CODEN:LUTEDX/(TEIE-4505)/1-033/(2024)

Industrial Electrical Engineering and Automation

Automatisering av kalibreringsförfarande för momentgivare

Ludwig Schreither

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Abstract

When using measurement equipment, it is not only important to use the right equipment for the job, but to also make sure that it is measuring correctly. Electronic devices, as well as mechanical devices, all deteriorate over time. This leads to a need for calibration at a regular interval. By calibrating a measurement device by comparing its measurements to a verified reference device, it is possible to (within a specific tolerance) make sure that the measured values are true to reality.

This thesis focuses on automating the calibration and verification of torque measurement devices at BorgWarner's testing lab in Landskrona. During the process of calibrating a measurement device, there is a specific set of instructions for every device that needs to be closely followed. This, together with the need of having a high repeatability, implies that an automatic process would be the most suitable way to proceed.

Förord

Först vill jag tacka min handledare från LTH, Samuel Estenlund, för jobbet med att hjälpa mig navigera mig igenom det akademiska och för den feedback under arbetets gång som har hjälpt forma denna rapport. Jag skulle också vilja tacka min examinator, Jörgen Svensson, med den feedback och ärliga kritik som har lett till en förbättring av rapporten på både det akademiska och innehållsmässiga planet.

Jag vill också rikta ett stort tack till alla mina kollegor på BorgWarner i Landskrona som har varit med under denna process. Förutom att vara min handledare på BorgWarner, har Måns Andersson också varit ett stort stöd i hela processens gång. Utan inbördes ordning vill jag också nämna Jon, Gustav, Christoffer, Niklas, Andreas, Karl, Björn, Henric, Nicklas och Magnus. Ni har alla hjälpt till antingen med feedback på det tekniska arbetet, rapporten eller bara genom att lyfta humöret när det inte är på topp.

Till sist skulle jag vilja tacka Ronja, min familj och mina vänner för det stöd jag har fått både innan och under detta arbete.

Ordlista

ADS Automation Device Specification. 6

CAN Controller Area Network. 6

cRIO Compact Reconfigurable Input/Output. 6, 20, 21

EtherCAT Ethernet for Control Automation Technology. 6

FIFO First in, first out. 18

HBM Hottinger Brüel & Kjær. 5, 11

IPC Industrial PC. 7, 16, 20

NI National Instruments. 6, 20, 21

OOP Object-oriented programming. 17

PLC Programmable Logic Controller. 7, 16–18

ST Structured Text. 7, 17, 20, 21

XAE eXtended Automation Environment. 7, 16

Innehåll

Kapitel 1

Inledning

1.1 Bakgrund

BorgWarner [9] är ett internationellt företag som utvecklar olika komponenter för bilindustrin. Hos BorgWarner i Landskrona finns ett av de testlaboratorium som företaget driver. I detta laboratorium sker många tester där vridmoment är en av de viktiga uppmätta storheterna. För att kunna veta att de uppmätta värdena stämmer överens med verkligheten kalibrerar man de vridmomentgivarna som används till dessa mätningar.

1.2 Syfte

Kalibreringsprocessen är för nuvarande en process som tar mycket tid, kräver en stor del manuellt arbete och kräver en operatör med ingående kunskap inom både processen och den nuvarande hårdvaran/mjukvaran. Examensarbetet ska försöka ta fram en lösning som kan automatisera denna process och därmed minska tiden och kunskapen som krävs för kalibreringen.

1.3 Mål

Utkastet till det första måldokumentet togs fram ett par månader innan arbetet började, och under den tiden hade de omkringliggande förutsättningarna för arbetet hunnit ändrats. Detta ledde till att målet fick revideras innan arbetet startades för att kunna anpassas efter de nya omständigheterna.

1.3.1 Reviderade mål

Examensarbetet ska undersöka vilken hårdvara som skulle krävas för att automatisera kalibreringsprocessen och att sedan ta fram en mjukvara med denna hårdvara simulerad för att kunna användas i framtiden.

1.3.2 Ursprungliga mål

De ursprungliga målet inkluderade allt som finns med i de reviderade målen, med tilläggen:

- Test av hårdvara i testrigg
- Utveckling av ett användargränssnitt

På grund av platsbrist och en uppskjuten planerad ombyggnad av den nuvarande kalibreringsriggen fanns inte möjligheten att testa den utvecklade mjukvaran på den faktiska kalibreringsriggen.

Användargränssnittet blev bortprioriterat då fokus av arbetet i slutändan skulle ligga på automatisering av processen och implementationen mellan hårdvara och mjukvara.

1.4 Avgränsningar

Examensarbetet kommer inte att ta fram en färdig produkt som kan användas i BorgWarners laboratorium, utan är tänkt som en grund för att bygga vidare på. Detta arbete kommer inte att gå in djupare på aspekter såsom personsäkerhet, mekanisk design, elektrisk design och användargränssnitt.

Kapitel 2

Teknisk bakgrund

Detta kapitel beskriver nödvändig teknisk bakgrund för att förstå genomförandet av projektet. Det innefattar definitioner, komponenter som är viktiga att förstå, de mjukvarubaserade verktyg som använts, samt beskrivningar kring hur det befintliga kalibreringssystemet är uppbyggt.

2.1 Testobjekt

Nedan beskrivs de komponenter som ska testas och kalibreras i processen.

2.1.1 Momentgivare och funktion

En momentgivare är i detta projekt definierad som ett mätinstrument kopplad till en kardanaxel och på så sätt kan mäta vridmomentet som påverkar denna axel. De momentgivare som används här är av typen trådtöjningsgivare. Den funktionella principen för dessa är ett variabelt motstånd som varierar beroende på hur ytan den sitter på töjs eller komprimeras.

2.1.2 Momentgivare

Företaget Hottinger Brüel & Kjær (HBM) är en tillverkare av ett antal olika testinstrument, givare och mätsystem. Den momentgivare som används i större delen av BorgWarners testriggar är tillverkad av HBM och är av modellen T12[12] (Se figur 2.1). Den tillhandahåller ett antal gränssnitt, bland annat CANopen (beskrivs i avsnitt 2.3.3) som används i detta projekt.

Givaren består av en stator och en rotor, där rotorn sitter kopplad på en axel och via en antenn överför de uppmätta värdena till statorn. Statorn ser sedan till att omvandla dessa värden för att sedan skicka de vidare över de olika möjliga kommunikationsgränssnitten. Rotorn är strömförsörjd induktivt via statorn, och kräver därför inget batteri för att kunna fungera.

2.1.3 Telemetrisystem

CAEMAX Dx [5] är ett telemetrisystem där en av de storheterna som kan mätas är vridmoment. Mätsystemet består av en basstation som via radio kan ta emot mätdata från olika sensorer. I BorgWarners fall består sensorerna av trådtöjningsgivare som går att limma fast på utsidan av axeln. Dessa kopplas sedan till en radiosändare som för över de uppmätta värdena till basstationen. Dessa strömförsörjs antingen med en ringstator eller ett batteripack som är fasttejpat på axeln. Fördelen med detta jämfört med en T12:a är att sensorerna tar mycket mindre plats och att det därför går att installera och använda i bilar.

2.2 Testrigg

Detta avsnitt behandlar hur den nuvarande kalibreringsriggen hårdvarumässigt är uppbyggd. Figur 2.2 visar en översiktbild över hur det nuvarande systemet är uppbyggt. Den befintliga elektriska hårdvaran (av-

snitt 2.2.2) används som grund för att jämföra med den elektriska hårdvaran som tagits fram under detta arbetets gång. ger en översikt över hur den nuvarande testriggen är uppbygd.

2.2.1 Befintlig Mekanisk Hårdvara

Den befintliga mekaniska hårdvaran innefattar en servomotor sammankopplad med en växellåda, ett par kardanaxlar med olika längder, mekaniska adaptrar för de olika användningsområdena och ett stativ där referensgivaren monteras (Se figur 2.3). Stativet är inte endast till för att kunna montera referensgivaren, utan detta fungerar även som ett mekaniskt mothåll då momentgivarna i denna rigg kalibreras utefter ett statiskt[1] vridmoment.

Växellådan används för att kunna få en noggrann kontroll över vridmomentet genom att växla ned rotationen av servomotorn, vilket också leder till att servomotorns maximala vridmoment tillåts vara mycket lägre än de vridmoment som kalibreringsriggen ska kunna åstadkomma.

2.2.2 Befintlig Elektrisk Hårdvara

Det befintliga mät- och styrsystemet består av en Compact Reconfigurable Input/Output (cRIO)-9045 [4] från National Instruments (NI). Denna FPGA-kontroller har åtta platser för expansionsmoduler, vilka används för att anpassa systemet för det tänkta användningsområdet. Ett exempel på en expansionsmodul är NI-9375, en modul med 16st digitala ingångskanaler och 16st digitala utgångskanaler. Denna används t. ex. för att styra indikatorlampor och läsa av status på tryckknappar.

I det befintliga styrsystemet används en servodrift från Kollmorgen av typen AKD-P01207-NBCC-E000 som styrs över TCP/IP.

2.2.3 Referensgivare

Som referensgivare i testriggen används en HBM T12 HP (High Precision). Testriggen har ett antal T12 HP med olika mätområden där det går att byta mellan de olika varianterna. Dessa referensgivare används oberoende av vilket testobjekt som ska kalibreras, och det är endast mätområdet som tas i beaktning när referensgivaren bestäms för ett specifikt testobjekt. Referensgivarna skickas på kalibrering hos ett externt testlabb för att få så liten mätosäkerhet i dessa som möjligt.

2.3 Kommunikationsgränssnitt

En av de stora komponenterna i arbetet är de olika typer av kommunikationsgränssnitt som används mellan de olika delarna. Nedan beskrivs dessa kort för att ge en bild av hur de olika gränssnitten fungerar.

2.3.1 ADS

Automation Device Specification (ADS) [1] är ett protokoll utvecklat av företaget Beckhoff Automation och används för att kommunicera med och konfigurera deras PLC:er. Det är ett protokoll som bygger på TCP/IP, och gör det möjligt att kommunicera med enheter som använder sig av ADS via en vanlig nätverksanslutning.

2.3.2 EtherCAT

Ethernet for Control Automation Technology (EtherCAT) [6] är ett Ethernet-baserat kommunikationsgränssnitt som är utvecklat av Beckhoff Automation. Grundprincipen för EtherCAT bygger på en kommunikationsbuss där varje enhet i bussen sitter i serie. Masterenheten skickar ut ett kommunikationstelegram där varje instruktion adresseras till slavenheterna baserat på vilken plats de har i kedjan.

