008. 456 pp. ISBN: 978-1-84800-336-1. URL: https://www.ebook.de/de/product/12469020/dae_hyuk_chung_seong_kyoon_kang_ian_stroud_suk_hwan_suh_theory_and_design_of_cnc_systems.html.
- [5] Cem Ünsalan, Hüseyin Deniz Gürhan, and Mehmet Erkin Yücel. *Embedded System Design with ARM Cortex-M Microcontrollers*. Springer International Publishing, Jan. 2022. 569 pp. ISBN: 978-3-030-88439-0. URL: https://www.ebook.de/de/product/42038562/cem_uensalan_hueseyin_deniz_guerhan_mehmet_erkin_yuecel_embedded_system_design_with_arm_cortex_m_microcontrollers.html.
- [6] Texas Instruments. *LAUNCHXL-F280049C*. <https://www.ti.com/tool/LAUNCHXL-F280049C>.
- [7] Raspberry Pi Foundation. *What is a Raspberry Pi?* <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>.
- [8] Mathworks. *Matlab - Overview*. <https://de.mathworks.com/products/matlab.html>. Accessed: 07.08.2021.
- [9] ISO and IEC. *ISO IEC JTC1 SC22 WG14 N1169 Programming languages - C*. Standart. ISO, Jan. 2022.
- [10] Python Software Foundation. *What is Python*. <https://docs.python.org/3/faq/general.html#what-is-python>. Jan. 2022.
- [11] Kivy.org. <https://kivy.org/doc/stable/gettingstarted/intro.html>. Feb. 2022.
- [12] Texas Instruments. *Code Composer Studio integrated development environment (IDE)*. <https://www.ti.com/tool/CCSTUDIO>. Jan. 2022.

Appendix

| | |
|---|------------|
| A. Additional Topics | II |
| A.1. Pin Out | II |
| A.2. External Reset | II |
| B. List of Companies | III |
| C. Network setup and configuration | V |
| D. Organisation Chart | VI |
| E. Source Code | VII |
| E.1. main.c | VIII |
| E.2. Configuration.h | XV |
| E.3. Configuration.c | XVII |
| E.4. MainUI.py | XXIV |

A. Additional Topics

A.1. Pin Out

A.2. External Reset

B. List of Companies



Company: Volvo Cars

Website: <https://www.volvocars.com/se>



Company: The MathWorks, Inc.

Website: <https://www.mathworks.com/>



Company: National Instruments

Website: <https://www.ni.com/>

TAMRON

Company: Tamron

Website: <https://www.tamron.com/>

The logo for MikroTik, featuring the word "Mikro" in a stylized, italicized font with a small arc above the 'i', followed by "Tik" in a bold, italicized font.

Company: MikroTik

Website: <https://mikrotik.com/>

C. Network setup and configuration

D. Organisation Chart

E. Source Code

E.1. main.c

```
1  //
2  // Included Files
3  //
4  #include "F28x_Project.h"
5  #include "Configuration.h"
6
7  #include "..\Matlab\RealTimeMachine_ert_rtw\RealTimeMachine.h"
8  #include "..\Matlab\RealTimeMachine_ert_rtw\rtwtypes.h"
9  #include "..\Matlab\RealTimeMachine_ert_rtw\
    zero_crossing_types.h"
10 #include "..\Matlab\RealTimeMachine_ert_rtw\
    RealTimeMachine_data.c"
11
12 #include "..\Matlab\StepperRTM_ert_rtw\StepperRTM.h"
13 #include "..\Matlab\StepperRTM_ert_rtw\rtwtypes.h"
14 #include "..\Matlab\StepperRTM_ert_rtw\zero_crossing_types.h"
15
16 #define FLASH
17
18 //
19 // Global Variables Inputs
20 //
21 static uint32_T arg_SpindelPos = 0U;
22 volatile real32_T arg_CountFactor = 0;
23 volatile uint16_T var_StepBacklog;
24
25 //
26 // Global Variables Outputs
27 //
28 static uint16_T arg_StepBit;
29 static uint16_T arg_Dir;
30 static uint16_T arg_DesSteps;
31
32 //
33 // Global Variables Statemachine Clock
34 //
35 static uint16_T System_Trigger[2] = { 0U, 0U };
36 static boolean_T System_Takt = 0;
37 static uint16_T Stepper_Trigger[2] = { 0U, 0U };
```

```

38 static boolean_T Stepper_Takt = 0;
39
40 //
41 // Global Variables Helpers
42 //
43 volatile uint32_T current = 0;
44 volatile uint32_T count = 0;
45 volatile uint32_T previous = 0;
46 volatile uint16_T RPM;
47
48 volatile int msg[] = {0, 0, 0, 0, 0};
49 volatile int i = 0;
50 volatile float feed = 0.0;
51 volatile int Mode = 1;
52 volatile int TransferComplete = 0;
53
54 void main(void)
55 {
56
57     #ifdef FLASH
58         // Copy time critical code and Flash setup code to RAM
59         // The RamfuncsLoadStart, RamfuncsLoadEnd, and
60         // RamfuncsRunStart
61         // symbols are created by the linker. Refer to the
62         // linker files.
63         memcpy(&RamfuncsRunStart, &RamfuncsLoadStart, (size_t)
64             &RamfuncsLoadSize);
65
66         // Initialize the flash instruction fetch pipeline
67         // This configures the MCU to pre-fetch instructions
68         // from flash.
69         InitFlash();
70     #endif
71
72     //
73     // Initialize Autocode
74     //
75     StepperRTM_initialize();
76     RealTimeMachine_initialize();
77     //

