Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Automatisering og elektronikkdesign	Vårsemesteret 2021 Åpen
Forfatter(e): Julie Tengs	
Fagansvarlig: Sveinung Tollefsen Veileder(e): Sven Ole Aase	
Tittel på bacheloroppgaven: Grensesnitt for touchpad med roterende del Engelsk tittel: Interface touchpad with rotary button	
Studiepoeng: 20	
Emneord: C++ Grensesnitt Navigasjonssystem	Sidetall: 43 + vedlegg/annet: 126 Egersund 15. mai 2021

Prosjektrapport

Grensesnitt for touchpad med roterende del

Marine Technologies LLC

En rapport presentert av
Julie Tengs

Automatisering og elektronikkdesign
Universitetet i Stavanger
Norge
7. mai 2021

Sammendrag

Hensikten med oppgaven Grensesnitt for Touchpad med Roterende Del var å lage et grensesnitt for en touch encoder med display fra Grayhill mot navigasjonssystemet til Marine Technologies i Egersund. Grensesnittet ble laget i Visual Studios i C# kodespråk med et HID bibliotek. Det er ulike funksjoner på touch encoderen. Den har en roterende kant, det er berøringssonser på den og det er mulig å svipe på skjermen. Dataene som kommer ut fra touch encoderen forteller hva som har blitt utført på den, og bitstrengen har bit satt av til ulike formål. Hvilke formål de har er gitt i en USB grensesnitt fil fra Grayhill. Den andre delen av oppgaven var å lage displaybilder til touch encoderen. De ble laget ved hjelp av en applikasjon som følgte med enheten.

De delene av navigasjonssystemet som skulle dekkes av grensesnittet var autopilot som styrer kursen, ECDIS som er en kartapplikasjon og radar som viser andre båter i nærheten. I den endelige koden i Visual Studios ble de satt opp som hver sin funksjon med tilhørende måter å styre dem på.

Displaybildene ble laget så enkle som mulig for å unngå feiltrykk og frustrasjon for brukeren i fremtiden. De ble laget med få og store touchsoner og det ble heller benyttet de andre mulige hendelsene på touch encoderen for å kunne utføre alt som var nødvendig for hver applikasjon.

Forord

Denne oppgaven ble skrevet på Universitet i Stavanger under linjen automatisering og elektronikkdesign. Den ble skrevet for Marine Technologies i Egersund og under hele perioden satt jeg på deres lokaler.

Jeg ønsker å takke Marine Technologies for muligheten til å skrive denne oppgaven og alt de bistod med. Jeg ønsker å si en spesiell takk til Sven Ole Aase som veileder fra universitet og god teknisk hjelp med rapporten og Sveinung Tollefsen som veileder fra Marine Technologies. Til Kjell Erik Larsen for å bistå med kontinuerlig veiledning innen programmering og software. Jeg ønsker også å takke Eirik Hovland og Kai Tonnesen som bistod med gode råd og grafisk innhold i rapporten. Til slutt ønsker jeg å takke Stein Aase som hjalp med hardware og koblinger og Sven Aamodt god veiledning og oppfølging.

Innholdsfortegnelse

Forkortelser

DP Dynamic Positioning. 6

ECDIS Electronic Chart Display and Information System. 3

GUI Graphical User Interface. 3

HID Hardware Interface Device. 19

IDE Integrert Utviklingsmiljø. 19

MT Marine Technologies. 2

VID Vendor ID. 18

Kapittel 1

Introduksjon

1.1 Oppgavebeskrivelse

Oppgaven går ut på å lage et grensesnitt mellom programvaren til Marine Technologies (MT) og en Grayhill Touch Encoder. Touch encoderen som er vist i figur 1.1 har en touch skjerm og en roterende kant, denne skal brukes til å blant annet styre kursen til båter som bruker navigasjonssystemet til MT. Touch Encoderen skal monteres på stolen hvor båten styres og den skal kunne navigere skjermen og dens ulike funksjoner.

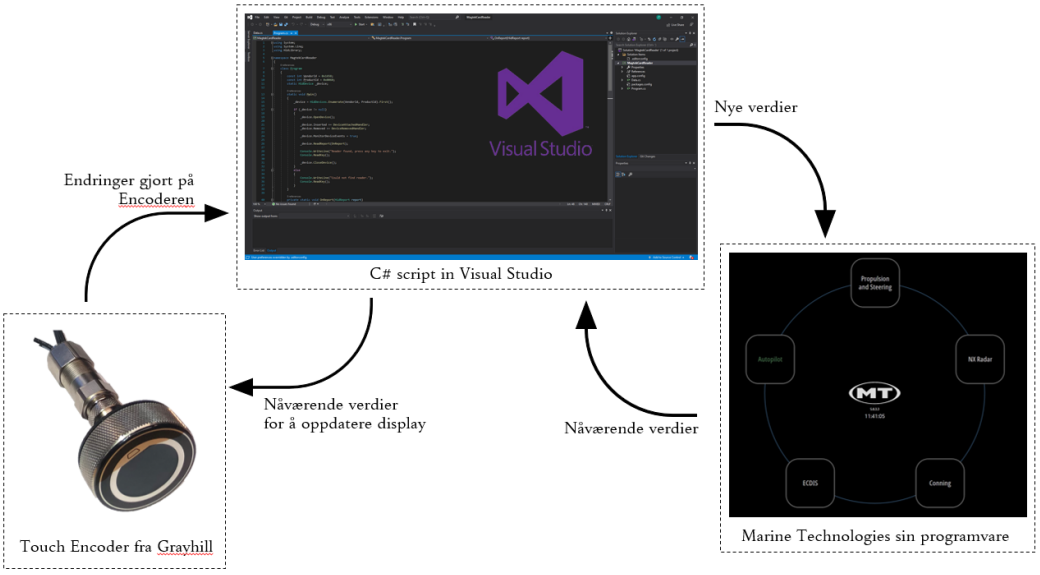
Figur 1.1: Touch encoder hentet fra [1]

1.1 Oppgavebeskrivelse

Grensesnittet skal lages ved hjelp av USB-kommunikasjon med en PC som har Windows 8 eller 10 operativsystem. Først og fremst skal den kommunisere med autopilot, for så å bli implementert på to andre applikasjoner siden. Autopilot angir kursen en båt skal seile mot. Denne innstillingen skal kunne endres både på encoderen og i applikasjonen ofr så å kommunisere dette til den andre parten.

Det skal etableres en toveis kommunikasjon mellom PC og enheten og det trengs derfor en kode som leser og oversetter dataene fra encoderen. Applikasjonene på PC er laget i C++ så det må implementeres støtte for dette. Overordnet oversikt over hvordan kommunikasjonen skal fungere er vist i figur 1.2.

Det skal også lages displaybilder med passede grafisk grensesnitt (Graphical User Interface (GUI)) symboler på enheten for de forskjellige funksjonene autopilot, som er kurs oppgitt i grader, Electronic Chart Display and Information System (ECDIS) og radar som angir posisjonen til andre båter i nærheten. ECDIS er et system som henter posisjonsinformajson fra posisjon, kurs og hastighet ved hjelp av vannreferansesystemer. Displaybildene skal lages via et verktøy som medfølger enheten.

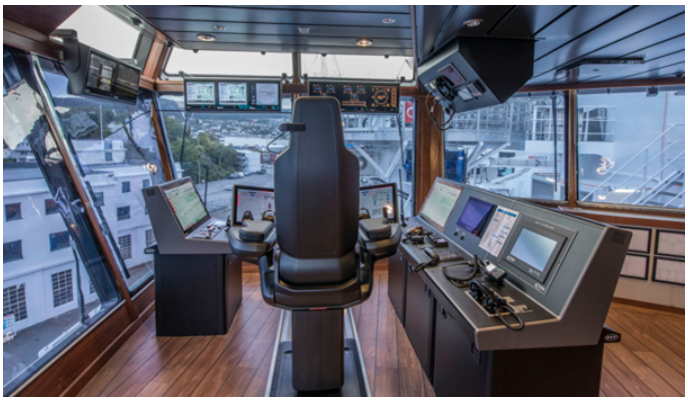


Figur 1.2: Hensikten med oppgaven.

1.2 Problemstilling

For å styre båtene som er utstyrt med MT sitt navigasjonssystem benyttes ulike dataskjermer som er montert foran kapteinen. Dataskjermene styres ved å reise seg fra stolen og gå bort til dem og endre f.eks. zoom i kart. Hvordan oppsettet typisk er i et styrrom på en båt er vist i figur 1.3.

1.2 Problemstilling



Figur 1.3: Styrrom på båt.

Foreløpig er det installert et lite hjul som lar deg stille på autopiloten. Hjulet er vist i figur 1.4. Siden dette hjulet har begrenset med muligheter og kun kan styre autopiloten, er det ønskelig å finne en ny enhet for å erstatte hjulet. Touch Encoderen skulle sørge for flere valg og muligheter til å styre mer enn kun autopilot. Encoderen er mer kompleks og kan derfor brukes over flere plattformer. Siden det var mulig å skreddersy displaybildene til hver enkelt båt er det lett å implementere de til ulike prosjekter.

Det skal leveres et komplekst system til en båt om et års tid og det hadde vært en fordel å kunne styre hele systemet med Touch Encoderen.

Figur 1.4: Nåværende hjul for å endre kurs i autopilot

1.3 Bakgrunn

1.3.1 Marine Technologies

MT er et utspring fra Simrad Robertson og Kongsberg's tidligere avdeling i Egersund, og styrker det maritime teknologimiljøet i Egersund. Selv om firmaets størrelse er beskjedent i forhold til sine konkurrenter, klarer MT å være innovative og tilpasningsdyktige og har bygget opp en sterk merkevare. Fra å levere egenutviklet elektronikk og programvare til nisjeprodukter innen Dynamic Positioning (DP) i offshoremarkedet har MT komplettert sin produktportefølje med egenutviklet og kan i dag levere fullintegrerte broløsninger med alt innen navigasjonsutstyr og manøvreringssystemer til de fleste typer fartøyer.

Det overordnede målet for MT er å støtte markedets etterspørsel med kostnadseffektive DP-systemer basert på disse punktene:

- Økt sikkerhet gjennom funksjonell design og programvare
- Redusert installasjonskostnad og mindre oppstartstid

- Reduserte servicekostnader og bedre tilgang på reservedeler
- Fjernbetjening og servicehjelp til flåteledelsen

MT tilbyr også løsninger for manøvrering av alle typer fartøy, fra store offshore spesialfartøyer, innkvarteringsfartøy og forsyningsfartøy for vindfelt til mindre slepefartøy og luksusyachter. Hovedkontoret deres ligger i Louisiana i USA, men de har også kontorer i Norge, Singapore og Brasil [2].

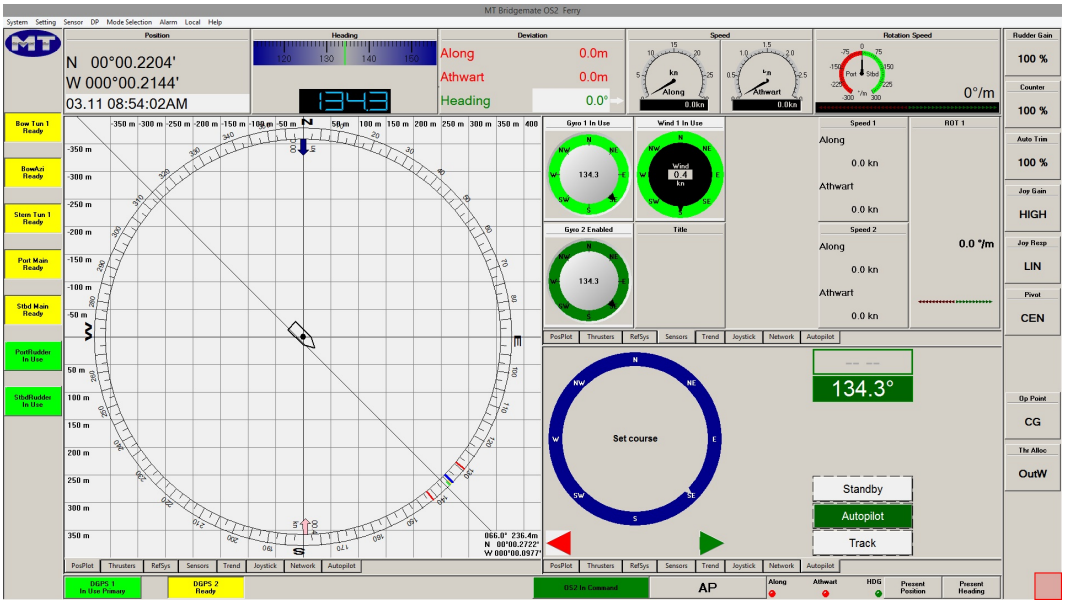
1.3.2 Tidligere Arbeid

Autopilot - applikajson fra Marine Technologies

Navigasjonssystemene til MT består av ulike moduler deriblant autopilot, ECDIS og radar. Autopilot sørger for at båten holder seg på en innstilt kurs, radar viser andre båter i nærheten og ECDIS er en kartfunksjon.

Autopilot er en applikasjon som MT selv har produsert og denne kan styres fra flere andre applikasjoner i tillegg til at det er en egen. I figur 1.6 kan du blant annet se at autopiloten kan stilles på mens kartet er åpent. Hvordan autopilotens egen applikasjon ser ut er vist i figur 1.5. Det er mulig å endre denne ved å trykke på pilene eller dra på det blå hjulet nede til høyre.

1.3 Bakgrunn



Figur 1.5: Skjermbilde av autopilot-applikasjonen

Oppgaven er å kunne lage et grensesnitt mellom den programvaren som finnes i Touch Encoderen og programvaren til MT. Grensesnittet skal lese inn informasjon fra Touch Encoderen, for eksempel hvor mange hakk den ble rotert, og oversette dette slik at det kan leses inn i systemet til MT. Om Encoderen ble rotert tre hakk, skulle grensesnittet oversette dette og gjøre at MT systemet kunne plukke opp at båten skulle endre kursen med 3 grader.

Touch Encoderen må også kunne lese inn nåværende kurs på båten for å kunne vise dette i displayet.

Programvaren for Touch Encoderen var som nevnt ovenfor allerede konfigurert, hovedprinsippet er at rotering og ulike touch soner blir lest inn på pcen som ulike byte, dette er nærmere forklart på side 2 til 8 i vedlegg B.

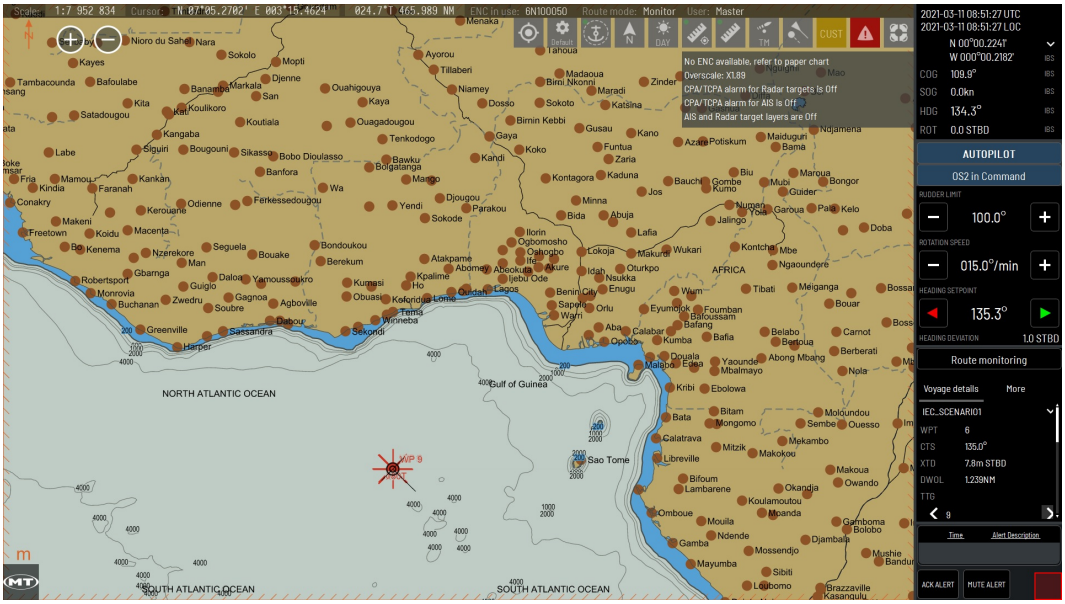
ECDIS - applikasjon fra Navtor

Hvordan ECDIS typisk ser ut er vist i figur 1.6. Det er mulig å zoome inn og ut i tillegg til å flytte på kartet for å se et nytt område eller flytte kartet ettersom

1.3 Bakgrunn

båten flytter seg. Det er også mulig å feste kartet slik at det følger båten når den flytter seg.

ECDIS er en applikasjon MT får fra Navtor. Navtor holder også til i Egersund og leverer innovative e-navigasjonsløsninger, navigasjonsprodukter og tjenester til den maritime sektoren.



Figur 1.6: Skjermbilde av ECDIS-applikasjonen

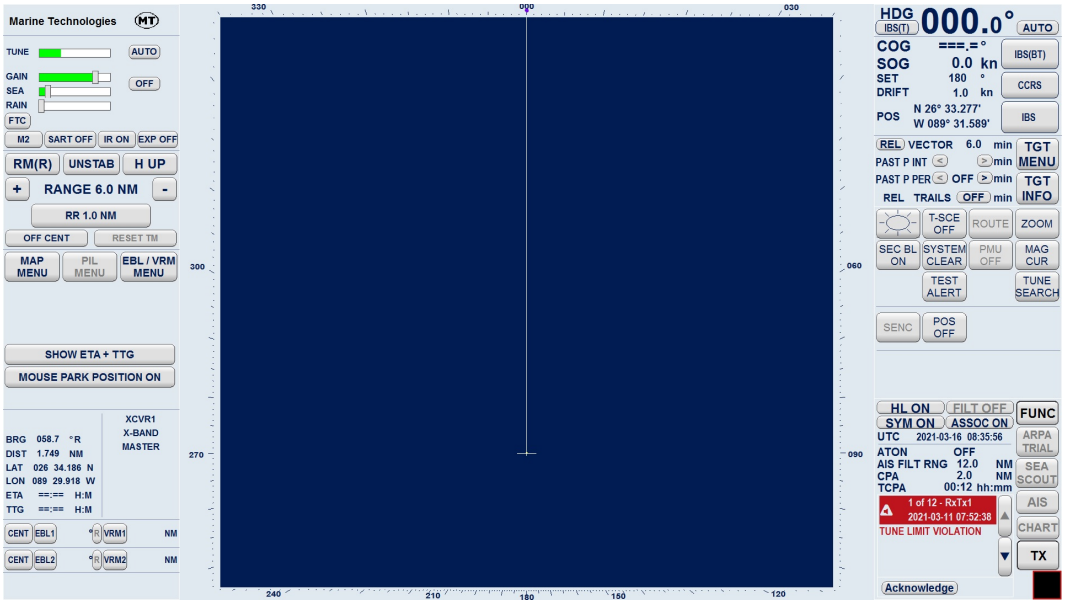
Radar - applikasjon fra Raytheon Technologies

Radarapplikajsonen er vist i figur 1.7 og i den er det mulig å endre blant annet "range", "rain", "sea" og "gain". En radar sender ut en kort radiopuls med høy pulseffekt. Når denne pulsen treffer et annet objekt blir noe av signalet reflektert tilbake. Radaren regnte så ut, ved hjelp av bølgene som returnerer, distansen til dette objektet. "Range" er rekkevidden, avstanden fra radaren til målet i luftlinje [3]. I applikasjonene til Raytheon har radaren en rekkevidde fra 0.125 til 96 nautiske mil. Er den mindre er det lettere å skille mellom to objekter som var nær hverandre. Er rekkeviden større kan du se skip som er elenger borte, men det vanskeligere for radaren å skjelne to skip som ikke var langt fra hverandre.

1.3 Bakgrunn

"Rain" og "sea" filtrerer ut ekko fra regn og hav, det kunne stilles på hvor mye som skal tas bort for å fjerne uønsket støy. "Gain" endrer mottaksensitiviteten til radaren. [4]

Radar-applikasjonen blir levert av Raytheon Technologies som forsker, utvikler og produserer avanserte teknologiprodukter i luftfarts- og forsvarsindustrien, inkludert flymotorer, flyelektronikk, luftstrukturer, cybersikkerhet, missiler, luftvernsystemer og droner.



Figur 1.7: Skjermbilde av radar-applikasjonen

Kapittel 2

En beskrivelse av hardware- og programvarefunksjonene til Touch Encoderen

I dette kapittelet er Touch Encoderen og dens funksjoner forklart for å gi en oversikt over mulighetene dens. Det er også gitt en oversikt over ulike programmer og verktøy som ble brukt for å løse oppgaven på best mulig måte. Noen av valgene er basert på bedriftens preferanser og noen fra erfaringer. Noen valg ble gjort etter vurderinger på hvilket verktøy som egnes best til oppgaven.

2.1 Touch Encoder

Touch Encoderen er av typen TE-M321 og kommuniserer via en M12 kobling. M12-kabler er sirkulære og brukes blant annet til sensorer, aktuatorer og industrielt Ethernet. Et eksempel av en slik kabel er vist i figur 2.1.

Figur 2.1: M12 kabel hentet fra [5]

Encoderen kan både hente inn og sende ut informasjon. For å laste opp nye displaybilder som er programmerte må den være i modus til å hente inn informasjon, altså programmeringsmodus. Når Encoderen blir satt til kjøremodus, kan den også hente inn informasjon fra en ekstern kilde i tillegg til å sende ut informasjonen som er på displayet dens.

På enheten er det mulig å styre ved å skru på kanten, trykke på displayet eller sveipe. I tillegg er det en hjem knapp. Det er derfor mulig å definere ulike funksjoner på samme displaybilde, bytte mellom displaybildene og lett komme tilbake til standarddisplayet ved hjelp av hjem-knappen.

2.1.1 Programmering av Touch Encoder

Det er mulig å kjøpe et utviklingssett med komponenter og ledninger for å programmere Touch Encoderen. I denne pakken er det blant annet inkludert en utviklingsmodul, en rød programmeringskabel, en strøm til USB kabel, en hvit USB Micro B til USB Type A adapterkabel og en minnepenn. Innholdet i pakken er vist i figur

2.1 Touch Encoder

Figur 2.2: Innhold i utviklingspakken fra Grayhill hentet fra D

Dette settet var ikke var tilgjengelig på bedriften og det ble derfor laget et eget oppsett. Enheten må hele tiden være koblet til strøm, og for å kunne bytte lett mellom programmeringsmodus og kjøremodus ble det laget en kabel med bryter. Kabelen har en hann USB, som var strøm til Encoderen, en hunn USB, som var til opplasting av program på Encoderen fra en minnepenn, og en bryter. Oppsettet er vist i figur 2.3. Ved å flytte bryteren, byttet Encoderen mellom programmeringsmodus for å laste opp displaybilder og kjøremodus for å bruke Encoderen.

Figur 2.3: Oppsett for programmering/kjøring av Touch Encoderen

2.1.2 Kjøring av Touch Encoder

Hvis bryteren stod slik at den var i kjøremodus vises de displaybildene som er lastet opp på Encoderen. Da kunne også de funksjonene som er lagt inn, for eksempel hjem knapp og sviping, testes og senere finjusteres.

2.1.3 Design av displaybilder til Touch Encoder

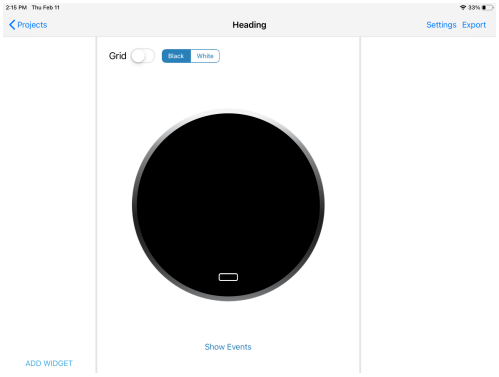
For å designe displaybildene som brukes på Encoderen, benyttes Grayhill sin egen app. Appen GH TE-MX ble lastet ned fra appstore og krevde programvare iPadOS 11 eller nyere. GH TE-MX er utstyrt med et utvalg standard displaybilder som kan endres etter behov. Figur 2.4 a) viser hvordan appen så ut etter et nytt prosjekt var startet opp, før noen endringer ble gjort. Det er her mulig å endre navn på prosjektet og legge til handlinger. En handling lagte snarvei til et annet displaybilde, enten ved berøringssoner som var mulig og skalere og flytte på, eller ved å svipe.

Når "ADD WIDGET" trykkes på, dukker menyen i figur 2.4 b) opp. Her vises ulike standard displaybilder og velges alt etter ønsket funksjon for Touch Encoderen. Det er noen basis displayer som bare et bilde, f.eks. logo, og ren tekst. Det er mulig å laste opp et eget bilde og deretter definere berøringssoner på disse og dermed lage en helt egendefinert meny. Andre muligheter er standardmenyer, for å velge mellom displaybilder, multi-verdi som viste flere verdier på

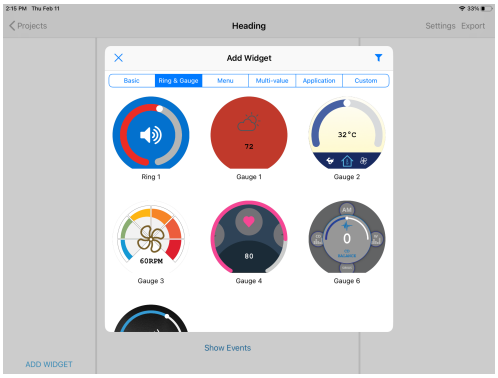
samme displaybilde og ring og måler. Alle bildene i de ulike displaybildene kunne erstattes med egne filer fra biblioteket på iPaden eller fra OneDrive.

Et eksempel fra "ring og måler" kategorien og hvordan den kunne redigeres er vist i figur 2.4 c). Her var det mulig å endre bakgrunn, farge, maksimum og minimumverdier på ringen, bytte ut pluss og minustegnet og endre farge på disse, bytte ut ikonet i midten og legge til prefiks og suffiks i tillegg til å endre font og farge på teksten. I eksempelet vist i figur 2.4 d) ble fargen på ringen endret til standardfarge for marine instrumenter, området ringen dekker ble satt til 0-360, ikonet ble byttet ut med marine ikonet for heading og ° tegnet ble lagt til etter teksten. Displaybildet vil da for eksempel kunne benyttes til å angi heading til en båt ut fra 360°. En nærmere forklaring på hvordan de ulike displaybildene kan redigeres er vist på side 3 i vedlegg **??**, den mest vesentlige delen av dette er vist i figur 2.5.