[1] I de flesta användningsfallen av momentgivarna sitter dessa kopplade till en axel som kan rotera flera tusen varv per minut, men när de kalibreras görs detta med en axel som inte tillåts rotera fritt

2.3.3 CANopen

CANopen är ett kommunikationsprotokoll som baseras på Controller Area Network (CAN). CAN är ett kommunikationsprotokoll där alla enheter kopplas ihop på samma buss, och därigenom har tillgång till alla meddelande som sänds på bussen. Varje CAN-meddelande innehåller ett unikt-id, vilket används för att bestämma vilket meddelande får prioritet om flera enheter försöker sända på bussen samtidigt. CANopen är ett protokoll som bygger på det fysiska CAN-lagret och som definierar hur enheter får kommunicera med varandra, vilka olika meddelandetyper det finns och hur enheter addresseras.

2.4 Styrsystem

Styrsystemet i detta projekt är utvecklat av Beckhoff Automation. Beckhoff Automation är ett tyskt företag som utvecklar styr- och kontrollsystem som baserar sig på Programmable Logic Controller (PLC). Dessa kommer i en stor mängd varianter, men de som används i detta projekt är DIN-monterade terminaler.

2.4.1 Beckhoff IPC

Beckhoff Industrial PC (IPC) [7] är en specialiserad dator. Denna har ett operativsystem och användargränssnitt, men har en inbyggd PLC för att kunna använda sig av olika mät- och styrmoduler.

2.4.2 Structured Text

Structured Text (ST) är ett textbaserat programmeringsspråk utvecklat för PLC-programmering. Det är ett av de fem språken som tillhör standarden IEC 61131-3:2013 [13], som är specifikt framtagen för att standardisera utvecklingen och användningen av PLC:er.

2.5 Mjukvaruverktyg

Nedan beskrivs de mjukvarubaserade verktyg som används i detta projekt.

2.5.1 TwinCAT XAE

TwinCAT EXtended Automation Environment (XAE) är Beckhoffs utvecklingsmiljö som baserar sig på Visual Studio. Detta används för att programmera och konfigurera Beckhoffs PLC:er. Verktyget tillåter en utvecklare att använda sig av flera olika programmeringsspråk, bland annat Structured Text (ST), vilket används i detta projekt.

2.5.2 TwinCAT Motion

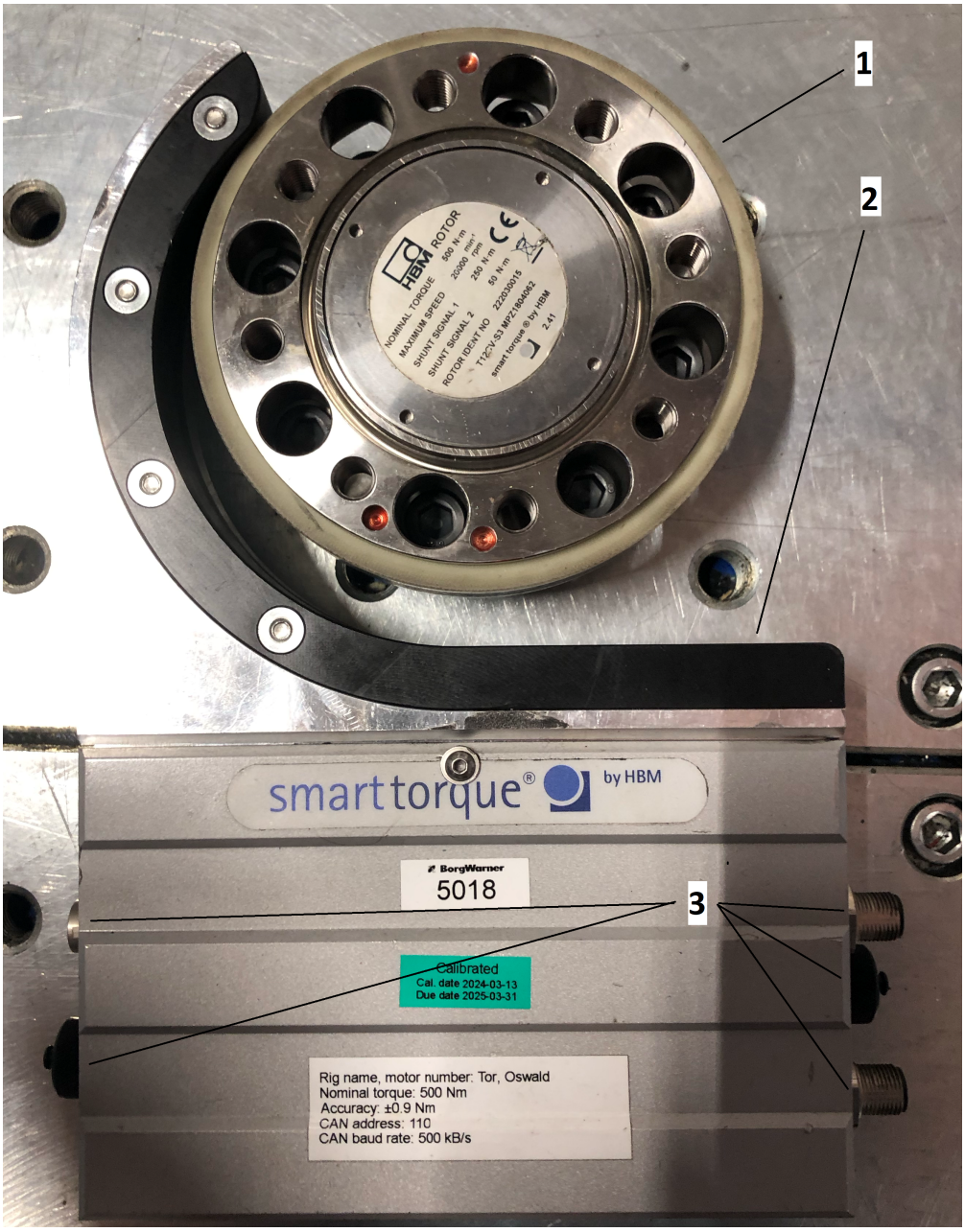
TwinCAT Motion är ett mjukvarubibliotek, tillsammans med ett konfiguration- och diagnostikverktyg, som används för att utveckla maskiner med rörliga delar. I detta projekt används TwinCAT Motion för att konfigurera och styra riggens servodrift, som i sin tur driver det servo i testriggen som ger upphov till vridmomentet som används för kalibreringen.

2.5.3 Robot Framework

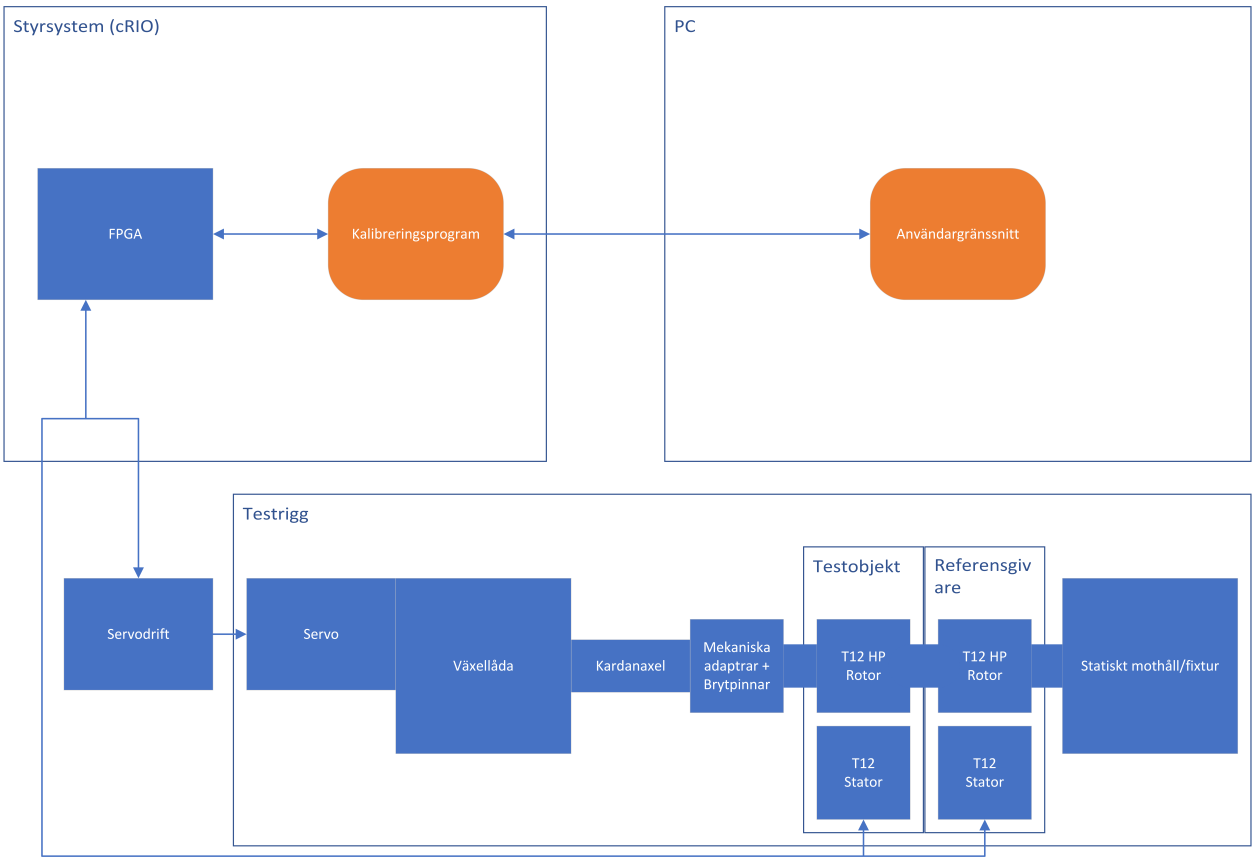
Robot Framework [10] är ett ramverk för att automatisera tester av mjukvara. Det är också möjligt att använda ramverket för att automatisera andra uppgifter, som t. ex. att köra igenom en testsekvens på en rigg.

