CHALMERS

Integrering av autopilotsystem och display för ruttstyrning av passagerarfärja

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Alice Persson
Pontus Nordström

INSTITUTIONEN FÖR ELEKTROTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2026
www.chalmers.se

CHALMERS

Integrering av autopilotsystem och display för ruttstyrning av passagerarfärja

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

ALICE PERSSON

PONTUS NORDSTRÖM

Institutionen för elektroteknik

CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2026

Integrering av autopilotsystem och display för ruttstyrning av passagerarfärja

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

ALICE PERSSON

PONTUS NORDSTRÖM

Institutionen för elektronikteknik

Chalmers tekniska högskola

FÖRORD

Vi vill börja med att tacka företaget Cstrider som gav oss möjligheten att genomföra ett examensarbete. Det har varit en spännande och lärorik process som gett oss inblick i arbetslivet och värdefull erfarenhet inom systemintegration.

Vi vill tacka Tobias Husberg, grundaren av företaget, samt handledaren Sebastian Pelto-Piri, för vägledning och stöd under arbetets gång.

Vi vill även tacka vår handledare och examinator från Chalmers, Veronica Olesen, för värdefull återkoppling och utformning av denna rapport.

Göteborg 2026
Alice Persson och Pontus Nordström

ABSTRACT

This degree project has been carried out in collaboration with the company Cstrider AB to investigate the possibility of integrating a display with an autopilot system. The report presents the methodology used in the work as well as the results achieved.

Cstrider AB is a company that aims to make it easier to commute over water and minimize the environmental impact by using small electrically powered passenger ferries. The work focuses on investigating the possibility of integrating a display with an autopilot system to reduce the number of displays in the ferry, as well as providing suggestions on how the integration should be implemented.

The project developed a user-friendly interface for a display adapted for controlling an autopilot system. The interface enables an operator to select, start and cancel routes, as well as perform course corrections during the ferry's journey. The communication between the display and the autopilot system is carried out via CAN using the communication standard NMEA 2000.

The work resulted in a working structure for receiving messages via CAN, which enabled navigation information to be read, such as the ferry's current heading. The transmission of messages via CAN was investigated but could not be verified and therefore resulted in a proposal for how it could be done.

SAMMANFATTNING

Detta examensarbete har genomförts i samarbete med företaget Cstrider AB för att undersöka möjligheten att integrera en display med ett autopilotsystem. Rapporten redovisar den metodik som använts i arbetet samt de resultat som uppnåtts.

Cstrider AB är ett företag som vill göra det lättare att pendla över vattnet och minimera klimatpåverkan genom att använda mindre eldrivna passagerarfärjor. Arbetet fokuserar på att undersöka möjligheten att integrera en display med ett autopilotsystem för att reducera antalet skärmar i färjan, samt ge förslag på hur en integration bör kunna genomföras.

Under arbetet utvecklades ett användarvänligt användargränssnitt för en display som är anpassat för styrning av ett autopilotsystem. Gränssnittet gör det möjligt för operatören att välja, starta och avbryta rutter, samt utföra kurskorrigeringar under färjans färd. Kommunikationen mellan displayen och autopilotsystemet sker via CAN med kommunikationsstandarden NMEA 2000.

Arbetet resulterade i en fungerande struktur för mottagning av meddelanden via CAN som möjliggjorde avläsning av navigationsinformation, exempelvis färjans aktuella kurs. Sändning av meddelanden via CAN undersöktes men kunde inte verifieras utan resulterade i ett förslag på hur det kan genomföras.

Sökord:
Integrering, systemintegration, CAN, NMEA 2000, autopilotsystem, Raymarine, display, ifm, passagerartransport

Innehållsförteckning

1 Inledning

Vattenburen transport har under lång tid varit en viktig del av infrastrukturen och är ett effektivt sätt att transportera gods. För passagerartransport är det inte lika effektivt. Enligt Naturvårdsverket kan utsläppen per passagerarkilometer från färjor vara 3-8 gånger högre jämfört med en fossilbil [1], beroende på faktorer som fartygstyp och hastighet. Detta tyder på att det finns potential att minska utsläppen inom vattenburen passagerartransport, bland annat genom elektrifiering. I takt med utvecklingen av mer hållbara, effektiva och miljövänliga passagerartransporter sätts ständigt nya krav på tillhörande teknik som navigationssystem, autopiloter och andra automatiserade lösningar. För att fartyg och färjor ska kunna förlita sig på den tekniken krävs system som är både säkra och enkla att hantera. I moderna båtar är navigationssystem, autopiloter och andra automatiserade lösningar ofta separata enheter med egna displayer. Att hantera flera enheter kan försvåra för bland annat styrning och övervakning under båtens färd, vilket kan påverka både säkerhet och arbetsmiljö. Detta examensarbete undersöker om olika enheter kan integreras, samt presenterar ett förslag på hur en sådan integrering kan genomföras för att förenkla ruttstyrning av en färja och bidra till en effektivare drift för framtida elektrifierade passagerarfärjor.

1.1 Bakgrund

Cstrider är ett företag som vill göra det enklare och mer lättillgängligt för människor att pendla över vatten med mindre klimatpåverkan. Företaget utvecklar små, eldrivna passagerarfärjor med automatiserad drift och tillhörande tekniska system, med mål att modernisera den vattenburna passagerartransporten [2]. De vill att deras passagerarfärja ska kunna användas i urbana miljöer, landsbygdsmiljöer och andra områden där vattenvägar kan utgöra ett alternativ till landbaserad kollektivtrafik. Deras lösning möjliggör bland annat för ett flexibelt resande vid behov.

Med mer tekniskt styrda färjor kommer allt större och komplexa kontrollpaneler med många skärmar. Cstrider vill i sina passagerarfärjor minimera antalet skärmar genom att integrera samtliga system i ett gemensamt användargränssnitt, där ett av systemen innefattar färjans autopilot. En integrering skulle bidra till minskad komplexitet samt enklare styrning för operatören.

1.2 Syfte

Syftet med arbetet är att undersöka om en ifm-display kan integreras med ett Raymarine autopilotsystem samt presentera förslag på hur en integrering kan genomföras för att reducera antalet skärmar i färjan.

1.3 Mål

Målet med integreringen är att ett användarvänligt användargränssnitt ska utvecklas på en ifm-display där en operatör enkelt kan välja, starta och avbryta färjans rutt. Utöver start och stopp ska operatören kunna utföra gradvisa kurskorrigeringar för att bibehålla färjans kurs vid yttre störningar. Utöver utvecklingen av användargränssnittet ska ett förslag ges på hur integreringen av en ifm-display och autopilotsystem kan genomföras.

1.4 Avgränsningar

För att konkretisera arbetet och säkerställa att det hålls inom given tidsram har avgränsningar fastställts. För det första kommer arbetet undersöka integrering mellan de tilldelade enheterna Raymarine-autopilot och en ifm-display, det vill säga att ingen utveckling eller modifiering av den befintliga autopiloten kommer att göras. Vidare kommer arbetet inte innefatta någon framtagning av nya rutter eller ruttinformation, utan arbetet kommer endast hantera kommunikation och synliggöring av ruttinformation. Utvecklingen av användargränssnittet kommer formas av befintliga system i färjan för att underlätta för operatör.