```

```

75     // Initialize device clock and peripherals
76     //
77     InitSysCtrl();
78
79     //
80     // Initialize GPIO, Timer and EQEP
81     //
82     InitGpio();
83     setupGPIO();
84     setupTimer();
85     setupEQEP();
86
87
88
89
90     //
91     // Initialize UART
92     //
93     initSCIAFIFO();
94     initSCIAEchoback();
95
96
97
98
99
100    while(1)
101    {
102        //
103        // Send RPM via UART
104        //
105        if(EQep1Regs.QFLG.bit.UTO==1)
106        {
107
108            Uint32 current = EQep1Regs.QPOSLAT;
109            Uint32 count = (current > previous) ? current -
                previous : previous - current;
110
111            // deal with over/underflow
112            if( count > _ENCODER_MAX_COUNT/2 )
113            {
114                count = _ENCODER_MAX_COUNT - count; // just

```

```

115         subtract from max value
116     }
117
118     RPM = count * 60 * RPMSampleTime / EncoderRes;
119
120     int highbyte = RPM >> 8;
121     int lowbyte = RPM & 0x00ff;
122
123     previous = current;
124     transmitSCIACChar(lowbyte);           // Send RPM
125         out via UART
126     transmitSCIACChar(highbyte);         // Send RPM
127         out via UART
128     EQep1Regs.QCLR.bit.UTO=1;           // Clear interrupt
129         flag
130 }
131
132 //
133 // Check for new Messages from the Raspberry Pi
134 //
135 if(SciaRegs.SCIFFRX.bit.RXFFST != 0)
136 {
137     msg[i] = SciaRegs.SCIRXBUF.bit.SAR;
138
139     if(i == 4)
140     {
141         i = 0;
142         TransferComplete = 1;
143         Mode = msg[1];
144         feed = msg[2] + (msg[3] * 0.01);
145     }
146     else
147     {
148         i++;
149     }
150 }
151
152 //
153 // Calculate Step-factor based on information received
154 // via UART
155 //

```

```

151
152     if(TransferComplete && (msg[0] == 0xff) && (msg[4] ==
153         0xff))
154     {
155         TransferComplete = 0;
156         // Normal Feed and metric thread cutting
157         if(Mode == 1 || Mode == 2)
158         {
159             arg_CountFactor = ((Steps * MotorTransmission *
160                 EncoderTransmission * feed)/(EncoderRes *
161                 LeadscrewSlope));
162         }
163
164         // Imperial thread cutting
165         else if(Mode == 3)
166         {
167             arg_CountFactor = ((Steps * MotorTransmission *
168                 EncoderTransmission * OneInch)/(EncoderRes
169                 * LeadscrewSlope * feed ));
170         }
171     }
172 }
173
174 //
175 // cpuTimer0ISR - CPU Timer0 ISR
176 //
177 _interrupt void cpuTimer0ISR(void)
178 {
179
180     Stepper_Takt = !Stepper_Takt; // Toggle System Clock
181     Stepper_Trigger[0] = (uint16_T)Stepper_Takt;
182
183     if(Stepper_Takt == 0)
184     {
185         Stepper_Trigger[1] = 1; //Power on Reset
186     }
187
188     StepperRTM_step(var_StepBacklog , (uint16_t*)&
189         Stepper_Trigger , &arg_StepBit);
190
191