2.5.4 Git

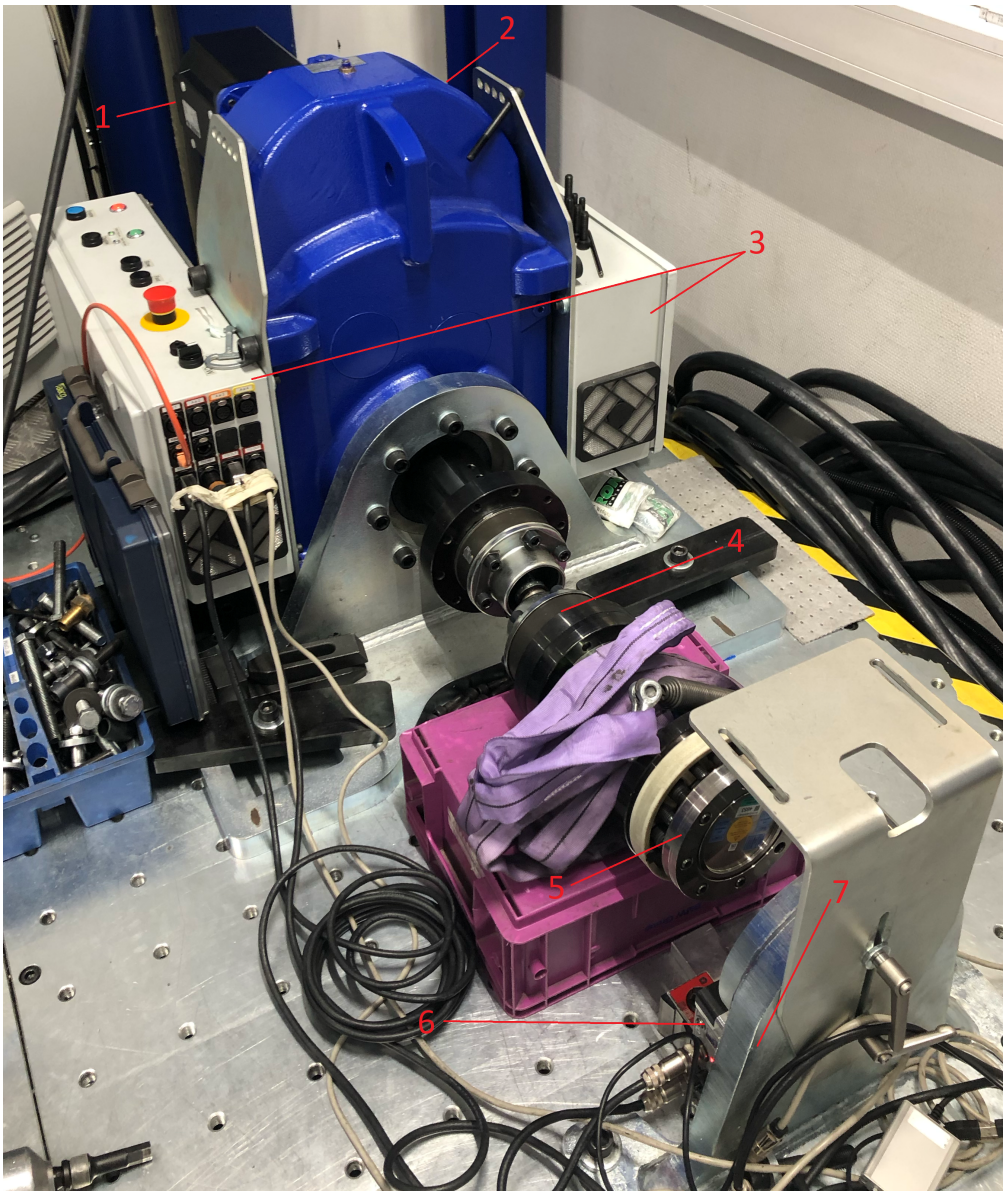
Git [8] är ett versionshanteringssystem som används för att hantera källkod. Förutom möjligheten att gå tillbaka till tidigare versioner, hanterar detta också skillnader i källkod och projekt när t. ex. flera personer arbetar tillsammans.



Figur 2.1: HBM T12: 1. Rotor med antenn, 2. Stator, 3. Kontakter för strömförsörjning och kommunikationsgränssnitt



Figur 2.2: Översiktsbild över det nuvarande systemet. De rektangulära blocken är hårdvarukomponenter, de rundade blocken mjukvarukomponenter och de orangefärgade blocken mjukvaruprocesser. För större bild, se bilaga A.1



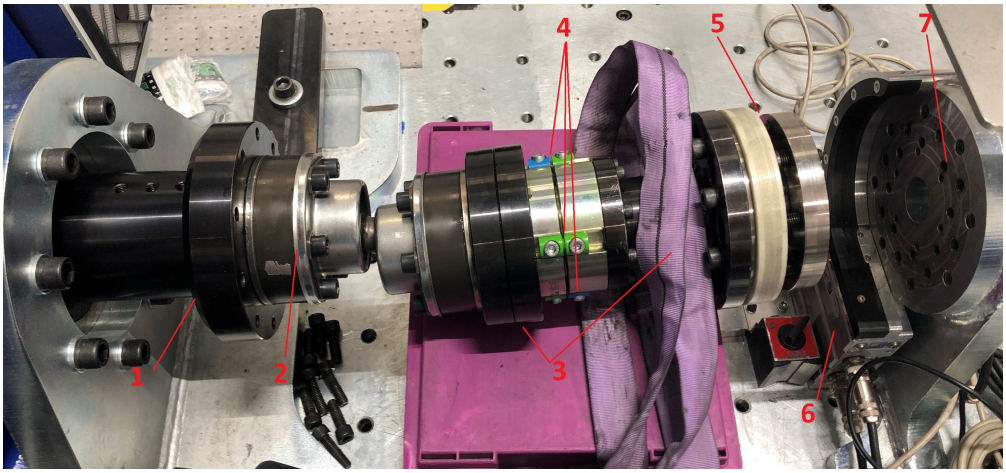
Figur 2.3: Nuvarande testrigg: 1. Servomotor, 2. Växellåda, 3. Elskåp för servodrift och FPGA-kontroller, 4. Kardanaxel (kort), 5. HBM T12 Rotor, 6. HBM T12 Stator, 7. Stativ/Mekaniskt mothåll

Kapitel 3

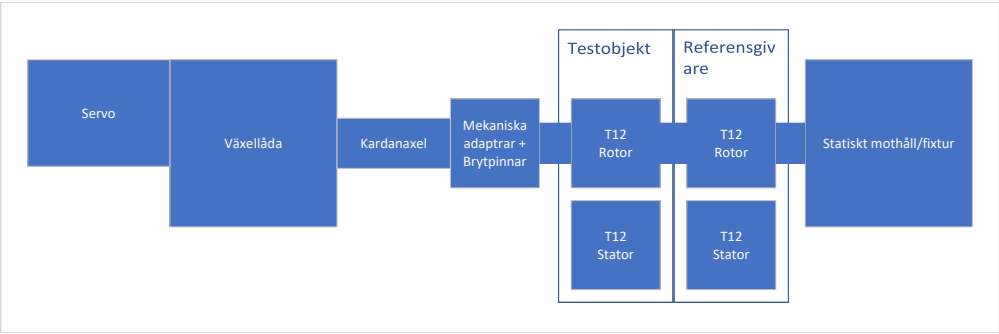
Metod

Detta kapitel förklarar själva kalibreringsprocessen, tillvägagångssättet som använts för att ta fram den hårdvara som behövts för detta projekt, samt hur de olika delarna av mjukvaran har utvecklats.

3.1 Förundersökning av kalibreringsprocess



Figur 3.1: Axelkoppling utan referensgivare: 1. Axel från växellåda, 2. Kardanaxel (kort), 3. Mekaniska adaptrar, 4. Vridmomentbegränsare med brytpinnar, 5. Rotor tillhörande T12 testobjekt, 6. Stator tillhörande T12 referensgivare, 7. Stativ/Mekaniskt mothåll

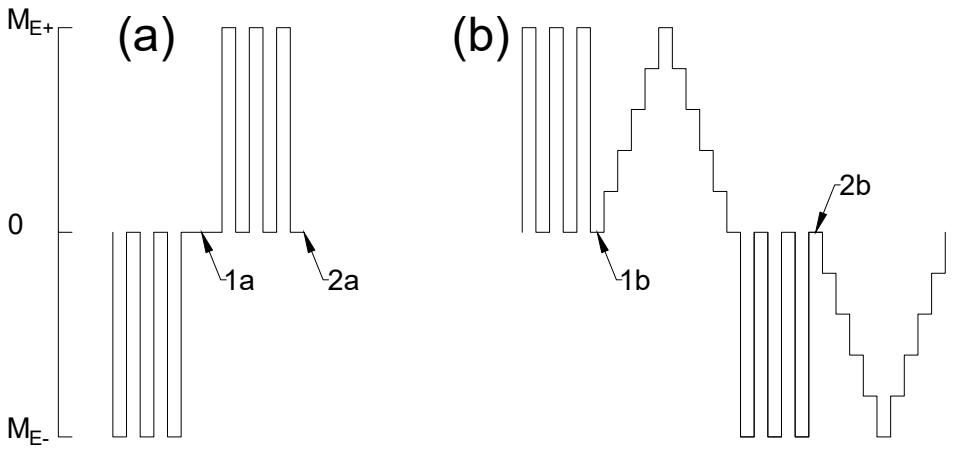


Figur 3.2: Grafisk representation av kalibreringsuppställningen i 3.1

Den nuvarande kalibreringsprocessen består av följande steg:

1. Referensgivaren för det relevanta mätområdet skruvas fast i en statisk fixtur.
2. Testobjektet skruvas fast i serie med referensgivaren.
3. Mätvärdet på de båda givarna nollas för att filtrera bort de externa krafterna som påverkar givarna.
4. Mellan servo och testobjektet kopplas en kardanaxel på, samt en vridmomentbegränsare bestående av två adaptrar med brytpinnar[1] (se figur 3.1) emellan.
5. Testobjektet belastas till ett par fördefinierade mätpunkter, där momentet hålls vid varje mätpunkt för att kunna ta en serie av mätvärden för senare utvärdering.
6. Föregående steg upprepas i negativ riktning.
7. Mätvärdena sätts in i en tabell där referensvärdet jämförs med det värdet som testobjektet rapporterade. Från denna tabell utvärderas statusen automatisk på momentgivaren enligt en ekvation som räknar ut risken att givaren ligger utanför sin specifikation.