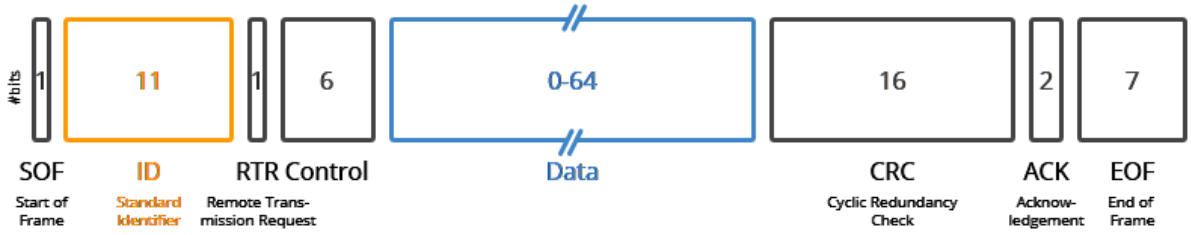
2 Teoretisk referensram

Under detta avsnitt beskrivs de hårdvaror och kommunikationsstandarder som används för att undersöka om en display och ett autopilotssystem kan integreras. Den hårdvara och kommunikationsstandard som kommer redogöras är: Controller Area Network (CAN-buss), ifm-display och Raymarine-autopilot.

2.1 Controller Area Network (CAN-buss)

Controller Area Network (CAN-buss) utvecklades av Robert Bosch år 1986 och fungerar som en delad kommunikationsledning, där olika elektroniska komponenter kommunicerar med varandra [3]. Kommunikationssystemet klassas som ett digitalt nätverk i bland annat moderna fordon och marina nätverk och möjliggör för elektroniska styrenheter (Electronic Control Unit, ECU) att kommunicera med varandra på ett säkert och effektivt sätt. En ECU är en inbyggd dator som ansvarar för att styra och övervaka en specifik funktion i ett system, exempelvis en motors drift i ett fartyg. Fortsatt ger detta möjlighet för en effektiv kommunikation mellan exempelvis en båts motor- och navigationssystem. Varje ECU på CAN-bussen kan förbereda, skicka och ta emot information, vilket skapar ett kommunikationssystem som inte kräver en central styrenhet.

De elektroniska styrenheterna kommunicerar via den gemensamma ledningen, CAN-buss, genom att skicka CAN-ramar [3]. En CAN-ram består av åtta delar. Se figur 1 nedan för uppbyggnaden av en standard CAN-ram.



Figur 1. Standarduppbyggnaden för en CAN-ram med 11-bitars identifierare. [3]. Återgiven med tillstånd.

De markerade delarna ID och Data är de mest väsentliga för hantering av meddelanden. ID:t anger vilket meddelande CAN-ramen representerar och vilken prioritet som meddelandet har, där lågt ID-nummer har högre prioritet. Data-fältet innehåller meddelandets värde och är den

information ECU:erna skickar till varandra. Utöver en standard CAN-ram med 11-bitars identifierare finns en utökad CAN-ram som använder en 29-bitars identifierare där skillnaden är längden på ID-numret. En 29-bitars identifierare möjliggör mer detaljerad meddelandeidentifiering.

CSS Electronics beskriver att en CAN-buss är en tvinnad tvåtrådsbuss bestående av CAN hög och CAN låg, två signalledningar som tillsammans används för att överföra data [3]. För att överföra data på CAN-bussen används två tillstånd, dominant och recessivt, som bestäms av potentialskillnaden mellan signalerna CAN-hög och CAN-låg [4]. I det dominanta tillståndet är potentialskillnaden mellan CAN-hög och CAN-låg ungefär 2V medan i det recessiva tillståndet är potentialskillnaden 0V. Informationen tolkas därefter baserat på potentialskillnaden vilket motverkar elektroniska störningar. Genom att överföra data på detta sätt är CAN-bussen lämplig för miljöer med mycket elektroniska komponenter där driftsäkerhet och snabb kommunikation är viktig, såsom fordon och marina nätverk.

Enligt CSS Electronics är CAN-buss standarden inom många fordon, maskiner och mycket mer, på grund av flera viktiga fördelar [3]. För det första erbjuder CAN-bussen en ingångspunkt för åtkomst och kommunikation till hela nätverket, vilket förenklar diagnostik utan att behöva samla in data från varje enhet individuellt. Ytterligare prioriteras CAN-ramar efter ID så att den data med lägst prioritet får omedelbar åtkomst till bussen. Fortsatt är systemet enligt CSS Electronics robust mot olika typer av störningar.

2.2 Raymarine-autopilot

Raymarine utvecklar och tar fram färdiga autopilotsystem och har varit delaktiga inom marin elektronik för fritidsbåtar och annan kommersiell marin i mer än 80 år [5]. Ett autopilotsystem är ett elektroniskt system som styr en båt för att ge en stabil kurs och enkel styrning under båtens färd [6].

Raymarines autopilotsystem bygger på komponenterna: kurssensor, styrenhet, drivenhet och styrhuvud [7]. Kurssensorn är hjärnan i systemet och matar styrenheten med information om kurshållning. Styrenheten strömsätter autopilotens kurssensor och styrhuvud, samt ger styrkommandon till drivenheten. För att hålla färjan på rätt kurs samverkar drivenheten med

fartygets styrsystem. Styrhuvudet bygger på en display utrustad med styrande knappar och är användargränssnitten för autopilotsystemen.

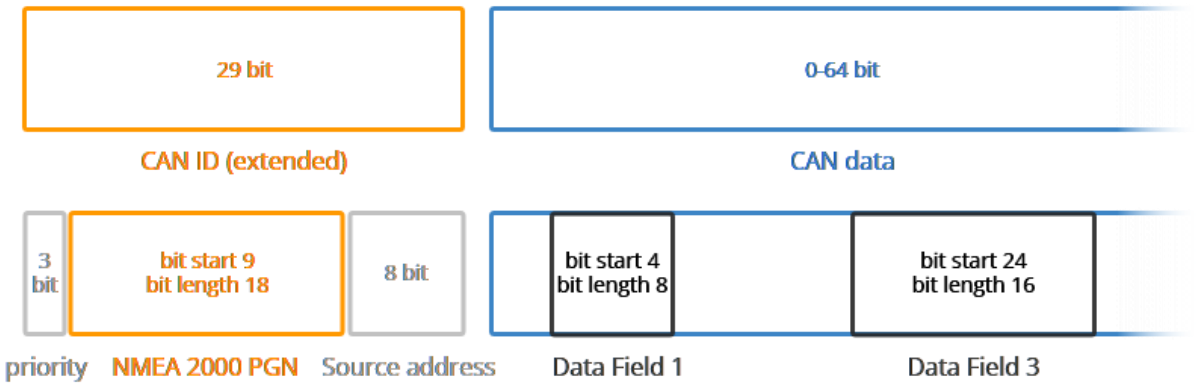
Utöver de komponenter som autopilotsystemet bygger på behövs en GNSS-mottagare (Global Navigation Satellite System) som ger autopiloten positionsdata, vilket används för att följa rutter och beräkna den optimala kursen [8].

2.2.1 NMEA 2000

Raymarine autopilotsystem använder NMEA 2000, en kommunikationsstandard som utvecklas och underhålls av organisationen NMEA. Organisationen grundades 1975 och samlar tillverkare, distributörer och återförsäljare inom marin elektronik. De arbetar för att olika produkter ska fungera tillsammans och där kommunikationsstandarden ska möjliggöra för anslutningar av elektroniska enheter på båtar [9], [10].

