Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Jon Martin Kristiansen
Tobias Giverholt

Plattform-uavhengig grensesnitt for Fitness Machine Service

Programmatisk fjernstyring av treningsapparater

Bacheloroppgave i ingeniørfag, data
Veileder: Tomas Holt
Mai 2023

NTNU
Kunnskap for en bedre verden

Jon Martin Kristiansen
Tobias Giverholt

Plattform-uavhengig grensesnitt for Fitness Machine Service

Programmatisk fjernstyring av treningsapparater

Bacheloroppgave i ingeniørfag, data
Veileder: Tomas Holt
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Denne oppgaven omhandler implementasjon av Fitness Machine Service profilen for Low Energy Bluetooth i et programvarebibliotek som kan integreres i større prosjekter. Tenkt bruksområdet for et slikt bibliotek er integrasjon mot spillutviklingsrammeverk som Unity med formålet om å utvikle spill som anvender treningsapparat for å øve koordinasjon og balanse, eller rehabilitering.

Oppgaven er gitt under forutsetningen av løsningen skal være plattform-uavhengig for enkel distribusjon av sluttprodukter som utvikles med biblioteket. Denne løsningen er et forsøk på å forene de vidt forskjellige implementasjonene av Bluetooth over ulike operativsystemer til et standardisert, intuitivt .NET bibliotek. Under utvikling ble det lagt vekt på ekstensibilitet, slik at utviklere kan utvide støtte for alternative implementasjoner av Bluetooth, andre treningsapparater og alternativ datahåndtering.

Løsningen utforsker bruksområder for plattform invokasjon (P/Invoke) for å utnytte tverrplattformstøtte fra et forvaltet programvarerammeverk som .NET, imens implementasjonsspesifikk Bluetooth funksjonalitet delegeres til et low-level programmeringsspråk som C++. Resultater er et system basert på to komponenter som kan brukes av alle applikasjoner som støtter .NET Standard API, med fleksibel støtte for ulike systemarkitekturer og operativsystemer.

Implementasjonen av et slikt grensesnitt er relevant for helsesektoren, hvor pasientbehandling og helsemotivasjon kan strømlinjes ved hjelp av interaktive opplevelser som utnytter treningsapparater. Slik kan alminnelig treningsutstyr som støtter FTMS brukes til formål ellers forbeholt spesialisert maskinvare.

Abstract

This thesis details the implementation of the Fitness Machine Service profile for Low Energy Bluetooth as a software library that can be further integrated into larger projects. One imagined use case for such a library is integration with game development frameworks such as Unity, with the intent of developing games that utilize fitness machines to strengthen coordination and balance, or to rehabilitate.

The assignment is developed under the assumption that the final solution should be platform independent for simplified distribution of end products developed with the library. This solution is an attempt at uniting the differing implementations of Bluetooth across different operating systems into a standardized, intuitive .NET library. During development, emphasis was put on extensibility, to give developers additional support to alternative implementations of the Bluetooth tech stack, other fitness equipment and alternative data handling.

The solution explores possibilities of platform invocation (P/Invoke) to utilize cross-platform functionality of managed software frameworks such as .NET, while delegating implementation-specific Bluetooth functionality to lower-level programming languages such as C++. The result is a system of two components that can be consumed by all applications which supports the .NET Standard API, with flexible support for different system architectures and operative systems.

The implementation of such an interface is relevant for the health sector, where patient treatment and health motivation can be streamlined by interactive experiences that utilize fitness equipment. This way, common exercise equipment that supports FTMS can be used for purposes otherwise reserved for specialized hardware.

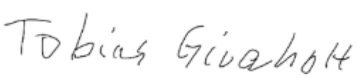
Forord

Denne bacheloroppgaven er skrevet i forbindelse med det avsluttende faget IDATT2900 for studieretning Bachelor i Ingeniørfag, Data (BIDATA) for vår 2023. Oppgaven er opprinnelig utsendt av 3D Motion Technologies, et selskap i regi av NTNU som spesialiserer seg i utvikling av programvareløsninger for VR-teknologi. Denne oppgaven var spesielt attraktiv da den var et avvik fra de nærmest rutinemessige web-utviklings prosjektene utført gjennom hele dataingeniørstudiet. Dette var et konsept som bygget på mye ukjent teknologi og nye rammeverk, og framsto som en ypperlig oppgave for å bli kjent med denne teknologien. I tillegg så vi på dette som en mulighet til å styrke våre evner til å tilnærme seg kunnskap forløpende over et prosjekt av denne omfanget.

Etter et oppstartsmøte med veileder og installasjon av tredemøllen som skulle brukes til oppgaven stod vi fritt til å utvikle løsningen i henhold til vår egen oppfatning av behovet. Oppgaven ble blitt utviklet over en periode på 4 måneder, under et fleksibelt prosjektrammeverk hvor teamet sto fritt til å undersøke relevant teknologi og jobbe tett på problemstillingen. Mange av de tekniske kravene var dermed ikke etablert ved prosjektstart, men heller blitt stiftet ut av nødvendighet gjennom prosjektperioden.

Vi vil takke 3D Motion Technologies for muligheten til å gjennomføre denne oppgaven med deres utstyr i deres laboratorium gjennom prosjektperioden. Vi vil også takke Tomas Holt, styreleder ved 3D Motion Technologies og universitetslektor ved institutt for datateknologi og informatikk, for veiledning under prosjektutvikling og bacheloroppgaven.

Trondheim 21. mai 2023



Jon Martin Kristiansen

Tobias Giverholt

Oppgavetekst

Oppgaven ble utsendt med følgene hensikt:

> Lage programvare (REST++) som enkelt gir tilgang og visualisering (web++) av treningsapparater og data ved rehabilitering/trening.