Denna process tar ej hänsyn till den mekaniska remanensen [2] hos momentgivarna. För att kunna få en noggrannare och mer representativ mätning hade en utvärdering av remanensen behövts läggas till i denna sekvens. Detta leder till en uppdelning i två separata sekvenser:



Figur 3.3: Kalibreringssekvens

(a): Sekvens för uppmätning av mekanisk remanens hos referensgivaren

1. Referensen skruvas fast i en statisk fixtur.
2. Mellan servo och referens kopplas en kardanaxel på.

[1] Brytpinnar är designade för att brytas av vid en fördefinierad sidokraft

[2] HBM beskriver den mekaniska remanensen som en typ av hysteres, där denna är den maximala avvikelsen mellan det uppmätta värdet och det faktiska värdet när givaren stegas fram och tillbaka [11]

3. Belasta referensen till M_{E-}, givarens nominella negativa momentvärde, och gå sedan tillbaka till viloläget. Detta upprepas tre gånger enligt figur 3.3 (a).

4. Koppla av kardanaxeln från referensgivaren.

5. Nollställ referensens mätvärde och koppla sedan på kardanaxeln igen.

6. Upprepa steg 3, där referensen belastas till M_{E+}, givarens nominella positiva momentvärde, denna gång.

7. Koppla av kardanaxeln från referensgivaren.

8. Spara ned mätvärdet hos referensen som mr_{ref}.

9. Nollställ referensens mätvärde och koppla sedan på kardanaxeln igen.

(b): Sekvens för kalibrering av testobjekt

1. Montera testobjektet i kalibreringsriggen.

2. Belasta testobjektet och referens enligt sekvensen 3.3 (b), där de tre topparna innan både den positiva och negativa trappan utesluts från resultatet. [3]

3. Alla mätvärden sparas i en tabell.

4. I punkt 1b i figur 3.3 noteras offset mellan referens och testobjekt, och denna offset adderas till alla mätvärden i tabellen.

5. I punkt 2b i figur 3.3 beräknas testobjektets remanensskillnad Δmr genom att subrahera testobjektets mätvärde från referensens mätvärde.

6. Den mekaniska remanensen hos testobjektet beräknas sedan enligt $mr_{obj} = \Delta mr + mr_{ref}$.

7. Alla mätvärden på den negativa momenttrappan förskjuts med Δmr så att värdena från både testobjekt och referens är noll vid punkt 2b i figur 3.3.

Förutom tillägget av en extra sekvens som har en påverkan på mätresultatet, finns det en stor del skillnader gällande den bakomliggande hanteringen av ett mätdon i sig som kommer att kunna förbättras. Dessa skillnader beskrivs på en högre nivå i figur 3.4.

I den nuvarande uppställningen går det ej att ändra på parametrar hos momentgivarna när de är elektriskt kopplade till testriggen. Detta gör att givarna måste kopplas fram och tillbaka ett antal gånger under processen när man gör uppställningen. Parameterändringarna görs manuellt av operatören, inklusive nollning av mätvärde, och skapar därför en risk för detta inte görs korrekt.

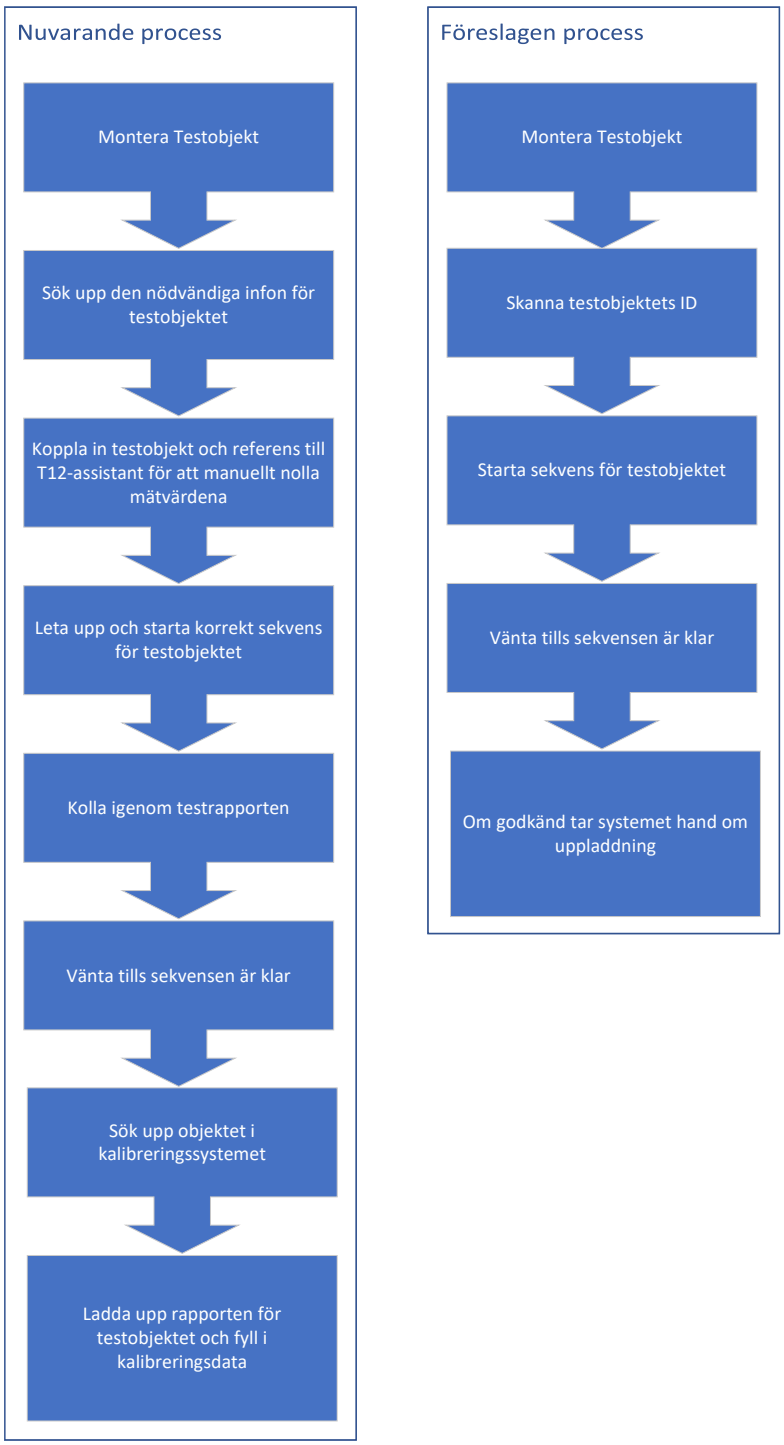
Behovet av loggningen av den mekaniska remanensen, tillsammans med större manuella arbete som detta tillför, gör att en automatiserad process hade både kunnat minska risken för fel, men också förkorta den tiden och specifika kunskapen som krävs för att kalibrera en momentgivare.

3.2 Val av hård- och mjukvaruplattform

Tre möjliga alternativ för kombinationer av hård- och mjukvara har undersökts. Dessa är:

- Nuvarande system, beskrivet i avsnitt 2.2.2.
- Beckhoff PLC med PC-baserad styrning implementerad med Python
- PC-baserad styrning implementerad med Python med CAN-dongle för kommunikation med servosystemet

[3] Dessa toppar används som 'pre-load' steg för att förminska påverkan av den mekaniska remanensen. Pre-load stegen utesluts ur resultaten då de endast används för att motverka den mekaniska remanensen



Figur 3.4: Flödesschema för kalibreringsprocessen

För och nackdelar för de olika alternativen togs fram och vägdes mot varandra

System	Fördelar	Nackdelar
Nuvarande system	• Hårdvaran finns installerad	• Svårt att versionshantera • Svårt att underhålla • Komplicerad lösning • Säkerhetshårdvara är inte integrerad
Beckhoff PLC + Python	• Enkelt att versionshantera • Enkelt att underhålla • Säkerhetshårdvara integrerad i PLC-miljön • Industriell lösning • Bibliotek för styrning av servo finns • Standardiserad hårdvara hos Borg-Warner	• Största hårdvarukostnaden
PC-baserad styrning med Dongle	• Enkelt att versionshantera • Enkelt att underhålla • CAN-kommunikationen kan styras helt av datorn • Minimal hårdvara som behövs	• Helt ny och obeprövad lösning • Kräver att styrning av servodriften implementeras i mjukvara • Säkerhetshårdvara måste implementeras • Ej industriell lösning

Den lösning som valts i detta arbete är lösningen som baseras på Beckhoff PLC och Python. Detta då det är en industriell lösning som är standardiserad hos BorgWarner och som är både enkel att underhålla och versionshantera.

3.3 Vald hårdvara

För att kunna styra servodriften krävs en Beckhoff PLC med en EtherCAT-busskopplare och stöd för TwinCAT Motion. Servodriften kräver ett par digitala signaler utöver kopplingen via EtherCAT, och både referensgivaren och givaren som ska kalibreras behöver sin egna CANbus. Förutom kringliggande komponenter för strömförsörjning och dylikt behövs då följande hårdvara från Beckhoff:

- IPC PLC, 1 st
- EL1859 (Digital IO-modul), 1 st
- EL6070 (Licensterminal), 1 st

- EL6751 (CANOpen mastermodul), 2 st
- EK1100 (Busskopplare från Beckhoffsystem till servodrift), 1 st

3.4 Implementering av mjukvara

Detta kapitel behandlar utvecklingen och implementeringen av mjukvaran. Kapitlet tar också upp den teoretiska implementeringen av reglersystemet.

3.4.1 Kommunikation med servosystem

Servosystemet kopplades in via EtherCAT och konfigurerades via TwinCAT Motion. För att kunna testa detta utan tillgång till den faktiska servodriften användes en annan drift med samma gränssnitt och tillhörande servo. Mellan TwinCAT Motion och huvudprogrammet användes Tc2_M2C biblioteket som ett gränssnitt. Detta bibliotek tillhandahåller ett antal funktionsblock som används för att kontrollera olika aspekter av servot. I detta projektet styrs och regleras servot baserat på axelvinkeln, där det går att begära en absolut vinkel, relativ vinkel, vinkelhastighet och vinkelacelleration.

3.4.2 Kommunikation med momentgivare

En HBM T12 momentgivare kopplades upp via en EL6751 CANopen mastermodul. Denna sattes upp som en CANopen-nod med ID 0x1. Det gjordes en del försök att kunna ändra ID på momentgivaren genom att använda LSS-protokollet som är specificerat i CANopen-standarden, men detta visade sig vara svårt att få till. Alla försök att göra detta med programvaran satte CANopen modulen i felläge. Efter diskussion med handledaren på BorgWarner bestämdes det att arbetet skulle fortsätta med att ändra ID på momentgivaren genom en CANopen-Dongle och HBM:s egna konfigurationsmjukvara T12-Assistant.