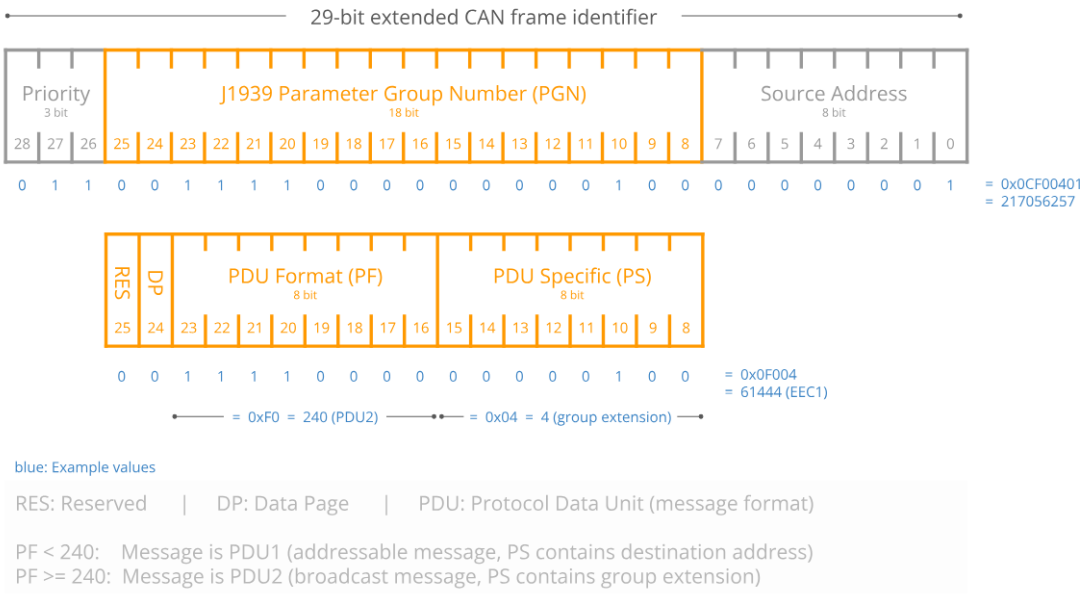
NMEA 2000 är baserad på CAN och kan på samma sätt förbereda, sända och ta emot information utan en central styrenhet [10]. Det är uppbyggt på samma fysiska principer som CAN med en tvinnad tvåtrådsbuss med signalledningarna CAN hög och CAN låg, robust kommunikation och tål brus och störningar. Utöver likheterna med CAN förklarar CSS Electronics att NMEA 2000 är utformad för att via samma nätverk kunna kopplas samman med andra enheter från andra tillverkare och fortfarande kunna utbyta data. Det enda som krävs är att nätverkets baudrate behöver vara 250 kbaud, vilket innebär att alla enheter på nätverket behöver ha samma hastighet för att utbyta data [10]. Fortsatt beskrivs det att NMEA 2000 kan stödja upp till 50 enheter, överföra data över längre avstånd och möjliggöra för äldre enheter med annan typ av NMEA-standard att integreras till befintligt NMEA 2000-nätverk [10].

NMEA 2000 använder den utökade CAN-ramen med en 29-bitars identifierare för att utbyta data [10]. Figur 2 visar hur CAN-ramen är uppbyggd för ett NMEA 2000-meddelande.



Figur 2. CAN ram för ett NMEA 2000-meddelande. [10]. Återgiven med tillstånd.

Ett NMEA 2000 meddelande består av två huvuddelar, CAN ID och CAN data, vilket illustreras i den övre delen av figur 2. CAN ID:t består i sin tur av tre delar: prioritet, NMEA 2000 PGN (Parameter Group Number) och källadress. PGN-numret är ett 18-bitars tal och informerar om vilket meddelande som CAN-ramen innehåller samt hur meddelandets information ska tolkas. De sista 8 bitarna i ID:t är källadressen som beskriver vilken enhet på CAN-bussen som skickat meddelandet. PGN-numret kan i sin tur delas upp i fyra delar, se figur 3.



Figur 3. Utökad 29-bitars CAN-ram identifierare [11]. Återgiven med tillstånd.

De två första bitarna i PGN-numret består av en reserverad bit följt av DP som står för Data Page [11]. Nästa del består av 8 bitar och benämns PDU Format (PF) i figuren. PDU står för Protocol Data Unit och beskriver vilken typ meddelandet är av, om det är adresserat eller inte.

Vid adresserat meddelande innehåller PF ett värde mindre än 240. Resterande 8 bitar av PGN-numret benämns PDU Specific (PS) och innehåller antingen en destinationsadress eller en gruppförlängning beroende på om det är ett adresserat meddelande eller inte.

Datafältet utgör den andra delen av CAN-ramen, se figur 2, och innehåller meddelandets information [10]. Det kan rymma upp till 64-bitar (8 bytes), vilket begränsar möjligheten att överföra större datamängder. Lösningen är att skicka CAN-meddelandet med ’Fast Packet’. Fast Packet ger möjlighet att skicka meddelanden som innehåller fler än 8 bytes genom att dela upp meddelandet i flera CAN-ramar. För att kunna sätta ihop meddelandet i rätt ordning vid mottagning behövs två typer av räknare i datafältet. En sekvensräknare som identifierar vilka ramar som innehåller samma meddelande och en räknare som håller koll på ordningen av ramarna i sekvensen. Dessutom innehåller den första ramen i varje sekvens den totala längden av datainnehållet i den andra byten. På detta sätt kan meddelanden av större storlek enkelt skickas mellan enheter.

2.2.1.1 NMEA 2000 Standard

NMEA 2000 har en officiell databas för bland annat PGN-beskrivningar, skalningsfaktorer och fältdefinitioner men är upphovsrättskyddad och begränsad till betalande medlemmar [12]. På grund av den slutna standarden startade nederländaren Kees Verruijt ett projekt år 2008 där han tog fram en öppen dokumentation för NMEA 2000 som innehåller PGN-dokumentation [13]. Framtagningen av dokumentationen gjordes genom ’reverse engineering’ och publikt tillgänglig information från NMEA.

2.3 ifm-display (CR1140)

I arbetet används en display och styrenhet från ifm electronic, ett företag som utvecklar hårdvara och mjukvara för styr- och övervakningssystem. I figur 4 nedan visas modellen CR1140 som används i arbetet. Det är en programmerbar grafisk display för styrning av mobila maskiner, konstruerad för att tåla temperaturväxlingar, vibrationer och kontinuerliga stötar [14]. Displayen är utrustad med en 4,3-tums status-LED-display, sex funktionsknappar och en navigeringsknapp med central bekräftelseknapp. På baksidan är enheten utrustad med två kommunikationsgränssnitt; ett CAN-gränssnitt samt ett Ethernet-gränssnitt, som visas i figur 5 nedan. Dessa gränssnitt möjliggör kommunikation mellan andra styrenheter samt

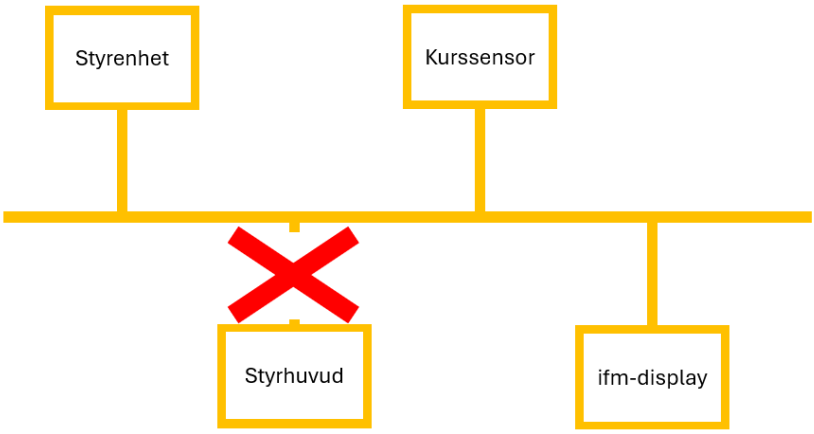
externa system. CR1140 är fritt programmerbar med programmeringsverktyget CODESYS V3.5 [14].