```

```

186 //
187 // Stepper Clock for Debugging
188 //
189
190 GpioDataRegs.GPASET.bit.GPIO6 = arg_StepBit;
191 GpioDataRegs.GPADAT.bit.GPIO6 = arg_StepBit;
192
193 //
194 // Acknowledge this interrupt to receive more
195 //
196 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
197 }
198
199
200
201 //
202 // cpuTimer0ISR - CPU Timer2 ISR
203 //
204 __interrupt void cpuTimer2ISR(void)
205 {
206     System_Takt = !System_Takt; // Toggle System Clock
207     System_Trigger[0] = (uint16_T)System_Takt;
208
209     if(System_Takt == 0)
210     {
211         System_Trigger[1] = 1; // Power on Reset
212     }
213
214     arg_SpindelPos = EQep1Regs.QPOSCNT;
215
216     RealTimeMachine_step(arg_SpindelPos, arg_CountFactor, (
217         uint16_t*)&System_Trigger,
218         &arg_DesSteps, &arg_Dir);
219
220     var_StepBacklog = var_StepBacklog + arg_DesSteps;
221
222     GpioDataRegs.GPBSET.bit.GPIO39 = !arg_Dir;
223     GpioDataRegs.GPBDAT.bit.GPIO39 = !arg_Dir;
224
225     //
226     // Acknowledge this interrupt to receive more

```

```
226 //  
227 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;  
228 }
```

E.2. Configuration.h

```
1  /*
2   * Configuration.h
3   *
4   * Created on: 26 Oct 2021
5   * Author: Lukas Schwoerer
6   */
7  #include "F28x_Project.h"
8
9  //
10 // Predefine Functions
11 //
12
13 void setupGPIO(void);
14 void setupTimer(void);
15 void setupEQEP(void);
16
17 void initSCIAEchoback(void);
18 void transmitSCIACChar(int msg);
19 void initSCIAFIFO(void);
20
21 __interrupt void cpuTimer0ISR(void);
22 __interrupt void cpuTimer2ISR(void);
23
24 #ifndef CONFIGURATION_H
25 #define CONFIGURATION_H
26
27 //
28 // Statemachine cycle times
29 //
30 #define Stepper_Clock 5
31 #define System_Clock 100
32
33 //
34 // Refreshrate RPM (DO NOT EDIT)
35 //
36 #define RefreshRate 100
37
38 //
39 // Hardware constants
```

```
40 //
41 #define _ENCODER_MAX_COUNT      0x00ffffff
42 #define MotorTransmission      3.2
43 #define EncoderTransmission    1
44 #define Steps                  2000
45 #define EncoderRes             4096
46 #define LeadscrewSlope        1.5
47 #define OneInch                25.4
48 #define RPMSampleTime          5           //Sample Rate RPM
                                         in Hz
49 #endif /* CONFIGURATION_H */
```

E.3. Configuration.c

```
1  /*
2   * Configuration.c
3   *
4   * Created on: 26 Oct 2021
5   * Author: Lukas Schwoerer
6   */
7
8  #include "Configuration.h"
9  #include "F28x_Project.h"
10
11
12 void setupTimer(void)
13 {
14
15     // Disable CPU interrupts
16     //
17     DINT;
18
19     //
20     // Initialize the PIE control registers to their default
21     // state.
22     // The default state is all PIE interrupts disabled and
23     // flags
24     // are cleared.
25     //
26     InitPieCtrl();
27
28     //
29     // Disable CPU interrupts and clear all CPU interrupt
30     // flags
31     //
32     IER = 0x0000;
33     IFR = 0x0000;
34
35     //
36     // Initialize the PIE vector table with pointers to the
37     // shell Interrupt
38     // Service Routines (ISR)
39     //
40     //
```

```

36     InitPieVectTable();
37
38     //
39     // Map ISR functions
40     //
41     EALLOW;
42     PieVectTable.TIMER0_INT = &cpuTimer0ISR;
43     PieVectTable.TIMER2_INT = &cpuTimer2ISR;
44     EDIS;
45
46     //
47     // Initialize the Device Peripheral. For this example,
48     // only initialize the
49     // Cpu Timers.
50     //
51     InitCpuTimers();
52
53     //
54     // Configure CPU-Timer 0 and 2
55     // 100MHz CPU Freq, Clock in uSeconds
56     //
57     ConfigCpuTimer(&CpuTimer0, 100, Stepper_Clock);
58     ConfigCpuTimer(&CpuTimer2, 100, System_Clock);
59
60     //
61     // To ensure precise timing, use write-only instructions
62     // to write to the
63     // entire register. Therefore, if any of the configuration
64     // bits are changed
65     // in ConfigCpuTimer and InitCpuTimers, the below settings
66     // must also be
67     // be updated.
68     //
69     CpuTimer0Regs.TCR.all = 0x4000;
70     CpuTimer2Regs.TCR.all = 0x4000;
71
72     //
73     // Enable CPU int1 which is connected to CPU-Timer 0, CPU
74     // int13
75     // which is connected to CPU-Timer 1, and CPU int 14,
76     // which is connected