Videre presiserer oppgaven at det er ønskelig å utvikle ett grensesnitt mot Fitness Machine Service profilen for Bluetooth Low Energy. Grensesnittet skal gjøre det mulig å koble seg opp mot et treningsapparat som støtter protokollen, og hente data og kontrollere apparatet over Bluetooth.

Det er spesiell interesse for at løsningen støtter kontroll og avlesning av tredemøller. Grensesnittet skal kunne brukes i forbindelse med et eksisterende prosjekt i spillutviklingsrammeverket Unity som er avhengig av tredemøllefunksjonalitet. Dette skal oppnås gjennom en plugin eller lignende bibliotek som kan importeres i prosjektet.

Videre skal grensesnittet brukes til en utvikling av et grafisk brukergrensesnitt som kunne vise og sende instrukser til et treningsapparat gjennom en skrivebords-applikasjon. I tillegg skal grensesnittet kunne grafisk visualisere et tenkt treningsapparat som mottar instrukser.

Grensesnittet skal også kunne generaliseres som et REST-API for mer standardisert grunnlag for kommunikasjon. Det skal undersøkes hvorvidt dette er en aktuell løsning og hvilke funksjonaliteter som kan støttes via denne generaliseringen.

Se forprosjektplan i vedlegg for mer informasjon om oppgavens krav og bakgrunn.

Innhold

1 Introduksjon og relevans

1.1 Bakgrunn

VR-teknologi er en gren av moderne maskinvareutvikling som har skapt nye muligheter for menneske-maskin interaksjon. Spesielt i underholdningsindustrien har VR funnet kommersiell suksess, med nye og spennende måter å levere audiovisuelle inntrykk. VR-spill som kombinerer interaktive elementer fra videospill med den dimensjonale fordypelsen til virtual reality er oppslukende for mange. I løpet av de siste tiårene har oppfatning av spill som alminnelige leketøy utvidet seg til å også se nytten av interaktive opplevelser i utdanning og velferd. Den samme oppfatningen har fulgt utviklingen av VR-spill.

3D Motion Technologies er et selskap som utvikler løsninger innenfor velferdsteknologi. Dette innebærer systemer som hjelper funksjonsnedsatte mennesker med hverdagslige oppgaver. I denne forbindelse har de tidligere utviklet et VR-spill til rehabilitering av slagpasienter som lider av sensorimotiriske forstyrrelser. (Påsche, 2019) Spillere blir plassert på en tredemølle og vist en lang gangvei i VR med hinder de skal tråkke over. Dette skal gi et fysioterapeutisk utbytte i et mer engasjerende miljø en tradisjonelle terapisituasjoner. (Tokgöz, et al., 2022)

Til dette formålet behøves en tredemølle som kan kontrolleres i samhold med spillet. 3D Motion Tech har tidligere løst dette problemet ved å manipulere kretsløpet i en tredemølle via en mikrokontroller som kunne motta instrukser. Ulempen ved en slik løsning er behovet for fysisk inngrep i maskinvare som kan gjøre ugyldiggjøre garantien og hindre normal bruk av apparatet. Dette gjør at spillet er avhengig av spesielt modifiserte tredemøller, og begrenser hvordan spillet distribueres.

1.2 Problemstillinger

Prosjektet bygger på flere problemstillinger med forskjellige forutsetninger avhengig av hvilket perspektiv man setter løsningen i. Den mest overveiende problemstillingen, basert på krav fra arbeidsgiver og bakgrunnen for prosjektet, er som følger:

Hvordan kan en tredemølle kontrolleres programmatisk uten fysisk modifikasjon av maskinvare?

En forutsetning for oppgaven er av vi benytter Fitness Machine Service; en protokoll som lar Bluetooth-enheter lese av og kontrollerte treningsapparater som støtter tjenesten. Det er derimot ingen implementasjon av et grensesnitt for denne tjenesten som kan brukes med spillutviklingsrammeverket som 3D Motion Technologies har utviklet spillet i. Dette antyder at det ikke kun er 3D Motion Technologies som har behov for en slik løsning. Dermed kan vi utlede en sekundær problemstilling, basert på et behov i økosystemet for åpen kildekode, som kan utrykkes slik:

Hvordan kan et grensesnitt for Fitness Machine Service implementeres, slik at det kan brukes i prosjekter som ønsker å integrere treningsapparater?

Skal løsningen gjøres tilgjengelig for alle utviklere som ønsker slik integrasjon forutsettes det at løsningen er tilgjengelig for så mange kjøremiljøer som mulig. Dette er også relevant for sluttbrukere av prosjekter utviklet med en slik løsning. Dermed utleder vi en tertiær problemstilling, basert på forutsetningen beskrevet over, som kan utrykkes slik:

Hvordan kan denne løsningen gjøres tilgjengelig for flest mulig rammeverk på flest mulige plattformer og systemarkitekturer?

1.3 Hovedrapportens struktur

Seksjon 1 – Introduksjon og relevans beskriver bakgrunnen for oppgavens utsendelse og hvilke overordnede problemstillinger som kan utledes fra dette.

Seksjon 2 – Teori og relevant litteratur beskriver teorien bak fundamentale teknologier brukt for å realisere løsningen.

Seksjon 3 – Metode beskriver hvordan prosjektarbeidet forløp og bakgrunn for mindre

Seksjon 4 – Resultater beskriver prosjektets funksjonelle, ingeniørfaglige og administrative resultater.

Seksjon 5 – Diskusjon drøfter vesentlige punkter fra seksjon 4 og inneholder ytterlige kommentarer til resultatene.

Seksjon 6 – Konklusjon og videre arbeid beskriver hvorvidt løsningen tilfredsstiller problemstillingene utledet i seksjon 1, og videre arbeid som burde utføres for en mer tilfredsstillende løsning.

Samfunnspåvirkning drøfter videre påvirkning et slikt prosjekt kan ha på samfunnet på stor skala.