Figur 4. Hemskärmen för CR1140-displayen. (Författarens egen bild).

Figur 5. Baksidan av CR1140-displayen. (Författarens egen bild).

3 Metod och systemöversikt

Det tilldelade systemet som hanterades i detta arbete bestod av autopilotsystemet EV-200 Sail Pilot från Raymarine och en separat ifm-display av modellen CR1140. Samtliga enheter i det tilldelade systemet kommunicerade via ett gemensamt CAN-nätverk vilket illustreras i figur 6. I systemet ingick ursprungligen ett styrhuvud men ersattes i arbetet med ifm-displayen, markerat med ett kryss i figur 6. Systemets uppbyggnad möjliggjorde för informationsbyte via ett gemensamt nätverk och skapade grunden för integration mellan enheterna.



Figur 6. Systemöversikt över CAN-nätverket. (Författarens egen bild).

Metoden för detta arbete baserades på en iterativ process, vilket innebar att samtliga delar av processen genomfördes med löpande utvärdering och omarbetning. Erfarenheter och resultat från tidigare moment och test användes kontinuerligt för att utveckla och förbättra arbetet. Nedan presenteras de två faserna i arbetet: design av användargränssnitt samt CAN-kommunikation och deras genomförande.

3.1 Design av användargränssnitt

I utvecklingsmiljön CODESYS V3.5 SP19 Patch 7 med mjukvarupaketet ecomatDisplay 4.3” (OS V2.0.0.11; Package V1.4.1.0) från ifm för CR1140-displayen utvecklades ett användargränssnitt. I syfte att skapa förståelse för plattformens möjligheter och begränsningar genomfördes en undersökning av vilka designlösningar som fanns tillgängliga i CODESYS visualiseringsverktyg. Informationen baserades huvudsakligen på tillgänglig dokumentation, CODESYS egna guider och genom att studera de grafiska komponenter och funktioner som erbjöds i utvecklingsmiljön.

För att säkerställa ett användargränssnitt med hög igenkänningsfaktor togs tidigare framtagna gränssnitt i beaktning. Utifrån dessa utvecklades det nya gränssnittet med liknande struktur och design anpassat efter vad CODESYS visualiseringsverktyg erbjöd.

3.2 CAN-kommunikation

Standardbibliotek med funktioner för CAN-kommunikation från ifm användes för att samla information om hur kommunikationen kunde hanteras. Genom manualer och tester av det befintliga systemet skapades förståelse för vilka PGN:er som var relevanta för en möjlig integrering. Testerna bestod av att det tilldelade autopilotsystemet EV-200 Sail Pilot kördes tillsammans med en GNSS-mottagare där enkla rutter fanns tillgängliga. Under testerna sammanställdes de PGN:er som skickades från GNSS-mottagaren och styrhuvudet och låg sedan till grund för arbetet. Med förståelse för datastrukturen hos relevanta PGN:er genom Kees Verruijts dokumentation [13] implementerades kod i CODESYS för att läsa och tolka CAN-meddelanden från autopilotsystemet. Därefter undersöktes möjligheten att från ifm-displayen skriva information tillbaka till autopilotsystemet genom sändning av CAN-meddelanden.

4 Resultat

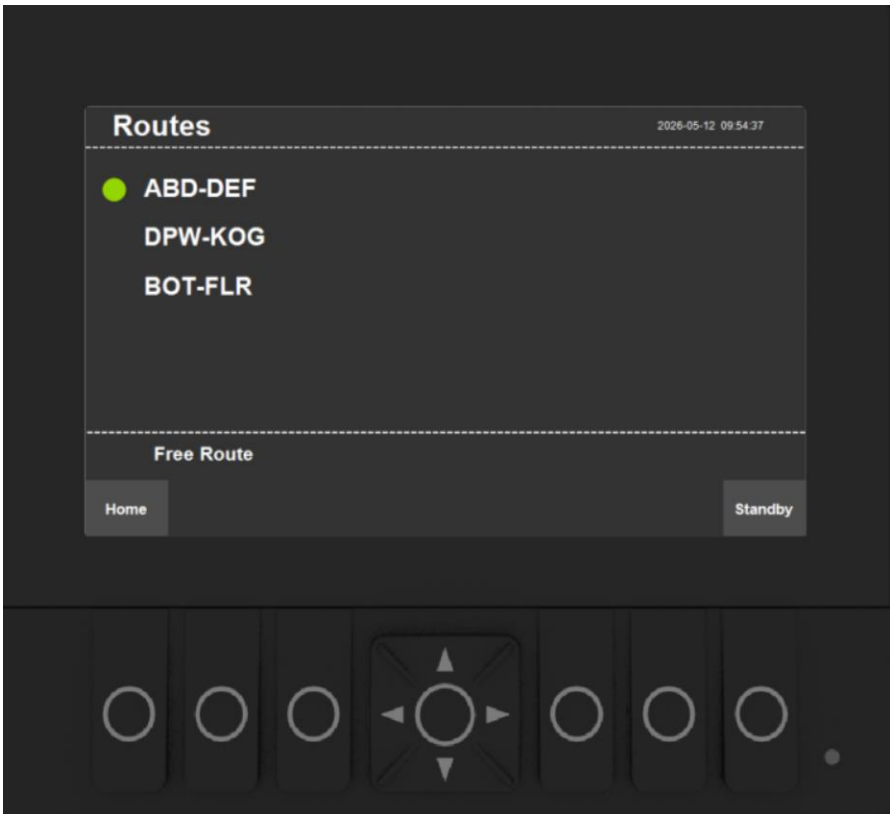
Nedan presenteras det resultat som erhållits från arbetet.

4.1 Utformning av displayen

Användargränssnittet för ifm-displayen har skapats i utvecklingsmiljön CODESYS med en hög igenkänningsfaktor från tidigare system i syfte att det ska vara enkelt för operatör att förstå och använda funktioner som presenteras. Användargränssnittet är uppbyggt i CODESYS visualiseringsverktyg där olika vyer har skapats och formats för att presentera information kopplat till en specifik användning. Navigationen mellan vyerna är programmerad och styrs av operatörens val via displayens olika knappar, se figur 4. Detta innebär att olika vyer aktiveras beroende på val av knapptryck och säkerställer att rätt information visas vid rätt tillfälle.

4.1.1 Ruttalternativ

I vyn Routes, som visas i figur 7, väljer operatören vilken rutt som ska köras eller alternativet Free Route. I listan synliggörs rutternas start- och slutdestination. När en rutt är aktiv, vilket innebär att färjan navigerar efter den, indikeras det genom att en grön prick tänds till vänster om rutten. Free Route innebär att autopiloten kör efter båtens nuvarande kurs och håller denna under fortsatt färd. Längst ner på skärmen finns även två knappalternativ, Home och Standby vilket tar operatören till hemskärmen respektive standby-läget.