Momentgivaren konfigurerades att cykliskt skicka momentdata över CAN bussen med en frekvens på 1200 Hz. Dessa momentvärden skickades ut som Processdataobjekt [4] och kunde länkas till variabler i TwinCAT.

3.4.3 Kommunikation med Beckhoff PLC

För att kommunicera med Beckhoff PLC:n användes ADS-protokollet i samband med python biblioteket pyads. Pyads sköter den fysiska kommunikationen mellan TwinCAT och python, men det krävs mycket kringliggande kod för att kunna implementera kommunikationen på programnivå.

Kommunikationen är implementerad med hjälp av en asynkron kö. Sekvensen kan begära ett kommando som sedan läggs till i denna kön. När en annan process märker av att kön inte längre är tom, hanterar denna process kommandot och kallar på metoden `TwinCatCommunicator.__send_command` för att skicka kommandot till PLC:n. För att se till att detta fungerar som det ska har denna metod flera olika steg. Kommunikationen sker i slutändan via två globala variabler i PLC-koden: `stMachineCommand` och `stMachineCommandAck`. Metoden `TwinCatCommunicator.__send_command` läser först från `stMachineCommand` för att se till att denna ej har något kommando liggandes som ej blivit processerat. Om `stMachineCommand` är tom, skriver metoden över den tomma variabeln med det kommandot som ska skickas. När PLC:n har processerat kommandot kopieras detta över till `stMachineCommandAck`, och detta signalerar till programmet att PLC:n har registrerat kommandot. När PLC:n är klar med att utföra kommandot, kopierar den över en tom kommandostruktur i `stMachineCommand` och på så sätt får programmet reda på att ett nytt kommando kan sändas.

3.4.4 Implementation av reglerstrategi

Nuvarande implementation av reglerstrategi

Den tidigare reglerstrategin var implementerad så att servot roterades i fördefinierade steg i olika storlekar tills att det uppmätta värdet var tillräckligt nära referensvärdet. Detta var en simpel implementation av en reglerstrategi, men har två stora nackdelar:

[4] Processdataobjekt (PDO) inom CANOpen används för att skicka högprioriterade meddelanden, t. ex. mätvärden, status och kontrollmeddelanden [3]

- Denna implementation tar lång tid på sig att komma tillräckligt nära det önskade värdet.
- Risken är stor att stegen blir för stora och momentgivarens nominella mätområde överskrids.

Dessa två punkter leder till en längre kalibreringsprocess som också innebär en risk att skada momentgivaren.

Föreslagen implementation av reglerstrategi

$$\varphi = \frac{M_v \cdot L}{G \cdot I_v} \tag{3.1}$$

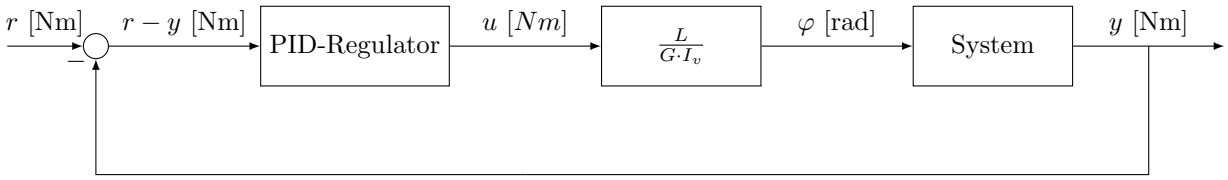
$$\frac{\varphi}{M_v} = \frac{L}{G \cdot I_v} \tag{3.2}$$

Med en typisk uppställning för kalibrering av en momentgivare kan vi förvänta oss ett någorlunda linjärt system. Detta efter att vi kommit till den punkt då kardanaxeln är uppspänd. Ekvationen (3.1) [2] beskriver förvridningsvinkeln φ (rad) mellan de två ändarna av en axel, och hur detta beror på vridmomentet M_v ($\mathrm{N} \cdot \mathrm{m}$), axellängden L (m), skjuvmodulen G (Pa) och vridtröghetsmomentet I_v ($\mathrm{kg} \cdot \mathrm{m}^2$). Flyttar vi sedan över vridmomentet till vänsterledet i (3.1) så får vi ett konstant förhållande mellan förvridningsvinkeln och vridmomentet enligt (3.2).

Eftersom systemet kan betraktas som linjärt när det är uppspänt och förhållandet mellan vridmomentet och förvridningsvinkeln är en konstant som endast beror på materiella egenskaper, går det att tillämpa följande reglerstrategi:

- Stega framåt tills att en tydlig ökning i vridmomentet fås.
- Spara ned momentvärdet och vinkeln på servot.
- Stega framåt 1-2 gånger med känd stegvinkel och spara ned de nya värdena på moment och vinkel.
- Subtrahera de första värdena från de senaste värden och på så sätt estimera konstanten $\frac{L}{G \cdot I_v}$

Det resulterande reglersystemet beskrivs i figur 3.5.



Figur 3.5: Reglersystemet från begärt moment r till uppmätt moment y

3.5 Simulering av systemet

Simulering av systemet gjordes med två separata uppställningar. Under de tester som gjordes av kommunikationen mellan TwinCAT och python användes endast en dator med TwinCAT XAE, där denna kommit installerad med en TwinCAT 'Runtime'. Den runtime som installerats tillåter en att direkt i utvecklingsmiljön ladda upp och köra TwinCAT-baserade PLC-program på den lokala datorn. Detta gör att ingen extern hårdvara behövs för att genomföra denna typ av simulering eller tester.

I den andra uppställningen användes fortfarande samma dator, men Beckhoff-terminaler kopplades in via datorns ethernet-port. Detta gjorde att datorn kunde fungera som en IPC och moduler som t. ex. servodriften kunde testas. Detta kräver endast att man har tillgång till de specifika fysiska terminalerna som behöver testas, men det kräver ingen annan hårdvara (förutom en busskopplare[5] och ethernetkabel).

[5]En busskopplare är en terminal som i detta fall har en ethernetport på ena sidan och terminalanslutning på andra sidan för att kunna koppla på flera terminaler efter denna

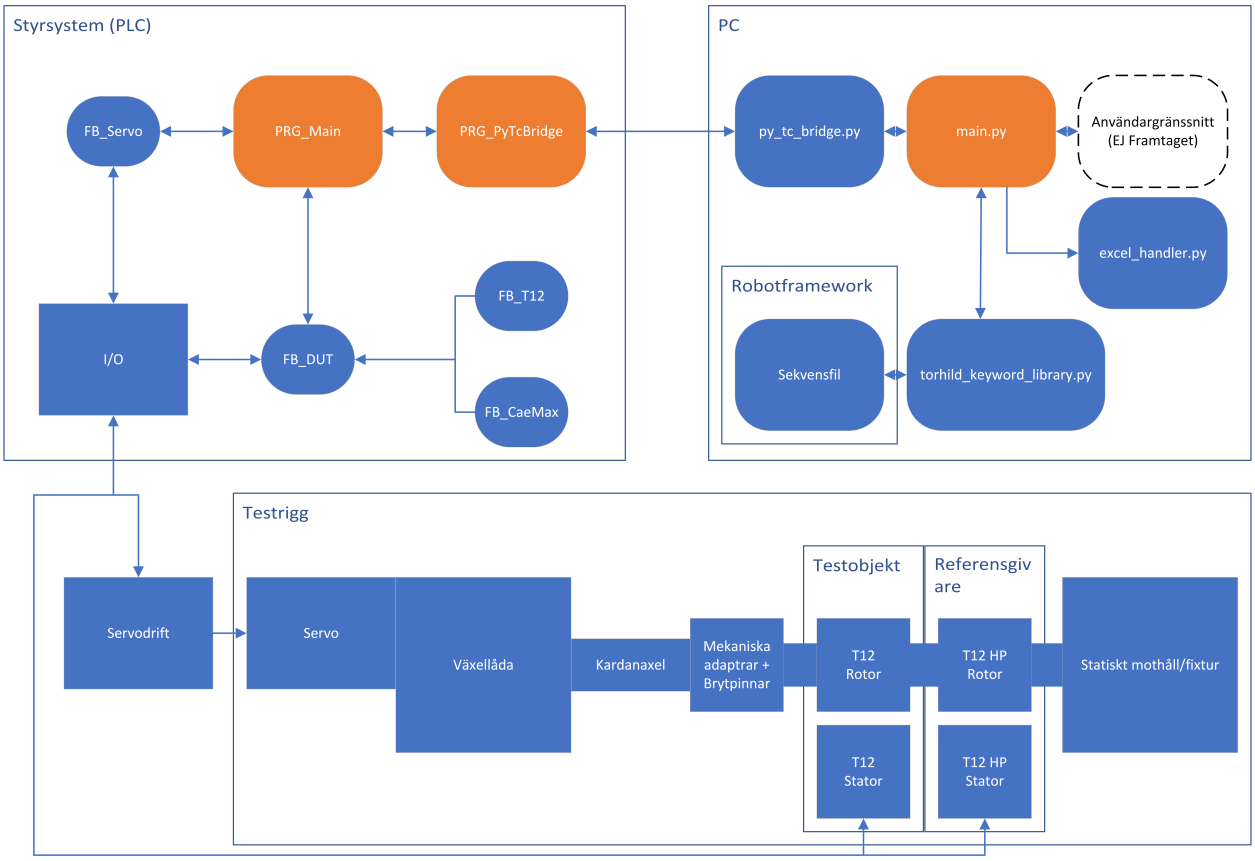
3.6 Test av projektet

PLC-projektet kördes på den lokala utvecklingsdatorn, vilken kopplades samman till servodriften över datorns ethernetport. Via pythonprogrammet startades en simpel sekvens som satte en momentbegränsning till 100 Nm och sedan stegade servot ett antal gånger. Under tiden observerades statusen på kommunikationen mellan PLC-runtime och pythonprogrammet, delvis genom att övervaka minnet och exekveringen hos PLC:n, men också genom loggar i konsolen över de meddelanden som skickats och tagits emot.

Efter detta test gick det att se att sekvensen exekverades som den skulle, att all kommunikation mellan sekvenshanteringen, pythonprogrammet och PLC:n var korrekt och att servot stegade framåt enligt sekvens med den efterfrågade momentbegränsningen.