2 Teori og relevant litteratur

Fitness Machine Service

Grensesnittet skal utvikles til bruk over Bluetooth Low Energy. Dette er en kommunikasjonsstandard designet av The Bluetooth Special Interest Group som en del av 4.0 spesifikasjon for Bluetooth protokollen. Bluetooth LE ble utviklet som et alternativ med lavere energiforbruk, rettet mot periferiutstyr med begrenset batterikapasitet som skal sende og motta strengt definert data. Dette innebærer blant annet apparater til bruk i helse, sport og treningssektor. Slik kan mindre eksterne enheter uten direkte strømtilførsel kobles til og lese av apparater som støtter protokollen uten betydelig utslag på enhetens batterilevetid.

For bruk av Bluetooth LE utviklet Bluetooth SIG et antall profiler som kan implementeres av leverandører som ønsker å produsere Bluetooth LE-kompatibelt utstyr. Den mest utbredte av disse er Generic Attribute (GATT) profilen. Denne profilen tilbyr en generell implementasjon av Attribute protokollen, hvor en maskin selv kringkaster en rekke tjenester den støtter, med flere karakteristikker per tjeneste som beskriver diverse funksjoner av

tjenesten. Hver karakteristikk er definert med hvilke egenskaper den har, om den for eksempel kan skrives til, leses av eller bli lyttet til.

Fitness Machine Service er en slik tjeneste implementert via GATT profilen. Den er forbeholdt sport- og treningsapparater, og lar en ekstern enhet tilkoble og interagere med et sett arketypiske apparater som tredemøller, trappemaskiner, romaskiner og ergometersykler. Tjenesten har karakteristikker for å lese av egenskaper som momentan fart, tråkkfrekvens, kaloriforbruk eller påløpende treningstid, samt karakteristikk for å sende instruksjoner. Disse instruksjonene kan sette målverdier som fart på en tredemølle eller motstand i ergometersykkel. (Bluetooth SIG, 2017)

Platform Invocation Services

Platform Invocation Services, også kjent som P/Invoke, er et sett med funksjoner funnet i implementasjoner av Common Language Infrastructure, Microsoft sin tekniske spesifikasjon for et kjøremiljø som tillater mange programmeringsspråk å kjøre den samme maskinkoden på systemer med ulik arkitektur og operativsystem. Kode som kjøres i et slikt kjøremiljø betraktes som «managed code» eller forvaltet kode. Prosesser som kjører denne type kode drar nytte av funksjonalitet som automatisk minnehåndtering, avfallsinnhenting og unntakshåndtering, men er også begrenset av restriksjonene innført av en slikt plattform-agnostisk paradigme.

Ønsker man å bruke systemkall som ikke er tilgjengelig via CLI spesifikasjonen kan man bruke P/Invoke for å importere funksjonsutrykk fra et eksternt bibliotek utenfor CLI kjøremiljøet. Denne koden betraktes som «unmanaged code» eller uforvaltet kode. Prosesser som kjører denne type kode står fritt til å bruke plattform-spesifikke systemkall, men er selv ansvarlig for å håndtere minne. Ved korrekt bruk av P/Invoke kan man utvikle systemer som tilbyr et generelt grensesnitt for alle plattformer med et CLI kjøremiljø, hvor plattform-spesifikk funksjonalitet importeres fra plattform-spesifikke biblioteker. (Microsoft, 2022)

3 Metode

Programvarebiblioteket er et resultat av flere individuelle prototyper utviklet gjennom prosjektperioden for å bli kjent med teknologien brukt. Oppgaven har klare funksjonelle krav, men definerer ikke nøyaktig hvordan disse kravene skal implementeres eller hvilken form sluttproduktet skal ha. Det var dermed behov for å undersøke hvilke løsninger som var best egnet til de funksjonelle kravene. Dette resulterte i en rekke konseptbevis: implementasjoner av individuelle krav med mulig egnede teknikker og rammeverk. Disse undersøkelsene forløp uten en spesifikk utviklingsmetodikk. Isteden fulgte prosjektet en iterativ prosess hvor ett og ett krav ble vurdert, mulige prototyper som tilfredsstilte kravet ble utviklet, og den mest fleksible og robuste løsningen ble integrert inn i sluttproduktet.

Prosjektframdrift ble modellert i grove trekk, gitt lite til ingen kjennskap med teknologien som ble brukt. Prosjektplanen ble tegnet opp med flere diskre milepæler som representerer individuelle komponenter av en komplett løsning. Hver komponent dekker et sett med krav, og er et produkt av prototyper som implementerer disse kravene. Hver komponent ble estimert til å ta en andel av prosjektperioden, men hvor mange iterasjoner som trengtes for en enkel komponent kunne ikke estimeres.

3.1 Valg at teknologi

Den første iterasjonen (grunnleggende Bluetooth funksjonalitet som søk og oppkobling mot andre Bluetooth-enheter) ble først utviklet i C# for .NET rammeverket med Windows Runtime komponenter. Prototypen var funksjonell og forholdsvis lett å implementere, men var sterkt koblet av dets avhengighet til Windows Runtime. Dette ville gjort det umulig å utvide løsningen for andre operativsystemer som ikke støtter Windows API. I tillegg viste tidligere eksperimenter at bruk av Bluetooth rammeverk i C# gjorde det vanskelig å integrere disse prosjektene inn i spillutviklingsrammeverk som Unity.