Figur 7. Vy för ruttalternativ. (Författarens egen bild).

4.1.2 Auto

När en rutt aktiverats i listan växlas vyn till auto-läge, se figur 8 nedan. I auto-läget synliggörs relevant navigeringsinformation för operatören.

Figur 8. Vy för autopilot-läget. (Författarens egen bild).

I den gula rutan ovanför bågen visas båtens nuvarande riktning, vilken alltid är centrerad ovanför bågen för att operatören enkelt ska se riktningen färjan färdas i. Värdet 158˚ under bågen är riktningen till nästa waypoint som autopiloten navigerar mot. Vid behov kan kurskorrigeringar göras genom att öka eller minska detta värde med hjälp av displayens knappar. Det finns två typer av korrigeringar, en mindre som ökar eller minskar värdet med en grad och en större som korrigerar med 10 grader. Differensen mellan de två riktningarna, båtens riktning och waypointens riktning, synliggörs i bågen som är centrerad på skärmen. Delen av bågen som är färgad gul indikerar hur långt ifrån den utsatta kursen båten färdas i. Längst ner på skärmen under waypointens riktning synliggörs vilken rutt som är aktiv, med information om ruttens start- och slutdestination. Detta fält alterneras var femte sekund mellan att visa den aktiva rutten och distans till waypoint (DTW).

Uppe i högra hörnet finns tre fält benämnda COG, SOG och STW. COG innebär kurs över grund vilket är den riktningen som båten färdas i relaterat till marken. SOG är motsvarande för fart, dvs fart över grund och visar vilken hastighet båten färdas i relaterat till marken. Sista fältet, STW, visar vilken hastighet båten har genom vattnet.

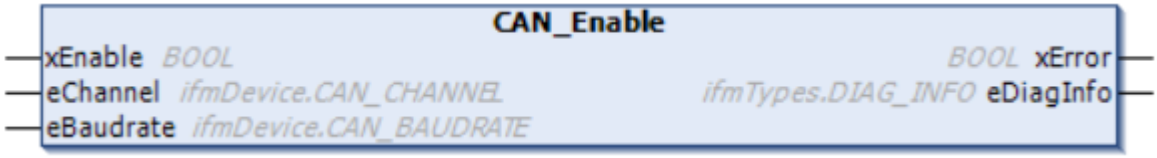
Knappen för Mode, som finns längst ner till vänster på skärmen tar operatören tillbaka till listan med ruttalternativ. Standby-knappen har samma funktion som tidigare.

4.2 Integrering av display och autopilot

Från ifm användes standardbiblioteket ifmRawCAN.library för att hantera CAN-kommunikationen i CODESYS. Nedan presenteras och förklaras de funktionsblock från ifm:s standardbibliotek vi kom fram till var nödvändiga för att möjliggöra integrering, följt av strukturen för mottagning av CAN-meddelande och förslag till struktur för sändning av CAN-meddelanden.

4.2.1 Funktionsblocket som aktiverar CAN-gränssnittet

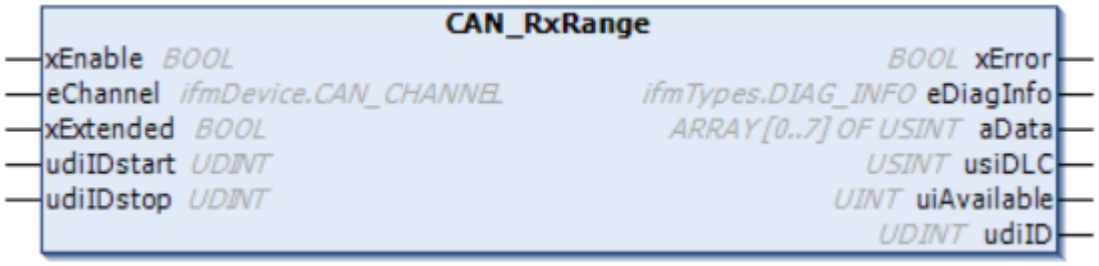
Funktionsblocket CAN_Enable används för att aktivera displayens CAN-gränssnitt och är det första steget för att möjliggöra för displayens kommunikation via CAN-bussen. Funktionsblocket är uppbyggd med ingångsparametrarna xEnable, eChannel och eBaudrate, se figur 9. Signalen xEnable aktiverar funktionsblocket, eChannel anger vilken CAN-kanal på displayen som ska användas och eBaudrate anger vilken hastighet data överförs på nätverket. Det är viktigt att alla enheter på CAN-bussen använder samma baudrate för att kunna tolka varandras meddelanden. Utan CAN_Enable aktiverat kan meddelanden varken tas emot eller skickas på nätverket. Utgångarna xError och eDiagInfo till höger i figur 9 indikerar om ett fel uppstått under exekvering samt ger information om felets orsak.



Figur 9. Funktionsblock CAN_Enable i CODESYS. [15]. Återgiven med tillstånd.

4.2.2 Funktionsblocket för mottagning av CAN-meddelanden

För mottagning av CAN-meddelanden används funktionsblocket CAN_RxRange. Funktionsblocket har 5 ingångsparametrar och 6 utgångar, se figur 10.



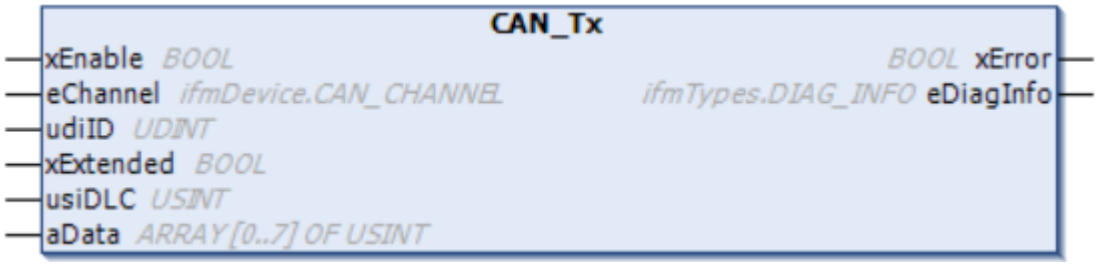
Figur 10. Funktionsblock CAN_RxRange i CODESYS [15]. Återgiven med tillstånd.

Till vänster i funktionsblocket är ingångarna placerade, xEnable aktiverar funktionsblocket och eChannel anger vilken CAN-kanal som används. Ingången xExtended används för att bestämma vilken typ av CAN-ram som ska tas emot, standard eller utökad med 29-bitars identifierare. Det önskade intervallet av CANID sätts med hjälp av udiIDstart och udiIDstop där startvärdet måste vara mindre än slutvärdet.

Funktionsblockets utgångar är placerade till höger i blocket, aData, usiDLC, uiAvailable och udiID är fyra av dem och ger information om det mottagna meddelandet. CAN-ramens identifierare som beskrivs närmare i avsnitt 2.2.1 placeras i udiID och CAN-ramens datainnehåll placeras i aData. Utgången usiDLC, som står för Data Length Count, anger hur många bytes datafältet i CAN-ramen innehåller och uiAvailable anger hur många meddelanden som mottagits sedan det sista funktionsanropet. xError och eDiaginfo indikerar som likt beskrivet i CAN_Enable om ett fel uppstått och felets orsak.