Kapitel 4

Konfiguration och resultat



Figur 4.1: Översiktsbild över det nya systemet. De rektangulära blocken är hårdvarukomponenter, de rundade blocken mjukvarukomponenter och de orangefärgade blocken mjukvaruprocesser. För större bild, se bilaga A.2

Detta kapitel presenterar den hårdvara som valts och mjukvara som tagits fram under projektets gång. Det förklarar hur systemet är uppbyggt, hur de olika modulerna fungerar och hur delarna kopplas samman. I figur 4.1 går det att se en översiktsbild över hela systemet.

4.1 Hårdvara

Den valda hårdvaran som användes under de ursprungliga testerna (se avsnitt 3.3) visade sig fungera väl för projektets ändamål. Den komponent som inte specificeras i detta projekt är själva IPC:n, där det finns många möjliga val beroende på vilken funktionalitet som krävs utöver projektets krav.

4.2 Mjukvara

Mjukvaran är implementerad med fokus på Object-oriented programming (OOP) och abstraktion för att skapa olika block. Förutom användningen av de olika bibliotek som använts (Varav de stora är TwinCAT motion, Pyads och Robotframework), har PLC-mjukvaran, Python-Mjukvaran och sekvensfilerna utvecklats inom detta arbete under projektets gång.

4.2.1 PLC

Mjukvaran på PLC:n består av två olika program som körs parallellt med varandra, och ett antal funktionsblock som tillhandahåller ett gränssnitt till de olika hårdvarukomponenterna i kalibreringsriggen. Figur 4.2 visar en översiktsbild över hur de olika programmen och funktionsblocken hör ihop. Namngivning av de olika programmen och funktionsblocken i PLC-programmet följer Beckhoffs konvention gällande ST.

Prefix	Typ	Användning
PRG_	Program	Program är de huvudprocesser som cycliskt exekveras
FB_	Funktionsblock	Funktionsblock (i OOP också kallat för klass) är ett program vars exekvering inte är knuten till en cyklisk process. Exekveringen av dessa sker utifrån att de specifikt kallas på från ett av de cykliska programmen.

Det första programmet är huvudloopen (`PRG_Main`) som koordinerar de olika hårdvarukomponenterna, uppdaterar alla bussterminaler och håller koll på riggens tillstånd. Detta program exekveras cykliskt med en frekvens på 100 Hz.

Det andra programmet är kommunikationsloopen (`PRG_PyTcBridge`) som hanterar kommunikationen med python och skickar vidare kommandon till huvudloopen. Detta program exekveras cykliskt med en frekvens på 200 Hz[1]. Uppdelningen av dessa komponenter tillåter kommunikationen att ske asynkront från huvudloopen, och ger möjligheten att öka frekvensen på kommunikationsloopen separat, vilket leder till en lägre belastning på systemet. De funktionsblocken som finns är `FB_CaeMax`, `FB_T12`, `FB_DUT` och `FB_Servo`. `FB_CaeMax` och `FB_T12` är block som beskriver funktionaliteten hos de två mätdonen som används i riggen, och innehåller metoder för att konfigurera dessa. `FB_DUT` är ett block som kapslar in instanser av `FB_T12` och `FB_CaeMax` för att kunna ha ett gemensamt block som kan användas för att hantera båda typerna av momentgivare som BorgWarner innehar. Detta gör att resten av programmet inte behöver ändras beroende på vilket typ av objekt vi kalibrerar. Till sist har vi `FB_Servo` som tillhandahåller ett gränssnitt mellan servodriften och resterande program.

4.2.2 Pythonprogram

Pythonprogrammet är uppdelat i följande fyra olika beståndsdelar:

- `main.py`
- `py_tc_bridge.py` [2]

[1] Denna uppställning kräver inte en hög frekvens på exekvering och sampling, då samplingen av momentet sker vid statiska punkter

[2] Inte att förväxlas med `PRG_PyTcBridge`, som är den del som `py_tc_bridge.py` kopplar ihop sig mot för att kunna kommunicera åt båda hållen.

- `torhild_keyword_library.py`
- `excel_handler.py`

En översiktbild för pythonprogrammet (som inkluderar kopplingen till Robotframework) går att se i figur 4.3.

Delen `main` initierar programmet i sig, ser till att starta upp alla nödvändiga loopar och ger användaren möjligheten att välja vilken kalibreringssekvens som ska köras. Denna del innehåller också en klass som används globalt i hela systemet för att tillåta Robotframework att interagera med PLC:n.

`py_tc_bridge.py` innehåller två klasser, `PLC` och `TwinCatCommunicator`. Klassen `PLC` skapar kopplingen mellan TwinCAT och python, och tillhandahåller ett gränssnitt för att kunna skicka data mellan dessa. `TwinCatCommunicator` hanterar kommunikationsflödet och innehåller en First in, first out (FIFO)-kö där de kommandon som programmet begär lagras för att sedan skickas över till PLC:n.

`torhild_keyword_library.py` fungerar som ett bibliotek för Robotframework och definierar de nyckelord som kan användas när man skriver en sekvens. Detta gör att sekvenserna går att skriva utan ingående förståelse för hur programmet i sig fungerar.

`excel_handler.py` genererar kalibreringsrapporter utefter fördefinierade rapportmallar.

4.2.3 Robotframework

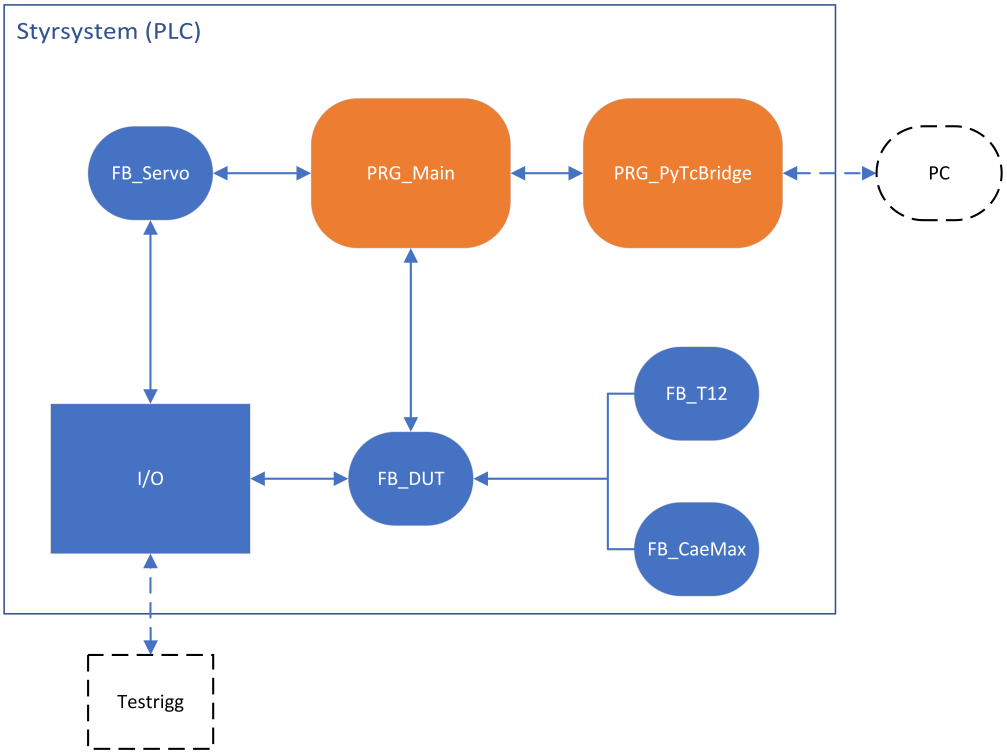
Denna del består endast av sekvensfiler som definierar i vilken ordning de kommandona som beskrivs i `torhild_keyword_library.py` ska köras. Sekvenserna skrivs i någorlunda klartext där syntax för att kalla på ett kommando kan se ut på följande vis: `Set torque limit to 100`.
Ett exempel på hur en väldigt simpel sekvens skulle kunna se ut går att se i figur 4.4.

4.3 Implementation av reglerstrategi

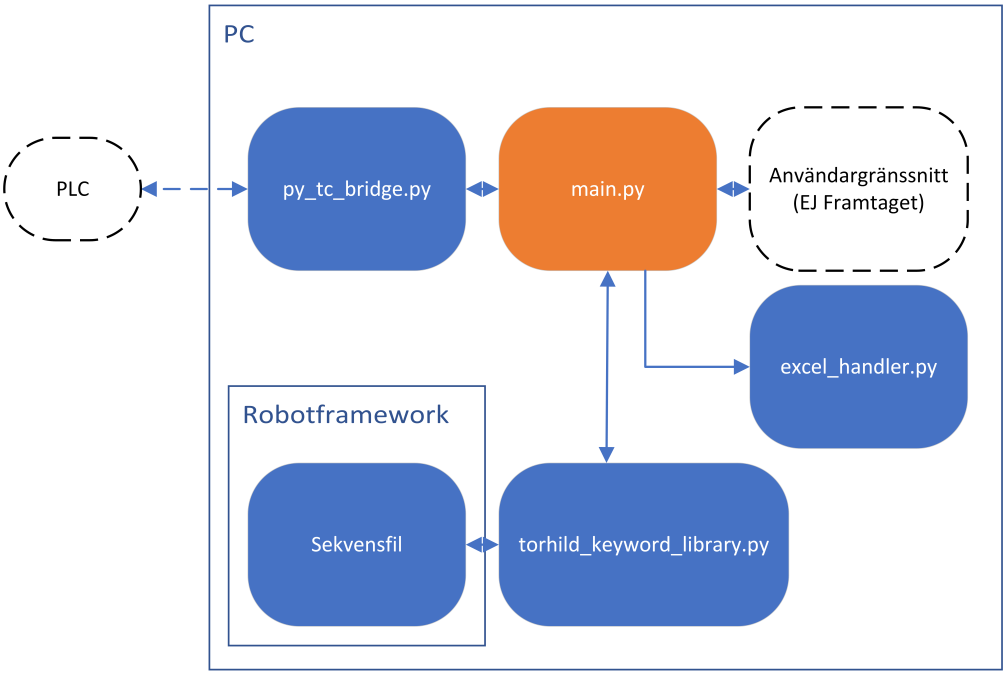
Teorin kring reglerstrategin och hur det bör fungera har tagits fram och utifrån detta har ett par grundläggande metoder skapats. Det har inte varit möjligt att testa detta i praktiken, vilket gör att utvecklingen av reglersystemet i detta projekt endast är grundläggande.