SimpleBLE
Den mest praktiske løsningen til dette problemet var å delegere Bluetooth funksjonalitet til uforvaltet kode som ble eksekvert av .NET gjennom P/Invoke. Dette ble i realisert gjennom et bibliotek skrevet i C++. Å implementere Bluetooth funksjonalitet for C++ er derimot en mye mer involvert prosess enn for C#, og ville alene ha krevet mye av prosjektperioden. I den anledning ble OpenBluetoothToolbox sitt SimpleBLE bibliotek brukt for Bluetooth. SimpleBLE tilbyr full støtte på tvers av plattformer for flere programmeringsspråk inkludert C++, og en fleksibel lisens for kommersielt bruk. (OpenBluetoothToolbox, 2023)

.NET Standard
Det resulterende programvarebiblioteket er utviklet etter .NET Standard 2.1 spesifikasjonen. Dette er en formell spesifikasjon av .NET API-er over flere .NET implementasjoner. Dette medfører at programvare skrevet med hensyn til .NET Standard spesifikasjonen kan konsumeres av blant annet .NET Core og .NET Framework. Dette har stor betydning for hvordan løsningen kan integreres i større prosjekter. For eksempel kan et Unity prosjekt bygges med skript fra .NET Framework, hvorav et Windows Form applikasjon kan bygges på .NET 6.0 rammeverket. .NET Standard garanterer at begge rammeverkene kan importere og kjøre kildekoden, som sikrer prosjektfleksibilitet i tillegg til plattformfleksibilitet.

Visual Studio 2022
Hele prosjektet er utviklet i Visual Studio 2022, hvor hver komponent ble satt opp som individuelt prosjekt med et individuelt GitLab repository. En Visual Studio Solution fil ble deretter konfigurert til å bygge hver komponent og kopiere over avhengigheter til de relevante kjøremiljøene. Slik kan man hyppig kjøre og teste løsninger som bygger på flere komponenter fra forskjellige miljøer og språk. I tillegg har Visual Studio støtte for Windows Forms Designer, et verktøy for strømlinjet design av skrivebords-applikasjoner som bruker .NET rammeverket.

4 Resultater

4.1 Funksjonelle krav

Visjonsdokumentet beskriver et sett med funksjonelle krav utledet ved prosjektets oppstart. De følgende tabellene beskriver krav for hver diskre komponent av prosjektet, og hvorvidt de er oppfylt eller ikke. Se 5.1 Funksjonelle resultater for ytterlige kommentarer utover det som står skrevet i tabellen.

FTMS grensesnitt

Funksjonelle krav	Oppfylt	Ikke oppfylt	Kommentar
Søke etter Bluetooth enhet etter navn	**X**		
Søke etter Bluetooth enhet etter adresse	**X**		
Koble seg til funnet Bluetooth enhet	**X**		Implementert av SimpleBLE
Koble seg fra tilkoblet Bluetooth enhet	**X**		Implementert av SimpleBLE
Lese tilgjengelige tjenester og karakteristikk	**X**		Implementert av SimpleBLE
Se hvilke egenskaper som kan leses av	**X**		
Se hvilke egenskaper som kan skrives til	**X**		
Lese av treningsdata	**X**		Implementert via callback funksjon, kun implementert for tredemølle
Lese av treningsdata delt over flere sendinger		**X**	
Lese av treningsstatus	**X**		Implementert via callback funksjon
Lese av maskinstatus	**X**		Implementert via callback funksjon, viser kun status, ikke assosiert verdi
Lese av grenseverdier for instrukser	**X**		
Sende instrukser	**X**		Kun implementert for egenskaper støttet av tredemøllen brukt til utvikling
Støtte for flere plattformer	**X**		Støtter både Windows og Linux

Plugin for spillutviklingsrammeverk

Funksjonelle krav	Oppfylt	Ikke oppfylt	Kommentar
Komplett grensesnitt i Unity	**X**		Grensesnittet kan importeres som et bibliotek i scripts.
Komplett grensesnitt i Godot		**X**	Ikke nok tid, nedprioritert til fordel for Unity kompatibilitet.

GUI applikasjon

Funksjonelle krav	Oppfylt	Ikke oppfylt	Kommentar
Koble til FTMS kompatibelt treningsapparat	**X**		Kan koble til treningsapparat gjennom FTMS protokollen.
Avlesing av data	**X**		Avlesing av data skjer i hver sin tekstboks med tilhørende tittel.
Kontrollering av FTMS kompatibelt treningsapparat	**X**		Kan styre hastighet og vinkel.
Eventlog som viser oppdateringer med tredemøllen og evt. error-meldinger	**X**		Implementert slik at alle oppdateringer og error-meldinger blir loggført
2D eller 3D visualisering av tredemøllen i applikasjonen		**X**	Mangel på tid og sett på som unødvendig siden tredemøllen må være nærme for å demonstrere programmet
Illustrasjon som viser status på tilkobling	**X**		Implementert som to bilder som sier status basert på tilkobling

REST grensesnitt

Funksjonelle krav	Oppfylt	Ikke oppfylt	Kommentar
Komplett grensesnitt over REST		**X**	Ikke nok tid, nedprioritert da det er lite behov for et REST grensesnitt for kontroll av lokale enheter.

4.2 Ikke-funksjonelle krav

Visjonsdokumentet beskriver også ikke-funksjonelle krav til en akseptabel løsning med hensyn til ingeniørfaglige prinsipper. Dette kapittelet beskriver hvorvidt disse kravene ble tilfredsstilt med grunnlag for mangler.

Brukervennlighet
Løsningen er utformet slik at en utvikler skal kunne bruke grensesnittet uten kjennskap til den eksterne FTMS spesifikasjonen. Dette innebærer utfyllende kildekodedokumentasjon for offentlige og interne metoder. Essensiell informasjon fra spesifikasjonen, som oppløsning på parameterverdier for maskininstrukser, er også dokumentert i kildekoden og kan synliggjøres av Visual Studio eller lignende IDE-er med IntelliSense eller lignede.

Pålitelighet
Alle unntak kan håndteres av try-catch kodemønster, selv de kastet fra forvaltet kode. I tillegg er utviklere oppfordret av løsningens design til å bruke kodemønster som sikrer at verdier er til stede før de aksesseres. Slik kan grensesnittet implementeres i større prosjekter uten at den underliggende logikken risikerer å stoppe programflyt.