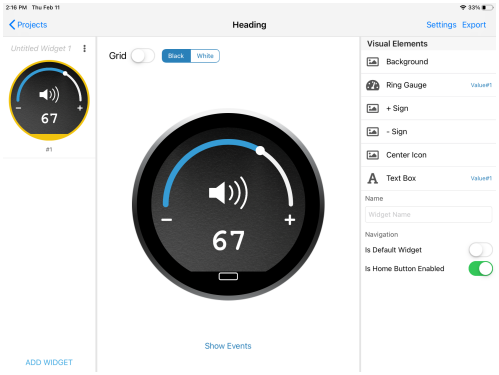
2.1 Touch Encoder



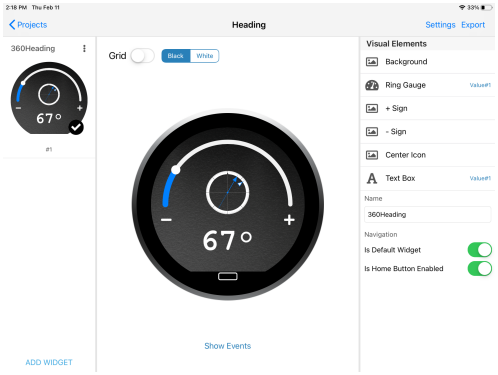
(a) Ny oppstartet prosjekt



(b) Legge til et displaybilde



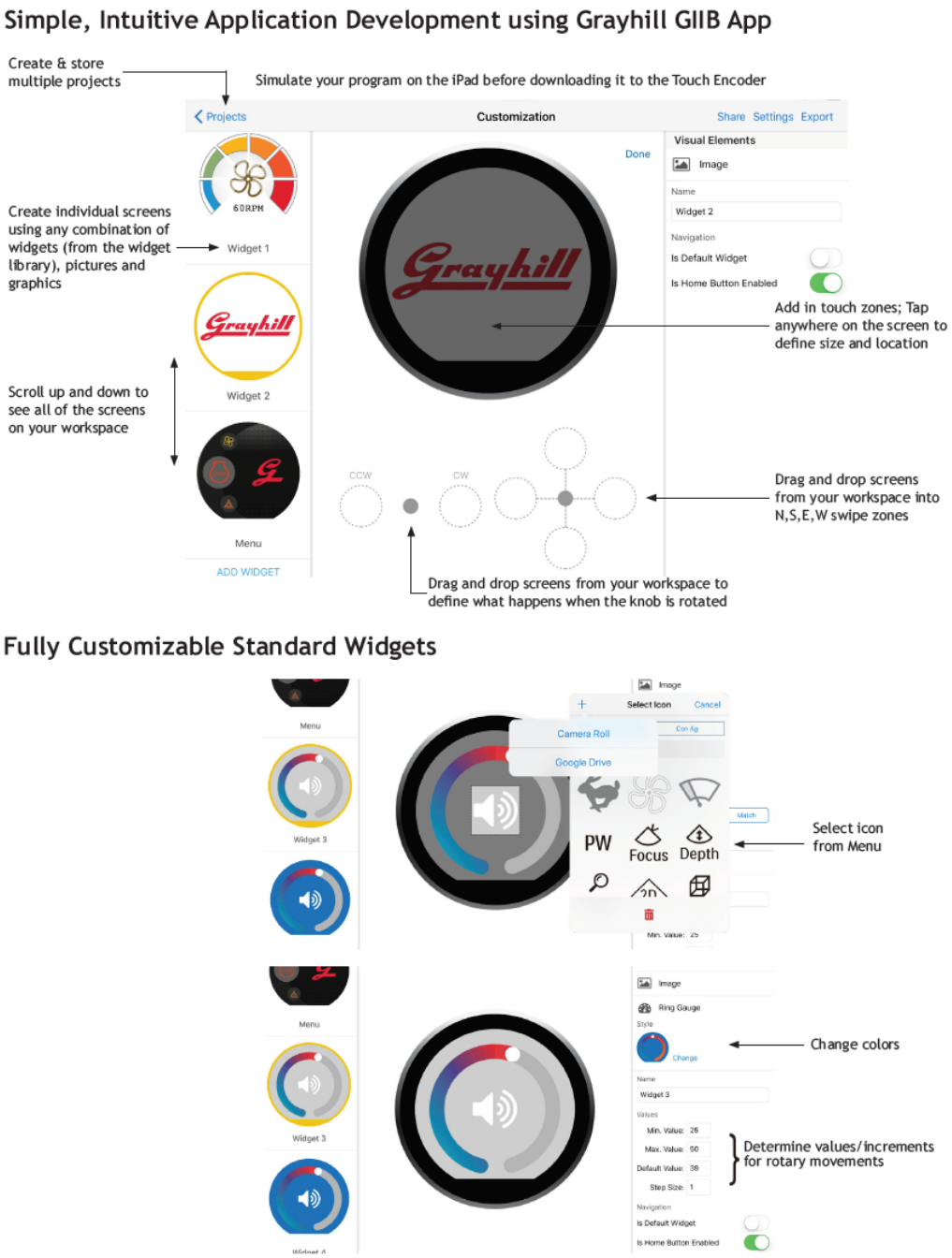
(c) Standard displaybilde



(d) Egendefinert displaybilde

Figur 2.4: Oppsett av prosjekt i Grayhill sin app

2.1 Touch Encoder



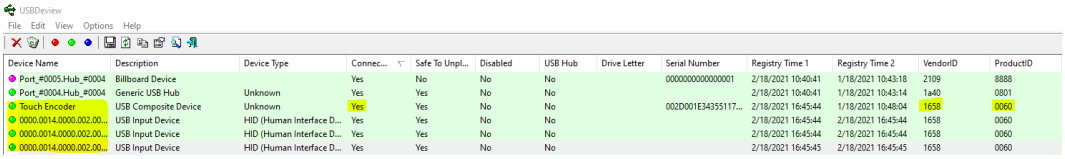
Figur 2.5: Utdrag av datablad

2.2 Lese- og programmeringsverktøy

2.2.1 USBDeview

For å kunne finne ut om Encoderen kommuniserer riktig med pcen ble USBDeview lastet ned. USBDeview lister alle USB enheter som er eller har vært tilkoblet pcen. Informasjonene som vises til hver enhet er blant annet navn/beskrivelse, type, serienummer, dato og tid den ble sist lagt til, VendorID og ProductID.

Det var mulig å identifisere Touch Encoderen med USBDeview og finne Vendor ID (VID) informasjon om den. Numrene, i heksadesimal, som var oppgitt på side 1 i USB dokumentet i vedlegg B ble bekreftet gjennom dette verktøyet. I tillegg ble status på om den var koblet til og kommuniserte med pcen monitorert gjennom dette verktøyet. Informasjonen som ble hentet ut er markert i gul i figur 2.6.



Figur 2.6: Touch Encoderen i USBDeview

2.2.2 Wireshark

Wireshark er et verktøy som lar deg se informasjonene du får fra internett og USB enheter koblet til pcen. Verktøyet ble brukt for å få en bedre forståelse av hvordan enheter kommuniserer med hverandre i tillegg til å skille ut de dataene fra Encoderen som var nødvendige i programmet. De dataene som var knyttet til beskrivelsen av selve enheten er vist i figur A.1 i vedlegg A. Disse bytene fortalte blant annet noe om størrelsen på dataene, hvilken enhet det er, klasser, protokoller og annen info som pcen trenger for å kunne finne enheten og laste opp og ned data.

De resterende dataene, som er markert i figur A.2 i vedlegg A, viser til selve dataene som enheten sendte ut, ut fra funksjon, endringer, osv. I dette tilfellet

viser det til hvilket displaybilde Encoderen var i, hvor mange hakk den var rotert, hvilke touch soner som var trykker på, om det var blitt svipet o.l. Bytene var oppgitt i heksadesimal slik at 03, eller binært som 0000 0011, viser til report ID 3, de neste to tallene viser til displaybilde 3 osv. Denne informasjonen ble hentet fra vedlegg B, side 5.

2.2.3 Visual Studio

Til å lage selve koden som skulle fortelle hvordan pc og Encoder ble Visual Studio benyttet i kombinasjon med standard Windows tastetrykk. Visual Studios er et Integrert Utviklingsmiljø (IDE) som hjelper utviklere å utvikle programmer for blant annet Windows. Det er inkludert et feilsøking-program som fungerer både som kilde feilsøker og en feilsøking på maskinnivå. For å vite hva endringer på Encoderen skulle oversettes til slik at applikasjonene til MT skulle forstå dem, ble de ulike funksjonene omgjort til standard Windows tastetrykk som "Enter".

2.3 HID bibliotek

For at koden for kommunikasjon mellom pc og Encoder skulle fungere optimalt, var det nødvendig med ulike funksjoner og variabler. Det var mange biblioteker som fantes som har en del slike funksjoner allerede definert. Valget for dette programmet ble et Hardware Interface Device (HID) bibliotek. Det er et populært bibliotek og derfor mye informasjon og tips å finne. Det hadde også de funksjonaliteter som var nødvendig for å kunne skrive koden.

Det følgte med en del ulike filer i HID biblioteket som ble lastet ned, blant dem var en del eksempler på hvordan det kunne brukes. Et av eksemplene var et skript for å lese av et "Magtek Card Reader". En "Magtek Card Reader" er en typisk kortleser en møter når en skal betale på butikken. Det kan enten være i form av tapping, putte inn eller dra kortet i en kortleser, se eksempel i figur 2.7. Denne sendte ut informasjon i lignende form som Encoderen og var derfor et godt utgangspunkt.

HID biblioteket har for eksempel allerede definert hvordan programmet skal hente ut "Report ID" og andre nyttige ting for å lese data fra Encoderen.

Figur 2.7: Magtek Card Reader

2.3.1 Dropbox

For å ha filene og bildene lett tilgjengelig over alle maskiner som ble brukt til å skrive oppgaven, hjemme eller på bedriften, ble Dropbox benyttet. Dropbox er en skylagrings tjeneste som gir tilgang til filer over flere plattformer. Det var da mulig å laste ned Dropbox applikasjonen til iPaden som ble brukt under prosjektet og lagre bilder og filer der. Mobiletelefonen ble brukt til å ta bilder av oppsett og utstyr som var på bedriften. Ved å laste også disse opp i Dropbox applikasjonen på mobilen, var bildene lett tilgjengelig på PC. Bildene kunne hentes ut og lastes opp i prosjektrapporten etter ønske.

Kapittel 3

Programmering og designing av displaybildene til Touch Encoderen

En del av oppgaven var å designe displaybilder som kunne egnes til de ulike applikasjonene bedriften ønsket å bruke Encoderen til, et innebar å bli kjent med enheten og dens tilhørende applikasjon. Dette kapittelet tar for seg fremgangsmåten for å kunne designe, laste ned og laste opp displaybildene. Til slutt er det en oversikt over hvordan testing ble gjennomført for disse.

3.1 Spesifikasjoner fra bedriften

Ulike ansatte i bedriften ble kontaktet angående ønsket utseende og funksjon for de ulike displaybildene. Det ble også formidlet hvor mange displaybilder de ønsket, hva de skulle brukes til og hvem som skulle bruke dem. Dette var nyttig informasjon for å kunne skreddersy displaybildene etter bedriften og kundene deres sine behov.

Først og fremst hadde de preferanser til et enkelt oppsett av Encoderen. Det skulle være lett å navigere displaybildene og touch-sonene skulle være så få

som mulig for hvert displaybilde. Dette ville begrense antall ganger det ble trykket feil og frustrasjon knyttet til bruk for kundene.

Bedriften etterspurte også at Touch Encoderen skulle fungere som en mus/tastatur for navigasjonssystemene deres. De hadde selv kun utbedret koden for autopilot slik at det var kun denne de kunne gå inn å endre. Målet var å kunne sende info fra Encoderen til PC og tilbake, og autopilot ble derfor det koden ble siktet inn mot i denne omgang.

Da grensesnittet var ferdigstilt var det enkelt for bedriften å tilrettelegge den mot de andre applikasjonene de hadde, når denne koden ble tilgjengelig. Displaybilder ble lagt til alle de tre applikasjonene som er nevnt tidligere, autopilot, ECDIS og radar, men bare autopilot ble testet.

Funksjonene bedriften ønsket for de andre applikasjonene ble tatt høyde for i skriptet som ble laget, men kun autopilot ble realisert og testet for denne oppgaven.

Touch Encoderen skulle monteres på stolen til kapteinen for å begrense antall ganger han måtte reise seg fra stolen for å manøvrere båten sin. Encoderen og skriptet måtte derfor være kompatibelt med de operativsystem som er på båtene. MT har og vil levere systemer der både C# og C++ er benyttet. Skriptet for oppgaven ble først skrevet i C#. Dette var mest kjent for både student og ansatt fra bedrift som bistod med hjelp. Skriptet ble senere også skrevet i C++ for å kunne tilfredstille kravet fra bedriften. Dette gjorde at skriptet ikke begrenset hvilke kunder som kunne benytte Touch Encoderen.

3.2 Grayhill applikasjon

For å bruke applikasjonen til Grayhill vist i figur 3.1 var det nødvendig med en iPad med operativsystem 11 eller nyere [6].

Applikasjonen var lett å navigere og ulike displaybilder ble testet før valget falt på de vist i kapittel 5. For å teste hvordan displaybildene opererer og hva som skjer når du for eksempel roteterer kanten, måtte de lastes opp på selve Encoderen, dette kunne ikke testes i applikasjonen alene.

Figur 3.1: Grayhill sin app; GH TE-MX

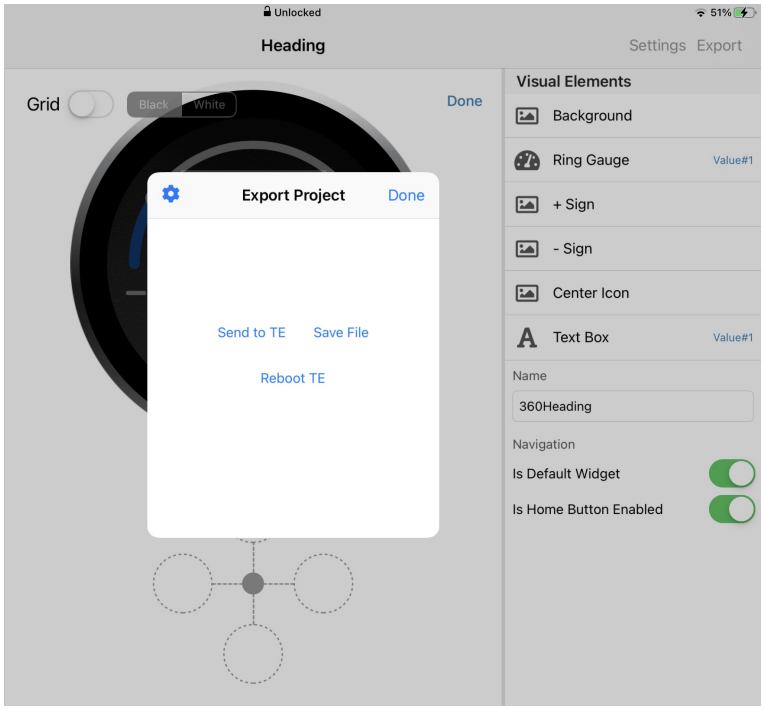
3.3 Displaybilde design

Displaybildene måtte være enkle og praktiske samtidig som de utfylte kravene til hver applikasjon. Bildene som ble brukt som bakgrunn for radar og ECDIS ble hentet fra applikasjonene selv. Det ble så tatt en avgjørelse på hva som skal være en del av displaybildet. For eksempel om det skulle være tall med på displaybildene eller ikke. Det ble også funnet ut hvilken metode for å gjøre endringene som egnet seg best, enten touch, sviping eller rotering av kanten, og hvor og hvor store touch sonene skulle være.

3.4 Eksportering til Encoder

Når skjermene var ferdigstilt i appen, var målet å overføre disse til Encoderen. Ved å klikke på eksport knappen oppe i høyre hjørnet dukket menyen i figur 3.2 opp. Her var det ulike valg, det som fungerte best var å lagre til fil på iPaden.

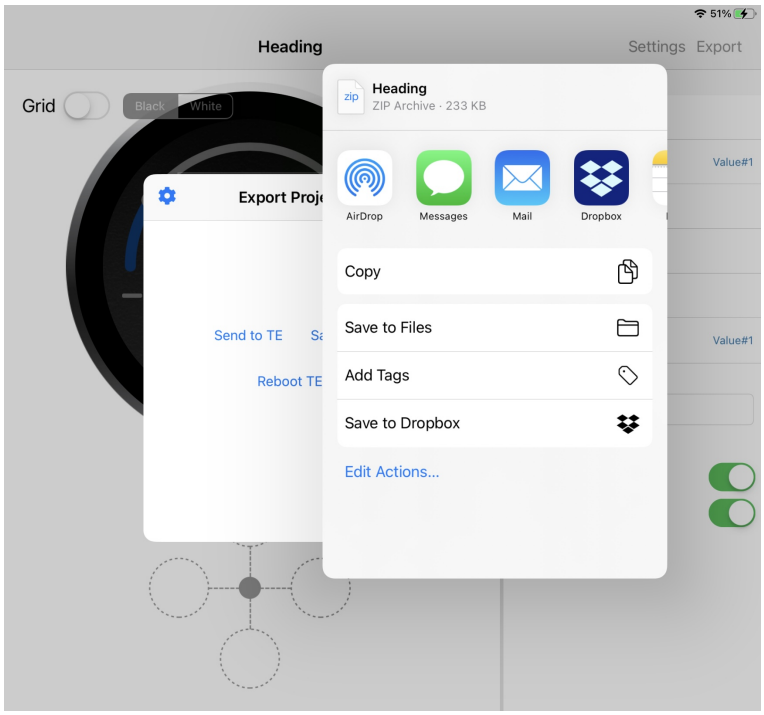
3.4 Eksportering til Encoder



Figur 3.2: Klikk; Export

Av de ulike valgene som dukket opp ved å trykke på "Save File", som er vist i figur 3.3, var det enklest å trykke Save To files". Ved å klikke direkte på Dropbox tegnet fra denne menyen, måtte filen sendes på mail og ha en mottaker. Dette ble sett på som unødvendig.

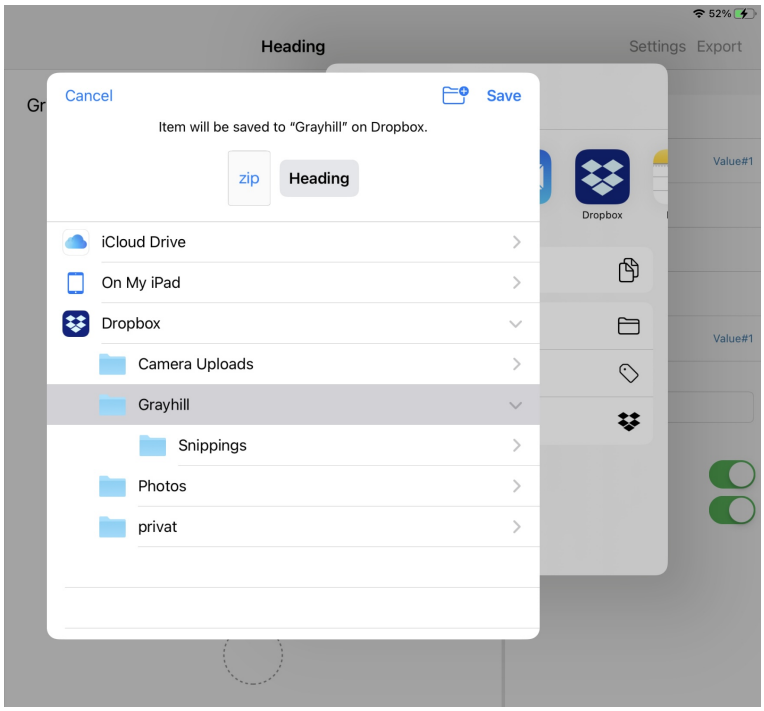
3.4 Eksportering til Encoder



Figur 3.3: Klikk; Save File

Den enkleste muligheten var å trykke på “Save to Files” som nevnt ovenfor, du fikk da opp menyen i figur 3.4. Siden applikasjonen til Dropbox var lastet ned på iPaden var det mulig å lagre filen direkte i en av mappene der. Det ble laget en egen mappe i Dropbox til prosjektet kalt Grayhillfor bedre oversikt.

3.4 Eksportering til Encoder



Figur 3.4: Klikk; Save to Files

Etter at filene ble lagret i Grayhill mappen på Dropbox, var det mulig å hente disse filene over på en minnepenn. Filene blir lagret som zip-filer fra Grayhill applikasjonen og for at de skal kunne lastes opp på Encoderen senere, måtte de beholdes slik.

Det var mulig å ha flere zip-filer på minnepinnen samtidig og bla seg gjennom disse inne i Encoderen. For å kunne gjennomføre en opplasting må Encoderen være i programmeringsmodus. displaybildet vil da se ut som i figur 3.5. For å spesifisere kommunikasjon, kalibrere enheten og få informasjon om nåværende versjon av programvaren på Encoderen ble "Utilities" knappen brukt. Det først som ble gjort med Encoderen var å laste opp siste versjon av programvaren på den. Dette for å sikre at alle funksjoner skulle være tilgjengelige og fungere som antatt.

For å laste opp filer for displaybildene som var designet og oppdatere program-

varen, ble “Update” knappen brukt.

Figur 3.5: Første display på Encoderen i programmeringsmodus

De to valgene som er representert i figur 3.6 som dukker opp om du trykker på “Utilities” viser til oppdatering av programvare og oppdatering av program som blir kjørt på Encoderen. Displaybildene som er designet ble lastet opp ved å trykke på “Project”.

Figur 3.6: Valg for å oppdatere Encoderen

Hvis minnepinnen var koblet i og inneholdt zip-filer, dukket de opp inne på "Project" og ble vist som i figur 3.7. SampleProject.zip er et standardprogram laget av Grayhill. Alle Encoderne kommer innstilt med dette for å ha et utgangspunkt. Det var gunstig å alltid ha liggende denne filen. Når det oppstod feilmeldinger eller opplastingen av displaybildene ikke fungerte som forventet, ble denne brukt som mal og det var da mulig å luke ut ulike feil.

Figur 3.7: zip-filer på minnepinne

3.5 Testing av designede displaybilder uten skript

Etter at opplastingen av displaybildene var velykket utført og de dukket opp på Encoderen når denne var i kjøremodus, kunne de testes. Dette ble gjort for å finne ut hvilke elementer det er mulig å stille på med Encoderen, hvilke som er statiske og hvilke som kun er avhengig av input fra en annen enhet som for eksempel PC.

3.5 Testing av designede displaybilder uten skript

Funksjonene som sviping mellom displaybildene og hjem knappen kunne også prøves ut. Hvilke funksjoner og hendelser som var gunstige å bruke i sluttproduktet og hvilke som burde byttes ut. Ønskelige touch soner ble definert selv og hvor omfattende disse måtte være for å være brukbare ble også funnet ut på dette stadiet.

Displaybildet på Encoderen er ganske lite og detaljer kunne derfor bli utydelige om de var for små. Det ble derfor forsket på hvor store de måtte være for å vise det som var nødvendig, hvilke farger som var tydelige nok til at det var enkelt å navigere rundt i programmet og hvilken tekst som egnet seg best.

Kapittel 4

Skriving av koden for kommunikasjon mellom pc og Encoder

I dette kapittelet blir det gjennomgått hvordan det ble gått frem for å kunne lage koden som skulle fortelle applikasjonene til MT hvilken informasjon som kom fra Encoderen og hvordan applikajsonene skulle sende informasjon itlbake til Encoderen. Blant annet hvordan de ulike eksempelkodene fra HID biblioteket ble benyttet for å nå et endelige resultatet og hvilke endringer og vurderinger som ble gjort underveis.

4.1 Identifisering av Touch Encoderen

Etter å ha blitt kjent med enheten og dens tilhørende verktøy, var det ønskelig å kommunisere med den ved hjelp av en PC. For å finne ut informasjonen som trengtes, ble USBDeview programmet benyttet. Noen ganger var det nødvendig å koble enheten ut og inn for å oppnå kontakt igjen. USBDeview gjorde det mulig å overvåke enhetens kontakt til en hver tid.

USBDeview ga også muligheten til å lese av VendorID og ProductID og be-

krefte at disse stemte med de tallene oppgitt i vedlegg B. Disse tallene var nødvendige å ha med i koden i skriptet for å hente ut riktig informasjon fra riktig enhet. Det er mange ulike USB enheter som vil være koblet i en PC, også på en båt. Det måtte derfor være mulig å skille ut de dataene vi trengte.

Siden enheten het Touch Encoder i USBDeview når den var koblet i, var det enkelt å identifisere den selv om flere USB enheter som tastatur og mus var koblet i samtidig. Mens koden ble skrevet var det derfor også enkelt å følge med på at enheten var koblet til slik den skulle og at ikke det var en av feilene som oppstod under debugging.

4.2 WireShark

Når kommunikasjon var bekreftet, var neste steg å forstå dataene som ble sendt ut fra Encoderen. Dette ble gjort i WireShark. Det ble funnet at dataene som i WireShark ligger under USB URB er informasjon om selve enheten. Hvor mye data den sender ut, identifisering av den osv.. Mens dataene som kom inn i WireShark under HID Data er dataene enheten sender ut om endringer som er gjort i den, hvilket displaybilde som er i bruk osv..

Det ble også oppdaget at disse dataene blir i WireShark oppgitt i heksadesimal. De måtte da omgjøres til binært for å kunne vite hvilke elementer fra Encoderen de tilsvarer. For å kun se den informasjonen som var vesentlig, ble kanten på Encoderen rotert mens WireShark kontinuerlig leste data, for så å sjekke hvilke nummer som dukket opp. Nummeret som WireShark identifiserte akkurat Touch Encoderen med varierte fra gang til gang den ble plugget i. Når riktig nummer var identifisert, ble datastrømmen filtrert slik at kun informasjonen som var nødvendig dukket opp. Det var da lettere å lete frem det som trengtes og det var mye mindre data å forholde seg til.

4.3 Visual Studio med HID bibliotek

Neste steg, når informasjonen for å detektere riktig enhet var funnet, var skrive koden. Det ble da tatt utgangspunkt i et HID bibliotek. Det ble lastet ned og importert i programmet og testprogrammene som ble brukt. For å finne svaret

på ulike utfordringer ble dette biblioteket brukt aktivt til å finne definisjonen til ulike variabler og funksjoner.