4.4 Resultat

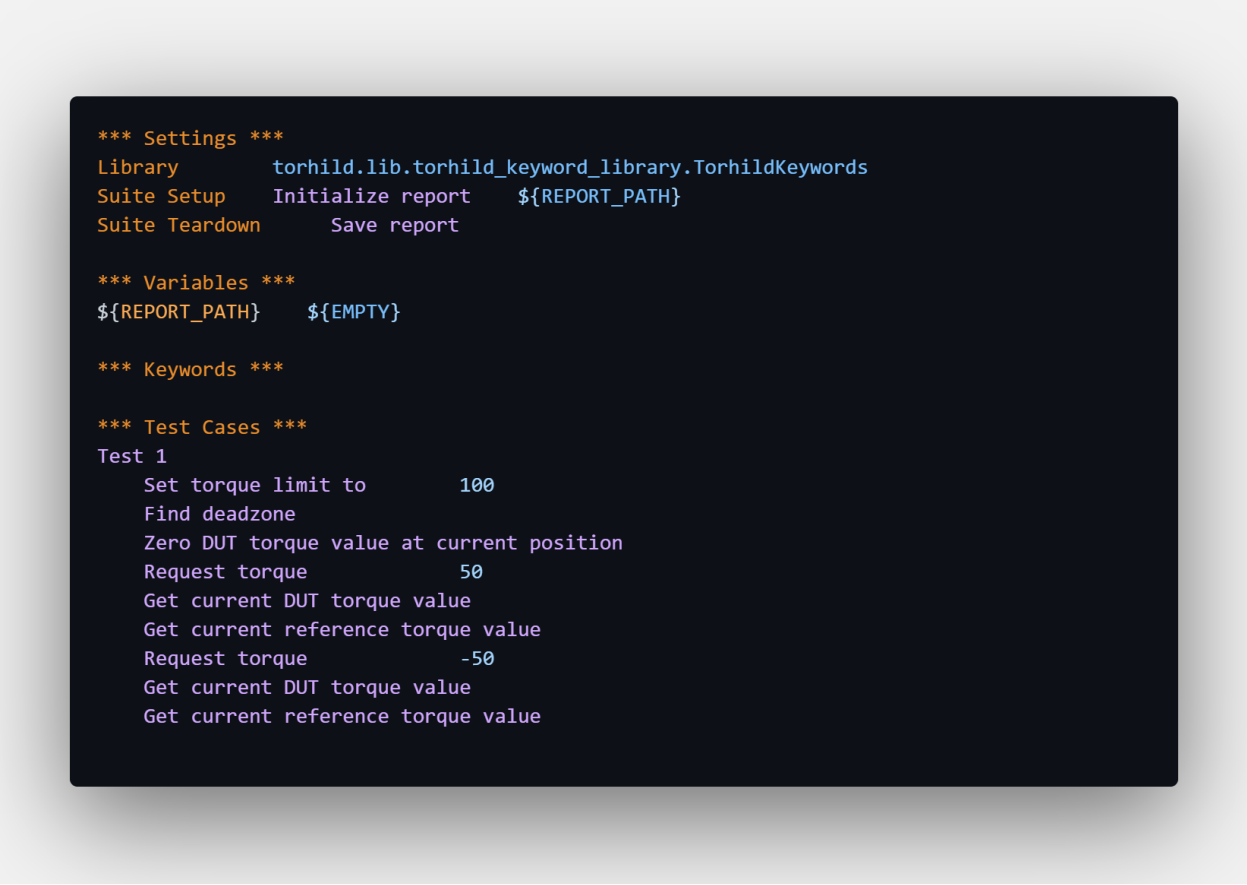
Enligt avsnitt 3.6 så testades konfigurationen beskriven i detta avsnitt med simulerad hårdvara. Testet visade på att sekvensfilen kunde styra servodriften, och att all kommunikation skedde på ett korrekt sätt.



Figur 4.2: PLC programstruktur



Figur 4.3: Python programstruktur



```
*** Settings ***
Library         torhild.lib.torhild_keyword_library.TorhildKeywords
Suite Setup     Initialize report    ${REPORT_PATH}
Suite Teardown      Save report

*** Variables ***
${REPORT_PATH}    ${EMPTY}

*** Keywords ***

*** Test Cases ***
Test 1
    Set torque limit to        100
    Find deadzone
    Zero DUT torque value at current position
    Request torque             50
    Get current DUT torque value
    Get current reference torque value
    Request torque             -50
    Get current DUT torque value
    Get current reference torque value
```

Figur 4.4: Exempel på Robotframework-sekvens

Kapitel 5

Diskussion och slutsatser

5.1 Hårdvara

Jämför man den tidigare hårdvaran med den nya går det att hitta både fördelar och nackdelar med båda systemen. Vi kan dela upp dessa i följande punkter: Kostnad, mätosäkerhet/precision, expansionsmöjlighet, prestanda

Kostnad: Kostnaden för dessa system kan delas upp i två separata delar: Kostnaden för styrsystemet (cRIO eller IPC) och kostnaden för modulerna (NI-moduler och Beckhoffterminaler). En cRIO kostar ungefär fyra gånger mer än en IPC. Om vi tar och jämför mätmoduler från de båda systemet med ungefär liknande mätfunktioner hamnar vi på samma faktor där NI-modulerna också kostar ungefär fyra gånger mer än Beckhoff-terminalerna. I de flesta fallen gör detta att hårdvarukostnaden för ett Beckhoff-system är mindre än hårdvarukostnaden för ett NI-System.

Mätosäkerhet och precision:
Här kan man rätt genomgående se att NI-modulerna har en betydligt mindre mätosäkerhet och bättre mätprecision än vad Beckhoffs terminaler erbjuder. Det finns ett par alternativ om man vill använda sig av Beckhoffterminaler, men ändå ha samma mätprestanda som NI-modulerna, men detta gör också att kostnaden ökar markant.

Expansionsmöjlighet:
En cRIO har ett fixt antal platser för moduler (1, 4 eller 8). För att kunna ansluta fler moduler krävs ett externt chassi som sedan måste synkas med cRIO:n. I riggens nuvarande konfiguration finns det endast en plats av 8 st kvar, vilket leder till att en expansion kräver en större investering i hårdvara.

Beckhoffterminalerna kopplas på efter varandra, och det kan teoretiskt sitta flera hundra terminaler i samma slinga.

De avsedda användningsområdena för dessa två system är rätt olika. NI-systemet är anpassat efter en dynamisk labbmiljö med flera olika mätuppställningar och där det snabbt ska gå att ändra på mycket med systemet. Beckhoff-systemet riktar sig generellt sett mer mot industrin där man inte förväntar sig många täta förändringar, utan fokus ligger på mer statiska uppställningar/maskiner som pålitligt ska arbeta under lång tid.

5.2 Mjukvara

Arbetets plan att kunna ändra på parametrarna hos momentgivarna utan att behöva använda externa verktyg gick inte som planerat. På grund av de problemen med att givarna gick in i felläge när detta försöktes, beslöts det att gå vidare med resterande mjukvara och lämna den biten manuell. Möjligheten att göra detta finns, men tiden för att felsöka detta ordentligt fanns inte med inom ramen för detta arbete.

Om man tittar på både utvecklingen av och funktionaliteten hos mjukvaran finns det många bra saker att ta med sig, men också ett par nackdelar som bör nämnas. Om vi börjar med att titta på hur utvecklingen och underhållet av programvaran, kan vi börja se många fördelar jämfört med det tidigare systemet (LabVIEW).

Git som versionshanteringssystem av python/ST fungerar utan extra konfiguration eller några externa skript som behöver köras. Det går också utan några större problem att sammanfoga kodändringar när man jobbar med flera olika versioner, eller när flera personer utvecklar samtidigt. Jämfört med LabVIEW där man behöver sätta upp externa flöden och förstå på vilka sätt som användandet av versionshantering skiljer sig, så är det mycket lättare att sätta sig in i och använda sig av Git tillsammans med python/ST.

När det gäller dokumentation av koden så har LabVIEW stöd för detta, men inte alls i samma utsträckning som med ett textbaserat programmeringsspråk. Dessutom går det i detta fall med python/ST att använda sig av externa dokumentationsverktyg som extraherar kommentarer och ritar upp förhållandet mellan olika kodmoduler.

Den sista och relativt viktiga skillnaden är att python är ett rätt välkänt programmeringsspråk med många hjälpresurser. LabVIEW är en någorlunda välkänd utvecklingsmiljö inom industrin, men kräver mycket tid för att kunna skriva funktionell och bra kod. Detta kan också leda till att det är väldigt svårt att sätta sig in i ett större LabVIEW-projekt som någon annan har programmerat, vilket oftast gör projektet svårt att underhålla eller vidareutveckla.

Om vi fortsätter med att jämföra funktionaliteten mellan LabVIEW och python/ST kan vi se att det finns för- och nackdelar med båda alternativen.

LabVIEW med tillhörande hårdvara tillåter att hela ekosystemet hamnar under en och samma utvecklingsmiljö där extern kommunikation mellan de olika programmen inte behövs. Mycket av utvecklingstiden gick åt att få igång en någorlunda stabil kommunikation mellan TwinCAT och python, vilket inte hade varit ett moment om man använt sig av LabVIEW tillsammans med en cRIO.

När det gäller att exekvera parallella processer är detta också mycket enklare med LabVIEW, då detta tas hand om av kompileringen. Att arbeta med parallella processer i python kräver mer planering och felsökning, och kan vara förvirrande när bibliotek använder sig av dessa metoder utan tydlig dokumentation avseende detta.

