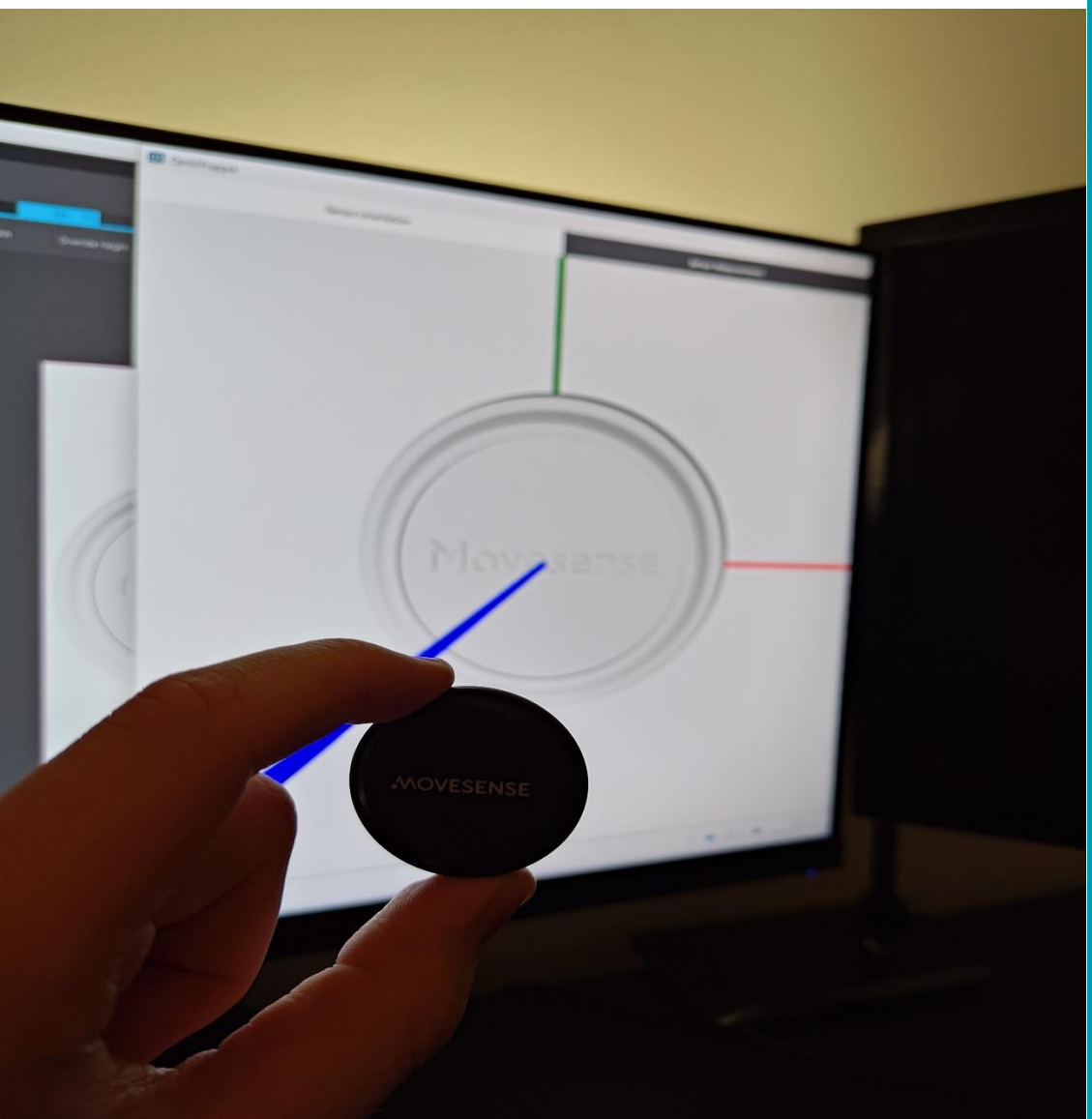
Koponen Tero

MotiQ Path construction using IMU measurements



Bachelor of Engineering, Information and communication technology

Education

Fall 2023

KAMK • University of Applied Sciences

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Koponen Tero

Työn nimi: MotiQ Path construction using IMU measurements

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka, Peliteknologia

Asiasanat: Ineartiamittausyksikkö, fuusioalgoritmit, ranneanturi, C++, ohjelmointi, QML