4.2.3 Funktionsblocket för sändning av CAN-meddelanden

Funktionsblocket CAN_Tx används för att sända meddelanden ut på CAN-nätverket och är uppbyggt med ingångsparametrarna xEnable, eChannel, udiID, xExtended, usiDLC och aData, se figur 11. Likt CAN_Enable i avsnitt 4.2.1 aktiverar xEnable funktionsblocket och eChannel anger vilken CAN-kanal som används. Resterande parametrar udiID, xExtended, usiDLC och aData beskrivs närmare under figur 10 i avsnitt 4.2.2. Vid aktivering av CAN_Tx skickas en CAN-ram med parametrar ut på nätverket, där anslutna enheter kan ta emot och tolka meddelandet.



Figur 11. Funktionsblock CAN_Tx i CODESYS. [15]. Återgiven med tillstånd.

4.2.4 Struktur för mottagning av CAN-meddelanden

För mottagning av CAN-meddelanden från autopilotsystemet utvecklades en kod uppdelad i fem block, se bilaga A. Det första blocket aktiverar displayens CAN-gränssnitt via funktionsblocket CAN_Enable, där tillhörande parametrar är satta till önskad kanal samt med en baudrate på 250 kbaud. Det andra blocket är uppbyggt med funktionsblocket CAN_RxRange för att ta emot CAN-ramar. Koden säkerställer via ingångsparametrarna udiIDstart och udiIDstop att funktionsblocket kan ta emot CAN-ramar från samtliga ID-nummer som skickas på nätverket. I det tredje blocket tolkas 29-bitars identifieraren från den mottagna CAN-ramen och bryts ned i sina beståndsdelar; prioritet, PGN och källadress. Ihop med PGN:ens olika delar; DP, PF och PS, som beskrivs i tidigare avsnitt 2.2.1, identifieras sedan PGN-numret för det mottagna meddelandet. Det fjärde blocket hanterar Fast Packet som används för meddelanden större än 8 bytes. Då ser koden till att meddelanden läses i rätt ordning för att senare kunna tolkas. I slutet av koden och i det sista blocket sorteras de mottagna meddelandena efter dess PGN-nummer för tolkning av meddelandets datainnehåll. Exempelvis PGN 127250 som innehåller vesselheading och berättar färjans aktuella kurs.

4.2.5 Test av befintligt autopilotsystem

Vid körning av det befintliga autopilotsystemet ihop med GNSS-mottagaren sammanställdes de PGN:er som skickades mellan autopilotens styrhuvud och GNSS-mottagaren. Den sammanställda listan med PGN:er finns att tillgå i bilaga B. Tillsammans med tillgänglig PGN dokumentation är slutsatsen att PGN 126993 (Heartbeat) och PGN 60928 (ISO Address Claim) är väsentliga för sändning av meddelanden på nätverket. ISO Address Claim används för att bestämma en nätverksadress för enheten, vilken inte får vara samma som någon annan enhet. När andra enheter försöker ta samma enhetsadress används ISO Address Claim för att svara enheten. I svaret skickas enhetsinformation ut som används för att avgöra vilken enhet

som får behålla adressen. Heartbeat används för att bekräfta att enheten fortfarande finns på nätverket. Vidare slutsats är att PGN 129284 (Navigation Data) behövs för att autopiloten ska kunna följa rutter samt att PGN 126720 (Garmin AHRS ATT/Seatalk 1) troligtvis kan användas för att styra autopilotens olika lägen.

4.2.6 Förslag till struktur för sändning av CAN-meddelanden

Under arbetet har ingen fullständig struktur för sändning av samtliga meddelanden utvecklats. Inte heller har det kunnat verifieras att enstaka meddelanden sänds ut korrekt på nätverket. Det som tagits fram utgör i stället förslag på hur sändning kan tänkas fungera och vad som behöver hanteras. Som tidigare nämnt används funktionsblocket CAN_Tx för att sända ut meddelanden på nätverket. För varje meddelande behöver ingångarna udiID och aData innehålla rätt information för att mottagande enhet ska tolka meddelandet korrekt. Därför behövs en struktur som kan bygga upp CAN-ramen som ska sändas med korrekt information. Beroende på meddelandets storlek måste hantering av Fast Packet finnas med i strukturen, där datainnehållet delas upp i flera CAN-ramar.

En del som gett positivt resultat är hur meddelandets 29-bitars identifierare byggs upp för sändning av CAN-meddelanden. I figur 12 redogörs strukturen för uppbyggnaden av identifieraren.

```
FUNCTION BuildCANID : DWORD
VAR_INPUT
    Priority    : BYTE;
    PGN         : DWORD;
    Destination : BYTE;
    SourceAddr  : BYTE;
END_VAR
VAR
    PF          : BYTE;
    PS          : BYTE;
    DP          : BYTE;
END_VAR
```

```
(* Extrahera fälten ur PGN-numret *)
DP := DWORD_TO_BYTE(SHR(PGN, 16) AND 16#01);
PF := DWORD_TO_BYTE(SHR(PGN, 8) AND 16#FF);
PS := DWORD_TO_BYTE(PGN AND 16#FF);

(* OM PF < 240 (16#F0) är meddelandet riktat, PS fältet används då som destination-adress, inte som del av PGN *)
IF PF < 16#F0 THEN
    PS := Destination;
END_IF

(* Sätt ihop 29-bitars identifierare *)
BuildCANID := SHL((BYTE_TO_DWORD(Priority) AND 16#07), 26)
              OR SHL((BYTE_TO_DWORD(DP) AND 16#01), 24)
              OR SHL(BYTE_TO_DWORD(PF), 16)
              OR SHL(BYTE_TO_DWORD(PS), 8)
              OR BYTE_TO_DWORD(SourceAddr);
```

Figur 12. Funktion för hopsättning av 29-bitars identifierare. (Författarens egen bild).

Funktionen extraherar först de ingående delarna av meddelandets PGN-nummer. Om meddelandet har en bestämd destinationsadress tilldelas PS-fältet meddelandets destinationsadress. Därefter byggs identifieraren upp med de ingående delarna prioritet, DP, PF, PS och källadress. Funktionen kan användas för att ge ingången udiID till funktionsblocket CAN_Tx rätt värde för varje meddelande som ska skickas.

Vad som återstår för en fullständig struktur för sändning är att ta fram funktioner som för varje meddelande bygger upp aData och skickar ut CAN-ramen på nätverket. Beroende på meddelandets storlek och om det behöver skickas med fast packet behöver datainnehållet hanteras olika. För ett meddelande som enbart består av 8 bytes och ryms i en CAN-ram kan meddelandets datainnehåll direkt placeras i aData. För meddelanden som inte ryms i en CAN-ram behöver meddelandet delas upp och skickas med Fast Packet. Förslagsvis så skapas en funktion som hanterar ’Fast Packet’ meddelanden. Funktionen behöver hantera de två räknarna som behövs, meddelandets längd samt dela upp meddelandets datainnehåll och fylla CAN-ramarna med rätt värden.

5 Diskussion

Att integrera en Raymarine-autopilot och en ifm-display för att reducera antalet styrande skärmar visade sig vara mer utmanande än vad som förutsågs till en början. Arbetet genomfördes till stora delar genom en iterativ arbetsprocess eftersom det under arbetets gång dök upp flera utmaningar kopplade till integration, dokumentation och befintliga system. Det var många antaganden kring de tre tidigare nämnda utmaningarna som behövde verifieras och ibland justeras. Samtidigt identifierades det under arbetets gång flera möjligheter och idéer som till stor del låg utanför ramen för arbetet men som kan utgöra grund för fortsatt arbete.