5.3 Slutsats

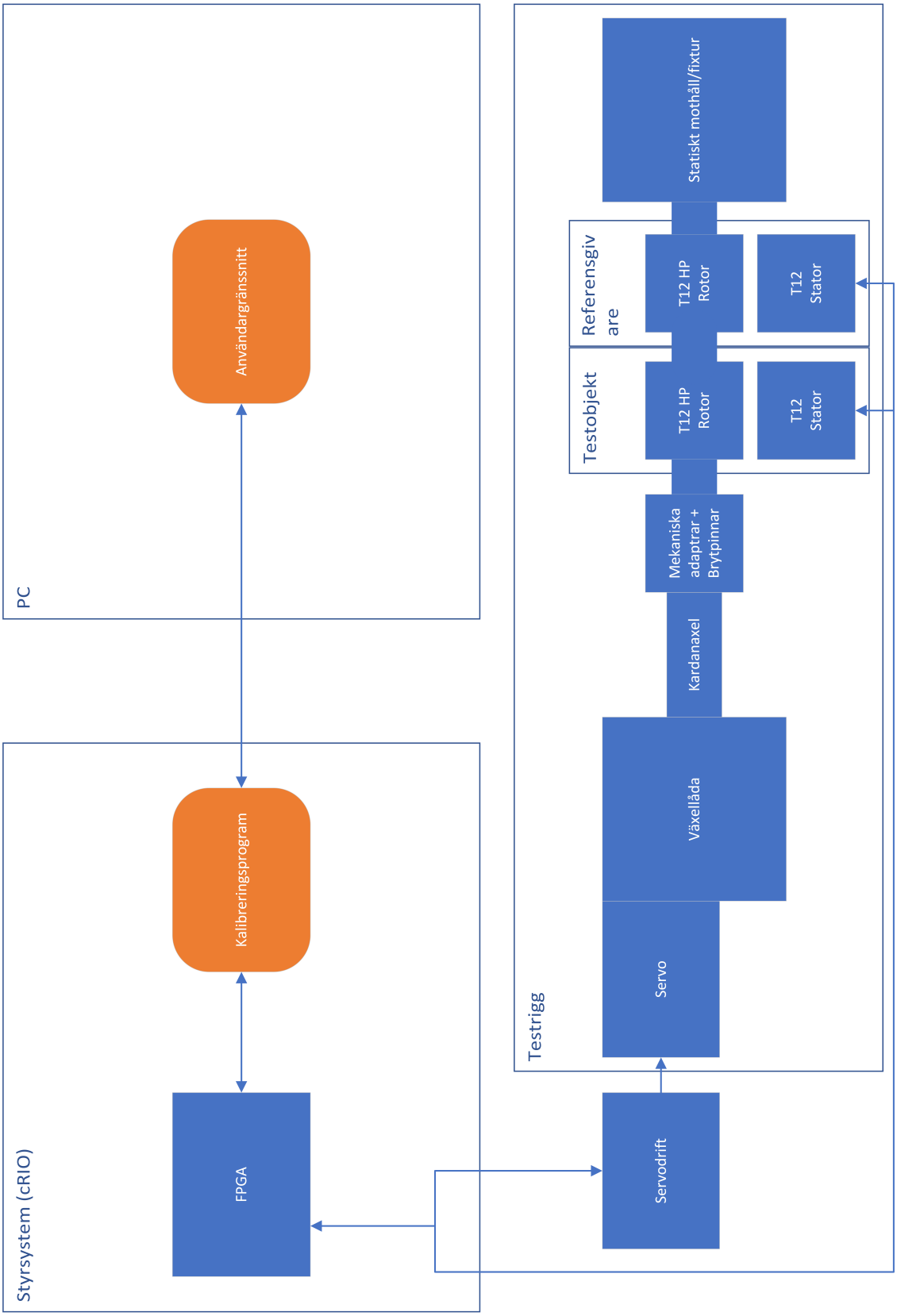
Den lösning som tagits fram under detta projekt har visats sig fungera för ändamålet, och har framåt stor potential inom både denna kalibreringsrigg och andra riggar hos Borgwarner. Det finns mycket att vinna i att göra utvecklingen och underhållet lättare. Med bättre dokumentationsverktyg, fler hjälpresurser och allmänt en bättre kunskap kring textbaserad programmering hos ingenjörer kommer det vara möjligt att öppna upp utvecklingen och felsökning till en bredare grupp, och på så sätt öka möjligheten för samarbete.

Det finns dock ett antal nackdelar med detta. Denna lösning är inte lika integrerad mellan hårdvara, gränssnitt och mjukvara som NI-systemet. Avsaknaden av denna integration kräver ett välbyggt ramverk som hela systemet kan vila på. Arbetet med kommunikationen mellan de olika delarna i detta projekt har visat att detta är en svår punkt att få välfungerande. Utvecklingen av ett ramverk för kommunikation, sekvenshantering och användargränssnitt är något som potentiellt sett kan ta lång tid och kan kräva utvecklare med hög kompetens inom området.

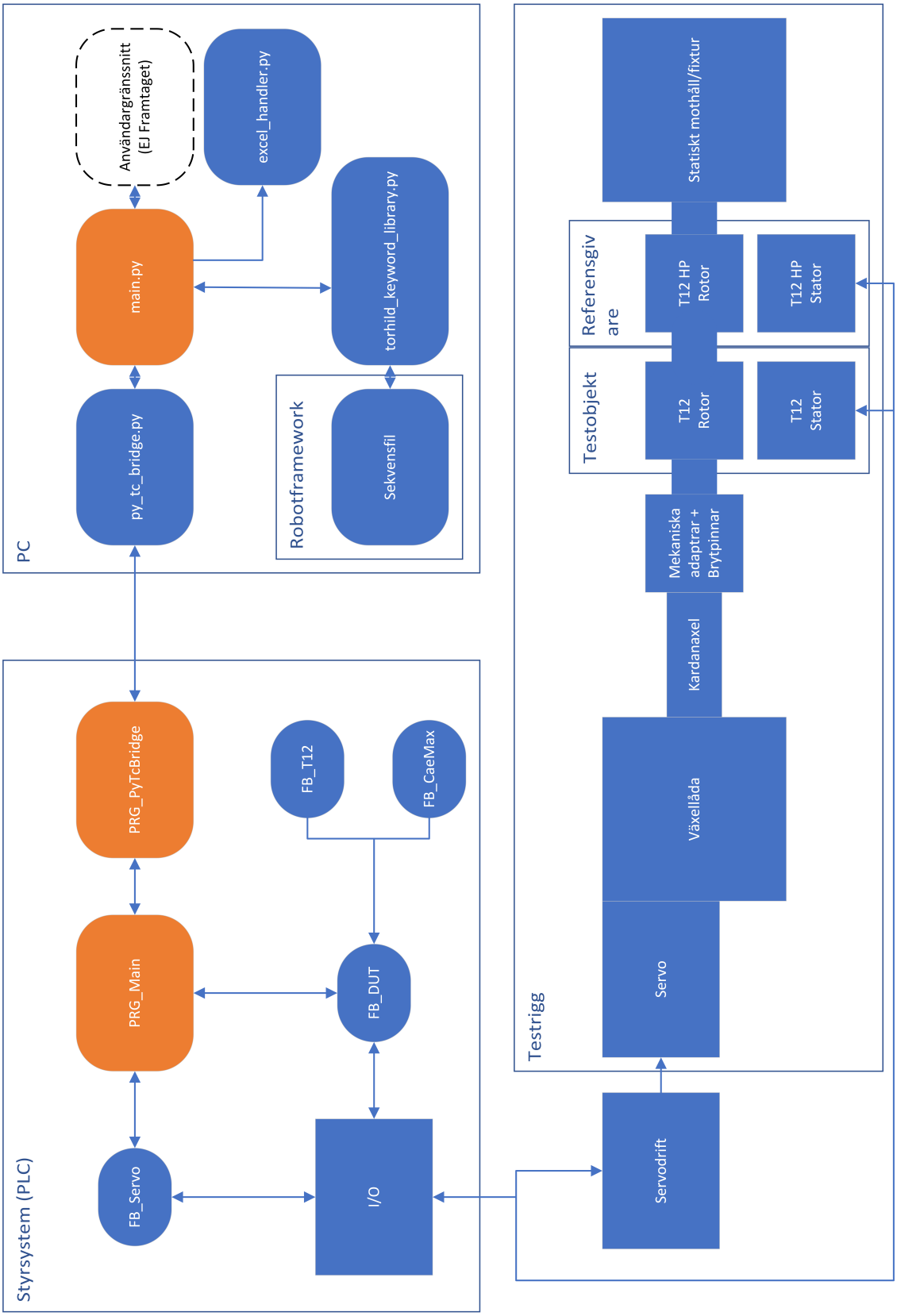
Kan man investera i den tiden som krävs för att kunna bygga ett ramverk för denna lösning, då kan man få en välfungerande och flexibel kalibreringsrigg med moduler som går att använda att bygga framtida projekt på.

Bilaga A

Översiktsbilder



Figur A.1: Översiktsbild över det nuvarande systemet. De rektangulära blocken är hårdvarukomponenter, de rundade blocken mjukvarukomponenter och de orangefärgade blocken mjukvaruprocesser.



Figur A.2: Översiktsbild över det nya systemet. De rektangulära blocken är hårdvarukomponenter, de rundade blocken mjukvarukomponenter och de orangefärgade blocken mjukvaruprocesser.

Litteratur

[1] *ADS Basics*. URL: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_ads_intro/index.html&id= (hämtad 2024-01-25).

[2] Karl Björk. *Formler Och Tabeller För Mekanisk Konstruktion*. 8. uppl. Spånga: Karl Björks förlag HB, 2017.

[3] *CAN in Automation (CiA): PDO Protocol*. URL: https://www.can-cia.org/can-knowledge/canopen/pdo-protocol/ (hämtad 2024-04-10).

[4] *CompactRIO Systems (cRIO)*. URL: https://www.ni.com/en/shop/compactrio.html (hämtad 2024-06-05).

[5] *Dx - Universal telemetry for up to 24 channels*. imc Test & Measurement GmbH. URL: https://www.imc-tm.com/products/sensor-solutions/telemetry/dx (hämtad 2024-06-05).

[6] Beckhoff Automation GmbH & Co KG Germany. *EtherCAT*. Beckhoff Automation - Ethercat. URL: https://www.beckhoff.com/sv-se/products/i-o/ethercat/ (hämtad 2024-06-05).

[7] Beckhoff Automation GmbH & Co KG Germany. *Industrial PC - IPCs for any application*. Beckhoff Automation - IPC. URL: https://www.beckhoff.com/sv-se/products/ipc/ (hämtad 2024-06-05).

[8] *Git*. URL: https://www.git-scm.com/ (hämtad 2024-06-05).

[9] *Home - BorgWarner*. URL: https://www.borgwarner.com/home (hämtad 2024-06-05).

[10] *Robot Framework*. URL: https://robotframework.org/ (hämtad 2024-01-25).

[11] Rainer Schicker och Georg Wegener. *Measuring Torque Correctly*. HBK World. URL: https://www.hbkworld.com/en/knowledge/resource-center/articles/tips-and-tricks-torque-reference-book (hämtad 2024-04-10).

[12] *T12HP Torque Transducer: High Precision Over Entire Range — HBM*. URL: https://www.hbm.com/en/6384/t12hp-torque-transducer-with-maximum-precision/ (hämtad 2024-06-05).

[13] wendelien. *IEC 61131-3*. 24 juli 2018. URL: https://plcopen.org/iec-61131-3 (hämtad 2024-04-26).