```

```

71 // to CPU-Timer 2
72 //
73 IER |= M_INT1;
74 IER |= M_INT14;
75
76 //
77 // Enable TINT0 in the PIE: Group 1 interrupt 7
78 //
79 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
80
81 //
82 // Enable global Interrupts and higher priority real-time
    debug events
83 //
84 EINT;
85 ERTM;
86
87 }
88
89 void setupGPIO(void)
90 {
91     EALLOW;
92     //
93     // Setup Port A
94     //
95     GpioCtrlRegs.GPAPUD.bit.GPIO6 = 0; // Enable pull-up
        on GPIO6 (DirPin)
96     GpioCtrlRegs.GPAQSEL1.bit.GPIO6 = 0; // Sync to
        SYSCLKOUT GPIO6 (DirPin)
97     GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0; // Configure GPIO6
        as GPIO
98     GpioCtrlRegs.GPAGMUX1.bit.GPIO6 = 0;
99     GpioDataRegs.GPASET.bit.GPIO6 = 0; // Configure GPIO6
        as Output
100    GpioCtrlRegs.GPADIR.bit.GPIO6 = 1;
101
102    GpioCtrlRegs.GPAPUD.bit.GPIO23 = 0; // Enable pull-up
        on GPIO23 (Step Pin)
103    GpioCtrlRegs.GPAQSEL2.bit.GPIO23 = 0; // Sync to
        SYSCLKOUT GPIO23 (Step Pin)
104    GpioCtrlRegs.GPAMUX2.bit.GPIO23 = 0; // Configure

```

```

105         GPIO23 as GPIO
106         GpioCtrlRegs.GPAGMUX2.bit.GPIO23 = 0;
107         GpioDataRegs.GPASET.bit.GPIO23 = 0;           // Configure
108         GPIO23 as Output
109         GpioCtrlRegs.GPADIR.bit.GPIO23 = 1;
110
111         //
112         // Setup Port B for EQEP1
113         //
114         GpioCtrlRegs.GPBPUD.bit.GPIO35 = 0;           // Enable pull-up
115         on GPIO35 (EQEP1A)
116         GpioCtrlRegs.GPBPUD.bit.GPIO37 = 0;           // Enable pull-up
117         on GPIO37 (EQEP1B)
118         GpioCtrlRegs.GPBPUD.bit.GPIO59 = 0;           // Enable pull-up
119         on GPIO59 (EQEP1I)
120
121         GpioCtrlRegs.GPBQSEL1.bit.GPIO35 = 0;         // Sync to
122         SYSCLKOUT GPIO35 (EQEP1A)
123         GpioCtrlRegs.GPBQSEL1.bit.GPIO37 = 0;         // Sync to
124         SYSCLKOUT GPIO37 (EQEP1B)
125         GpioCtrlRegs.GPBQSEL2.bit.GPIO59 = 0;         // Sync to
126         SYSCLKOUT GPIO59 (EQEP1I)
127
128         GpioCtrlRegs.GPBMUX1.bit.GPIO35 = 1;          // Configure
129         GPIO35 as EQEP1A
130         GpioCtrlRegs.GPBGMUX1.bit.GPIO35 = 2;
131         GpioCtrlRegs.GPBMUX1.bit.GPIO37 = 1;          // Configure
132         GPIO37 as EQEP1B
133         GpioCtrlRegs.GPBGMUX1.bit.GPIO37 = 2;
134         GpioCtrlRegs.GPBMUX2.bit.GPIO59 = 3;          // Configure
135         GPIO59 as EQEP1I
136         GpioCtrlRegs.GPBGMUX2.bit.GPIO59 = 2;
137
138         GpioCtrlRegs.GPBPUD.bit.GPIO39 = 0;           // Enable pull-up
139         on GPIO39 (EnablePin)
140         GpioCtrlRegs.GPBQSEL1.bit.GPIO39 = 0;         // Sync to
141         SYSCLKOUT GPIO39 (EnablePin)
142         GpioCtrlRegs.GPBMUX1.bit.GPIO39 = 0;          // Configure
143         GPIO39 as GPIO
144         GpioCtrlRegs.GPBGMUX1.bit.GPIO39 = 0;
145         GpioDataRegs.GPBSET.bit.GPIO39 = 0;           // Configure