I arbetet valdes kurskorrigeringarna på ±1° och ±10° utifrån önskemål och bedömdes vara lämpliga för en färja. Den mindre korrigeringen är till för finjusteringar och den större korrigeringen för snabbare kursändringar. I det tidigare systemet skiftade fältet för aktiv rutt och DTW var tionde sekund, vilket bedömdes vara för långt och kortades därför ned till fem sekunder. Detta för att under färd ge operatören snabbare tillgång till relevant information. Ytterligare relevant information för operatören att ta del av ansågs vara SOG och STW, därför inkluderades dessa tillsammans med COG i vyn Auto, se figur 8. I Auto-vyn och Routes-vyn finns även knappar för läge Standby och Free route utsatta, men saknar i detta arbete funktionalitet och möjliggör för fortsatt arbete. Standby och Free Route ska förslagsvis ha egna vyer och efterlikna Auto-vyn med tillhörande båge, fälten under bågen och fälten uppe i högra hörnet.

I standardbiblioteket från ifm fanns det fyra funktionsblock som hanterar mottagning på olika sätt. Två av dem kunde enbart ta emot meddelanden från ett enda CAN-ID, vilket var begränsning för arbetet. Ett annat funktionsblock hade likt CAN_RxRange egenskaper att ta emot meddelanden från samtliga CAN-ID inom ett bestämt intervall, men hanterade enbart CAN-ramar med 29-bitars identifierare. Vid fortsatt arbete bör möjligheten att använda funktionsblocket som enbart hanterar CAN-ramar med 29-bitars identifierare testas för att minimera antalet ingångsparametrar som behöver tas hänsyn till.

Trots tester och befintlig NMEA 2000 dokumentation har inte tillräckligt med information kunnat sammanställas för att få en exakt förståelse för vilka PGN:er som behöver hanteras i ifm displayen för att möjliggöra en fullständig integration. Ytterligare information om kommunikationen mellan enheterna i Raymarine EV-200 Sail Pilot behövs, förslagsvis

genom fler tester av systemet där kommunikationen loggas med en CAN-loggare, eller från Raymarine om det finns tillgängligt. Detta kvarstår även för sändning av CAN-meddelanden som behöver genomgå fler tester för att verifiera kommunikationen innan systemet testas ombord på färjan. Även om sändning ännu inte verifierats finns god förståelse för uppbyggnad av relevanta PGN:er, och förslag hur kod för sändning funkar som företaget kan bygga vidare på. Om sändning av CAN-meddelanden lyckas återstår slutverifiering och planering för vilka funktioner som är viktiga för en fungerande integration.

Inför ett liknande arbete bör en förstudie genomföras för att ge bättre förståelse för vilka förutsättningar och begränsningar arbetet väntas ha. I detta arbete tilldelades ett autopilotsystem och en display och lämnade inget utrymme för val av enheter, men hade med en förstudie kunnat ge viktig information om vad som är möjligt och inte. Vid ett liknande arbete bör valet av enheter övervägas, där exempelvis ett autopilotsystem med mer tillgänglig dokumentation hade kunnat underlätta integreringen.

6 Slutsats

Slutsatsen från detta arbete är att en CR1140-display från ifm möjligen kan integreras med ett EV-200 Sail Pilot autopilotsystem från Raymarine vid fortsatt arbete. Ett förslag på användargränssnitt har tagits fram och fungerande struktur för mottagning av CAN-meddelanden har implementerats. Det krävs dock mer information om autopilotsystemet, mer specifikt vilka PGN-meddelanden som skickas mellan samtliga enheter samt hur datapaketet för varje meddelande ser ut. För en fullständig integration krävs även fortsatt utveckling av kod som hanterar både sändning och mottagning av PGN-meddelanden i samspel med displayens olika vyer.

Referenser

[1] ”Klimatet och transporterna”. Åtkomstdatum: 28 maj 2026. [Online]. Tillgänglig vid: https://www.naturvardsverket.se/amnesomraden/klimatomstallningen/omraden/klimatet-och-transporterna

[2] Cstrider, ”Cstrider - Innovation on Water”. Åtkomstdatum: 16 februari 2026. [Online]. Tillgänglig vid: https://www.cstrider.com/

[3] M. Falch, ”CAN Bus Explained - A Simple Intro [2025] – CSS Electronics”. Åtkomstdatum: 08 april 2026. [Online]. Tillgänglig vid: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

[4] J. Shah, ”Understanding CAN: A Beginner’s Guide to the Controller Area Network Protocol | CircuitBread”. Åtkomstdatum: 20 maj 2026. [Online]. Tillgänglig vid: https://www.circuitbread.com/tutorials/understanding-can-a-beginners-guide-to-the-controller-area-network-protocol

[5] Raymarine, ”Global Leader of Innovative Marine Electronics Technology | Raymarine”. Åtkomstdatum: 08 april 2026. [Online]. Tillgänglig vid: https://www.raymarine.com/en-us/about-raymarine

[6] Digital Skipper, ”Autopiloter - Raymarine - Tillverkare - Digital Skipper”. Åtkomstdatum: 21 maj 2026. [Online]. Tillgänglig vid: https://digital-skipper.se/tillverkare/raymarine/autopiloter

[7] Raymarine, ”Båtautopiloter | Onlineguider | Raymarine”. Åtkomstdatum: 21 maj 2026. [Online]. Tillgänglig vid: https://www.raymarine.com/sv-se/guider/onlineguider/vilka-ar-de-olika-typerna-av-autopiloter-for-batar

[8] *Evolution Autopilot Installation Instructions*, Raymarine, Storbritannien, 2024.

[9] National Marine Electronics Assosiation, ”National Marine Electronics Association (NMEA) - Marine Electronics Standards, Training & Certification - National Marine Electronics Association”. Åtkomstdatum: 24 april 2026. [Online]. Tillgänglig vid: https://www.nmea.org/

[10] M. Falch, ”NMEA 2000 Explained - A Simple Intro [2024] – CSS Electronics”. Åtkomstdatum: 20 mars 2026. [Online]. Tillgänglig vid: https://www.csselectronics.com/pages/nmea-2000-n2k-intro-tutorial#nmea-pgn

[11] M. Falch, ”J1939 Explained - A Simple Intro [2025] – CSS Electronics”. Åtkomstdatum: 13 maj 2026. [Online]. Tillgänglig vid: https://www.csselectronics.com/pages/j1939-explained-simple-intro-tutorial

[12] National Marine Electronics Assosiation, ”NMEA 2000 Standards - National Marine Electronics Association (NMEA) - Marine Electronics Standards, Training & Certification”. Åtkomstdatum: 08 maj 2026. [Online]. Tillgänglig vid: https://www.nmea.org/nmea-2000.html

[13] K. Verruijt, ”Reverse engineering the NMEA 2000 standardized and manufacturer property data”. Åtkomstdatum: 08 maj 2026. [Online]. Tillgänglig vid: https://canboat.github.io/canboat/canboat.html#ft-TIME

[14] ifm electronic, ”CR1140 - Programmerbar grafisk display för styrning av mobila maskiner - ifm”. Åtkomstdatum: 05 mars 2026. [Online]. Tillgänglig vid: https://www.ifm.com/se/sv/product/CR1140#documents

[15] ifm electronic gmbh, ”Programming manual CR1140 CR1141”.