"Magtek Card Reader" eksempelet ble benyttet til å se hvordan oppsettet kunne være for å lese av en USB enhet. Eksempelet ble skrevet om for å gjelde Touch Encoderen. I første omgang gjaldt dette kun å forandre VID og "Product ID". Senere ble det skrevet inn kode for å kunne vise hva de ulike dataene tilsvarte. Et eksempel er vist i figur 4.1. Koden her går gjennom dataene som ble sent inn fra Encoderen og sorterte det i de ulike kategoriene.

```
if(int.Parse(report.ReportId.ToString()) == 1)
{
    int tapMask = int.Parse(((report.Data[4] << 8) + report.Data[5]).ToString()); //MSB is if tap occured
    Console.WriteLine("Report ID: " + report.ReportId.ToString() + " Screen number: " + report.Data[0].ToString() + " Reserved Byte "
        + report.Data[1].ToString() + " Event ID: " + report.Data[2].ToString() + " Encode Value " + report.Data[3].ToString()
        + " Tap Mask: " + tapMask.ToString() + "Swipe Mask: " + report.Data[6].ToString()); //MSB is if swipe occured
}
```

Figur 4.1: Magtek Card Reader eksempel skrevet om

4.4 Standard windows tastetrykk implementert for Encoderen

Det var ulike ting bedriften ønsket at Encoderen skulle gjøre, men de ønsket at alle skulle være allerede definerte funksjoner fra Windows. Dette førte til at dataene Encoderen sendte ut ble tenkt, ved hjelp av koden, oversatt og omformulert til standard Windows tastetrykk. Det var da mulig å anvende Encoderen med en rekke programmer. I tillegg er det lettere å implementere Encoderen over flere plattformer, applikasjoner og kunder. De standard tastetrykk som ble brukt i koden var enter for valg av merkert applikasjon på dataskjermen. Berøring av encoderskjermen ble implementert som enter. Det ble også brukt piltasttrykk for å navigere menyen til navigasjonssystemet til MT. Rotering av kanten på encoderen ble implementert til p bety piltasttrykk. Rotering med klokka ble høyre piltast, mens rotering mot klkka ble venstre piltast.

Kapittel 5

Resultat

I dette kapittelet er resultatene av oppgaven gitt. Det er forklart hva som ble den endelige avgjørelsene og hvorfor valgene ble tatt. Videre arbeid er gitt i neste seksjon.

5.1 Ferdigstilte displaybilder

De ferdig designede displaybildene er vist nedenfor. Displaybildene ble satt opp slik at det var mulig å svipe til høyre å venstre mellom dem. Dette med unntak av ECDIS displaybildet som benytter svipe bevegelsene til å flytte kartet. For ECDIS er det ikke mulig å svipe tilbake til forrige eller til neste displaybildet. I dette bildet må hjemknappen berøres for å komme ut av funksjonen. Siden Encoderen skulle først brukes til å navigere i menyen på navigasjonssystemet til MT, ble det valgt å lage en hjemskjerm som i figur 5.1. Det er ikke definert noen touchsoner, da et touch hvor som helst på displaybildet skal tilsvare et enter klikk på tastaturet for å velge applikasjonen som er markert på dataskjermen. De vil bytte mellom de ulike ikonene på dataskjermen ved å rotere på hjulet, det er derfor ikke nødvendig med en mer kompleks displaybilde enn denne.

Autopiloten kan styres fra ulike applikasjoner og det som skal endres er gradene. displaybildet i figur 5.2 tar høyde for dette. Ved å rotere på kanten, endres gradene. Det er mulig å gå tilbake til applikasjonen til Grayhill og endre slik

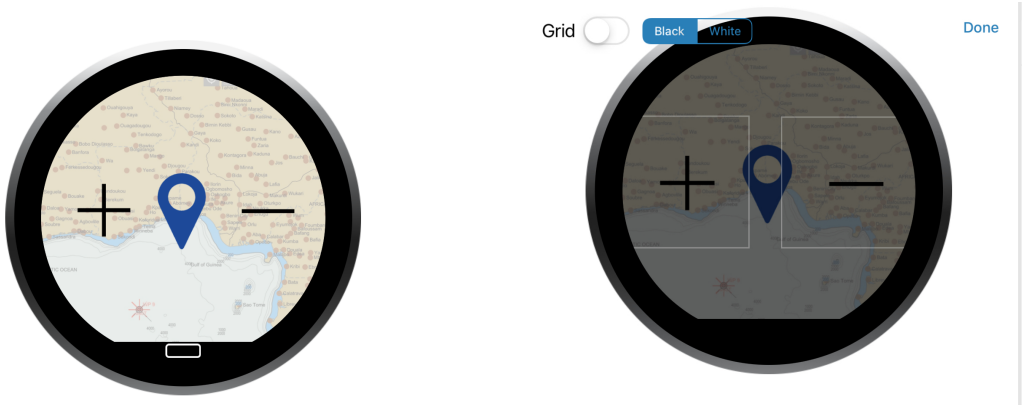
at gradene endres med ulike intervaller. Den som er på figuren er satt til å hoppe 1 grad av gangen og går fra 0 til 360 grader. I visse tilfeller vil det være nødvendig å endre intervallet til 0.5 grad o.l.

Figur 5.1: Hjemskjerm

Figur 5.2: Displaybilde egnet for autopilot endring

5.1 Ferdigstilte displaybilder

Hvordan displaybildet for ECDIS ble seende ut er vist i figur 5.3 (a). På dette displaybildet måtte båten kunne zoome inn og ut i kartet, dette ble løst ved å legge inn touchsoner på displaybildet som vises i figur 5.3 (b). Ytterligere var det ønskelig å svipe rundt i kartet for å endre området som vises. Sviping på displayet vil derfor flytte kartet en definert distanse. Denne distansen vil kunne spesifiseres for hver båt, alt etter hva som er nødvendig. Bytene som tilsier hvilken vei på displayet som har blitt svipet er definert i vedlegg B og kunne direkte hentes inn i koden.



(a) Displaybilde egnet for ECDIS (kart) (b) Touch soner på displaybilde til ECDIS

Figur 5.3: ECDIS skjerm

Den siste applikasjonen som bedriften etterspurte displaybilde til, var radar. Displaybildet er vist i figur 5.4. (a) viser hvordan displaybildet ser ut mens (b) viser touch sonene på displaybildet. I applikasjonen var det mulig, ved hjelp av touchskjerm eller mus, å trykke på pluss eller minus for å minke eller øke range. Dette displaybildet kunne yte samme funksjon. “Gain”, “rain” og “sea” er barer som kunne bli styrt ved hjelp av displaybildet i figur 5.5. Denne går fra 0 til 100% og kan derfor implementeres til alle 3.

5.1 Ferdigstilte displaybilder

(a) Displaybilde egnet for radar (b) Touch soner på displaybilde til radar

Figur 5.4: Radar skjerm

Figur 5.5: Displaybilde egnet for styring av radar

5.2 Ferdigstilt grensesnitt for kommunikasjon mellom encoder og pc

Grensesnittet ble lagt i Visiual Studios og den ferdigstilte koden er vist i vedlagt zipfil representert av bindersen.  . Denne er også lagt ved på slutten av besvarelsen i vedlegg E.

5.2.1 Oppsett

Den mest ryddige og oversiktlige måten og ha oppsettet på ble funnet til å være at grensesnittet først sjekker hvilken skjerm som er aktiv på encoderen. Hvert alternativ for skjermene ble laget som en funksjon. Inne i hver funksjon ble det lagt inn de nødvendige handlinger for gjeldende applikasjon. For eksempel ble det i ECDIS lagt inn kode som reagerer på rotering av kanten, dette skal resultere i at det blir zoomet inn og ut i kartet. De to berøringssonene som er lagt inn for pluss og minus ble også implementert i fiksjonen for ECDIS. Berøring av disse skal også resultere i at kartet blir zoomet inn eller ut. En siste ting som ble lagt inn i ECDIS funksjonen var sviping. Ved å svipe enten opp, ned eller til en av sidene, skal karetet flyttes tilsvarende.

I slutten av hver funksjon ble det sjekket om den gjeldene displaybildet enda var aktiv på encoderen. Hele funksjonen for hver skjerm ble satt opp som en if-løkke slik at den hele tiden kjørte gjennom if-løkken om det var gjeldene dislpaybildet var aktivt. Så snart det ikke var aktivt lenger ble løkken brutt og grensesnittet returnerte til hovedkoden for å sjekke hvilken skjerm som nå var aktiv.

Siden sviping ble valgt på encoderen for å bytte mellom de ulike skjermene og det er en av hendelsene som blir brukt av ECDIS funksjonen, ble dette displaybildet låst til at du kun kommer ut av det på encoderen ved å trykke på hjemknappen. På resterende displaybilder er det mulig å svipe mellom de ulike displaybildene.

5.2.2 Overordnet innblikk i skriptet for grensensittet mellom encoder og pc

Det ble valgt å bruke funksjoner for hver av de ulike dislplaybildene eller applikasjonene da denne koden kun blir utført når den blir kalt på. Det er i tillegg mulig å samle en bolk med kode som skal kjøres flere ganger og bare kalle på funksjonen når kodebolken trengs, i stedet for å skrive hele koden hver gang den skal kjøres. Dette fører til at koden er mer oversiktlig og ryddig og dermed enklere for andre og endre på i senere tid.

Dataene som kommer inn fra encoderen er i form av en lang bitstreng. I denne bitstrengen er all informasjon som trengs for å oversette dataene som kommer fra encoderen. Bittene er satt av til ulike formål, som å fortelle hvilken Report ID som er aktiv og om ulike funksjoner er aktive. De bittene som forteller om for eksempel berøring av skjermen har skjedd, er etterfulgt av bit som forteller hvor på skjermen berøringen skjedde.

De bittene som forteller om hendelsen har tatt sted er brukt på starten av if-løkkene for å fortelle skriptet om det skal kjøre løkken, de påfølgende bittene er satt inn i løkken for å fortelle hvor mye endring som har tatt sted og da hvor mye verdien på gjeldende displaybilde skal justeres.

5.2.3 Oversikt over applikasjonene implementert fra MT sitt navigasjonssystem

Utenom ECDIS applikasjonen som er nevnt tidligere i resultat seksjonen, er det lagt inn kode for autopilot, radar, med range og rain, sea og gain, og hjemskjerm. Det er også lagt inn en funksjon for sviping.

Når navigasjonssystemet og encoderen er i hjemskjermen er det ønskelig å kunne navigere menyen på dataskjermen med MT sitt navigasjonssystem på. Det ble derfor lagt inn at berøring hvor som helst på skjermen tilsvarer valg av applikasjonen som er markert på dataskjermen. Det ble også lagt inn at å rotere kanten med klokka tilsvarer å markere applikasjonen til høyre for den som er markert på dataskjermen, og rotere mot klokka tilsvarer å markere applikasjonen til ventre for den som er markert. Dette ble gjort fordi da kan kapteinen rotere kanten for å navigere seg gjennom applikasjonene på data-

skjermen og deretter velge den markerte applikasjonen ved å berøre encoder displayet.

For autopilot ble det kun lagt inn funksjon som reagerer på rotering av kanten på encoderen. Rotering med klokka justerer gradene for autopilot, eller da kurs, oppover like mange hakk som kanten er rotert. Samme for å rotere kanten mot klokka, da minker gradene. Det er lagt inn at det skal være mulig å rotere gradene mer enn en gang hver gang encoder kanten roteres for at det skal gå fortere å justere kursen. Koden legger derfor inn ett hopp på like mange hakk som er rotert, enten i positiv eller negativ retning.

For radar er det laget en funksjon, men inne i denne funksjonen sjekkes det om encoderen er i displaybildet for range eller det som er lagt for å stille på rain, gain og sea. For range ble det lagt inn samme konsept som i autopilot. Rangen øker eller minker like mange hakk som kanten blir rotert. I dette tilfellet økes range med andre intervaller, men det er intervallet som stiger med en for hver gang kanten roteres ett hakk. På skjermen vises disse intervallene som null til hundre prosent av maks range. I tillegg til å kunne justere range med å rotere kanten, er det mulig å berøre de to touch sonene for å minke eller øke rangen. Ved å berøre skjermen økes rangen kun med ett intervall hver gang displayet blir berørt.

For å justere rain, gain og sea ble det lagt inn en bar som går fra 0 til 100 prosent. Denne kan stilles på ved å rotere kanten som for autopilot og range. Dette for at det skal være mulig å justere det fortere enn bare en og en prosent av gangen, ved å rotere encoder kanten flere hakk.

Det ble laget en egen funksjon for sviping for å gjøre koden i ECDIS enklere og mer oversiktlig. I denne funksjonen ligger definisjonene for hva som skal skje når de ulike sviperetningene blir detektert. Det ligger inne til å flytte kartet i retningen som blir svipet.

Kapittel 6

Diskusjon og konklusjon

6.1 Diskusjon

Grensesnittet for informasjonen ut av koden er laget med C# kodespråk. Det er laget funksjoner for de ulike applikasjonene som var ønskelig og det er lagt inn hvordan disse kan styres ved hjelp av touch encoderen. De ulike hendelsene som kan gjøres på touch encoderen er implementert på forskjellige måter for å kunne navigere i og justere MT sitt navigasjonssystem. Applikasjonene som koden er implementert fro er autopilot, ECDIS og radar. Det er ikke samme måte som er lagt inn for å justere hver av alternativene, men heller det som er mest gunstig for gitt applikasjon. I noen tilfeller var det nødvendig å justere små sprang og navigasjonssystemet trengte å være følsomt for dataene som kom fra encoderen. I andre tilfeller som for å justere range til radaren, er det normalt sett ett og ett steg som skal justeres om gangen og det er derfor ikke like viktig at navigasjonssystemet er like følsomt for hvor mye endringer som er gjort på encoderen, men heller at det er gjort en endring for så å justere da med ett steg som følge av det.

Hovedproblemet med oppgaven ble å kunne implementere dataene som PCen leste inn fra encoderen i en kode. Det ble benyttet et HID bibliotek og tatt utgangspunkt fra et allerede laget eksempel til å komme frem til den endelige koden som kan lese inn og bruke dataene.

Det ble laget displaybilder passende til de tre applikasjonene det var ønskelig for. De er alle skreddersydde slik at det er mulig å utføre nødvendige justeringer i dem under manøvrering av båten. Verktøyet, eller da applikasjonen til grayhill, som følgte med encoderen ble brukt for å lage displaybildene. Det er også mulig å gå tilbake til applikasjonen å gjøre nødvendige justeringer i displaybildene for å tilpasse for eksempel intervalllengde o.l.

6.2 Konklusjon

Arbeidet startet med å bli kjent med encoderen og den tilhørende applikasjon for å vite mer om hva som var mulig å gjøre. Neste steg var å få til en lesbar kommunikasjon mellom encoderen og en pc. For å komme frem til det endelige oppsettet av koden ble det prøvd ulike oppsett, kommandoer og fremgangsmåter og Kjell Erik Larsen bidro med mange gode forslag her. Når det kommer til design av skjermene ble disse laget til det bruket som de i dag er ønsket til. De er mulige å utbedre om ønskelig lett ved hjelp av applikasjonen til Grayhill. Dette hvis det er ønskelig å fjerne skjermer da noen av fartøyene ikke trenger alle, det er også mulig å redigere steglengder og lignende.

Etter den informasjonen som ble gitt av bedriften er det er et godt oppsett av kode som er lett å gå inn å legge til de ønskelige utdata og snarveier. Det er gode og utfyllende kommentarer lagt ved slik at en person som senere ønsker å utbedre, endre eller videreutvikle koden kan forstå hva som er gjort og hensikten med de ulike funksjonene. det er da mulig å legge til flere funksjoner og endre de som alt er lagt til hvis det blir endringer i navigasjonssystemet til MT. Det er også mulig å legge til den informasjonen som mangler fra applikasjonene som ikke er tilgjengelig per dags dato.

Skjermene som ble lagtet er enkle i utformingen for å hindre mest mulig frustrasjon og feiltrykk ved bruk. Det er enkle deler av applikasjonen som er ønskelig å endre og justere med disse skjermene og det passer derfor at de også er enkle i utseende og bruk. Det er få touchsoner og store områder til å gjøre justeringene med og de ser enkle ut.

6.3 Videre arbeid

Siden bedriften får to av applikasjonene fra andre bedrifter, er ikke denne koden tilgjengelig per dags dato. Grensesnittet og encoderen kan implementeres først når koden er tilgjengelig sli kat det er mulig å vit hva dataene fra encoderen skal oversettes til, eller da hva endringer gjort på encoderen skal føre til i praksis i navigasjonssystemet. Hvis rotering av kanten på encoderen skal bety at applikasjonen i navigasjonssystemet skal øke eller minke kursen, må det være klart hvordan dette kommuniseres på dataskjermen allerede. Når den informasjonen er klar, er det mulig å legge inn at en endring i disse bittene i dataene fra encoderen skal bety en justering av kursen i navigasjonssystemet.

Det vil også være mulig å implementere de inndata og utdata funksjonene som allerede er laget i navigasjonssystemet til MT. Det er ulike koder og tall som betyr ulike ting i systemer deres. Med et litt mer omfattende grensesnitt kan disse tas høyde for å benyttes. Hvilke koder de har definert allerede er vist i vedlegg C.

6.3.1 Grensesnitt fra navigasjonssystemet til standard Windows tastetrykk

Som nevnt i oppgaveteksten er det ønskelig at kommunikasjonen mellom navigasjonssystemet og encoderen skal være toveis. Altså at du skal kunne justere verdiene på dataskjermen med encoderen, men også at encoderen kan lese inn den nåværende eller nye verdien på dataskjermen sli kat den kan vises på encoderdisplayet. Dette ønsker bedriften skulle oppnås ved hjelp av standard Windows tastetrykk som enter, piltast og snarveier. Foreløpig er ikke standard Windows tastetrykk implementert i navigasjonssystemet. For at det skal være mulig for navigasjonssystemet og forstå den informasjonene som kommer fra grensesnittet fra encoderen med standard Windows tastetrykk må det først lages et grensesnitt fra systemet som sier hva de tastetrykkene skal bety. Ved å benytte standard Windows tastetrykk vil det være enkelt å implementere encoderen på flere bruksområder og lett å lage den til for de ulike prosjektene som dukker opp i fremtiden.

Det vil i tillegg være mulig å lage til grensenittet til at encoderen henter inn

informasjonen som kommer fra grensesnittet til navigasjonssystemet og bruke det til og oppdatere informasjonen på encoderdisplayet. Dette vil være nyttig når du åpner displaybildet som for eksempel hører til autopilot på encoderen, da er det nødvendig for encoderen å vite hvilken kurs som skipet allerede er på. Hvis den vet det vil den kunne vise riktig informasjon på displayet. Det er ikke essensielt for å bruke encoderen for å justere kursen til båten da koden sørger for at det er mulig å justere den opp og ned uansett hvilken verdi som vises på skjermen. Det vil dog være nyttig for kapteinen å ha riktig informasjon begge steder og kunne lese kursen på encoderen.

6.3.2 Videre utbedring av alledere implementerte hendelser fra encoderen

For ECDIS er det lagt inn at sviping skal flytte kartet et hakk opp, ned, eller til en av sidene. Det må også legges inn hvor langt ett hakk skal tilsvare. Distansen som er ønskelig å flytte vil kunne variere fra prosjekt til prosjekt og skulle ha endret seg etter hvilken zoom som er i kartet. Dette grunnet at steget vil flytte skjermen mye om den er langt zoomet inn og lite om den er langt zoomet ut. Steget som blir satt inn kan derfor bli lagt til hvor mange cm på skjermen det skal flyttes for eksempel i stedet for distanse i kartet.

Det er også mulig å benytte alle berøringssonene i stedet for sviping til å flytte kartet. Det som ville vært ønskelig da er å berøre displayet på encoderen og bruke denne på samme måte som en datamus. Det vil kreve høyfrekvent oppdatering av informasjonen fra encoderen, noe den allerede gjør og detaljert kode for at flytting over de ulike berøringssonene skal føre til at musepekeren flyttes over dataskjermen. Ved å implementere kode som utfører dette i grensesnittet fra encoderen, er det mulig å legge denne inn overordnet slik at den kan brukes over alle applikasjonene.

Det vil også være mulig å for eksempel velge hvilken av de tre valgene, gain, rain og sea, du ønsker å stille på. Da disse er i samme applikasjon og det ikke per dags dato er kode tilgjengelig for å finne ut hvordan de opereres, er det vanskelig å vite hvordan de skal legges inn i grensensittet fra encoderen. Når bedriften får denne koden tilgjengelig vil det være fullt mulig å gå inn i koden å fortelle hva som skal skje på dataskjermen, hvilken av de tre alternativene som skal justeres, når det skjer endringer på encoderen.

6.3 Videre arbeid

Slik koden i grensensittet er satt opp nå med kommentarer og beskrivelser er det fullt mulig å gå inn å gjøre endringer siden. Dette er gjort for at det enkelt skal være mulig å gå inn å skreddersy grensesnittet for hver båt eller hvert prosjekt. Det er også fullt mulig å bruke dokumentet i vedlegg B for å legge til eller endre ønsket følge av de ulike endringene som kan gjøres på encoderen, enten det er å benytte sviping, berøring, rotere kanten eller hjem knappen.

Det er også enkelt å gå inn å endre, legge til og fjerne displayskjermer for båtene den skal bli brukt til og bare laste opp det nye prosjektet på encoderen.

Bibliografi

[1] Grayhill, *Grayhill Touch Encoder article Featured on Gadgeteer*, Tilgjengelig ved https://www.grayhill.com/post/grayhill-touch-encoder-article-featured-on-gadgeteer (2020).

[2] M. Technologies, *About Marine Technologies*, Tilgjengelig ved http://www.marine-technologies.com/company.html (2021).

[3] C. Wolff, *Range or distance measurement*, Tilgjengelig ved https://www.radartutorial.eu/01.basics/Distance-determination.en.html (2021).

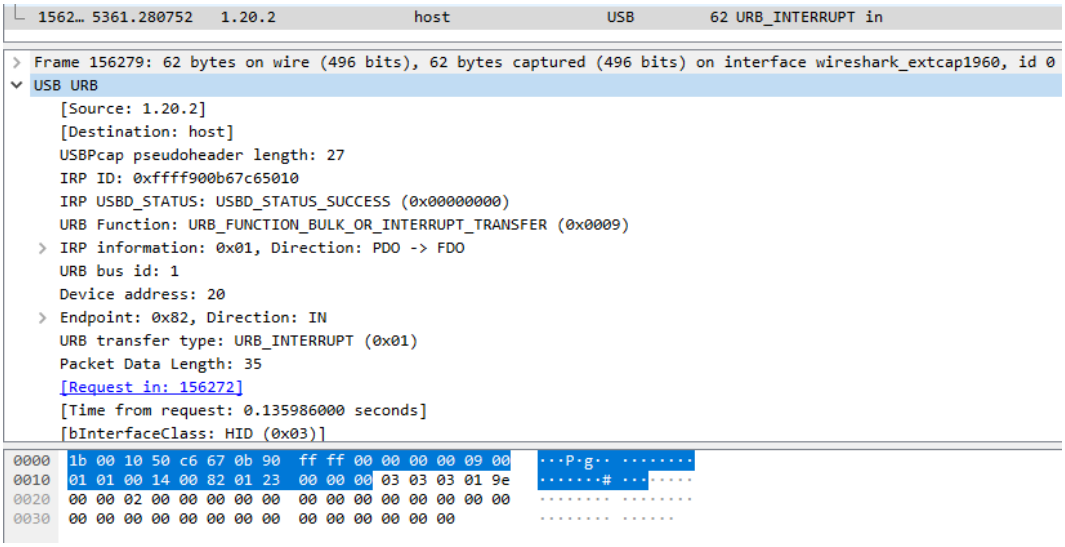
[4] C. R. Jassal, *Marine radar: How best to set up to have it plot perfect targets on screen*, Tilgjengelig ved https://www.myseatime.com/blog/detail/marine-radar-how-best-to-set-up-to-have-a-perfect-targets-on-screen (2016).

[5] L-com, *M12 5 Position A-Coded Male/Female Cable, 3.0m*, Tilgjengelig ved https://www.l-com.com/ethernet-m12-5-position-a-coded-male-female-cable-30m (2021).

[6] I. Grayhill, *GH TE-MX*, Tilgjengelig ved https://apps.apple.com/us/app/gh-te-mx/id1459459728 (2021).

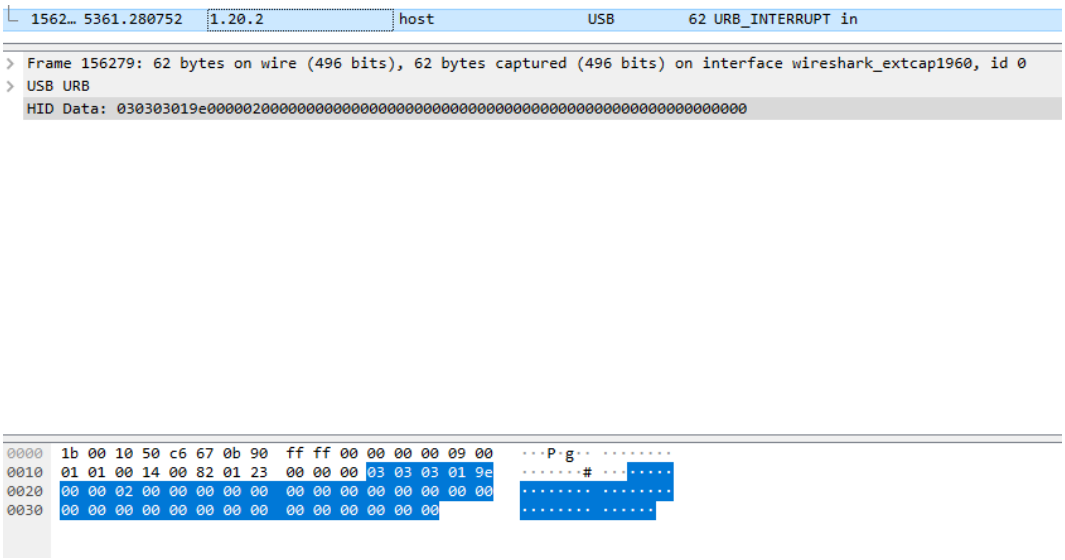
Vedlegg A

Data i Wireshark



Figur A.1: Standard USB kommunikajsondata i Wireshark

Data i Wireshark



Figur A.2: Data fra Encoderen i Wireshark

Vedlegg B

Interface Control Document (USB)

Grayhill

Intuitive Human Interface Solutions

Touch Encoder

Interface Control Document (USB)

TE OS Version 1.0.0 or greater (includes UtilityApp)

Revision History

Revision	Date	Description
A	3/27/2018	Original Release
B	7/12/2018	Changed pinout on M12 to reflect production pinout
C	8/24/2018	Added Programming Harness Information
D	10/25/2018	Updated Run/Programming Mode Schematics
E	12/7/2018	Updated Touch Encoder Image
F	4/23/2019	Added Command, Multi-value, and Display Code Information

Table of Contents

1. Overview

The USB interface of the Touch Encoder product is designed to conform to the USB 2.0 specification and, specifically, the USB HID and the USB Mass Storage Device protocols. Both protocols allows system integrators to retrieve the data from, or send data to, the device using the ubiquitous USB HID/Mass Storage Device support in the host OS. For more information about the USB standard or the USB HID/Mass Storage Device support in your specific host OS, please visit the USB web site (usb.org) or contact the OS vendor.