Tämä opinnäytetyö keskittyi ensisijaisesti Qt-sovelluskehystä hyödyntävän mobiilisovelluksen kehittämiseen. Se sisälsi myös ohjelmiston kehitystä Movesense-sensorille C++-ohjelmointikielellä. Tavoitteena oli kehittää sovellus, joka visualisoi Movesense-sensorilta saapuvaa dataa. Aluksi tutkittiin työn taustoja sekä työn suorituspaikan kontekstia. Teoreettisessa osuudessa käsiteltiin AHRS-teknologiaa, paikan ja kiihtyvyyden integrointia, laitteen suuntaa, kalibrointia sekä GATT-protokollaa. Teorian pohjustuksen jälkeen siirryttiin ohjelmointivaiheeseen.

Käytännön toteutuksessa syntyi sovellus, joka mahdollistaa Movesense-sensorin lähettämän datan visualisoinnin. Sensori välittää sovellukselle tietoa sen sijainnista, orientaatiosta sekä raakaa dataa kiihtyvyysanturilta, gyroskoopilta ja magneettimetriltä ennen ja jälkeen kalibroinnin. Tämän avulla voitiin vertailla datan muutosta kalibroinnin yhteydessä. Sijaintitietoa hyödynnettiin objektin liikuttamisessa näytöllä, kun taas orientaatiota käytettiin objektin kääntämiseen.

Projektin edetessä havaittiin, että reaaliaikaisen liikkeen visualisointi pelkän IMU-sensorin avulla on haastavaa ja vaatii tarkan sensorin sekä huolellisen kalibroinnin. Vaikka alkuperäistä tavoitetta ei saavutettu täysin, opittiin paljon ohjelmiston kehittämisestä Movesense-sensorille ja Qt ohjelmisto kehitysympäristöstä.

KAMK • University of Applied Sciences

Abstract

Author(s): Koponen Tero

Title of the Publication: MotiQ Path construction using IMU measurements

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: Inertial measurement unit, fusion algorithms, wrist-mounted sensor, C++, programming, QML

This thesis primarily centered on the development of a mobile application utilizing the Qt application framework. It also involved developing software for the Movesense sensor using C++ programming language. The goal was to develop an application that visualizes the data coming from the Movesense sensor. At first, the background of the work and the context of the place where the work was performed were studied. The theoretical part covered AHRS technology, position and acceleration integration, device direction, calibration, and GATT protocol. After priming the theory, the programming phase followed.

The practical implementation resulted in an application that enables the visualization of the data sent by the Movesense sensor. The sensor transmits information to the application about its location, orientation, and raw data from the accelerometer, gyroscope, and magnetometer before and after calibration. This made it possible to compare the change in data during calibration. Location information was used to move the object on the screen, while orientation was used to rotate the object.

As the project progressed, it was discovered that visualizing real-time movement using only an IMU sensor is challenging and requires a precise sensor and careful calibration. Although the original goal was not fully achieved, a lot was learned about software development for the Movesense sensor and the Qt framework development environment.

Starting words

I am deeply grateful to Qt Group for their instrumental role in making this thesis project possible. Additionally, I extend my appreciation to Movesense for their valuable assistance and help.

Table of contents

Attachments

List of symbols

IMU	Inertial Measurement Unit
AHRS	Attitude and Heading Reference System
BLE	Bluetooth Low Energy
API	Application Programming Interface
GPS	Global Positioning System
GATT	Generic Attribute Profile
UI	User Interface
GUI	Graphical User Interface
UX	User Experience
IDE	Integrated Development Environment
MEMS	Microelectromechanical System
MOC	Meta-Object Compiler
SIG	Special Interest Group
DFU	Device Firmware Update
VAT	Value-Added Tax
RAM	Random-Access Memory

1 Introduction

The purpose of this bachelor's thesis is to test the Movesense sensor and find out if it can be used to obtain accurate enough data to track movement in 3D space. For this purpose, the objective is to find and develop an AHRS-algorithm for the Movesense HR2 sensor that gives information of the sensor's orientation while using the sensor's accelerometer to determine the direction to which the sensor is moving. Additionally, double integrating the accelerometer data to gain position with drift removed.

In this thesis, the Qt framework is used as the application development framework. Additionally, the plan is to connect the Movesense sensor to a mobile phone via Bluetooth. In the end of the thesis, an application will be made with Qt that helps in using the sensor and offers the user an opportunity to change the measurement frequency, calibrate the sensor and spectate the data coming from the sensor. Custom software is flashed to the sensor, it listens and answers to the commands coming from the application.