Bilaga A – Kod för mottagning av CAN-meddelanden

```
PROGRAM CAN_Main
VAR
(* funktionsblock CAN_Enable *)
fbCAN_Enable            : ifmRCAN.CAN_Enable;
bInitDone                                   : BOOL := FALSE;
bInitError                                  : BOOL := FALSE;

(* Mottagning av CAN-meddelanden *)
fbCAN_RxRange           : ifmRCAN.CAN_RxRange;

dwRawID     : DWORD;
nDLC                    : BYTE;
aData                   : ARRAY [0..7] OF USINT;
i                       : INT;
nIdx                    : INT;
(* ID-TOLKNING *)
nPriority       : BYTE;
nDP                     : BYTE;
nPF             : BYTE;
nPS             : BYTE;
nSA             : BYTE;
dwPGN           : DWORD;

(* Fast-Packet hanterare *)
nSeqID_current          : BYTE;
nExpectedFrame          : BYTE;
nBytesReceived          : INT;
bAssembling             : BOOL;
nPayloadLen             : BYTE;
aPayload                : ARRAY[0..222] OF BYTE;
xPacketReady                                : BOOL;
tonFP_Timeout           : TON;
nWritePos               : INT;
nBytesToCopy            : INT;
nSeqID_frame                                : BYTE;
nFrameCounter           : BYTE;

(* PGN 127250 – Vesselheading *)
VesselHeadingDeg        : REAL;

END_VAR


(* Block 1: Aktivera CAN-kanalen *)
fbCAN_Enable(
        xEnable         := TRUE,
                eChannel        := 0,
```

```
            eBaudrate    := ifmDevice.CAN_BAUDRATE.KBAUD_250
);

(* Block 2: Ta emot en CAN-ram *)
FOR i := 0 TO 50 DO

        fbCAN_RxRange(
                        eChannel    := ifmDevice.CAN_CHANNEL.CHAN_0,
                        xExtended   := TRUE,
                        xEnable     := TRUE,
                        udiIDstart  := 16#00000000,
                        udiIDstop   := 16#1FFFFFFF
                );

                (* Ingen ny frame? Avvakta. *)

                IF fbCAN_RxRange.uiAvailable = 0 THEN
                        RETURN;
                END_IF

                (* Hämta frame – ID kommer från udiID-outputen på CAN_RxRange *)

                dwRawID := UDINT_TO_DWORD(fbCAN_RxRange.udiID);
                nDLC    := USINT_TO_BYTE(fbCAN_RxRange.usiDLC);
                aData := fbCAN_RxRange.aData;

(* Block 3: Tolka 29-bitars ID *)
                nPriority   := DWORD_TO_BYTE(SHR(dwRawID, 26) AND 16#07);
                nDP                     := DWORD_TO_BYTE(SHR(dwRawID, 24) AND
16#03);
                nPF                     := DWORD_TO_BYTE(SHR(dwRawID, 16) AND
16#FF);
                nPS                     := DWORD_TO_BYTE(SHR(dwRawID, 8) AND
16#FF);
                nSA                     := DWORD_TO_BYTE(dwRawID AND 16#FF);

                IF nPF < 240 THEN
                        dwPGN := SHL(BYTE_TO_DWORD(nDP), 16)
                                OR SHL(BYTE_TO_DWORD(nPF), 8);
                ELSE
                        dwPGN := SHL(BYTE_TO_DWORD(nDP), 16)
                                OR SHL(BYTE_TO_DWORD(nPF), 8)
                                OR BYTE_TO_DWORD(nPS);
                END_IF

(* Block 4: Fast-Packet hanterare *)
                nSeqID_frame            := SHR(aData[0], 5) AND 16#07;
                nFrameCounter           := aData[0] AND 16#1F;

                IF nFrameCounter = 0 THEN
```

```
                    nSeqID_Current          := nSeqID_frame;
                    nPayloadLen             := aData[1];
                    nBytesReceived          := 0;
                    nExpectedFrame          := 1;
                    bAssembling             := TRUE;

                    FOR nIdx := 0 TO 222 DO
                            aPayload[nIdx] := 0;
                    END_FOR

                    nBytesToCopy := MIN(6, BYTE_TO_INT(nPayloadLen));
                    FOR nIdx := 0 TO nBytesToCopy - 1 DO
                            aPayload[nIdx] := aData[2 + nIdx];
                    END_FOR
                    nBytesreceived := nBytesToCopy;

                    tonFP_Timeout(IN := FALSE);
                    tonFP_Timeout(IN := TRUE, PT := T#500MS);

            ELSIF bAssembling THEN
                    tonFP_Timeout(IN := TRUE, PT := T#500MS);
                    IF tonFP_Timeout.Q THEN
                            bAssembling             := FALSE;
                            tonFP_Timeout(IN := FALSE);
                            RETURN;
                    END_IF

                    IF nSeqID_frame <> nSeqID_current OR nFrameCounter <>
nExpectedFrame THEN
                            bAssembling := FALSE;
                            RETURN;
                    END_IF

                    nWritePos    := 6 + (BYTE_TO_INT(nFrameCounter) - 1) *7;
                    nBytesToCopy:= MIN(7, BYTE_TO_INT(nPayloadLen) -
nBytesReceived);

                    FOR nIdx := 0 TO nBytesToCopy - 1 DO
                            IF (nWritePos + nIdx) <= 222 THEN
                                    aPayload[nWritePos + nIdx] := aData[1
+ nIdx];
                            END_IF
                    END_FOR
                    nBytesReceived := nBytesReceived + nBytesToCopy;
                    nExpectedFrame := nExpectedFrame + 1;

                    IF nBytesReceived >= BYTE_TO_INT(nPayloadLen) THEN
                            xPacketReady            := TRUE;
                            bAssembling             := FALSE;
                            tonFP_Timeout(IN := FALSE);
```

```
                    END_IF
                END_IF

 (* Block 5: Sortera PGN *)
   CASE DWORD_TO_UDINT(dwPGN) OF
                    127250:     (* VESSEL HEADING *)
                    VesselHeadingDeg := ParsePGN127250(aData);

                    (* Övriga PGN:er fylls på här *)

   END_CASE
END_FOR
```

Bilaga B – Sammanställning av observerade PGN:er

PGN-nummer Styrhuvud	PGN beskrivning	PGN-nummer GNSS-mottagare	PGN beskrivning
126208	Group Function	129283	Cross Track Error
65384		126992	System Time
126720	Garmin AHRS ATT/Seatalk 1	126208	Group Function
126993	Heartbeat	129025	Position, Rapid Update
		65535	
		129285	Navigation-Route/WP Information
		129539	GNSS DOPs
		129547	GNSS Pseudorande Error Statistics
		129540	GNSS Stats in View
		129033	Time & Date
		129284	Navigation Data
		59904	ISO Request
		129029	GNSS Position Data
		130577	Direction Data
		130848	SeaTalk:Waypoint Information
		130918	SeaTalk: Route Information
		129044	Date
		129542	GNSS Pseudorange Noise Statistics
		129026	COG & SOG, Rapid Update
		127258	Magnetic Variation
		65288	SeaTalk: Alarm
		65311	BEP Marine: Proprietary PGN 65311
		126993	Heartbeat
		60928	ISO Address Claim
		126996	Product Information