The Touch Encoder device connects to the host as a composite USB device with two Generic HID interfaces and one USB Mass Storage Device interface. The HID interfaces consist of several top-level collections (TLC) to virtually separate different device functions. Each report type generated by either the host (OUT) or the device (IN) will have a unique report ID to denote which interface and TLC it is associated with.

The Generic HID interfaces are used to send data to the host application. These interfaces also allow the host application to make changes to the configuration of the device, including screen/widget settings.

The Mass Storage Device interface is used to support the upload of firmware updates or configuration settings to the device.

In order to identify the Touch Encoder within the OS, use the following USB Vendor ID (VID), Product ID (PID) combination:

VID:	0x1658	Grayhill, Inc.
	0x0060	Touch Encoder, 2nd generation Touch Encoder

2. Generic HID Interfaces

The Touch Encoder device's Generic HID Interfaces are designed to interface with the generic HID support in the host OS. This means that the host OS does not consume the data itself, but that the data needs to be retrieved (IN report) or transmitted (OUT report) by an application or a driver running on the host device.

When new information is available from the device, a new report of the appropriate type is generated. Each report type can be generated by the device at a maximum rate of every 10 ms, although it is possible that the device generates mixed-type reports at a faster rate.

The different types of USB reports for this interface are explained below.

3. USB Reports

3.1 IN: Events Data Report (Interface #1, Collection #1, Report ID 1)

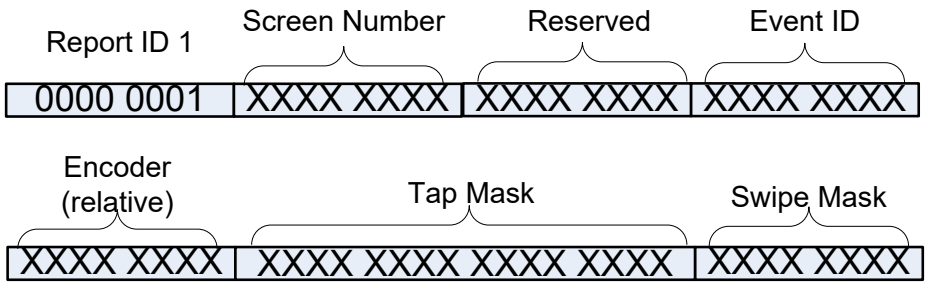


Figure 1 – Events Data Report

The Events Report is the lowest level report generated by the Touch Encoder device. Even though the Touch Encoder is designed to automatically respond to events occurring on the device, it will use this type of report to keep the host application informed about those events as well. In addition to the Events Data report, the device will also send a Widgets Data (IN) report to the host when an event causes a new widget/screen to be displayed on the device or causes one of the widget Values to change. This method of sending a widget report as well as the underlying event report is implemented in order to allow the host to closely monitor the activity of the device, and for the host to respond appropriately should a data mismatch occur.

The Events report is 8 bytes long and contains the report ID byte followed by the current event data. The Event data consists of a Screen Number byte, a reserved byte, an Event ID byte, an Encoder byte, a Tap Mask word and a Swipe byte.

The Encoder byte is a signed 8-bit value containing the relative change in encoder position since the last event report. It uses a special signing, like that used in the Touch Encoder CAN protocol, where the 0 value is 0x80. For example, if the encoder was turned two detents in the positive (CW) direction since the last report, the Encoder byte would be 0x82. Similarly, if the encoder was turned three detents in the negative (CCW) direction since the last report, the Encoder byte would be 0x7D (Grayhilll's notation for -3). In case the encoder was not turned but the Events report was transmitted because another event value changed, then the encoder byte would remain 0x80.

The Tap Mask word is a 16-bit value containing the tap information currently available. The most significant bit of the 16-bit value shows whether or not a tap occurred since the last Event report. The remaining 15 bits contain a bit mask of the tap zones the tap occurred in. The bit mask is defined as follows:

Bit0	Zone 0
Bit1	Zone 1
Bit2	Zone 2
Bit3	Zone 3
Bit4	Zone 4
Bit5	Zone 5
Bit6	Zone 6
Bit7	Zone 7
Bit8	Zone 8
Bit9	Zone 9
Bit10	Zone 10
Bit11	Zone 11
Bit12	Zone 12
Bit13	Zone 13
Bit14	Zone 14

The Swipe Mask byte is an 8-byte value containing the swipe information currently available. The most significant bit of this byte shows whether or not a swipe occurred since the last Event report. The least significant 4 bits contain a bit mask of the direction in which the swipe occurred. The bit mask is defined as follows:

Bit0	Up
Bit1	Down
Bit2	Left
Bit3	Right

3.2 OUT: Command Report (Interface #1, Collection #2, Report ID 2)

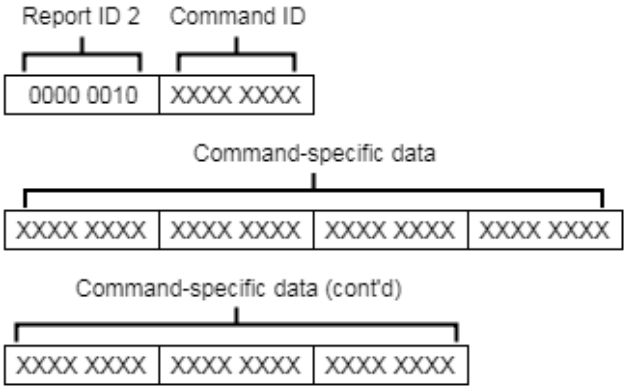


Figure 2 – Command Report

The Command Report is the lowest level control available to the USB host controller. It allows modification of system-level attributes, such as the display's backlight level.

The Command Report is 9 bytes long, with the first byte being the Report ID, the second byte being the Command ID, and the remaining 7 bytes being reserved for any Command-specific data.

The 8 bytes following the Report ID are meant to emulate the 8 byte Command messages in the Touch Encoder's CAN J1939 protocol, simplifying documentation and implementation for both the host device and the Touch Encoder.

3.2.1 Backlight Control

The backlight control command report uses a Command ID of 0x80 (128) and the second Command-specific data byte as the new percentage of backlight brightness.

The permitted range for the new percentage (data byte 2) is from 0x00 (0%) to 0x64 (100%).

As an example, the report to change the backlight to 100% would look like:

Interface #1, Collection #2, OUT: [02 xx 64 xx xx xx xx xx xx]

(where 'xx' bytes can be any value)

Similarly, the report to change the backlight to 30% would look like:

Interface #1, Collection #2, OUT: [02 xx 1E xx xx xx xx xx xx]

3.3 IN: Widget Data Report (Interface #2, Collection #1, Report ID 3)

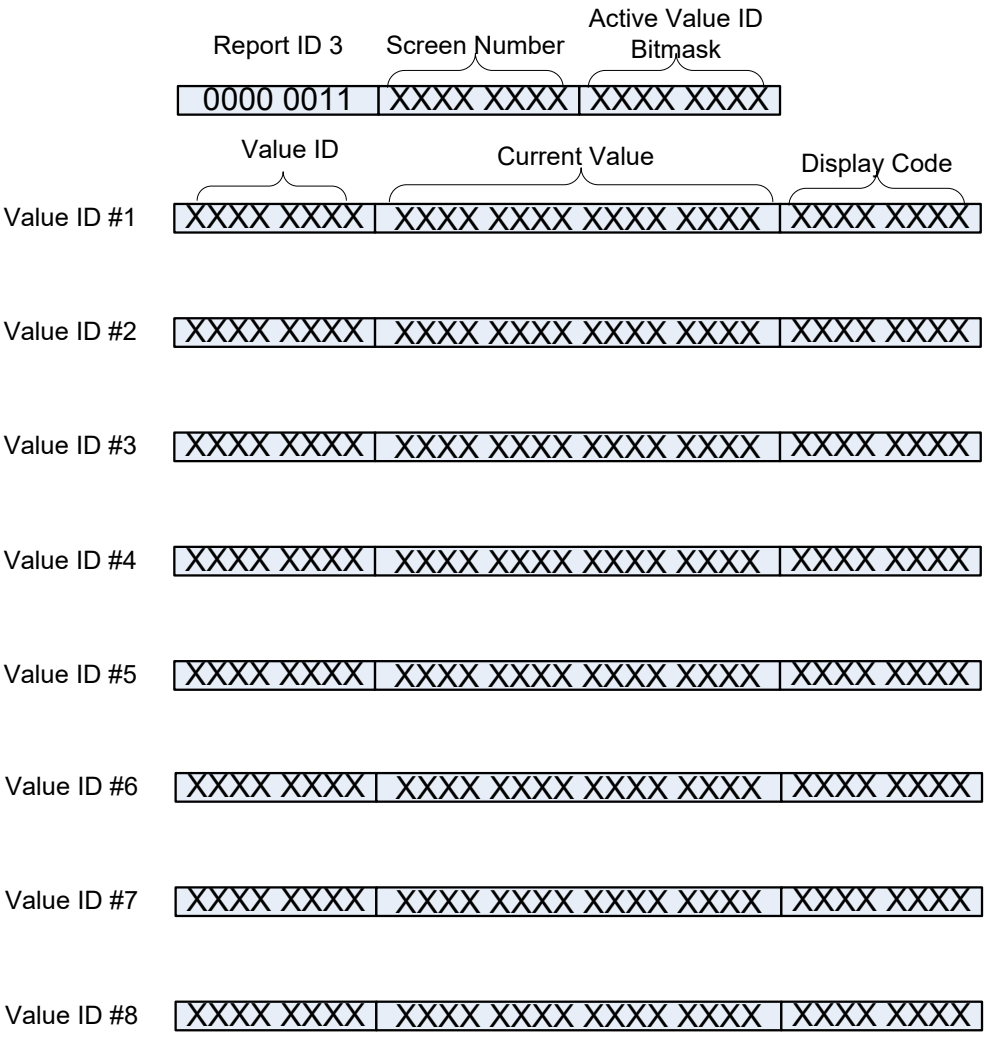


Figure 3 – Widget Data Report (IN)

This type of report is sent automatically by the device to the host whenever an event on the device causes a new screen to be displayed or causes one of the current screen's "Value ID" values to change. This type of report is used to keep the host informed about the widgets and "in-sync" with the device.

The Screen Number byte is an 8-bit value containing the screen number currently being displayed on the device.

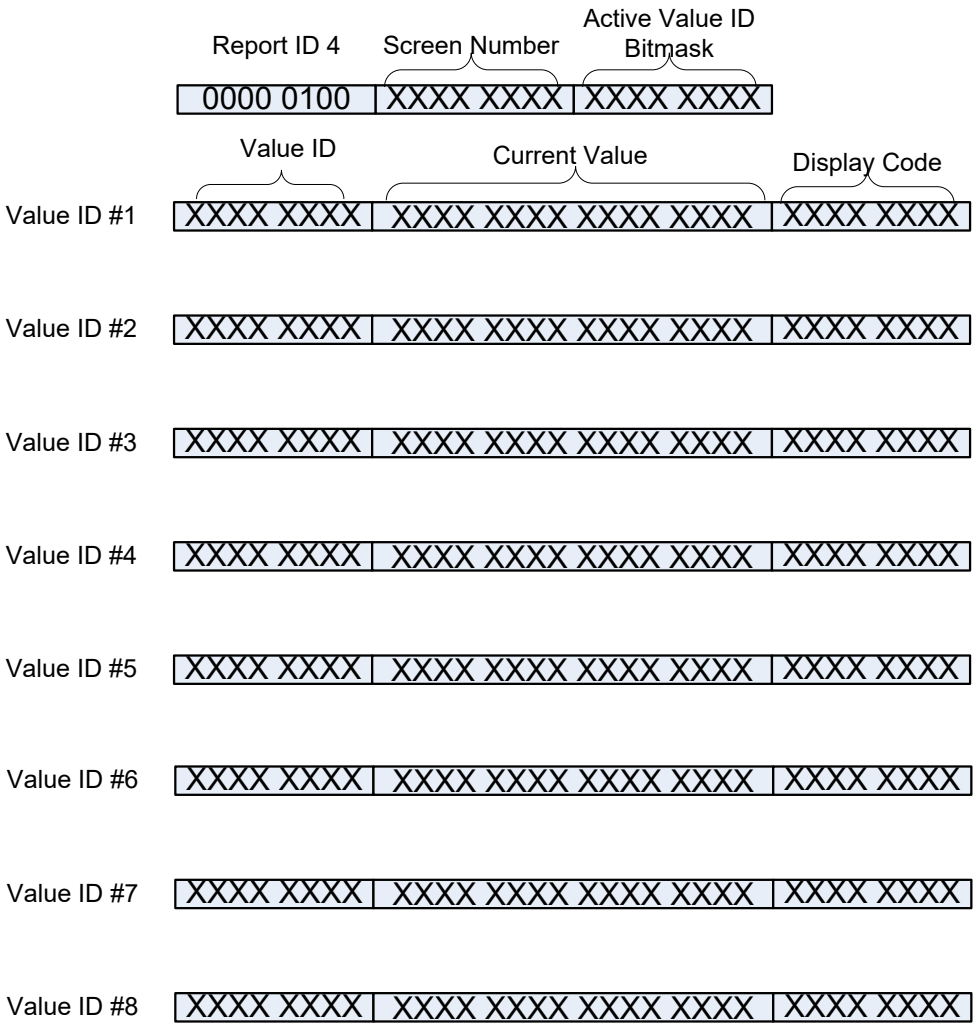
The Active Value ID Bitmask byte is a bitmask of the individual Value IDs that are currently active on the screen. There are 8 Value IDs available per screen ranging from 0x01 to 0x80. For each of the Value IDs listed in the Active Value ID Bitmask, one of the following Value ID Information Sections is populated.

The Value ID Information Section contains current value information for each of the up to eight active Value IDs on the current screen. Each field in this section contains a Value ID byte, a Current Value associated with that Value ID and a Display Code.

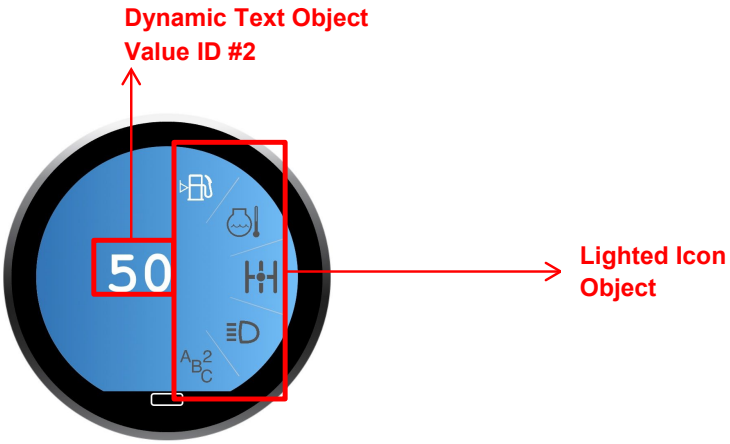
The Current Value field is constrained by the initial conditions, minimum and maximum values, and the step constraints defined during the design stage in the IDE. However, the host has the ability to overwrite or initialize the Current Value field.

The Display Code field contains a code specifying the format to apply to the Current Value before it is displayed on the widget.

3.4 OUT: Force Widget Data Report (Interface #2, Collection #2, Report ID 4



3.4.1 Multi-Value Data Example



Example: The figure above displays an example of a multi-value widget.

The dynamic text object is designated at Value ID #2. The lighted icon object is designated as Value ID #3.

Below is the sequence of messages to turn on the top lighted icon and change the dynamic text to 100 (note that the lighted icon object has an offset of 0x8000).

Interface #2, Collection #2, OUT: [04 03 06
xx xx xx xx
02 64 00 00
04 01 80 00
xx xx xx xx
xx xx xx xx
xx xx xx xx
xx xx xx xx]

(where 'xx' bytes can be any value)

3.4.2 Display Code and Decimal Code

The Display Code can allow the Touch Encoder to display decimal values within text boxes, while still using the 16-bit integer values for the respective Value ID's.

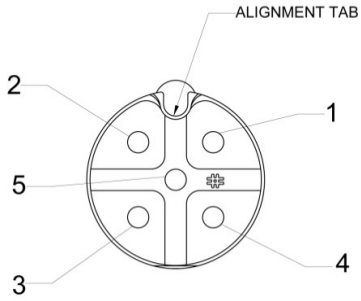
To do this, we use the top 4 bits of the Display Code as a signed 4-bit integer (which we call the Decimal Code). This integer is used as an exponent value with a base of 10, like in scientific notation.

As a quick note, this Decimal Code is only relevant for Value ID's which are used by text boxes and the remaining 4 bits are currently reserved for future uses.

Display Code Byte

Display Code	Input	Output
0x1X	Integer x 10	e.g. 10=100
0x2X	Integer x 100	e.g. 10=1000
0xEX	Integer ÷ 100	e.g. 10=0.10
0xFX	Integer ÷ 10	e.g. 10=1.0

3.5 Physical Layer



	USB	CAN
1	MODE	MODE
2	VIN	VIN
3	GND	GND
4	USB_D+	CAN_H
5	USB_D-	CAN_L

For Touch Encoder serial numbers less than A100000 please contact Grayhill for pinout detail.

If MODE Pin is floating at power up, the Touch Encoder will assume run mode operation. If Mode pin is connected to GND externally at startup, the Touch Encoder will assume programming mode. The Touch Encoder will download updates from a USB mass storage device (if connected) and update the Touch Encoder accordingly.

Run Mode:

See previous page: Connector Pinouts

USB RUN MODE

TOUCH ENCODER		USB HOST
VIN **2**	———	**1** 4.5V-5.5V
USB_D- **5**	———	**2** D-
USB_D+ **4**	———	**3** D+
GND **3**	———	**4** GND
MODE **1**		

Programming Mode:

PROGRAMMING MODE

See previous page: Connector Pinouts

TOUCH ENCODER		USB FLASH
VIN **2**	———	**1** 4.5V-5.5V
USB_D-/CAN_L **5**	———	**2** USB_D-
USB_D+/CAN_H **4**	———	**3** USB_D+
GND **3**	———	**4** GND
MODE **1**		

POWER SUPPLY

1 4.5V-5.5V

2 GND

Recommended Flash Drives:
Verbatim 49304
SanDisk SDCZ60-016G-B35

Note: Load files to FLASH drive and plug into socket. Touch Encoder only samples MODE pin during power up, so be sure this occurs after connecting harness and FLASH drive.

4. Mouse HID Interface (Coming Soon)

The Mouse HID interface of the Touch Encoder is designed to interface directly with the mouse driver of the host OS. The interface generates USB reports that contain relative motion, as well as left, and right mouse button click data. Since only one type of report is used in this interface, no Report ID is included in Mouse HID reports.

The firmware on the Touch Encoder is responsible for processing the hardware button data and the touch data reported by the touch controller, and converting this data into the mouse data format. This processing includes converting individual touch points (as reported by the touch controller) to relative mouse motion data, smoothing the motion data to reduce noise while keeping the processing latency as low as possible, and calculating the duration of individual touches to determine if a tap or other single-touch gesture occurred.

The supported single-touch gestures are as follows:

Tap – touch did not move significantly and was shorter than approximately 360 milliseconds in duration

Drag Enable – touch did not move significantly and was longer than 1 second in but shorter than 2 seconds in duration

Right-Click – touch did not move significantly and was longer than 2 seconds in duration

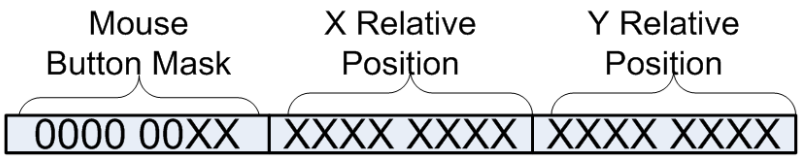
The “Tap” gesture is realized by sending a single report with the left-click button active, which also allows the use of a double-tap to generate a double-click.

If the “Drag Enable” gesture mentioned above is followed by (significant) relative motion of the touch on the touch pad surface, then the firmware will send persistent left-click reports for as long as the touch is active on the pad. This allows for items to be “pinned” by performing the gesture and then moving them across the screen. The dragging stops when the touch is lifted off the screen.

The “Right-Click” gesture mentioned above causes the device to send a single report with the right-click button active. This results in a right-click menu being opened. The user can then use simple touch motion to move the mouse cursor to the desired menu selection.

5. USB Reports

5.1 IN: Mouse Report (Interface #2)



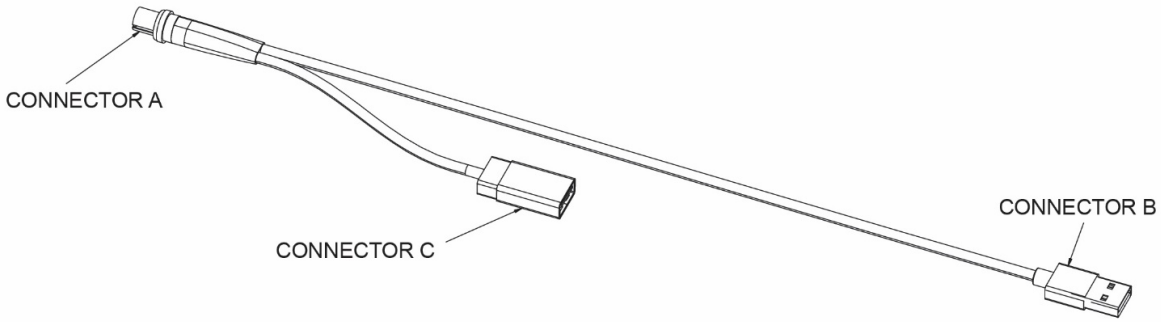
This type of report is 3 bytes long and contains 1 byte of button mask data, followed by 1 byte of relative X position data and 1 byte of relative Y position data. The button mask indicates whether the left (bit 0) or right (bit 1) buttons are either active (1) or inactive (0). The button bits reflect the output of the single-touch gesture recognition mentioned above.

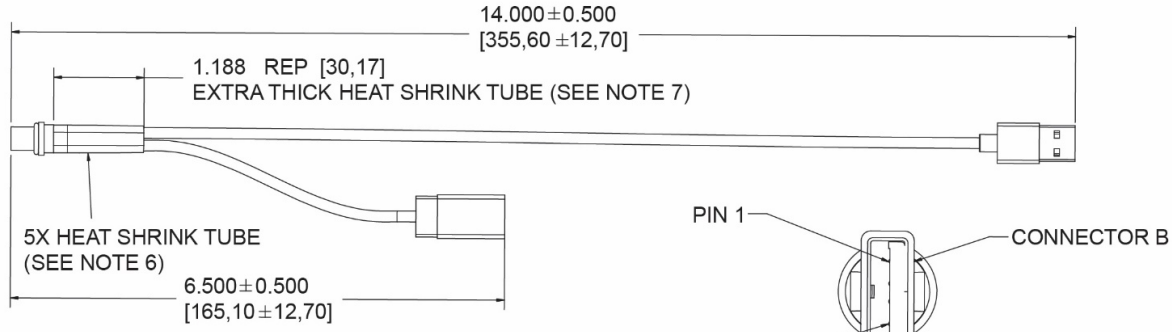
While touches are being sensed on the touch surface, this report is generated every 20 ms.

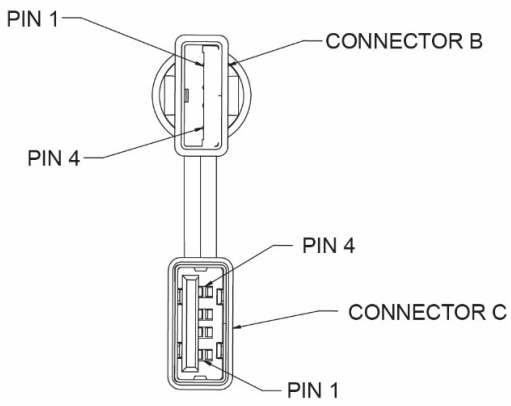
6. Appendix

6.1 Programming Harness

GRAYHILL P/N T18908 - USED FOR PROGRAMMING MODE ONLY



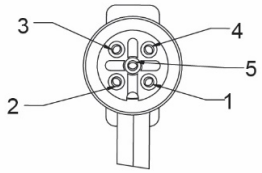




NOTES:

1. CONNECTOR A IS NORCOMP INC. P/N: 858-005-203RSU4 WITH METAL NUT REMOVED.
2. WIRE WITH USB CONNECTORS B & C IS TRIPP LITE P/N: U024-003 OR EQUIVALENT
3. CONNECTOR A PINS TO CONNECT TO CORRESPONDING CONNECTOR B & C TERMINALS ON THE CONNECTOR (SEE TABLE)..
4. WIRE LENGTH SHOULD BE MEASURED WITH WIRES STRAIGHT.
5. WIRE PULLOUT FORCE MUST BE 2 LBS. MINIMUM.
6. 5X HEAT SHRINK TUBING TO BE USED OVER CONNECTOR A CONNECTIONS.
7. EXTRA THICK HEAT SHRINK TUBING TO BE USED OVER CONNECTOR A CONNECTIONS.

WIRE #	CONNECTOR A PIN	CONNECTOR B PIN	CONNECTOR C PIN	COLOR	DESCRIPTION
1	1	N/A	4	BLACK	MODE
2	2	1	1	RED	Vin
3	3	4	4	BLACK	GND
4	4	N/A	3	GREEN	USB+
5	5	N/A	2	WHITE	USB-



Connector A Pinout

561 Hillgrove Avenue
La Grange, IL 60525
web: www.grayhill.com
e-mail: te@grayhill.com
phone: +1 (708) 354-1040

About Grayhill

Grayhill, Inc. is a privately held firm which designs and manufacturers intuitive human interface solutions that make life simpler, safer and more efficient. Standard products include optical and Hall Effect encoders, discrete and Hall Effect joysticks, rotary switches, keypads, and pushbuttons; all with finely tuned haptics. Grayhill specializes in creating ergonomic panels and product shells that integrate various interface technologies, including displays, our components, and gesture recognizing multi-touch technology. With headquarters in La Grange, Illinois, and multiple state-of-the-art facilities around the world, Grayhill's team has the full engineering, product development and manufacturing expertise to deliver both standard and customized products quickly and cost-effectively. To learn more about Grayhill's products and capabilities, visit www.grayhill.com.