Krav om robust behandling av rå binærdata er ikke tilstrekkelig oppfylt. Nåværende løsning har ikke systemer implementert for å forsikre at data mottatt fra treningsapparater tolkes korrekt. Se *5.1 Ikke-funksjonelle resultater* for flere detaljer.

Ytelse
.NET grensesnittet, komponenten som kjøres som forvaltet kode, bygger på den asynkrone programmeringsmodellen "async/await" for C#. Dette gir utviklere mulighet til å implementere grensesnittet i prosjekter som er avhengig av responsive GUI elementer eller behandling av parallelliserte datasett. Lignede implementasjon for kjernekomponenten, som kjøres som uforvaltet kode, er ikke oppfylt. Se 5.1 Ikke-funksjonelle resultater for flere detaljer.

Utvidbarhet
Grensesnittet er utviklet slik at all Bluetooth-funksjonalitet må implementeres gjennom en oppgitt interface klasse. Løsningen tilbyr en implementasjon av denne klassen som bruker P/Invoke. Utviklere som ønsker andre tekniske implementasjoner av Bluetooth kan dermed bruke samme interface, og utnytte resten av grensesnittet for å motta og sende instrukser. Interface for andre typer treningsapparater er også inkludert, utover den nåværende implementasjonen for tredemøller.

4.3 Administrative krav

Dette kapittelet beskriver resultater med hensyn til prosjekthåndboken, inkludert timeplan, timeliste og veiledningsmøter. Denne dokumentasjonen, samt møteinnkallinger og referat finnes i vedlegg.

Timeplan
Prosjektframdrift samsvarte med timeplanen utledet under prosjektplanlegging for den første måneden av prosjektarbeidet. Etter første iterasjon av den grunnleggende grensesnittet ble det oppdaget flere uforutsette integrasjonskompilasjoner med neste milepæl. Tiden investert i å overkomme disse komplikasjonene medførte store avvik fra timeplanen, som videre ble grunnlag for å nedprioritere REST-API som en oppnåelig milepæl. Dette avviket var derimot forventet fra prosjektets oppstart, da teamets manglende kjennskap til den brukte teknologien gjorde det umulig å fastslå estimert tidsbruk for hvert delprosjekt.

Timeliste
Prosjektarbeidet konkluderer med anslagsvis 400 timer per teammedlem. Dette avviker fra den oppgitte arbeidsbelastningen på 500 timer beskrevet i bacheloroppgavens generelle retningslinjer. Den største faktoren for dette avviket er emnet Ingeniørfaglig Systemtenkning, som kjørte prosjektoppgave parallelt med bachelorprosjekt fra uke 3 til uke 10. Det estimeres at 50 timer gikk med til prosjektoppgaven. Videre årsaker til avvik skyldes kurs i Vitenskapsteori og metode ved prosjektoppstart, som ikke er loggført da det ikke er i direkte forbindelse med bacheloroppgaven.

Veiledningsmøter
Under prosjektperioden ble det avholdt 3 møter med veileder; et oppstartsmøte, et møte halvveis gjennom prosjektperioden, og et avsluttende møte 3 uker før prosjektets innlevering. Dette skyldes en fleksibel oppgavebeskrivelse som har gitt teamet frihet til å utforme oppgaven etter eget initiativ. Oppstartsmøte la grunnlaget for oppgaven og ønskene til veileder, som for denne oppgaven også stod som oppdragsgiver. Andre møte var forbeholt demonstrasjon av daværende framdrift og videre betraktninger for prosjektets fortsettelse. Siste møte la grunnlaget for prosjektdokumentasjon og krav til ferdigstilling av hovedrapport med vedlegg.

5 Diskusjon

Den følgende seksjonen diskuterer resultatene fra prosjektarbeidet, med hensyn til prosjektkrav og erfaringer med teknologien brukt, samt refleksjon over gruppearbeidet innad i teamet. Arbeidet har blitt fordelt etter praktisk kunnskap slik at hvert medlem tar ansvar for deler av prosjektet etter erfaring, interesse eller disponibel arbeidskraft. Dette har resultert i en teamdynamikk hvor ett medlem fokuserer på utvikling av grensesnittet, og hvor et annet medlem fokuserer på implementasjon mot det grafiske brukergrensesnittet og Unity kompatibilitet. I begynnelsen av prosjektet var det størst fokus på å utvikle det generelle grensesnittet. Ved dette stadiet jobbet teamet tett med hverandre for å samkjøre forventinger til den endelige løsningen. Videre utover i prosjektet oppstod det todelte skille mellom grensesnitt og implementasjon, hvor arbeidsoppgaver ble mer uavhengige.

5.1 Funksjonelle resultater

Resultater oppnådd for dette prosjektet er tilfredsstillende med hensyn til kravene utledet av problemstillingen. Som et produkt oppfyller grensesnittet kravene som er nødvendige for å bli brukt av applikasjonsutviklere, som er målgruppen for programvaren.

FTMSLib og FTMSCore
Det grunnleggende grensesnittet implementeres gjennom to komponenter. FTMSCore er utviklet i C++ og håndterer all Bluetooth-funksjonalitet gjennom dynamisk linket kildekode. Denne koden brukes av FTMSLib i .NET, og danner et grensesnitt for komplett implementasjon av FTMS. Samspill mellom disse komponentene er muliggjort av P/Invoke. Bruk av P/Invoke på dette vis har vært utfordrende, da det krever viderekommen kjennskap til minnehåndtering for den relevante systemarkitekturen. Alle argumenter som utveksles mellom de to komponentene må tolkes på samme måte, for at ikke minneregioner skal mistolkes. Dette er spesielt viktig for strenger med variabel koding som tar ulike antall bytes per karakter. Støtte for disse konversjonene håndteres i stor grad av .NET sitt *InteropServices* navnerom, men medfører til tross for dette utallige særegenheter.