The result can be, for example, shown to students who are interested in data, or developers of exercise applications, who could use precise movement data to enhance their applications. Additionally, the work is usable for organizations that are interested in developed motion data technology. This thesis could bring up new and exciting points of view and use cases.

This work sets out to figure out the problems that lie in motion tracking and develop a functional work that offers precise and trustworthy data from the Movesense sensor. The suitability of the Movesense sensor for motion tracking will be studied, along with improvements to its usability.

Limitations will be encountered in the thesis, such as possible technical challenges and hardware limitations. The goal is to produce a high-quality solution that offers accurate and reliable sensor information. As qualitative criteria, high measurement and monitoring accuracy and user-friendliness when using the application is aimed for.

2 Theoretical background

The theoretical background chapter lays the groundwork for understanding the various components and concepts crucial to the research. It begins by delving into the foundational principles of Qt Company, Qt Creator, and Qt Designer, which serve as essential tools in software development for the client interface. Moving forward, the discussion expands to encompass the physical principles underlying moment of inertia and inertial measurement units (IMUs), including the operation of Microelectromechanical systems (MEMS).

Next, the chapter explores the concept of attitude and heading reference systems (AHRS) and the vital process of sensor fusion, which integrates data from multiple sensors to improve accuracy and reliability. It further addresses techniques for determining position from acceleration and the inherent challenges associated with acceleration integration and orientation estimation. Calibration procedures are then elucidated to ensure the accuracy of sensor measurements.

Shifting focus, the next chapters explore the communication protocols of Bluetooth Low Energy (BLE), including Generic Attribute Profile (GATT), services, and characteristics, which facilitate seamless interaction between devices. Throughout the discussion, emphasis is placed on elucidating the interplay between these components and their significance in achieving the research objectives.

2.1 The Qt Company

For the client in this thesis, Qt serves as a pivotal multiplatform framework library facilitating application development. Qt, renowned for its versatility, is designed to support desktop, embedded, and mobile platforms seamlessly. Written in C++, it extends the language's capabilities through features like signals and slots using the Meta-Object Compiler (MOC) preprocessor.

Initially developed by Trolltech, now known as The Qt Company, its robustness and adaptability make it an indispensable tool in software development. More detailed insights into Qt Creator and Designer will be provided in the subsequent chapter. [1]

2.1.1 Qt Creator & Qt Designer

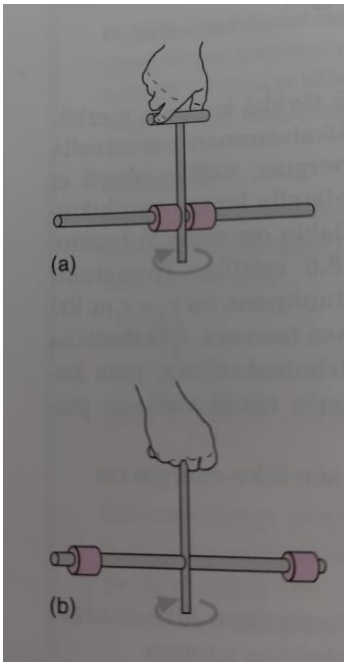
Qt Creator is an integrated development environment (IDE) designed for efficient and streamlined application development using the Qt framework. It provides a comprehensive set of tools for writing, compiling, and debugging Qt applications.

Qt Designer, on the other hand, is a graphical user interface (GUI) design tool that works in conjunction with Qt Creator. It is a visual layout designer allowing developers to design and customize the UI of their Qt applications without writing code manually.

2.2 Moment of inertia

According to Britannica, moment of inertia is defined as the quantitative measure of an object's resistance to angular acceleration in a specific axis. It is calculated based on the mass of the object and the distribution of that mass around the axis of rotation. The farther the mass is distributed from the axis of rotation, the larger the moment of inertia. Moment of inertia plays a crucial role in rotational dynamics, determining how objects respond to rotational forces. [2]

For example, a device can rotate around its vertical axis. Two symmetrical masses can be locked to the horizontal bar at any point. In picture 1 (a) locking the mass close to the vertical bar, the moment of inertia is small thus making the rotational moment easy to produce. In picture 1 (b) the mass is in the ends of the bar, making the moment of inertia large and changing the motion to be more laborious. [3]



Picture 1. Example of inertia.