Vedlegg C

Third Party Operator Interface

MARINE TECHNOLOGIES

Document title:

Functional Description
Third Party Operator Interface

Document description:

This document describes how a third party can operate the system by using a custom NMEA interface.

	4/11/2014	Added functions	TK		
1.0	12/3/2013	First issue	HH		
Rev.	**Date** mm/dd/yyyy	**Reason for issue**	**Issued by**	**Checked by**	**Approved by**

Table of Content

Document History

Issue no	*Affected Para-graphs*	*Document History*	*Reason for Change*
1.0		First issue	

1. Preface

Document Version Control:

It is the reader's responsibility to ensure they have the latest version of this document. Questions should be directed to the owner of this document, or the project manager.

Document Owner:

The primary contact for questions regarding this document is:

Håvard Hellvik

Marine Technologies LLC

Privacy Information:

This document may contain information of a sensitive nature. This information should not be given to persons other than those who are involved in the project or who will become involved during the lifecycle.

2. Introduction

The Introduction section provides for an executive level overview.

2.1 Overview

An online simulation requires the simulator to be able to control the DP system automatically. This document describes how a third party can operate the system by using a custom NMEA interface.

The custom NMEA messages will be sent/received via UDP Ethernet messages.

2.2 Requirements

Function	Priority	Implemented	ID
Monitor/acknowledge alarms	1	Yes	
Monitor/set thruster enable/disable	1	Yes	600
Monitor/set posref and sensor enable/disable/primary	1	Yes	604, 605
Monitor/set setpoint (position/heading)	1	Yes	602, 603
Monitor/set mode: Standby/Joystick/DP	2	Yes	601
Monitor/set alarm limits	2	Yes	610
Set present position	2	Yes	603
Monitor/set consequence analysis mode	3	Yes	611
Monitor/set joystick mode (along, athwart, heading)	3	Yes	601
Monitor/set joystick command	3	Yes	502
Monitor/set gain settings	2	Yes	606
Monitor/set pivot point	3	Yes	607
Monitor thrust force estimates	3		
Monitor current force estimates	3		
Monitor wind force estimates	3		
Monitor leverarm compensated posref data	3		
Monitor/Set thruster allocation (set default)	3		

The requirements table gives an overview of the functions needed and priority of development.

3. Message format

The message format is based on proprietary NMEA 0183 messages.

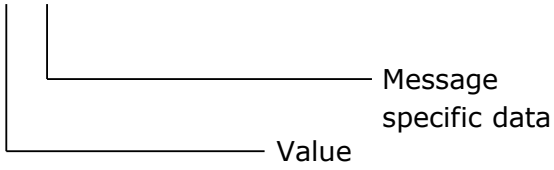
The feedback section defines how the DP system will generate the necessary feedback messages, and the command section defines how the third party should define the command messages sent to the DP system.

All transmitted data shall be interpreted as ASCII characters. Specific data fields shall only be located within a sentence by observing the fields delimiters ",".

3.1 Feedback

The beginning of a sentence is marked with $MTFBA followed by the ID of the function required and the end of the sentence is marked with <CR><LF>.

$MTFBA,XX,x,xxx,xxx.xx,x,xxx,x<CR><LF>



Each field is marked with:

1. ID
2. Message specific data

3.2 Command

The beginning of a sentence is marked with $MTCMD followed by the ID of the function required and the end of the sentence is marked with <CR><LF>.

$MTCMD,XX,x,xxx,xxx.XX,x,xxx,x<CR><LF>

4. Available functions

4.1 Thruster enable/disable

ID,DeviceId,EnableStatus,SimRpmFeedb,SimPitchFeedb,SimAngleFeedb,DeviceId,EnableStatus,SimRpmFeedb,SimPitchFeedb,SimAngleFeedb …

ID = 600
DeviceID = 1-12 (Thruster Id)
EnableStatus = 0 or 1
SimulateRpmFeedback = 0 or 1
SimulatePitchFeedback = 0 or 1
SimulateAngleFeedback = 0 or 1

4.1.1 Feedback – $MTFBA

Example:

$MTFBA,600,1,1,0,0,0,2,1,1,0,0,3,0,0,0,0,4,1,0,1,1

Thruster 1 Enabled, no simulated feedback
Thruster 2 Enabled, simulate rpm feedback
Thruster 3 Disabled, no simulated feedback
Thruster 4 Enabled, simulate pitch and angle feedback

4.1.2 Command - $MTCMD

Example:

$MTCMD,600,1,1,1,0,0,2,1,0,0,0,3,0,0,0,0,4,1,1,1,0

Thruster 1 Enable command, simulate rpm feedback
Thruster 2 Enable command, don’t simulate feedback
Thruster 3 Disable command, don’t simulate feedback
Thruster 4 Enable command, simulate rpm and pitch feedback

Example:

$MTCMD,600,3,1,0,0,1

Thruster 3 Enable, simulate angle feedback

4.2 Sensor enable/disable

ID, DeviceId, EnableStatus, DeviceId, EnableStatus…

ID = 617
DeviceID = DeviceType + DeviceNumber
EnableStatus = 0 or 1

DeviceType	100	Gyro
	200	Wind
	300	VRU
	400	Speed
	500	ROT

4.2.1 Feedback – $MTFBA
Example:

$MTFBA,617,101,1,102,1,103,0,201,1

Gyro 1	Enabled
Gyro 2	Enabled
Gyro 3	Disabled
Wind 1	Enabled

4.2.2 Command - $MTCMD
Example:

$MTCMD,617,101,1,102,1,103,0,201,1

Gyro 1	Enable command
Gyro 2	Enable command
Gyro 3	Disable command
Wind 1	Enable command

4.3 Reference system enable/disable

ID,DeviceId,EnableStatus,DeviceId,EnableStatus…

ID = 604
DeviceID = 1-9 (Sensor Id)
EnableStatus = 0 or 1

4.3.1 Feedback – $MTFBA

Example:

$MTFBA,604,1,1,2,1,3,0,4,1

Sensor 1 Enabled
Sensor 2 Enabled
Sensor 3 Disabled
Sensor 4 Enabled

4.3.2 Command - $MTCMD

Example:

$MTCMD,604,1,1,2,1,3,0,4,1

Sensor 1 Enable command
Sensor 2 Enable command
Sensor 3 Disable command
Sensor 4 Enable command

4.4 Position setpoint

ID,Lat,Lon,Speed,PresentPos

ID = 603
Lat = Latitude of current setpoint (DD) +/-
Lon = Longitude of current setpoint (DD) +/-
Speed = Speed setpoint (m/s) 0-2.0
PresentPos = Present Position status 0 or 1

4.4.1 Feedback – $MTFBA
Example:

$MTFBA,603,58.61234,6.32154,0.25,0

Latitude setpoint 58.61234
Longitude setpoint 6.32154
Speed setpoint 0.25 m/s
Present position No, in a position move.

4.4.2 Command - $MTCMD

Example 1:

$MTCMD,603,58.61234,6.32154,0.25,1

Latitude setpoint 58.61234
Longitude setpoint 6.32154
Speed setpoint 0.25 m/s
Present position Yes, ignore setpoint and issue a present position command.

Example 2:

$MTCMD,603,58.61234,6.32154,0.25,0

Latitude setpoint 58.61234
Longitude setpoint 6.32154
Speed setpoint 0.25 m/s
Present position No.

4.5 Heading setpoint

ID,HeadingSetpoint,Speed,TurnDirection

ID = 602
HeadingSetpoint = Heading setpoint (°) 0-360.0
Speed = Rotation Speed setpoint (°/min) 0-200.0
TurnDirection = Turn Direction:

0	Shortest
1	Starboard
2	Port

4.5.1 Feedback – $MTFBA

Example:

$MTFBA,602,291.0,15.0,0

Heading setpoint	291.0
Speed setpoint	15.0 °/min
Turn direction	Shortest

4.5.2 Command - $MTCMD

Example 1:

$MTCMD,602,291.0,15.0,0

Heading setpoint	291.0
Speed setpoint	15.0 °/min
Turn direction	Shortest

4.6 Mode selector

ID, MainMode, SubMode, AlongshipsControl, AthwartshipsControl, HeadingControl

ID = 601

MainMode = **2** **Standby**
Sub = 0 No Sub Mode

MainMode = **4** **Joystick**
Sub = 0 No Sub Mode

MainMode = **8** **DP**
Sub = 0 No Sub Mode
Sub = 4 Follow Line
Sub = 1 Relaxed DP
Sub = 2 Follow Target
Sub = 64 DP Track

MainMode = **16** **Transit**
Sub = 0 Autopilot
Sub = 4 Follow Line
Sub = 128 Seismic Track
Sub = 64 Track

4.6.1 Feedback - $MTFBA

Example:

$MTFBA,601,8,0,0,1,0

Mode	DP
SubMode	No Sub Mode
Alongships control	Off
Athwartships control	On
Heading control	Off

4.7 Alarm limits

ID, HeadingWarningActive, HeadingWarningLimit, HeadingAlarmActive, HeadingAlarmLimit, PositionWarningActive, PositionWarningLimit, PositionAlarmActive, PositionAlarmLimit

HeadingWarningActive	0 False, 1 True
HeadingAlarmActive	0 False, 1 True
PositionWarningActive	0 False, 1 True
PositionAlarmActive	0 False, 1 True

4.7.1 Feedback - $MTFBA
Example:

$MTFBA,610,0,3.0,1,5.0,0,3.0,1,5.0

Heading Warning Active:	false
Heading Warning Limit:	3.0 °
Heading Alarm Active:	true
Heading Alarm Limit:	5.0 °

Position Warning Active:	false
Position Warning Limit:	3.0 m
Position Alarm Active:	true
Position Alarm Limit:	5.0 m

4.7.2 Feedback - $MTFBA
Example:

$MTFBA,610,1,2.0,1,6.0,1,2.0,1,5.0

Heading Warning Active:	true
Heading Warning Limit:	2.0 °
Heading Alarm Active:	true
Heading Alarm Limit:	6.0 °

Position Warning Active:	true
Position Warning Limit:	2.0 m
Position Alarm Active:	true
Position Alarm Limit:	5.0 m

4.8 Gain settings

ID, PositionGain, VelocityGain, IntegralGain, HeadingPriority, GainType

GainType	1	DP Gain
	2	Transit Gain
	67	XTE Gain + DP Track
	131	XTE Gain + Seismic Track

4.8.1 Feedback - $MTFBA

Example:

$MTFBA,606,150,100,100,0,1

PositionGain:	150%
VelocityGain:	100%
IntegralGain:	100%
HeadingPriority:	false
GainType:	DP Gain

4.8.2 Command - $MTCMD

Example:

$MTFBA,606,110,105,100,1,2

PositionGain:	110%
VelocityGain:	150%
IntegralGain:	100%
HeadingPriority:	true
GainType:	Transit Gain

4.9 Joystick Command

ID, Surge, Sway, Yaw

Surge	-100.0	0.0	100.0
	-100%	0%	100%
Sway	-100.0	0.0	100.0
	-100%	0%	100%
Yaw	-100.0	0.0	100.0
	-100%	0%	100%

4.9.1 Feedback - $MTFBA

Example:

$MTFBA,502,0.0,50.0,-20.0

Surge:	0%
Sway:	50%
Yaw:	-20%

4.9.2 Command - $MTCMD

Example:

$MTCMD,502,30.0,-20.0,0.0

Surge:	30%
Sway:	-20%
Yaw:	0%

4.10 Joystick Settings

ID, Sensitivity, Response, Pivot Centre

Sensitivity	0	High
	1	Low
Response	0	Linear
	1	Progressive
Pivot Centre	0	Fore
	1	Centre
	1	Aft

4.10.1 Feedback - $MTFBA

Example:

$MTFBA,607,0,0,1

Sensitivity:	High
Response:	Linear
Pivot Centre:	Centre

4.10.2 Command - $MTCMD

Example:

$MTCMD,607,1,1,0

Sensitivity:	Low
Response:	Progressive
Pivot Centre:	Aft

4.11 DP Class

ID, Class

Class:	0	Off
	1	DP2
	2	DP3

4.11.1 Feedback - $MTFBA

Example:

$MTFBA,611,0

Class: Off

4.11.2 Command - $MTCMD

Example:

$MTCMD,611,1

Class: DP2

Vedlegg D

Programming Manual

Grayhill

Intuitive Human Interface Solutions

Touch Encoder

Programming Manual

TE OS Version 1.0.0 or greater (includes UtilityApp)

Revision History

Revision	Date	Description
A	2/19/2018	Original Release
B	2/27/2018	Changes to the example message in section 2.5.2 Rephrased the meaning of Byte 7 in section 2.5.2
C	3/6/2018	Changed Byte message in section 3.1
D	7/25/2018	Changed pinout on M12 to reflect production pinout
E	8/24/2018	Added Programming Harness Information
F	10/17/2018	Updated images to reflect latest app update
G	1/31/2019	Updated Touch Encoder Image
H	4/22/2019	Added Touch Zones in Photo Widget
I	7/10/2019	Added New Menu Widget in section 13
J	8/28/2019	Updated with Latest Firmware update instructions

Table of Contents

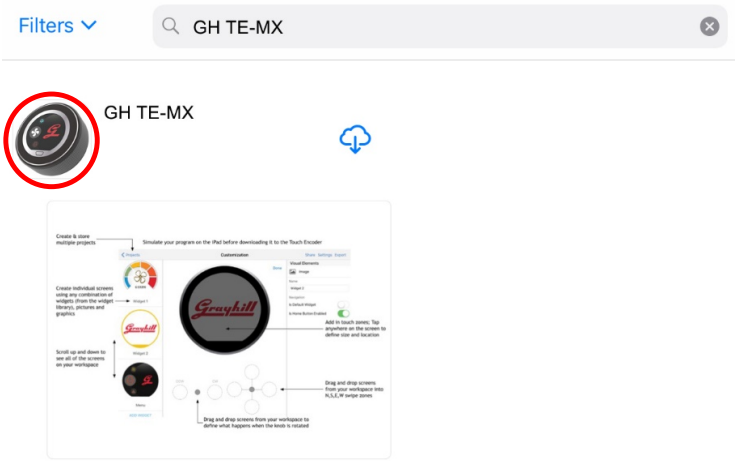
1 What's included?

1. Touch Encoder Development Module
2. CANbus Interface Cable
3. Red Programming Cable
4. Power to USB Cable
5. Power Supply Wall Mount
6. White USB Micro B to USB Type A Adapter Cable
7. Thumb drive

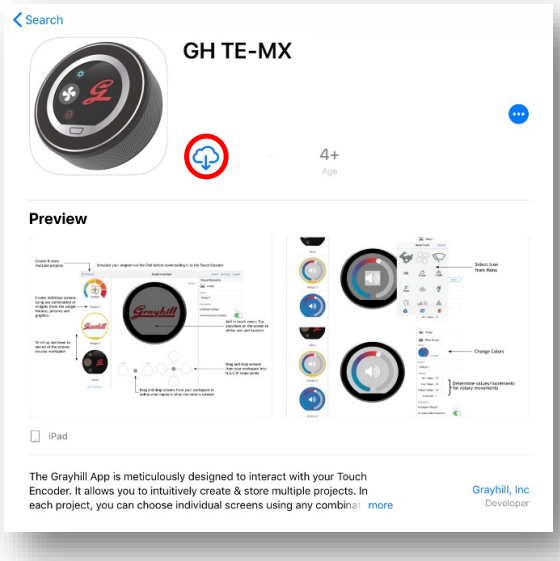
* iPad is included if tablet version is ordered

2 Download App

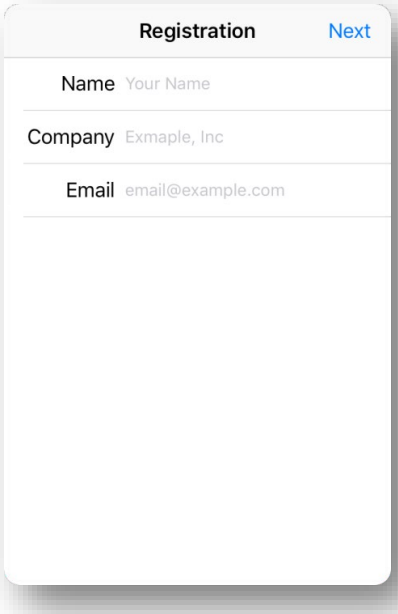
2.1 Search for "Grayhill" on App Store with iPad

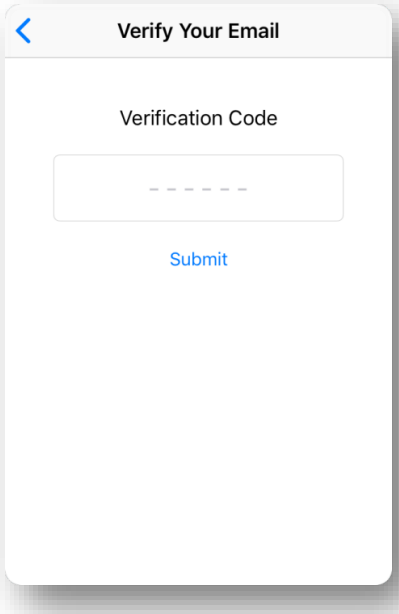


2.2 Download App



2.3 Registration: Enter info and receive verification code in email





3 Utility Application

3.1 TE Utility App

Power the Touch Encoder SDK with the red programming cable and USB thumb drive. If screen appears as below, and you wish to install the newest firmware and app, please visit www.touchencoder.com/programming

For latest Touch Encoders, the screen will appear as below. To update your Touch Encoder firmware, visit www.touchencoder.com/programming for instructions.

Update
1. **Project**
2. **System**

Utilities
1. **Info**
2. **Comm**
3. **Calibrate**

3.2 Utility Functions

3.2.1 Info

Tap the Info button to get the current Touch Encoder firmware revision.

3.2.2 Comm

Tap the COMM button to set the communication protocol and baudrate.

3.2.3 Calibrate

Please contact us per information on the last page.

3.3 Update Functions

3.3.1 Project

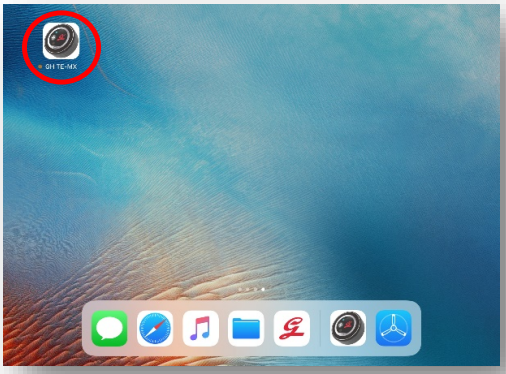
Go to the project button to update the project on the Touch Encoder. There can be multiple projects loaded on the USB. Scroll to select the .zip file of you choosing.

3.3.2 System

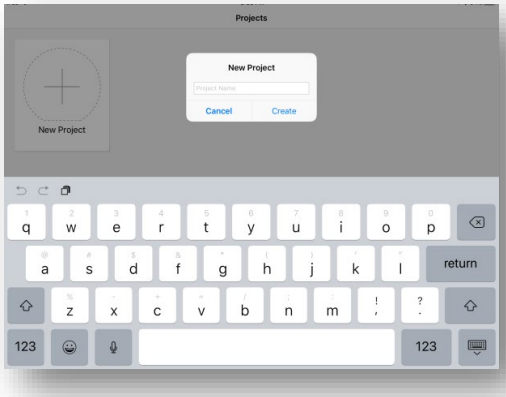
Go to the system button to update the operating system and other required system files on the Touch Encoder. For update instructions, Please visit www.touchencoder.com/programming

4 Create and Save a new project

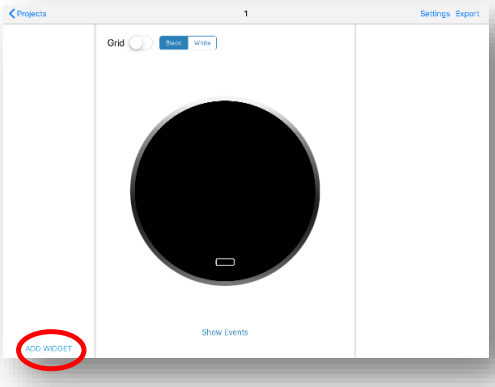
Open App



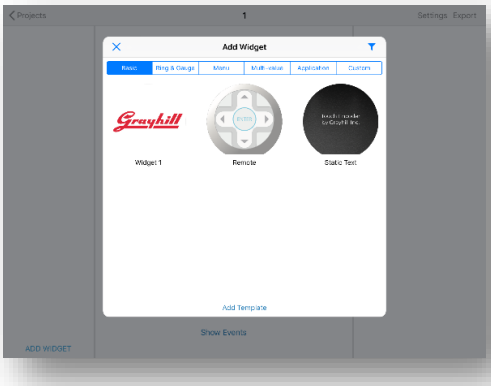
Name Project



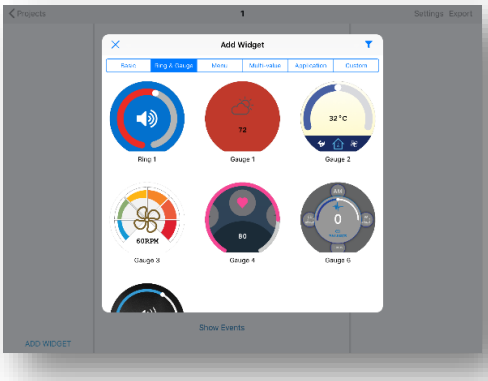
Add Widget



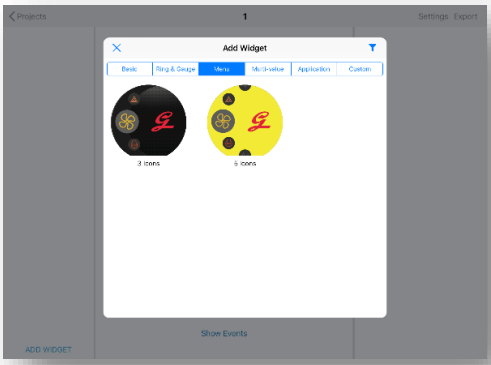
Basic Widgets



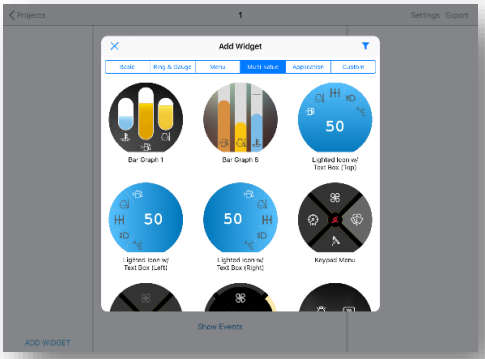
Ring& Gauge Widgets



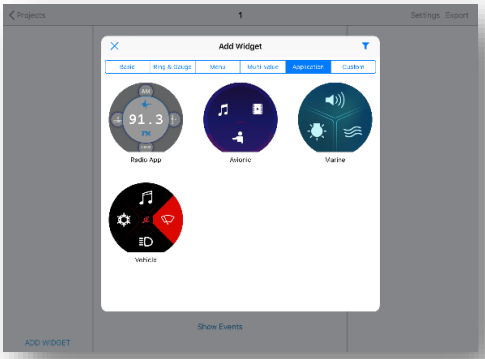
Menu Widgets



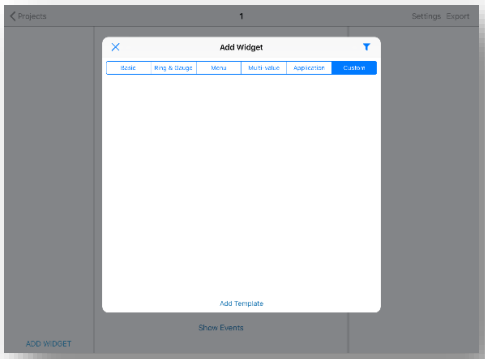
Multi-value Widgets



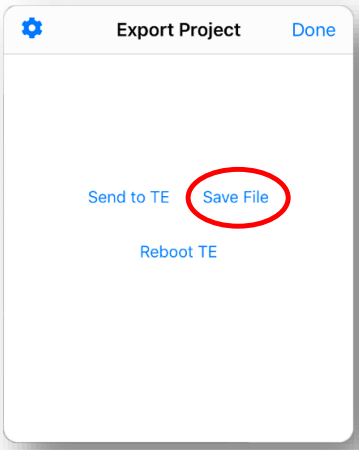
Application Widgets: Market Dependent

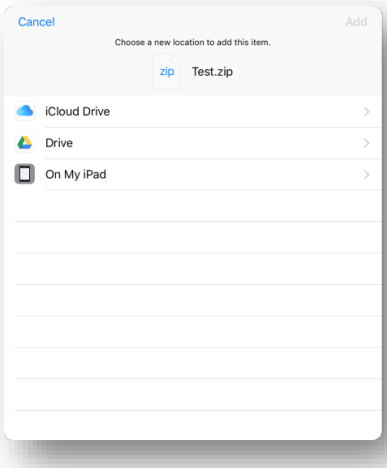


For Complete Custom Widgets, Please Contact Us per Info on the Last Page

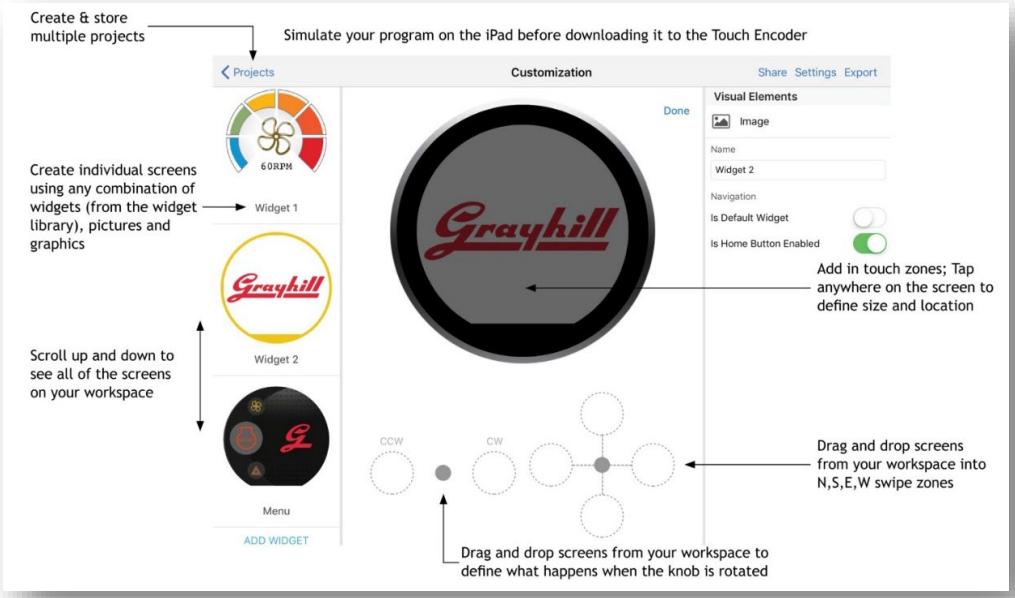


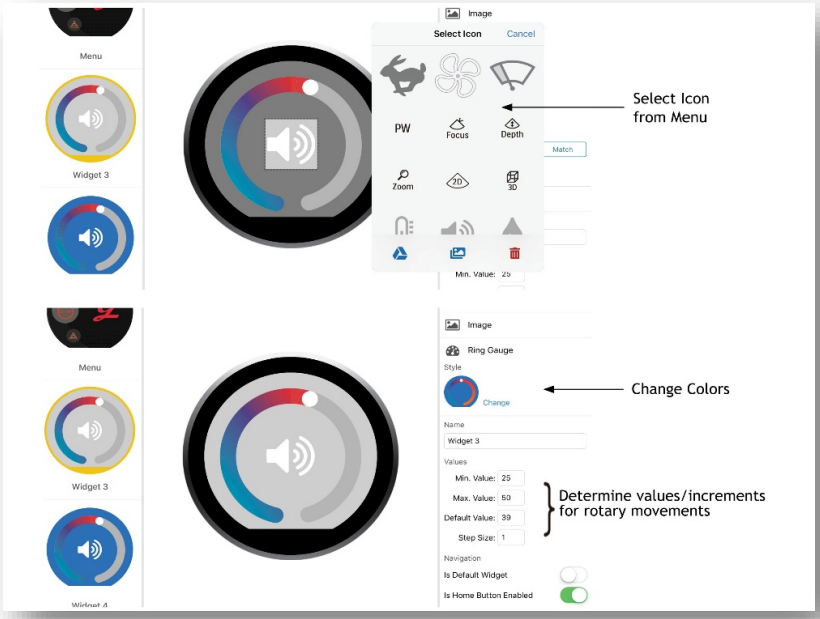
Choose to save in iCloud, Google Drive, or on your iPad