Bruk av SimpleBLE
Å ta utgangspunkt i et eksternt bibliotek som SimpleBLE har gitt teamet mulighet til å fokusere på den opprinnelige problemstillingen, framfor å utvikle en egen BLE implementasjon. Slik vil støtte for Bluetooth vedlikeholdes indirekte gjennom SimpleBLE blir, selv etter oppløsning av det opprinnelige bachelorgruppen. Dette forutsetter at valgt bibliotek blir aktivt vedlikeholdt for de relevante plattformene. Å ta utgangspunkt i et bibliotek for å implementere funksjonalitet teamet mangler kjennskap til risikerer at det bygges en avhengighet til et tredje parti. Dette tredje partiet er ikke bundet til arbeidskontrakten som skal forsikre prosjektgang, og dette kan forstyrre framdrift.

SimpleBLE har i skrivende stund kun en aktiv bidragsyter til biblioteket. I løpet av prosjektperioden støtet teamet på udefinert adferd i forbindelse med biblioteket og tok kontakt med bidragsyteren. Tidsforskjell og personlige affærer i denne personens liv resulterte i en ukes passasje før problemet ble løst. Hadde problemet vært mer omfattende kunne det skapt betydelig forsinkelser i utvikling, og vært grunnlag for å omstrukturere kildekoden til å ekskludere biblioteket. Toleranse for en slik risiko avhenger av oppdragsgiver. I dette tilfellet hadde oppdragsgiver godkjent bruk av SimpleBLE, og dermed akseptert risiko for slike komplikasjoner.

Treningsdata mottatt over flere sendinger
Individuelle treningsapparater støtter kringkasting av ulike mengder data. For enkelte apparater vil denne datamengden overskride det som støttes for én sendelse over BLE. For denne situasjonen har FTMS spesifikasjonen definert et unntakstilfelle for å tolke flere diskree sendelser som en kontinuerlig og øyeblikkelig sending. Tredemøllen brukt til utvikling av grensesnittet støtter ikke nok egenskaper til å sende data som overskrider denne grensen. Dette har medført at grensesnittet er utviklet uten hensyn til dette unntakstilfellet. Konsekvensen av dette betyr at enkelte treningsapparat ikke vil kunne samhandle med grensesnittet. Problemet ble anerkjent sent i prosjektperioden og det antas at en betydelig restrukturering av kildekoden må gjøres for å løse det. Løsningen leveres derfor uten dette funksjonelle kravet oppfylt.

Visualisering av tredemølle i GUI
Kravet om en 2D eller 3D modell som illustrerer tredemøllen ikke fullført av flere grunner. Programmet er utviklet for å kun kunne endre på data etter det har blitt koblet til et treningsapparat. Brukeren må dermed befinne seg innenfor rekkevidde til Bluetooth signalet. treningsapparatet så på det som unødvendig i forhold til andre prioriteter. En slik modell ville også vært meget utfordrende å implementere, og ble dermed nedprioritert da GUI applikasjonen ble kontekstualisert til å kun tjene som et implementasjonseksempel.

5.2 Ikke-funksjonelle resultater

Pålitelighet
Nåværende løsning har ingen systemer som forsikrer integritet og korrekt tolkning av treningsdata. Dette skyldes i hovedsak mangelfull dokumentasjon i FTMS spesifikasjonen, hvor ulike verdier som sendes fra maskinen ikke er strengt definerte datatyper. Antall bytes som inngår i hver verdi er dermed ukjent og må estimeres etter en "prøv-og-feil" metodikk. Løsningen er utviklet og testet mot kun én tredemølle, og det er ukjent hvorvidt andre apparater vil avduke flere særegenheter.

Ytelse
FTMSCore, komponenten som kjøres som uforvaltet kode, bygger ikke på en tilsvarende asynkron programmeringsmodell som FTMSLib og kan forårsake blokkasje om den blir kalt på fra flere asynkrone prosesser. Realistisk sett vil denne blokkasjen ha et neglisjerbart tidsutslag, med hensyn til det motoriske begrensningene til treningsapparatet. Det er få tilfeller hvor en utvikler behøver flere asynkrone kjernekall, og hvor nåværende ytelse ikke vil være tilstrekkelig.

Utvidbarhet
Implementasjon av forskjellige typer treningsmaskiner forutsetter at hvert type apparat har en klasse som utvider en parametrisert instans den abstrakte treningsapparatklassen. Parameteren må være en klasse med en konstruktør som tolker et datalast fra treningsapparatet for en type apparat. Treningsapparatklassen bruker så refleksjon til å initialisere en dataklassen for hver datalast som mottas.

Bruk av refleksjon på denne måten skaper et svakt ledd i utformingen til løsningen. Det er ingen eksplisitte mønster som forsikrer at parameteren har en gyldig konstruktør. En klasse uten gyldig konstruktør medfører at applikasjonen kompilerer, men feiler ved kjøretid. Det forutsettes dermed at en utvikler som ønsker å støtte andre typer treningsapparater har implisitt kunnskap til dette designvalget, enten gjennom systemdokumentasjon eller analyse av de allerede implementerte løsningene. Dette designvalget ble valgt som et kompromiss for å nærmest etterligne adferden til FTMS, hvor det eneste skillet mellom individuelle typer maskiner er treningsdata som mottas via et C# event. Framfor å lage en unik klasse for hvert treningsapparat, var det heller hensiktsmessig å gi event signaturen et argument av en type som blir parametrisert på superklassen.