```

```

132         GPIO39 as Output
133         GpioCtrlRegs.GPBDIR.bit.GPIO39 = 1;
134
135         EDIS;
136     }
137
138     void setupEQEP(void)
139     {
140
141         EQep1Regs.QDECCTL.bit.QSRC = 0;           // QEP quadrature
142             count mode
143         EQep1Regs.QDECCTL.bit.IGATE = 1;          // gate the index
144             pin
145         EQep1Regs.QDECCTL.bit.QAP = 1;            // invert A input
146         EQep1Regs.QDECCTL.bit.QBP = 1;            // invert B input
147         EQep1Regs.QDECCTL.bit.QIP = 1;            // invert index
148             input
149         EQep1Regs.QEPCTL.bit.FREE_SOFT = 2;        // unaffected by
150             emulation suspend
151         EQep1Regs.QEPCTL.bit.PCRM = 1;             // position count
152             reset on maximum position
153         EQep1Regs.QPOSMAX = 0x00ffffff;
154
155         EQep1Regs.QUPRD = 100000000/RPMsSampleTime; // Unit Timer
156             latch at RPM_CALC_RATE_HZ Hz
157         EQep1Regs.QEPCTL.bit.UTE=1;                // Unit Timeout
158             Enable
159         EQep1Regs.QEPCTL.bit.QCLM=1;               // Latch on unit
160             time out
161         EQep1Regs.QEPCTL.bit.QPEN=1;               // QEP enable
162     }
163
164     void initSCIAFIFO(void)
165     {
166         GPIO_SetupPinMux(28, GPIO_MUX_CPU1, 1);
167         GPIO_SetupPinOptions(28, GPIO_INPUT, GPIO_PUSH_PULL);
168         GPIO_SetupPinMux(29, GPIO_MUX_CPU1, 1);
169         GPIO_SetupPinOptions(29, GPIO_OUTPUT, GPIO_ASYNC);
170
171         SciaRegs.SCIFFTX.all = 0xE040;

```

```

164     SciaRegs.SCIFFRX.all = 0x2044;
165     SciaRegs.SCIFFCT.all = 0x0;
166 }
167
168 void initSCIAEchoback(void)
169 {
170     //
171     // Note: Clocks were turned on to the SCIA peripheral
172     // in the InitSysCtrl() function
173     //
174     SciaRegs.SCICCR.all = 0x0007;           // 1 stop bit, No
175                                             // No parity, 8
176                                             // char bits,
177                                             // async mode,
178                                             // idle-line
179                                             // protocol
180     SciaRegs.SCICTL1.all = 0x0003;         // enable TX, RX,
181                                             // Disable RX ERR,
182                                             // SLEEP, TXWAKE
183
184     SciaRegs.SCICTL2.all = 0x0003;
185     SciaRegs.SCICTL2.bit.TXINTENA = 1;
186     SciaRegs.SCICTL2.bit.RXBKINTENA = 1;
187
188     //
189     // SCIA at 9600 baud
190     // @LSPCLK = 25 MHz (100 MHz SYSCLK) HBAUD = 0x01 and
191     // LBAUD = 0x44.
192     //
193     SciaRegs.SCIHBAUD.all = 0x0001;
194     SciaRegs.SCILBAUD.all = 0x0044;
195
196     SciaRegs.SCICTL1.all = 0x0023;         // Relinquish SCI
197                                             // from Reset
198 }
199
200 //
201 // transmitSCIAChar - Transmit a character from the SCI
202 //
203 void transmitSCIAChar(int msg)

```

```
197 {
198     while (SciaRegs.SCIFFTX.bit.TXFFST != 0)
199     {
200     }
201     SciaRegs.SCITXBUF.all = msg;
202 }
```