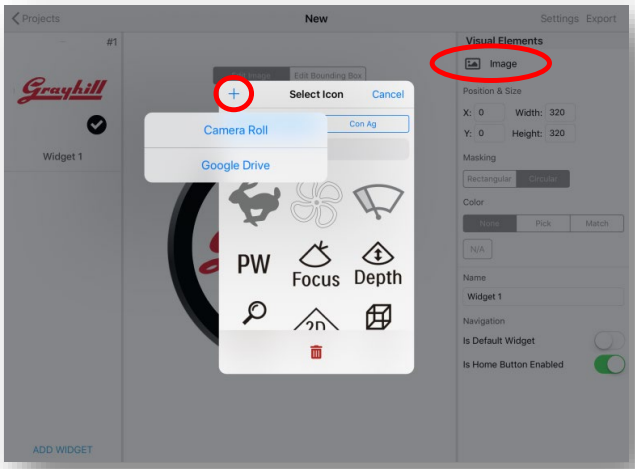
5 Basic App Navigation



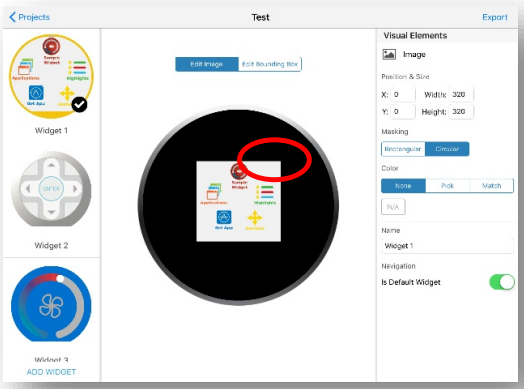


6 Uploading and Sizing an Image

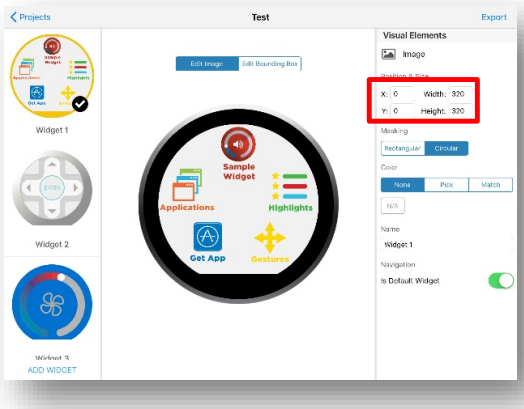
Tap on “Image”, then “+” to fetch images from Google Drive or Camera Roll



Select Image



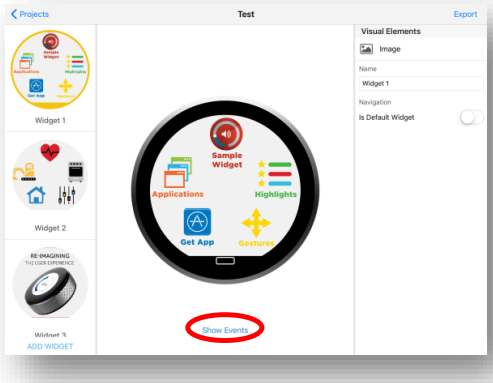
Reposition and Resize Image



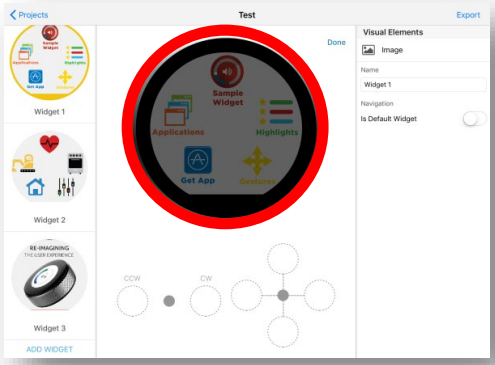
*** Repositioning and resizing can be done by tapping and dragging on image corners or inputting locational and dimensional values**

7 Defining Touch Zones in Photo

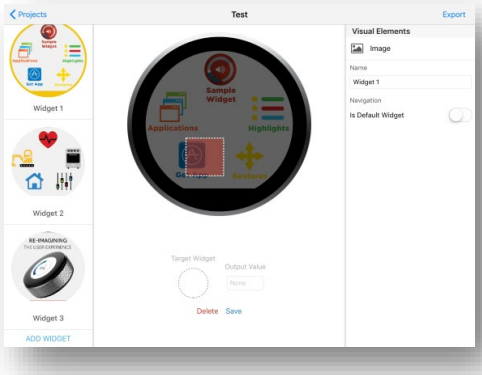
Tap on ‘Show Events’



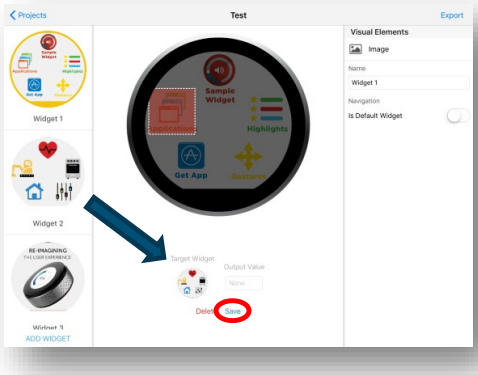
Tap anywhere on Touch Encoder Screen



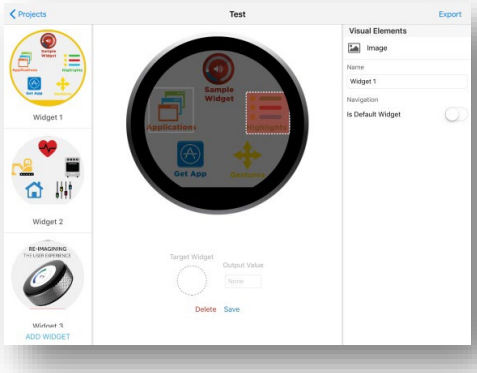
Drag corners to resize, and drag box to reposition



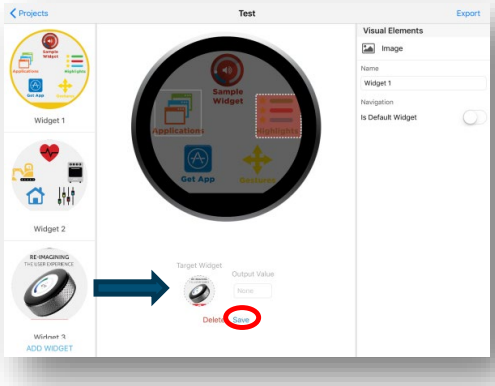
Drag Widget on the left to Target, and tap ‘Save’



For more Touch Zones, Tap on screen again

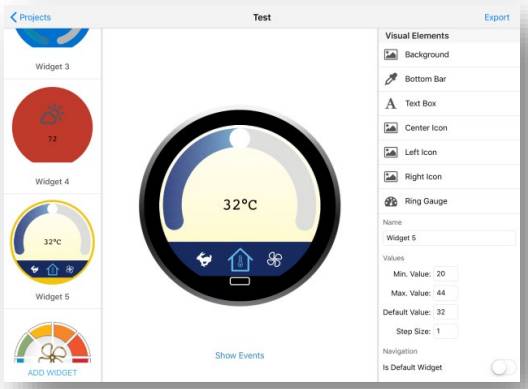


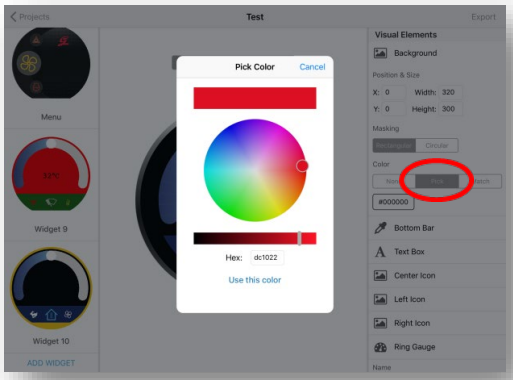
Drag Widget on the left to Target, and tap ‘Save’



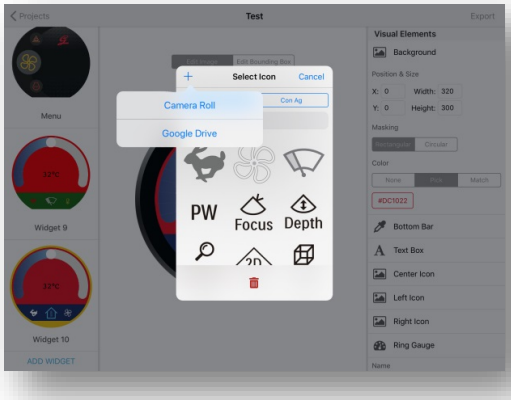
8 Widget Configuration

Pick any widget and select background colors

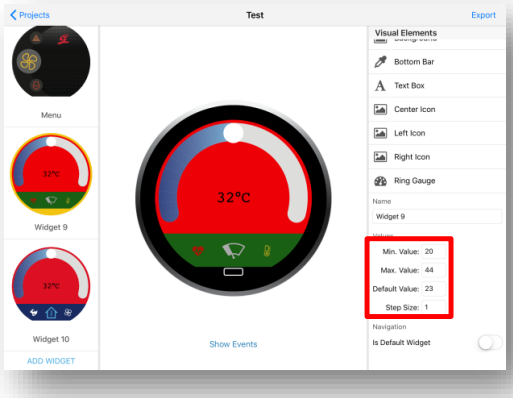




Choose Icons from Library

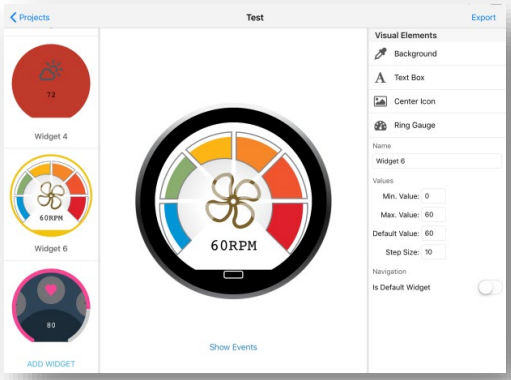


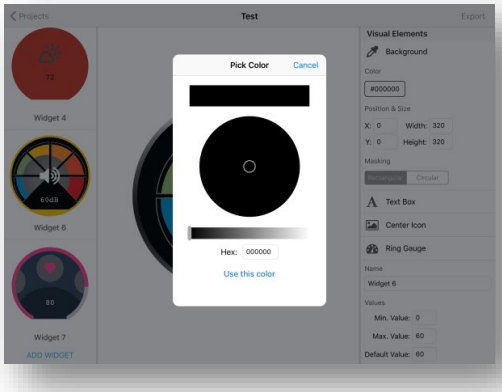
Customize Values



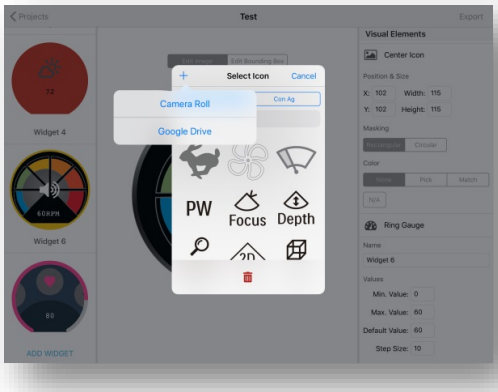
*** Customizable values include Max, Min, Default, and Step Size**

Pick any widget and select background color

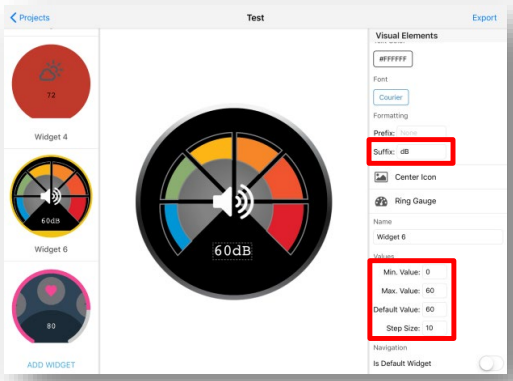




Choose Icons from Library

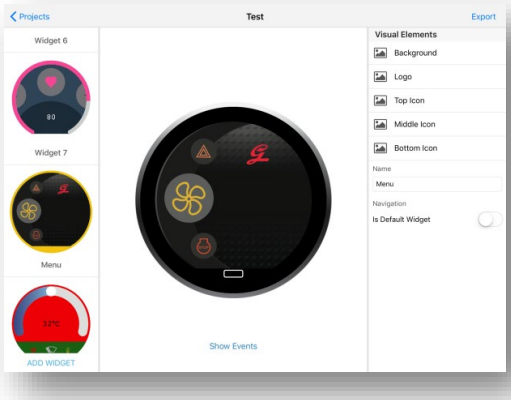


Customize Values

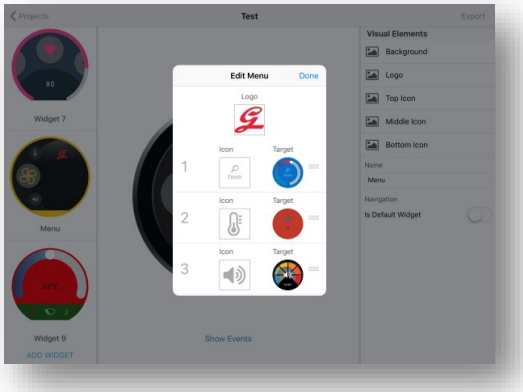


*** Customizable values include Max, Min, Default, incremental Step Size values, and Suffix (measurement unit)**

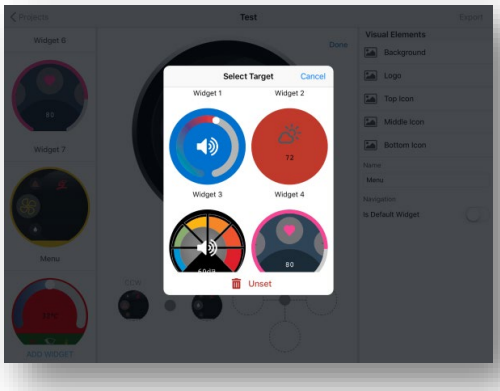
Select Menu Widget



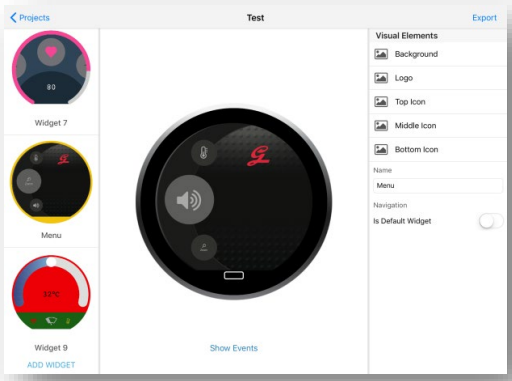
Customize Icons and Target Widgets



Choose Widgets from Workspace

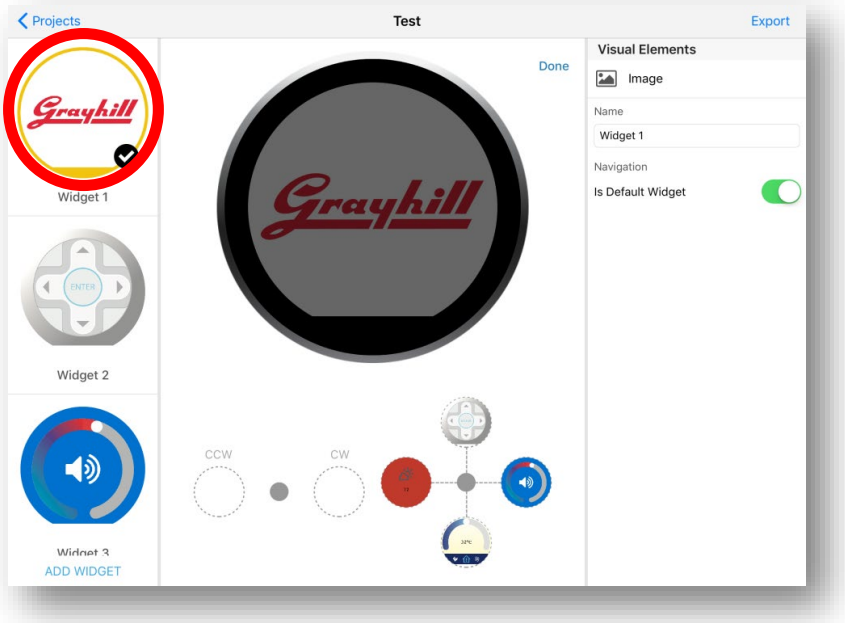


New Menu



9 Setting Home Screen

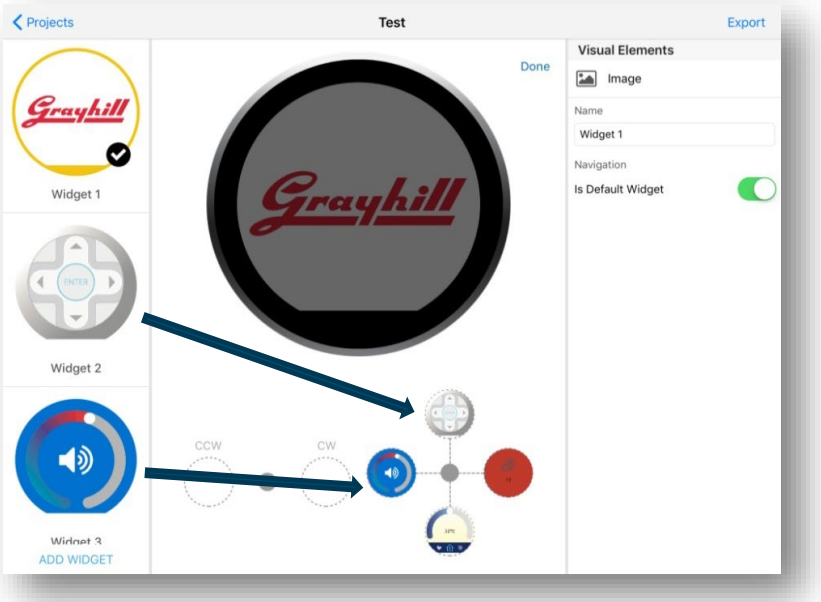
Double tap any widget to set as home screen