6 Konklusjon og videre arbeid

Med utgangspunkt i den mest overveiende problemstillingen, utledet i seksjon 1.2:

Hvordan kan en tredemølle kontrolleres programmatisk uten fysisk modifikasjon av maskinvare?

kan det konkluderes at en tilfredsstillende løsning er oppnåelig ved å benytte seg av FTMS profilen, forutsett at det finnes apparater som implementerer profilen. Dette er demonstrert av teamet, som produserte et bibliotek ved bruk av verktøyene beskrevet i denne rapporten. Biblioteket skal gjøre det mulig å observere og kontrollere treningsapparat innenfor begrensinger definert av selve apparatet. Med unntak av støtte før splittede datalaster er biblioteket en komplett implementasjon av FTMS for tredemølle. Videre arbeid burde derfor omhandle denne mangelen, for å forsikre universell kompatibilitet.

Med utgangspunkt i den sekundære problemstillingen:

Hvordan kan et grensesnitt for Fitness Machine Service implementeres, slik at det kan brukes i prosjekter som ønsker å integrere treningsapparater?

kan det konkluderes at et slikt grensesnitt er oppnåelig ved å skille logikk mellom forvaltet og uforvaltet kode. For komplekse prosjekter er det ofte ønskelig å jobbe med robuste og "high-level" programmeringsrammeverk. Samtidig er systemfunksjonalitet som Bluetooth mest fleksibelt i "low-level" programmeringsrammeverk. Bruk av P/Invoke for å samkjøre disse miljøene gir en samlet implementasjon som kjører uavhengig av overordnet rammeverk. Dette illustreres i vedlagte implementasjoner av et WinForms grensesnitt og en Unity demonstrasjon. Her utnyttes støtte for .NET Standard, slik at begge rammeverkene kan utnytte funksjonaliteten til grensesnittet. Videre arbeid burde fokusere på P/Invoke og hvordan denne teknologien kan brukes i prosjektet på en tryggere og mer strømlinjet måte. Dette nevnes spesielt med tanke på minnehåndtering, som kan forårsake udefinert adferd om et prosjekt som implementerer grensesnittet blir tilstrekkelig komplekst.

Med utgangspunkt i den tertiære problemstillingen:

Hvordan kan denne løsningen gjøres tilgjengelig for flest mulig rammeverk på flest mulige plattformer og systemarkitekturer?

kan det konkluderes at en slik løsning kan utnytte det logiske skillet mellom forvaltet og uforvaltet kode for å oppnå plattform-uavhengighet. .NET bygger på Common Language Runtime, som kan kjøre forvaltet kode for alle arkitekturer som støtter rammeverket. Ved å delegere Bluetooth til et ekstern bibliotek som tilbyr low-level implementasjoner av nødvendig funksjonalitet for en rekke plattformer og arkitekturer kan løsningen brukes i flest mulig kjøremiljøer med minst mulig modifikasjon. Videre arbeid burde innebære mer omfattende undersøkelser over systemkriterier for forskjellige plattformer, og hvordan grensesnittet kan utvides for størst mulig tilgjengelighet.

Samfunnspåvirkning

Den opprinnelige problemstillingen som beskriver fjernkontroll at treningsapparater er basert på 3D Motion Technologies sitt ønske om å utvikle et VR-spill for rehabilitering av slagpasienter. Dette er et av flere type tilfeller hvor pasientbehandling kan gjennomføres med VR-teknologi. Spesielt hos de med nedsatt motoriske evner pga. slag eller cerebral parese er det aktuelt med VR til rehabilitering. Det er mulighet til å motivere pasienter med interaktive mål og audiosensorisk feedback, og kan installeres i hjemmet til pasienter hvor det kan brukes med minimalt oppsyn, som vil avlaste helsepersonell. Avhengig av valgt teknologi kan slike løsninger kutte kostnader på individuell behandling og gi flere pasienter nødvending behandling. (Tokgöz, et al., 2022)

Grensesnitt har også bruk utenfor motorisk terapi og rehabilitering. Overvekt og fedme er et påløpende problem i verden, og skyldes blant annet økende mangel på fysisk aktivitet pga. sittestillende arbeidsformer og økt urbanisering. (World Health Organization, 2021). Treningsapparater er relativt plassbesparende og kan bidra til økt mosjonering i private hjem, men flere mangler motivasjonen til å investere i og aktivt bruke slike apparater. Her har VR mulighet til å supplere denne motivasjonen gjennom sitt iboende underholdningspotensial. Enkelte VR-spill som *Beat Saber* har allerede demonstrert muligheter for kardiovaskulær trening og har oppnådd betydelig kommersiell suksess, og står som eksempel til mulighetene virtual reality kan ha for personlig helse. (Virtual Reality Institue of Health and Exercise, 2018).

Mulighet for å programmatisk kontrollere og avlese treningsutstyr har betydning for produksjonen av disse apparatene. En produsent kan ønske å utvikle treningsapparat som oppfyller visse funksjonelle krav, for eksempel til bruk ved rehabilitering, integrasjon mot Fitbit eller Apple Watch, eller proprietær spillutvikling. Uten en universell protokoll er det mulig disse kravene implementeres i maskiner uten hensyn til gjenbruk til andre formål. Med en protokoll som FTMS og et generelt tilgjengelig grensesnitt kan derimot same maskinvare brukes til flere formål, potensielt utover det prosjekterte livsløp.

Referanser

Bluetooth SIG, 2017. *Fitness Machine Service.* [Internett]
Available at: https://www.bluetooth.com/specifications/specs/fitness-machine-service-1-0/

Microsoft, 2022. *Platform Invoke (P/Invoke).* [Internett]
Available at: https://learn.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke

OpenBluetoothToolbox, 2023. *SimpleBLE.* [Internett]
Available at: https://github.com/OpenBluetoothToolbox/SimpleBLE

Påsche, A. S., 2019. *Virtual Reality i tredemølletrening av slagpasienter.* s.l.:Bachelor thesis at NTNU for 3D Motion Technologies AS.

Tokgöz, P. et al., 2022. Virtual Reality in the Rehabilitation of Patients with Injuries and Diseases of Upper Extremities. *Healthcare (Basel, Switzerland).*

Virtual Reality Institue of Health and Exercise, 2018. *Beat Saber.* [Internett]
Available at: https://vrhealth.institute/portfolio/beat-saber/
[Funnet 25 04 2023].