E.4. MainUI.py

```
1  #!/usr/bin/python
2
3  ## Libraries Import
4  from logging import Manager
5  from time import sleep , time
6  from kivy.app import App
7  from kivy.uix.widget import Widget
8  from kivy.lang import Builder
9  from kivy.uix.screenmanager import ScreenManager , Screen
10 from kivy.core.window import Window
11 from kivy.clock import Clock
12 import serial
13 import RPi.GPIO as GPIO
14
15 class CommunicationClass(object):
16
17     def __init__(self):
18         self.Mode = int(1)
19         self.Feed = 0.09
20         self.serialIndicator = 0
21         self.RPM = 0
22         self.Metric_BTN = 0
23         self.Imperial_BTN = 0
24         self.FeedFeed = 0.09
25
26     def SetBTN(self , Screen , BTN, State):
27
28         if Screen == 1:
29             self.Metric_BTN = BTN
30
31         elif Screen == 2:
32             self.Imperial_BTN = BTN
33
34         if State:
35             kv.screens [ Screen ] . ids [ str(BTN) ] .
36                 enabled = 1
37
38         else:
39             for n in range(0,4):
```

```

39         for m in range(0,14):
40             try:
41                 kv.screens[n].
42                     ids[str(m)
43                         ].enabled =
44                             0
45             except:
46                 pass
47
48     def getStatus(self):
49         return self.Mode, self.Feed, self.
50             serialIndicator, self.Metric_BTN, self.
51             Imperial_BTN, self.FeedFeed
52
53     def initCom(self):
54
55         if self.serialIndicator == 0:
56             try:
57                 self.ser = serial.Serial('/dev
58                     /ttyACM0', 9600, timeout =
59                         0.2)
60                 sleep(1)
61                 self.serialIndicator = 1
62                 self.Mode = 'Normal'
63
64             except:
65                 self.serialIndicator = 0
66             pass
67
68     def TX(self, Mode, Feed):
69         if Mode == 1:
70             self.FeedFeed = Feed
71
72             self.Mode = int(Mode)
73             self.Feed = round(Feed, 2)
74             self.FeedInt = int(Feed)
75             self.FeedDez = int((Feed - int(Feed))*100)
76
77             if self.serialIndicator:
78                 self.ser.write(b'\xff')
79                 self.ser.write(self.Mode.to_bytes(1,

```

```

73         byteorder='big'))
74     self.ser.write(self.FeedInt.to_bytes
75         (1, byteorder='big'))
76     self.ser.write(self.FeedDez.to_bytes
77         (1, byteorder='big'))
78     self.ser.write(b'\xff')
79     print('Transmission completed')
80
81     def RX(self):
82
83         lowbyte = 0
84         highbyte = 0
85         TransferComplete = 0
86
87         if self.serialIndicator:
88             while not TransferComplete:
89                 if self.ser.in_waiting == 2:
90                     lowbyte = int(
91                         from_bytes(self.ser
92                             .read(1), 'big',
93                             signed=False)
94                     highbyte = int(
95                         from_bytes(self.ser
96                             .read(1), 'big',
97                             signed=False)
98                     highbyte = highbyte <<
99                         8
100                     self.RPM = lowbyte +
101                         highbyte
102                     self.ser.flushInput()
103                     TransferComplete = 1
104                     return str(self.RPM)
105
106                 elif self.ser.in_waiting > 2:
107                     self.ser.flushInput()
108
109                 else:
110                     return str(self.RPM)
111
112         else:
113             return 'Not connected'

```

```

103
104 class Startseite(Screen):
105     def btn_defone(self):
106         MainApp.MainCom.SetBTN(0,0,0)
107         MainApp.MainCom.TX(1, 0.09)
108
109     def btn_deftwo(self):
110         MainApp.MainCom.SetBTN(0,0,0)
111         MainApp.MainCom.TX(1, 0.18)
112
113     def btn_normal(self):
114         feed = MainApp.MainCom.getStatus()[5]
115         MainApp.MainCom.TX(1, feed)
116         MainApp.MainCom.SetBTN(0,0,0)
117
118     def btn_gewinde(self):
119         if MainApp.MainCom.getStatus()[0] == 1:
120             kv.screens[1].ids[str(MainApp.MainCom.
121                                     getStatus()[3])].dispatch('on_press
122                                     ')
123
124             elif MainApp.MainCom.getStatus()[0] == 2:
125                 kv.screens[2].ids[str(MainApp.MainCom.
126                                         getStatus()[4])].dispatch('on_press
127                                         ')
128
129                 elif MainApp.MainCom.getStatus()[0] == 3:
130                     kv.screens[1].ids[str(MainApp.MainCom.
131                                             getStatus()[3])].dispatch('on_press
132                                             ')
133
134     def btn_prev(self):
135         if MainApp.MainCom.getStatus()[0] == 1:
136             MainApp.MainCom.TX(1,MainApp.MainCom.
137                                     getStatus()[1] - 0.01)
138             global release_event
139             release_event = Clock.
140                 schedule_interval(self.Decrement,
141                                   0.2)
142
143     elif MainApp.MainCom.getStatus()[0] == 2:

```

```

135         try:
136             kv.screens[1].ids[str(MainApp.
                MainCom.getStatus()[3] - 1)
                ].dispatch('on_press')
137         except:
138             pass
139
140     elif MainApp.MainCom.getStatus()[0] == 3:
141         try:
142             kv.screens[2].ids[str(MainApp.
                MainCom.getStatus()[4] - 1)
                ].dispatch('on_press')
143         except:
144             pass
145
146     def Decrement(self, *args):
147         MainApp.MainCom.TX(1, MainApp.MainCom.getStatus
            () [1] - 0.01)
148
149     def cancelDec(self):
150         if MainApp.MainCom.getStatus()[0] == 1:
151             release_event.cancel()
152
153     def btn_next(self):
154         if MainApp.MainCom.getStatus()[0] == 1:
155             MainApp.MainCom.TX(1, MainApp.MainCom.
                getStatus()[1] + 0.01)
156             global down_event
157             down_event = Clock.schedule_interval(
                self.Increment, 0.2)
158
159     elif MainApp.MainCom.getStatus()[0] == 2:
160         try:
161             kv.screens[1].ids[str(MainApp.
                MainCom.getStatus()[3] + 1)
                ].dispatch('on_press')
162         except:
163             pass
164
165     elif MainApp.MainCom.getStatus()[0] == 3:
166         try:

```

```

167         kv.screens[2].ids[str(MainApp.
                                MainCom.getStatus()[4] + 1)
                                ].dispatch('on_press')
168     except:
169         pass
170
171     def Increment(self, *args):
172         MainApp.MainCom.TX(1, MainApp.MainCom.getStatus
                                () [1] + 0.01)
173
174     def cancelInc(self):
175         if MainApp.MainCom.getStatus()[0] == 1:
176             down_event.cancel()
177
178 class MetrischeGewinde(Screen):
179     def btn_zero_four(self):
180         MainApp.MainCom.TX(2, 0.4)
181         MainApp.MainCom.SetBTN(1, 0, 0)
182         MainApp.MainCom.SetBTN(1, 0, 1)
183         pass
184
185     def btn_zero_five(self):
186         MainApp.MainCom.TX(2, 0.5)
187         MainApp.MainCom.SetBTN(1, 1, 0)
188         MainApp.MainCom.SetBTN(1, 1, 1)
189         pass
190
191     def btn_zero_seven(self):
192         MainApp.MainCom.TX(2, 0.7)
193         MainApp.MainCom.SetBTN(1, 2, 0)
194         MainApp.MainCom.SetBTN(1, 2, 1)
195         pass
196
197     def btn_zero_eight(self):
198         MainApp.MainCom.TX(2, 0.8)
199         MainApp.MainCom.SetBTN(1, 3, 0)
200         MainApp.MainCom.SetBTN(1, 3, 1)
201         pass
202
203     def btn_one(self):
204         MainApp.MainCom.TX(2, 1.0)