10 Defining Program Flow

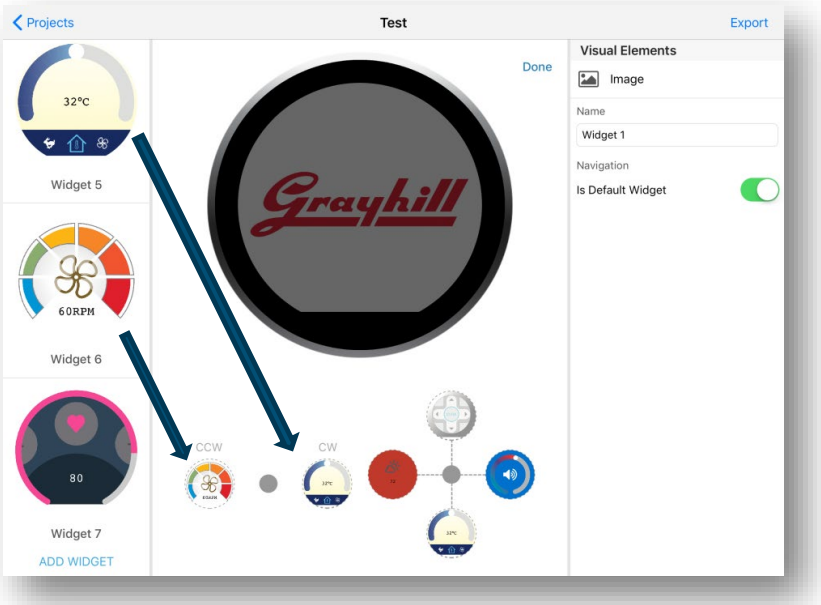
10.1 Swipe Logic

Drag and Drop any widgets to Up, Down, Left, and Right



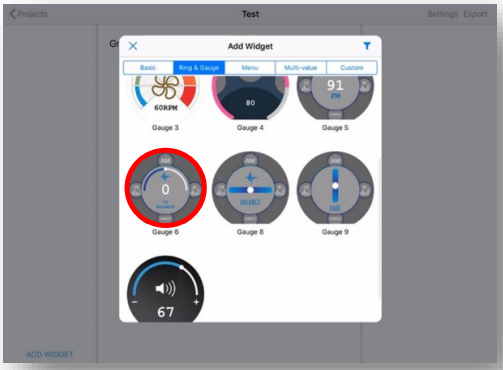
10.2 Rotation Logic

Drag and Drop any widgets to Clockwise and Counter-Clockwise target locations

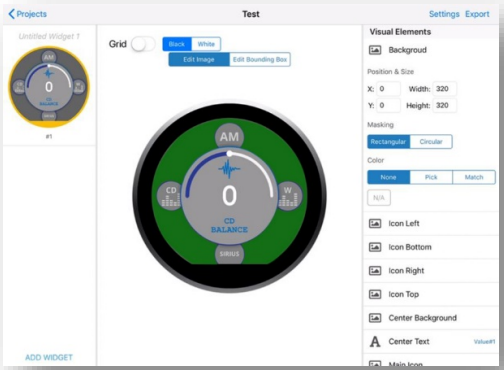


11 Exploring New Ring Widget

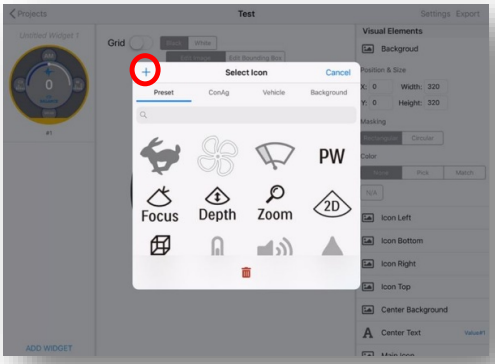
Select Widget



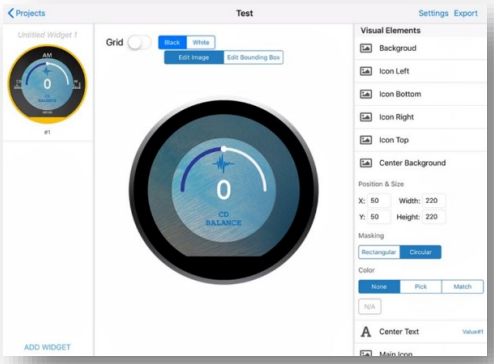
Tap 'Background' and Green Zone



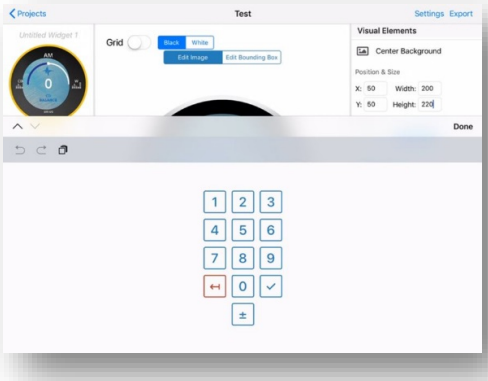
Import Image or Select Icons below



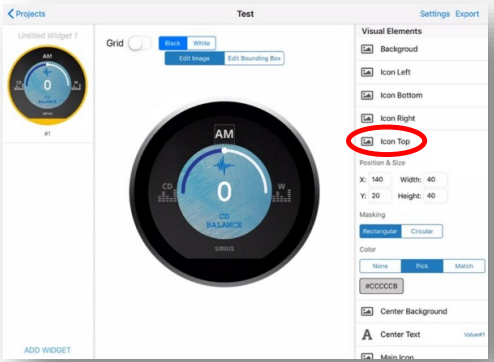
Define Masking Shape



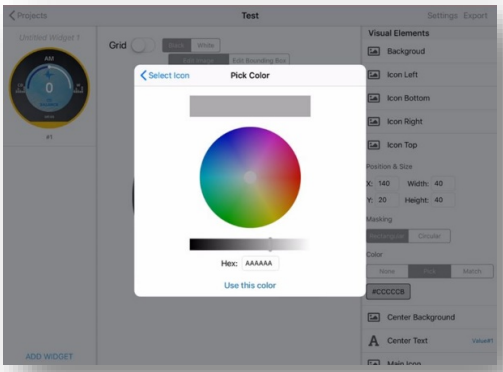
Resize Masking Measured in Pixels



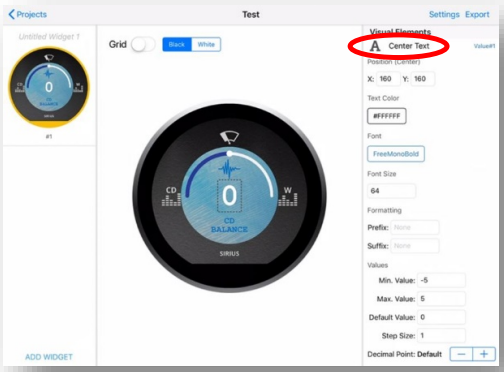
Edit Top Icon



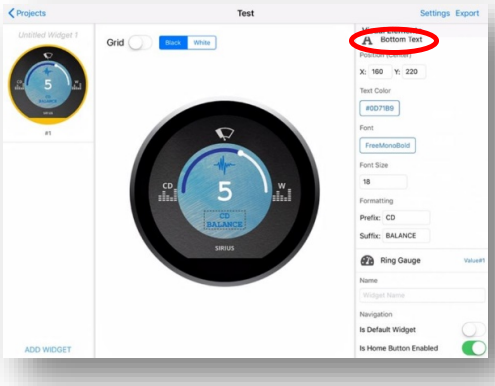
Pick Icon Color



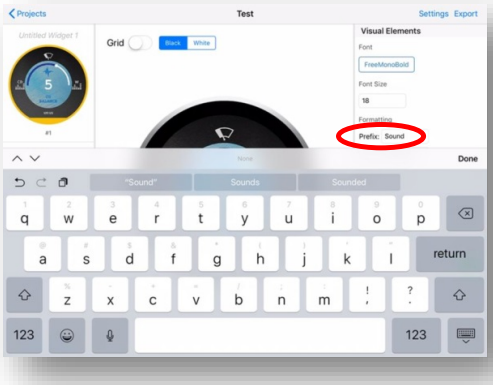
Modify Center Text with Options Listed



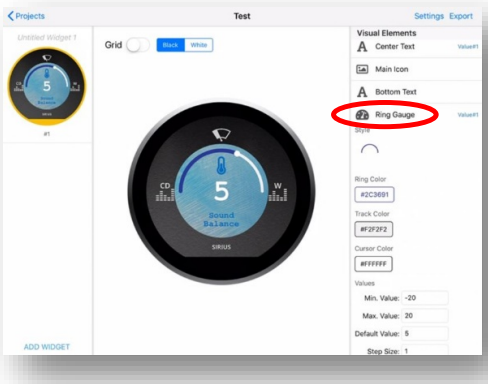
Change Bottom Text



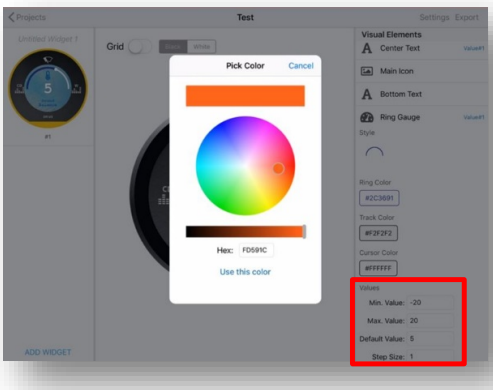
Customize Prefix and Suffix



Modify Ring Gauge

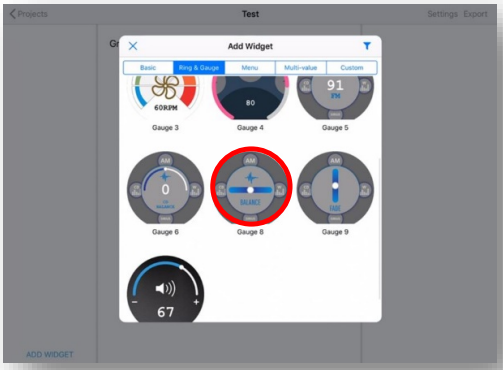


Select Colors and Edit Displayed Values

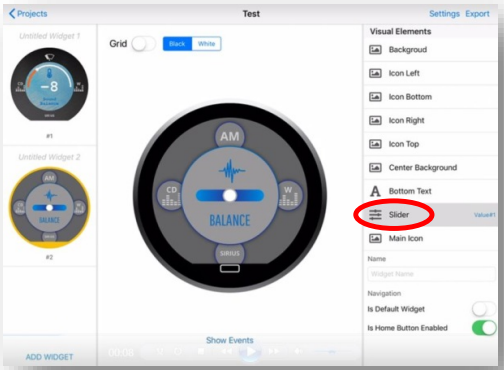


12 Slider Gauge

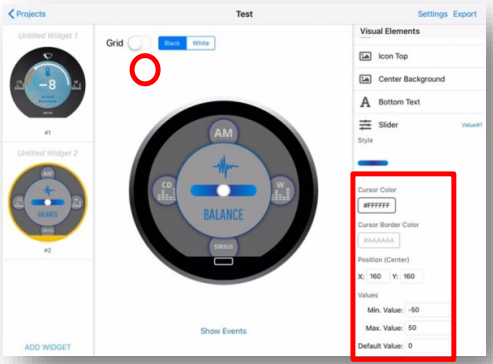
Select Widget



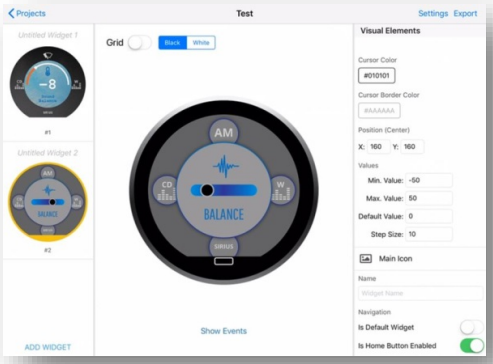
Tap on 'Slider'



Change Cursor Color, Position, and Values

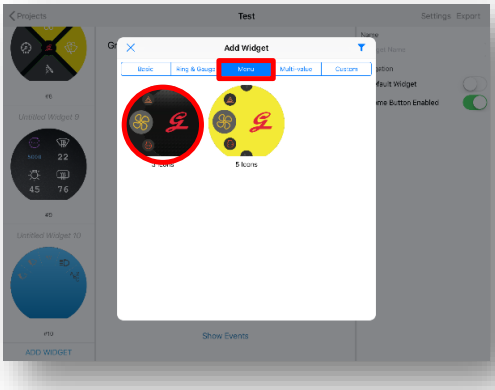


New Parameters

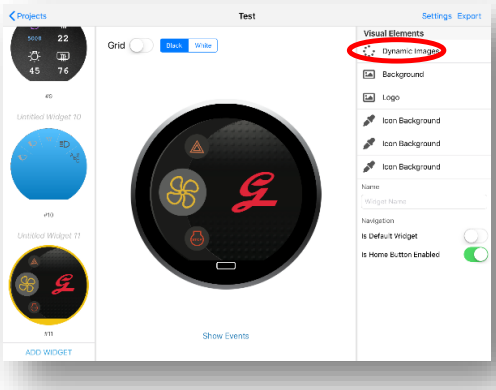


13 Menu Widget

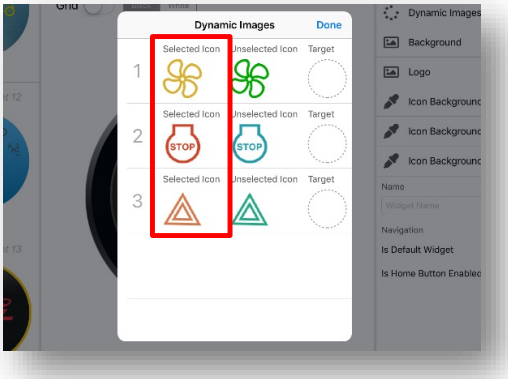
Select Widget under 'Menu'



Tap on 'Dynamic Images'

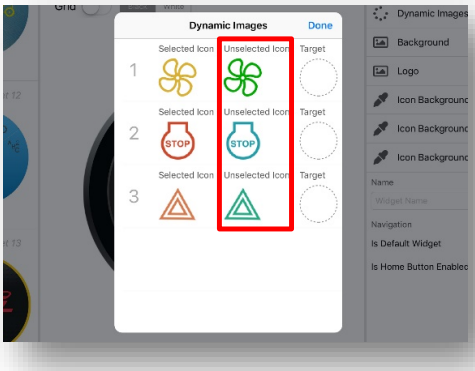


Configure Active Icons

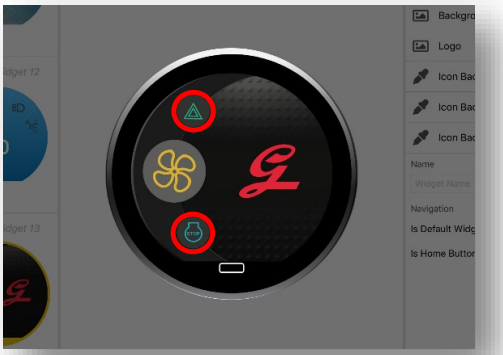


Active Icon appears in the Center

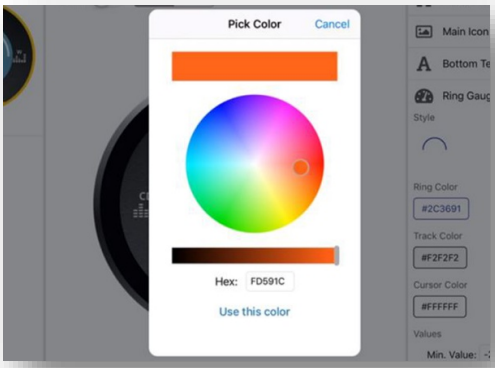
Configure Inactive Icons



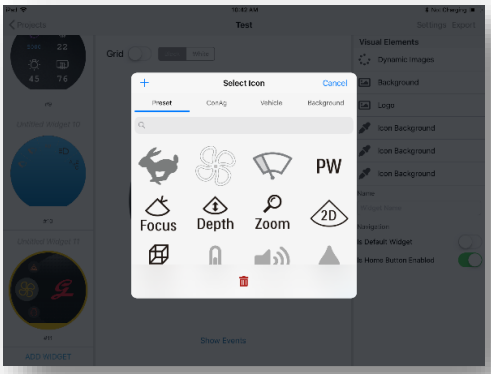
Inactive Icons appear above and below Active Icon



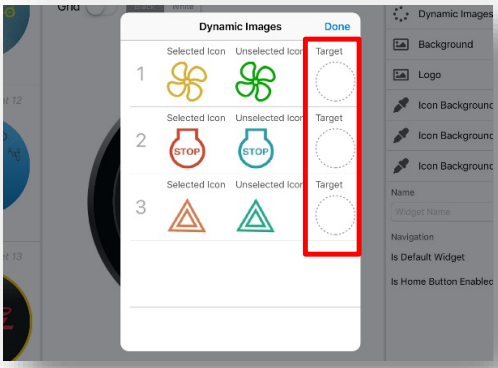
Change Icon Color



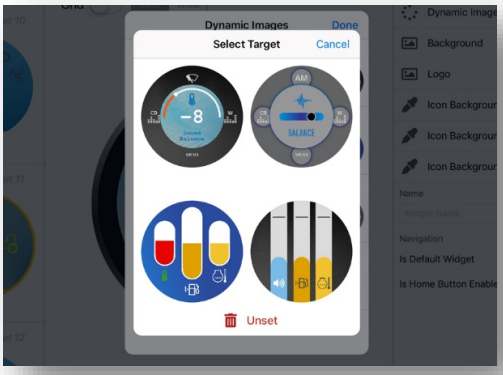
Choose other Icon Designs



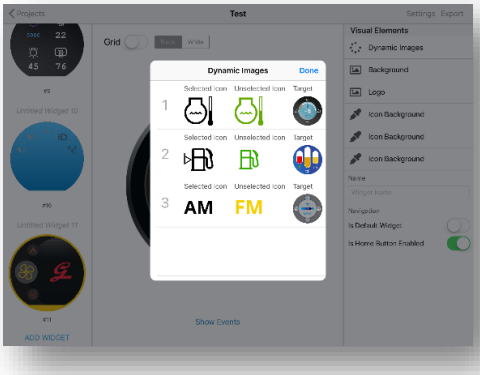
Configure Icon Targets



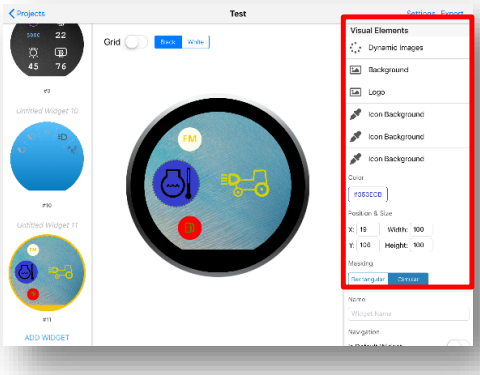
Choose from Created Widgets as Targets



Sample Configuration

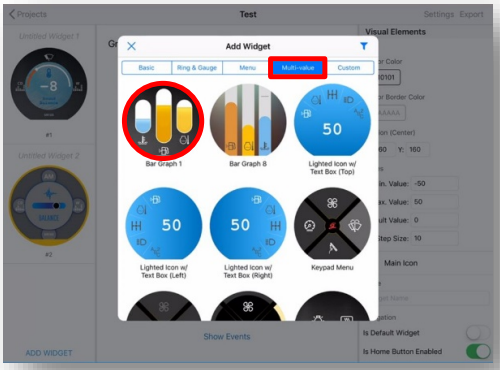


Change Backgrounds for Icons and Entire Widget

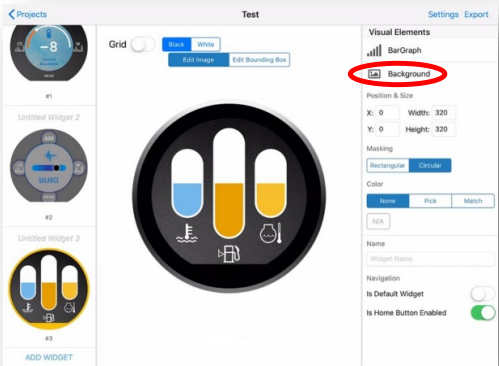


14 Bar Widget

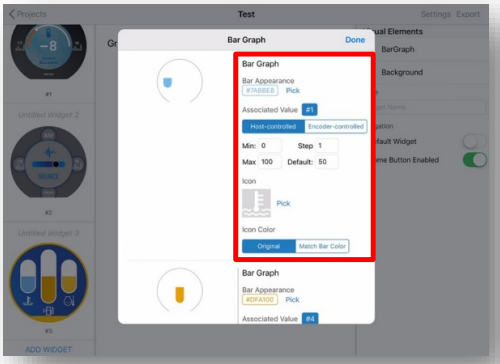
Select Widget under ‘Multi-value’



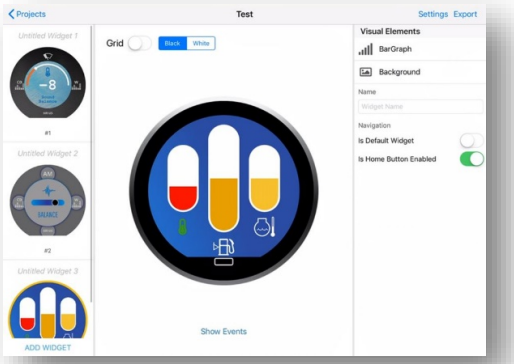
Edit ‘Background’ Settings



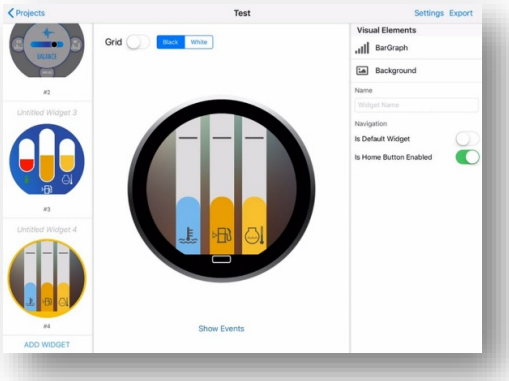
Customize Bar Graph Aesthetics, Values, and Control Device



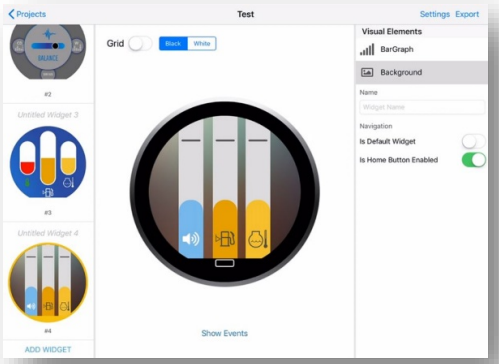
Notice changes on Background and Left Bar Graph



Configure another Widget Similarly

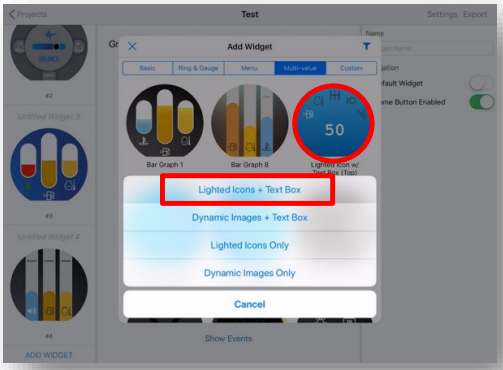


Notice changes on Left Icon and Its Color

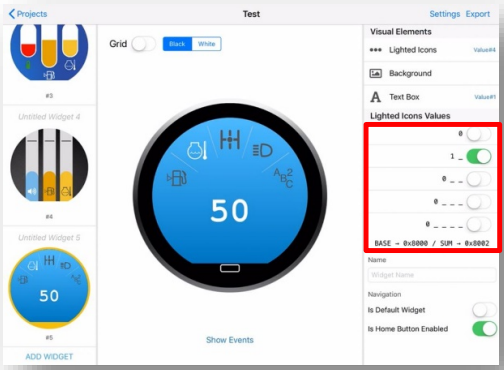


15 Lighted Icon Multi-value Widget

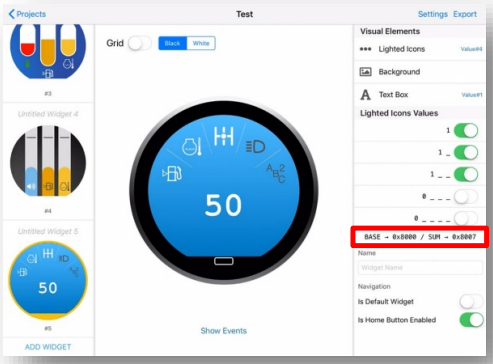
Select Widget



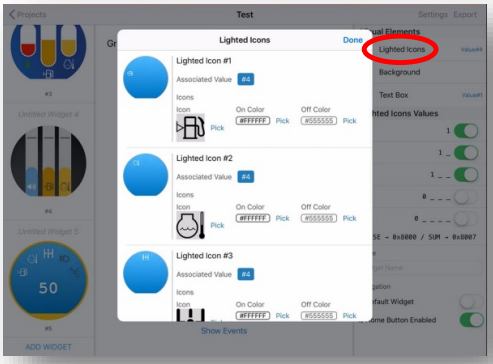
Set Icons 'On' and 'Off"



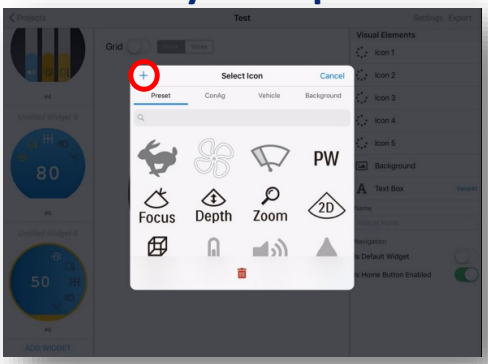
Notice Changes on Bit Masking



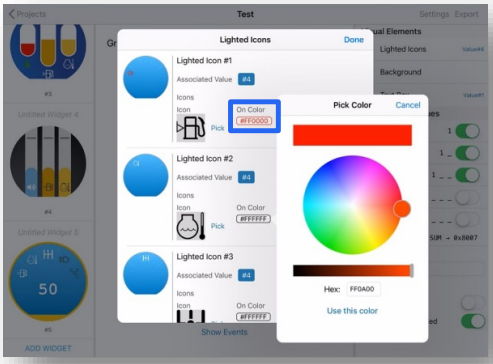
Tap on 'Lighted Icons'



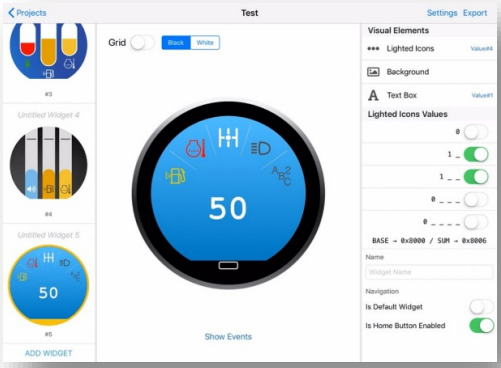
Customize Icons from Library or Import



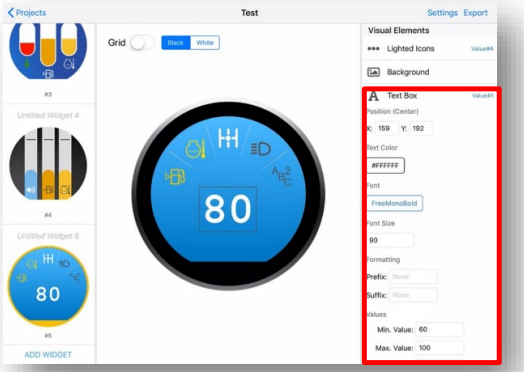
Change Icon Colors for 'On' and 'Off' States



Notice Icon Color Changes per Previous Customization

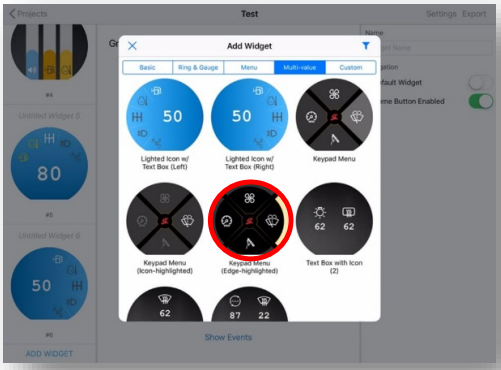


Configure Text Box with Settings Below

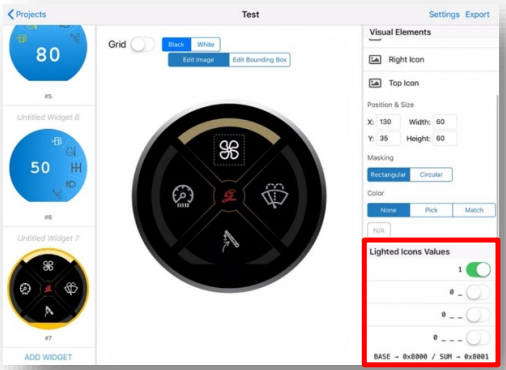


16 Configure Keypad Widget

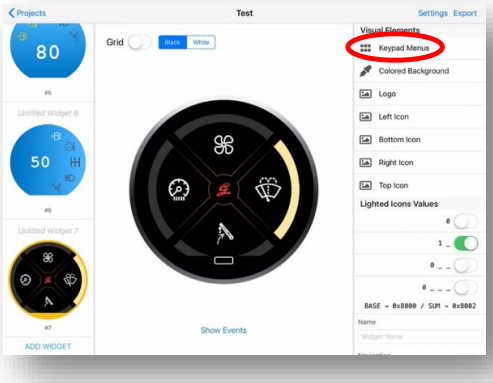
Select Widget



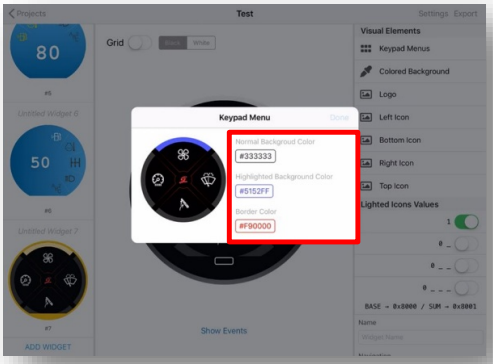
Set Icons ‘On’ and ‘Off’



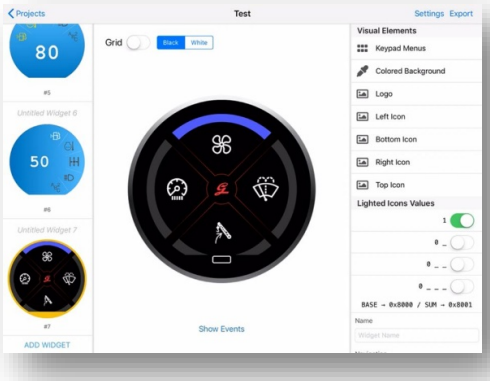
Tap on ‘Keypad Menus’