World Health Organization, 2021. *World Health Organization.* [Internett]
Available at: https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight

Vedlegg

Denne seksjonen skal gi oversikt over relevante vedlegg til prosjektdokumentasjon. Alle vedlegg er tilgjengelig via NTNU Open.

Forprosjektplan

Dette dokumentet ble utledet i begynnelsen av prosjektperioden og beskriver overordnede krav og forventninger til prosjektet.

Visjonsdokument

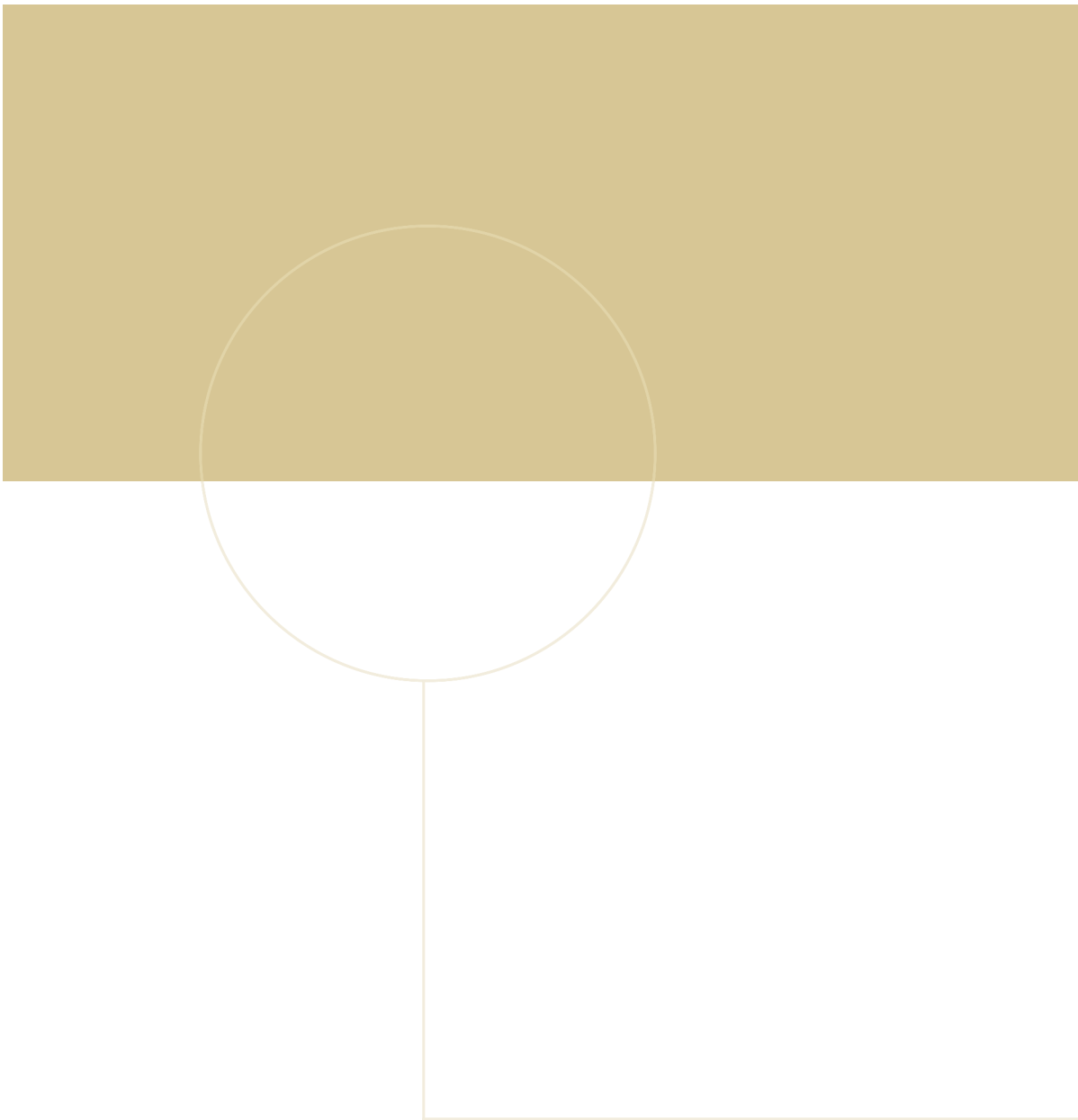
Dette dokumentet beskriver problemstilling og interessenter/brukere for en mulig løsning. I tillegg beskrives user stories for en tenkt løsning, og hvilke funksjonelle krav en slik løsning må oppfylle med hensyn til oppdragsgivers ønsker, user stories og ingeniørfaglig integritet. Visjonsdokumentet omfatter dokumentasjon man ellers ville ført i et separat kravdokument. Dette gjøres da et slikt prosjekt, hvor sluttprodukt er et kildekodebibliotek istedenfor en applikasjon for sluttbrukere, ikke produserer dokumentasjon egnet for et typisk kravdokument.

Systemdokumentasjon

Dette dokumentet beskriver prosjektarkitekturen og inneholder instrukser for bygging, installasjon og bruk av biblioteket. I tillegg beskriver dokumentet betraktninger for videre implementasjoner og særegenheter til den spesifikke tredemøllen brukt til utvikling av biblioteket. Systemdokumentasjonen er skrevet på engelsk for å best kommunisere utviklingsterminologi, og for å støtte internasjonal bruk.

Prosjekthåndbok

Prosjekthåndboken inneholder prosjektadministrativ dokumentasjon som møteinnkallinger, timelister og framdriftsplan. Merk at timeliste og framdriftsplan er vedlagt som separate Excel-ark.



NTNU

Kunnskap for en bedre verden