```

```
205         MainApp.MainCom.SetBTN(1, 4, 0)
206         MainApp.MainCom.SetBTN(1, 4, 1)
207         pass
208
209     def btn_one_two_five(self):
210         MainApp.MainCom.TX(2,1.25)
211         MainApp.MainCom.SetBTN(1, 5, 0)
212         MainApp.MainCom.SetBTN(1, 5, 1)
213         pass
214
215     def btn_one_five(self):
216         MainApp.MainCom.TX(2,1.5)
217         MainApp.MainCom.SetBTN(1, 6, 0)
218         MainApp.MainCom.SetBTN(1, 6, 1)
219         pass
220
221     def btn_one_seven_five(self):
222         MainApp.MainCom.TX(2,1.75)
223         MainApp.MainCom.SetBTN(1, 7, 0)
224         MainApp.MainCom.SetBTN(1, 7, 1)
225         pass
226
227     def btn_two(self):
228         MainApp.MainCom.TX(2,2.0)
229         MainApp.MainCom.SetBTN(1, 8, 0)
230         MainApp.MainCom.SetBTN(1, 8, 1)
231         pass
232
233     def btn_two_five(self):
234         MainApp.MainCom.TX(2,2.5)
235         MainApp.MainCom.SetBTN(1, 9, 0)
236         MainApp.MainCom.SetBTN(1, 9, 1)
237         pass
238
239     def btn_three(self):
240         MainApp.MainCom.TX(2,3.0)
241         MainApp.MainCom.SetBTN(1, 10, 0)
242         MainApp.MainCom.SetBTN(1, 10, 1)
243         pass
244     pass
245
```

```

246 class ZollGewinde(Screen):
247     def btn_ten(self):
248         MainApp.MainCom.TX(3,10.0)
249         MainApp.MainCom.SetBTN(2, 0, 0)
250         MainApp.MainCom.SetBTN(2, 0, 1)
251         pass
252
253     def btn_eleven(self):
254         MainApp.MainCom.TX(3,11.0)
255         MainApp.MainCom.SetBTN(2, 1, 0)
256         MainApp.MainCom.SetBTN(2, 1, 1)
257         pass
258
259     def btn_thirteen(self):
260         MainApp.MainCom.TX(3,13.0)
261         MainApp.MainCom.SetBTN(2, 2, 0)
262         MainApp.MainCom.SetBTN(2, 2, 1)
263         pass
264
265     def btn_nineteen(self):
266         MainApp.MainCom.TX(3,19.0)
267         MainApp.MainCom.SetBTN(2, 3, 0)
268         MainApp.MainCom.SetBTN(2, 3, 1)
269         pass
270
271     def btn_twenty(self):
272         MainApp.MainCom.TX(3,20.0)
273         MainApp.MainCom.SetBTN(2, 4, 0)
274         MainApp.MainCom.SetBTN(2, 4, 1)
275         pass
276
277     def btn_twentytwo(self):
278         MainApp.MainCom.TX(3,22.0)
279         MainApp.MainCom.SetBTN(2, 5, 0)
280         MainApp.MainCom.SetBTN(2, 5, 1)
281         pass
282
283     def btn_fourty(self):
284         MainApp.MainCom.TX(3,40.0)
285         MainApp.MainCom.SetBTN(2, 6, 0)
286         MainApp.MainCom.SetBTN(2, 6, 1)

```

```

287         pass
288
289     def btn_fourtyfour(self):
290         MainApp.MainCom.TX(3,44.0)
291         MainApp.MainCom.SetBTN(2, 7, 0)
292         MainApp.MainCom.SetBTN(2, 7, 1)
293         pass
294     pass
295
296 class SpezialGewinde(Screen):
297     pass
298
299 class SchnittdatenRechner(Screen):
300     pass
301
302 class Einstellungen(Screen):
303     pass
304
305 class WindowManager(ScreenManager):
306     pass
307
308 kv = Builder.load_file("kvroot.kv")
309
310 class MainApp(App):
311
312     MainCom = CommunicationClass()
313
314     def on_start(self):
315         Clock.schedule_interval(self.Cyclic, 0.1)
316
317     def Cyclic(self, *args):
318         MainApp.MainCom.initCom()
319         self.root.screens[0].ids.rpm_lable.text =
320             MainApp.MainCom.RX()
321
322         if MainApp.MainCom.getStatus()[0] == 1:
323             self.root.screens[0].ids.btn_gewinde.
324                 enabled = 0
325             self.root.screens[0].ids.btn_normal.
326                 enabled = 1
327             self.root.screens[0].ids.lable_feed.

```

```

325         text = str(MainApp.MainCom.
326             getStatus()[1])+ '_mm/rev'
327
328     elif MainApp.MainCom.getStatus()[0] == 2:
329         self.root.screens[0].ids.btn_normal.
330             enabled = 0
331         self.root.screens[0].ids.btn_gewinde.
332             enabled = 1
333         self.root.screens[0].ids.btn_gewinde.
334             text = 'Metrisch'
335         self.root.screens[0].ids.lable_feed.
336             text = str(MainApp.MainCom.
337                 getStatus()[1])+ '_mm'
338
339     elif MainApp.MainCom.getStatus()[0] == 3:
340         self.root.screens[0].ids.btn_normal.
341             enabled = 0
342         self.root.screens[0].ids.btn_gewinde.
343             enabled = 1
344         self.root.screens[0].ids.btn_gewinde.
345             text = 'Zoll'
346         self.root.screens[0].ids.lable_feed.
347             text = str(MainApp.MainCom.
348                 getStatus()[1])+ '_TPI'
349
350     def build(self):
351         return kv
352
353 if __name__ == "__main__":
354     Raspi = True
355     if Raspi == True:
356         GPIO.setmode(GPIO.BCM)
357         GPIO.setup(2, GPIO.OUT, initial=GPIO.HIGH)
358         sleep(1)
359         GPIO.output(2, GPIO.LOW)
360         sleep(1)
361         GPIO.output(2, GPIO.HIGH)
362     MainApp().run()

```