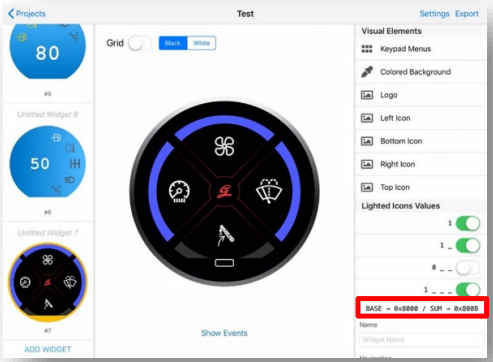
Change Color Settings



See Updates take Effect

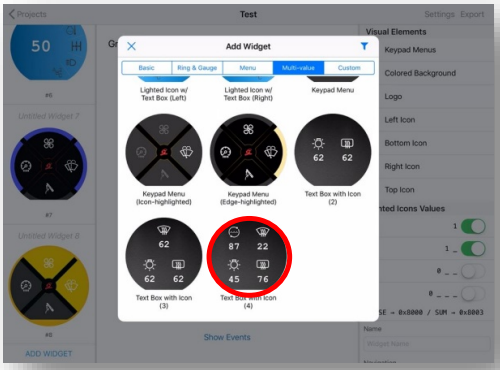


Notice Changes in Bit Masking

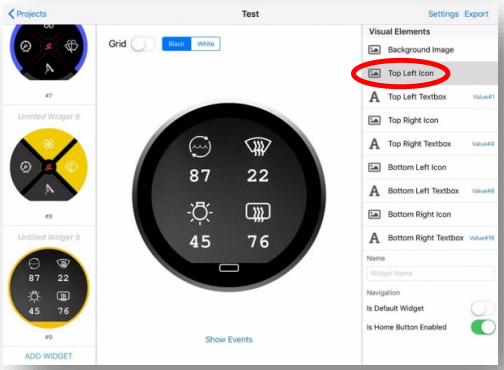


17 Text Box with Icon Widget

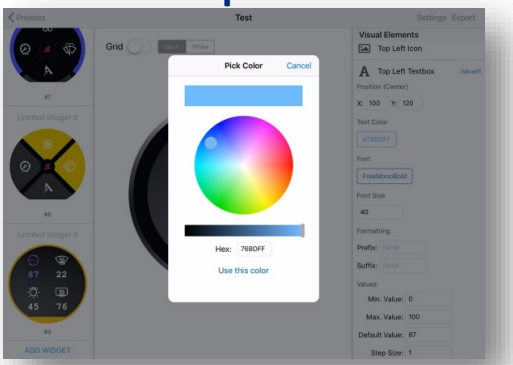
Select Widget



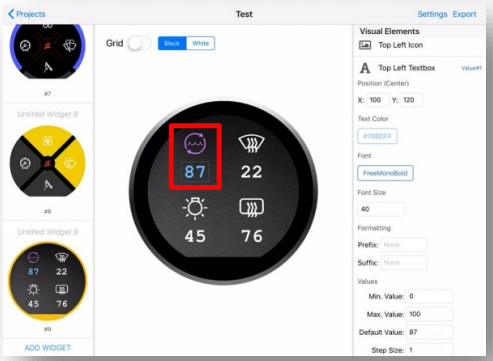
Tap on 'Top Left Icon'



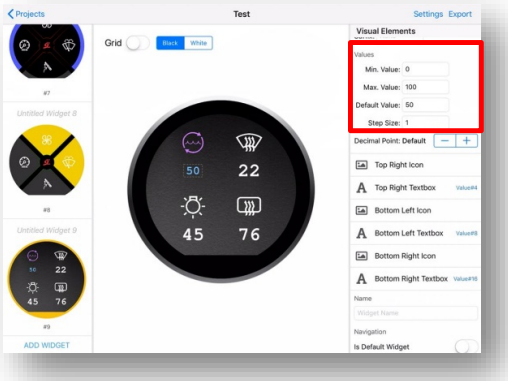
Make Selection from Color Scale or Input Hex Code



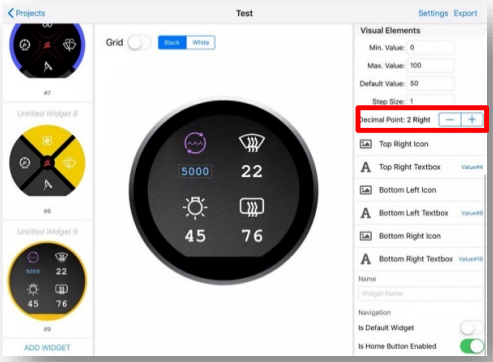
Tap on 'Top Left Textbox'



Change Values and Font



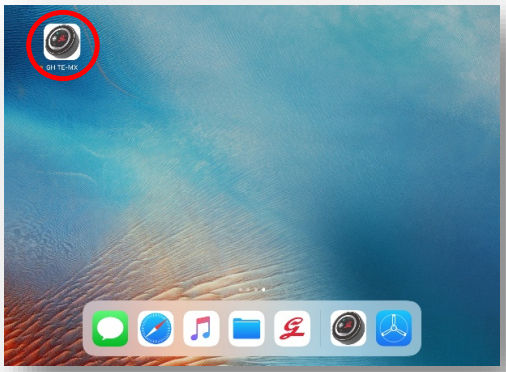
Change number of Digits in Textbox



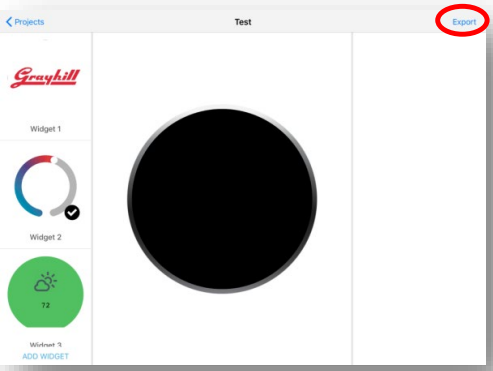
18 Download Program to SDK via USB

Plug in USB

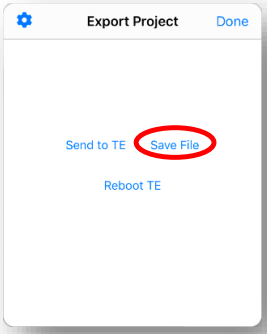
Open App



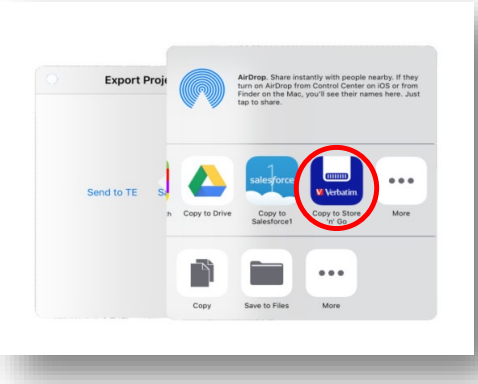
Tap on 'Export'



Tap on 'Save File'



Tap on Storage Icon

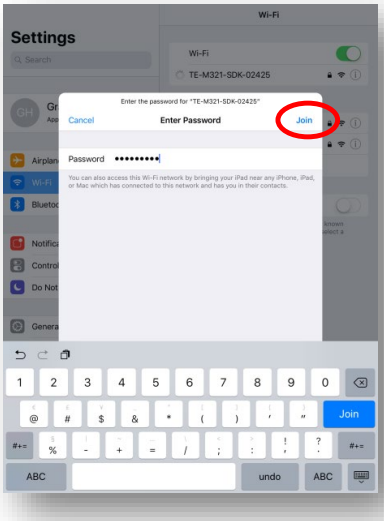


Connect USB to Red Cable, Power on

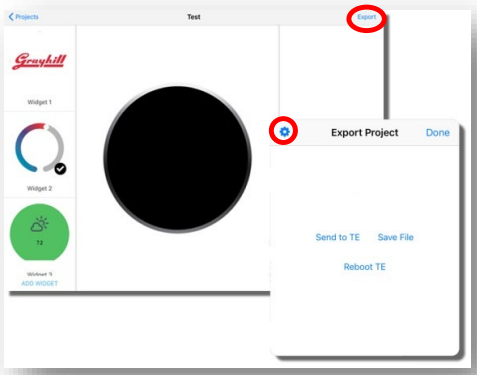
Follow Instructions. Once update is complete, reboot SDK by removing power cable for at least 5 seconds

19 Download Program to SDK via Wi-Fi

Go to Settings → Wi-Fi → Choose network that begins with "TE-M321"; Enter 9-digit S/N as password located on SDK label: reference next page; Tap on "Join" which might take several attempts to establish connection. Then go back to Grayhill app

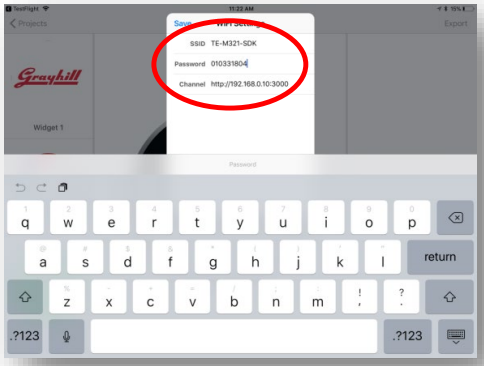


"Export", then hit "gear" icon

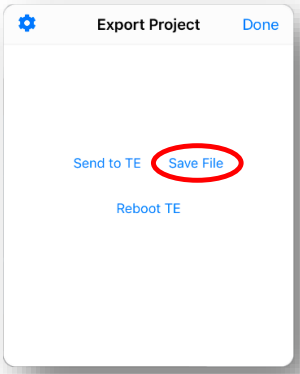


Model Number and Serial Number

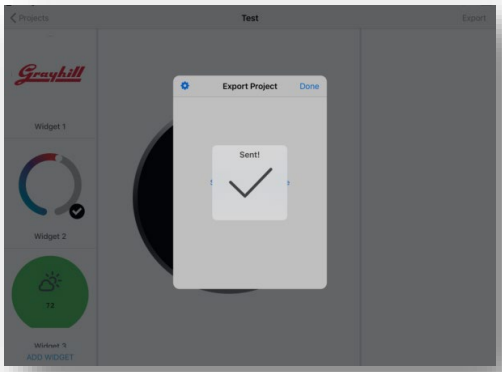
Input from previous step and save

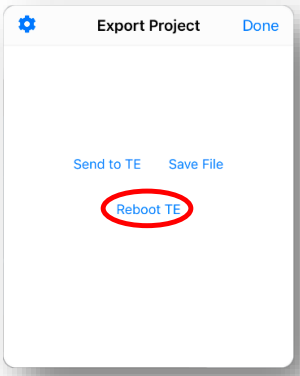


Tap on 'Send to TE'



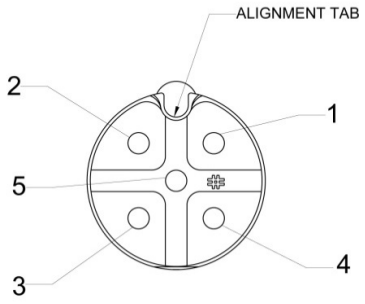
Follow Instructions on SDK. Once update is complete, reboot by tapping on 'Reboot TE'





20 Programming Harness

Connector Pinouts



M12 Mating Connector:
AMPHENOL HDM12PF05A1STM

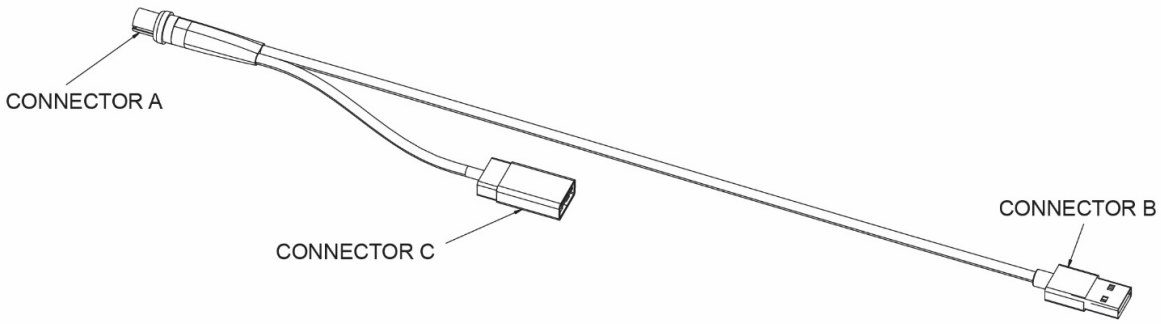
	USB	CAN
1	MODE	MODE
2	VIN	VIN
3	GND	GND
4	USB_D+	CAN_H
5	USB_D-	CAN_L

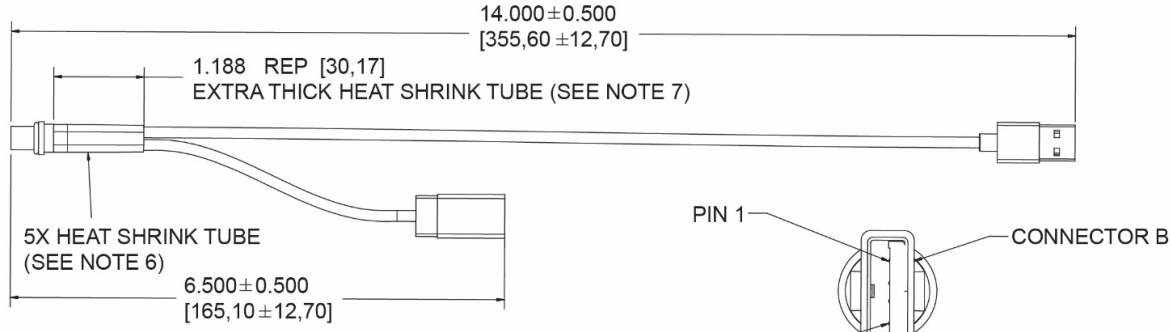
For Touch Encoder serial numbers less than A100000 please contact Grayhill for pinout detail.

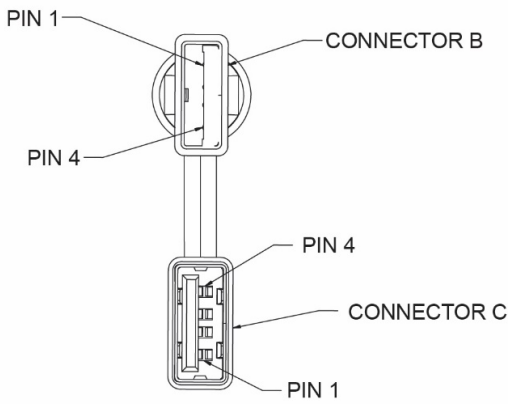
If MODE Pin is floating at power up, the Touch Encoder will assume run mode operation. If Mode pin is connected to GND externally at startup, the Touch Encoder will assume programming mode. The Touch Encoder will download updates from a USB mass storage device (if connected) and update the Touch Encoder accordingly.

Programming Harness for Production Units

GRAYHILL P/N T18908 - USED FOR PROGRAMMING MODE ONLY



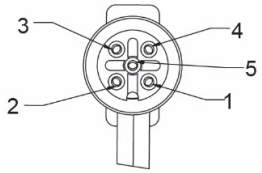




NOTES:

1. CONNECTOR A IS NORCOMP INC. P/N: 858-005-203RSU4 WITH METAL NUT REMOVED.
2. WIRE WITH USB CONNECTORS B & C IS TRIPP LITE P/N: U024-003 OR EQUIVALENT
3. CONNECTOR A PINS TO CONNECT TO CORRESPONDING CONNECTOR B & C TERMINALS ON THE CONNECTOR (SEE TABLE)..
4. WIRE LENGTH SHOULD BE MEASURED WITH WIRES STRAIGHT.
5. WIRE PULLOUT FORCE MUST BE 2 LBS. MINIMUM.
6. 5X HEAT SHRINK TUBING TO BE USED OVER CONNECTOR A CONNECTIONS.
7. EXTRA THICK HEAT SHRINK TUBING TO BE USED OVER CONNECTOR A CONNECTIONS.

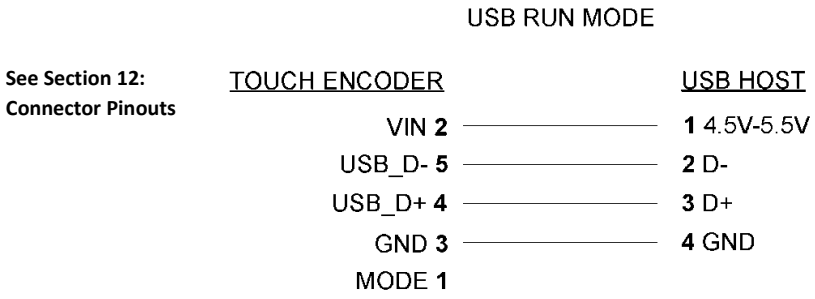
WIRE #	CONNECTOR A PIN	CONNECTOR B PIN	CONNECTOR C PIN	COLOR	DESCRIPTION
1	1	N/A	4	BLACK	MODE
2	2	1	1	RED	Vin
3	3	4	4	BLACK	GND
4	4	N/A	3	GREEN	USB+
5	5	N/A	2	WHITE	USB-



Connector A Pinout

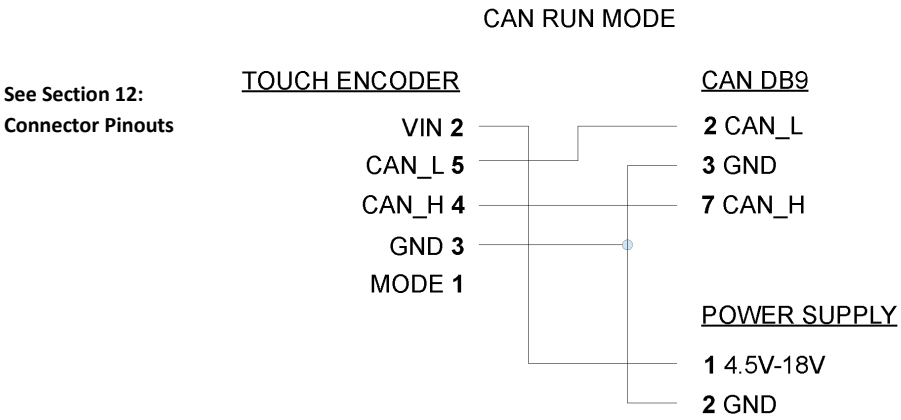
21 Connections

USB run mode:

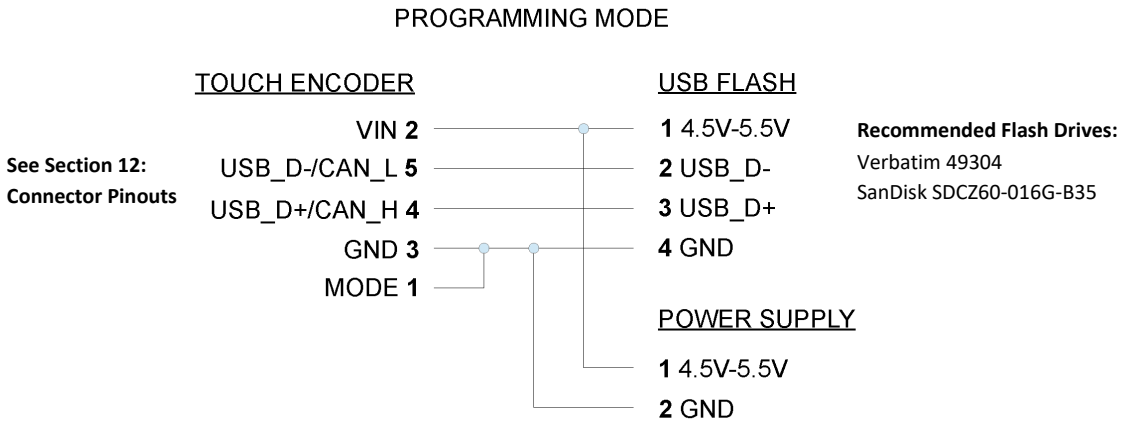


Note: The Touch Encoder can be powered from an external source or directly from the USB if the source capability is high enough (>250mA.)

CAN run mode:



Programming mode:



See Section 12: Connector Pinouts

Recommended Flash Drives:
Verbatim 49304
SanDisk SDCZ60-016G-B35

Note: Load files to FLASH drive and plug into socket. Touch Encoder only samples MODE pin during power up, so be sure this occurs after connecting harness and FLASH drive.

Grayhill
Intuitive Human Interface Solutions

561 Hillgrove Avenue
La Grange, IL 60525
web: www.grayhill.com
e-mail: te@grayhill.com
phone: +1 (708) 354-1040

About Grayhill

Grayhill, Inc. is a privately held firm which designs and manufacturers intuitive human interface solutions that make life simpler, safer and more efficient. Standard products include optical and Hall Effect encoders, discrete and Hall Effect joysticks, rotary switches, keypads, and pushbuttons; all with finely tuned haptics. Grayhill specializes in creating ergonomic panels and product shells that integrate various interface technologies, including displays, our components, and gesture recognizing multi-touch technology. With headquarters in La Grange, Illinois, and multiple state-of-the-art facilities around the world, Grayhill's team has the full engineering, product development and manufacturing expertise to deliver both standard and customized products quickly and cost-effectively. To learn more about Grayhill's products and capabilities, visit www.grayhill.com.

Vedlegg E

Kode

```
using System;
using System.Linq;
using HidLibrary;

namespace MagtekCardReader
{
    class Program
    {
        const int VendorId = 0x1658;
        const int ProductId = 0x0060;
        static HidDevice _device;

        static void Main()
        {
            _device = HidDevices.Enumerate(VendorId,
ProductId).FirstOrDefault();

            if (_device != null)
            {
                _device.OpenDevice();

                _device.MonitorDeviceEvents = true;

                _device.ReadReport(OnReport);

                Console.WriteLine("Reader found, press any key to exit.");
                Console.ReadKey();

                _device.CloseDevice();
            }
            else
            {
                Console.WriteLine("Could not find reader.");
                Console.ReadKey();
            }
        }

        private static void OnReport(HidReport report)
        {
            if (!_device.IsConnected) { return; }
            if (int.Parse(report.ReportId.ToString()) == 1)
            {
                EncoderIsHomescreen();

                EncoderIsAutopilot();

                EncoderIsECDIS();

                EncoderIsRadar();
            }

            var cardData = new Data(report.Data);


            _device.ReadReport(OnReport);
        }

        private static void EncoderIsHomescreen()
        {
            if (Data[1] == 3)
            {
```

```
                Console.WriteLine("System is in Homescreen.");

                if (Data[9] > 0x0)
                {
                    Console.WriteLine("Enter click");
                }
                else if (Data[8] > 0x80)
                {
                    Console.WriteLine("Right arrow click");
                }
                else if (Data[8] < 0x80)
                {
                    Console.WriteLine("Left arrow click");
                }
            }
            else
                return;

            _device.ReadReport(OnReport);
        }
        /**
         * Checks if the encoder is in homescreen
         * if in homescreen, it detects touches as enter clicks ot choose
programs
         * and rotations of the edge to navigate the menu window
         * if it is no longer in the screennumber for homescreen it breaks
the loop
         * once out of the loop, it enters the function for the right
screennumber
         * @return nothing
         */

        private static void EncoderIsAutopilot()
        {   if (Data[1] == 1)
            {
                Console.WriteLine("System is in Autopilot.");

                if (Data[8] > 0x80)
                {
                    int increaseDegrees = Data[8] - 0x80;
                    Console.WriteLine("Increase heading degrees with" +
increaseDegrees);
                }
                else if (Data[8] < 0x80)
                {
                    int decreaseDegrees = 0x80 - Data[8];
                    Console.WriteLine("Decrease heading degrees with" +
decreaseDegrees);
                }
            }
            else
                return;


            _device.ReadReport(OnReport);
        }
        /**
        * Checks if the encoder is in Autopilot
        * if in autopilot, it detects rotations of the edge to increase or
decrease degrees on autopilot
```

```
        * if it is no longer in the screennumber for autopilot it breaks
the loop
        * once out of the loop, it enters the function for the right
screennumber
        * @param decreseDegrees the number of degrees to adjust auopilot
down
        * @param increseDegrees the number of degrees to adjust auopilot up
        * @return The new value of autopilot
        */

        private static void EncoderIsRadar()
        {
            if (Data[1] == 5)
            {
                Console.WriteLine("System is in Radar: range");
                if (Data[9] > 0x0)
                {

                    if (Data[9] = 0x84)
                    {
                        Console.WriteLine("Increase range");
                    }
                    else if (Data[9] = 0x86)
                    {
                        Console.WriteLine("Decrease range");
                    }
                }
                else if (Data[8] > 0x80)
                {
                    int increaseRain = Data[8] - 0x80;
                    Console.WriteLine("Increase radar range with" +
increaseRain);
                }
                else if (Data[8] < 0x80)
                {
                    int decreaseRain = 0x80 - Data[8];
                    Console.WriteLine("Decrease radar range with" +
decreaseRain);
                }
            }
            else if (Data[1] == 4)
            {
                Console.WriteLine("System is in Radar: gain, rain, sea");
                if (Data[8] > 0x80)
                {
                    int increaseGRS = Data[8] - 0x80;
                    Console.WriteLine("Increase heading degrees with" +
increaseGRS);
                }
                else if (Data[8] < 0x80)
                {
                    int decreaseGRS = 0x80 - Data[8];
                    Console.WriteLine("Decrease heading degrees with" +
decreaseGRS);
                }
            }
            else
                return;
        }
        /**
        * Checks if the encoder is in Radar, both screens
```

```
        * it checks if it is in the screen for range or the screen for
gain, rain and sea
        * detects rotations of the edge to increase or decrease degrees or
gain
        * if it is no longer in the screennumbers for radar it breaks the
loop
        * once out of the loop, it enters the function for the right
screennumber
        * @param increaseRain the number of times to increase range
        * @param decreaseRain the number of times to decrease range
        * @param increaseGRS the number of times to increase rain, gain or
sea
        * @param decreaseGRS the number of times to decrease rain, gain or
sea
        * @return The new value of gain, rain, sea or range
        */

        private static void EncoderIsECDIS()
        {
            if (Data[1] == 5)
            {
                Console.WriteLine("System is in ECDIS");
                SwipingDetected();
                if (Data[8] > 0x80)
                {
                    int increaseZoom = Data[8] - 0x80;
                    Console.WriteLine("Increase heading degrees with" +
increaseZoom);
                }
                else if (Data[8] < 0x80)
                {
                    int decreaseZoom = 0x80 - Data[8];
                    Console.WriteLine("Decrease heading degrees with" +
decreaseZoom);
                }
                if (Data[9] > 0x0)
                {
                    if (Data[9] = 0x84)
                    {
                        Console.WriteLine("Increase zoom");
                    }
                    else if (Data[9] = 0x86)
                    {
                        Console.WriteLine("Decrease zoom");
                    }
                }
            }
            else
                return;
        }
        /**
        * Checks if the encoder is in ECDIS
        * if in ECDIS, it detects rotations of the edge and touchzones to
increase or decrease zoom
        * and swiping to move map
        * if it is no longer in the screennumber for ECDIS it breaks the
loop
        * once out of the loop, it enters the function for the right
screennumber
        * @param decreseZoom the number of times to zoom out
        * @param increseZoom the number of times to zoom in
```

```
        * @return The new value of zoom and map posision
        */

        private static void SwipingDetected()
        {
            if (Data[13] > 0x0)
            {
                Console.WriteLine("Swiping detected");
            }
            else if (Data[14] == 0x1) {

                Console.WriteLine("Swipe up on map");
            }
            else if (Data[14] == 0x2)
            {
                Console.WriteLine("Swipe down on map");
            }
            else if (Data[14] == 0x4)
            {
                Console.WriteLine("Swipe left on map");
            }
            else if (Data[14] == 0x8)
            {
                Console.WriteLine("Swipe right on map");
            }
        }
        /**
        * Checks if swiping has occurred
        * if swiping has been detected, it checks in which direction
        */
    }


}
```