Understanding moment of inertia is essential for accurately interpreting the data from the inertial measurement unit (IMU) in terms of rotation and position. Since the IMU measures angular velocity and acceleration, which are directly related to rotational motion, knowledge of moment of inertia helps in interpreting these measurements in the context of the physical properties of the rotating object or system.

2.3 Inertial measurement unit

An IMU is a device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, utilizing a combination of accelerometers, gyroscopes, and magnetometers. These measurements can be used to calculate the device's orientation, velocity, and position relative to a known starting point or coordinate system. IMUs find applications in various fields such as automotive, and robotics for navigation, motion tracking, and stabilization purposes.

The IMU serves as a critical component within the work, tasked with acquiring precise measurements pertaining to the device's orientation and acceleration. IMU provides real-time data regarding the device's spatial orientation in three-dimensional space, including its pitch, roll, and

yaw angles. This information is indispensable for accurately determining the device's position and orientation relative to a reference frame.

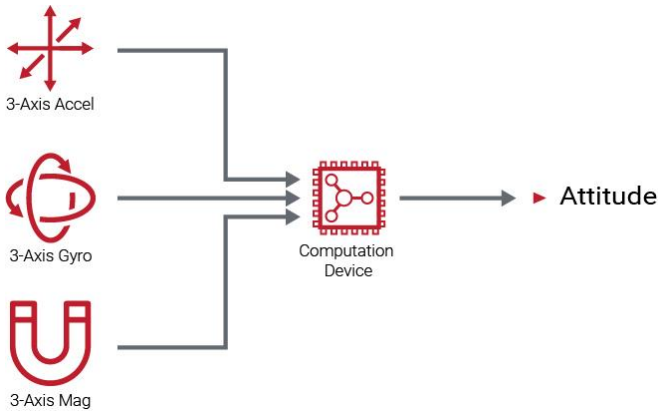
2.4 Microelectromechanical system

MEMS are miniature integrated systems that combine mechanical and electrical components on a microscale. They are fabricated using semiconductor manufacturing techniques, allowing for the creation of tiny sensors, actuators, and other devices. MEMS devices can sense and manipulate physical phenomena such as motion, pressure, temperature, and light. [4]

MEMS have applications across various fields, including consumer electronics, automotive, healthcare, and aerospace. MEMS technology enables the development of compact, low-power, and cost-effective devices with high precision and reliability. Overall, MEMS play a crucial role in advancing technology by providing miniaturized solutions for a wide range of applications. [4]

2.5 Attitude and heading reference system

AHRS uses an inertial measurement unit consisting of MEMS sensors to measure the angular rate, acceleration, and Earth's magnetic field. These measurements determine the orientation (attitude) and direction (heading) of an object, such as an aircraft or a vehicle. These systems employ sensor fusion algorithms to integrate data from multiple sensors and provide an accurate estimation of the object's attitude and heading. [5]



Picture 2. Example of AHRS component Diagram. [5]

In Picture 2, the AHRS receives raw data from IMU accelerometers, gyroscopes, and magnetometers. This data is then processed by a sensor equipped with the AHRS algorithm, converting intricate motion measurements into practical orientation information.

2.6 Sensor fusion

Sensor fusion is a process that involves combining data from multiple sensors to improve accuracy, reliability, and robustness in estimating the state of a system. This technique is commonly used in various fields such as robotics, navigation, and augmented reality to obtain a more comprehensive understanding of the surroundings or the system's state. [6]

In sensor fusion, different types of sensors are often employed, including accelerometers, gyroscopes, magnetometers, GPS receivers, cameras, and lidars. Each sensor type provides unique information about the system, but they also have limitations and can be affected by noise, biases, and environmental conditions. By fusing data from multiple sensors using sophisticated algorithms such as Kalman filters, complementary filters, or particle filters, it's possible to compensate for the limitations of individual sensors and obtain a more accurate and reliable estimation of the system's state or the environment's characteristics. [7]

2.7 Position from acceleration

Position estimation from acceleration integration is a fundamental technique utilized in IMUs for determining an object's spatial displacement over time. The principles, challenges, and applications of position estimation through acceleration integration in IMUs will be explored.

Acceleration integration involves integrating accelerometer measurements over time to estimate velocity and subsequently integrating velocity to compute position. This process relies on the fundamental principles of calculus, where acceleration represents the rate of change of velocity, and velocity represents the rate of change of position.

2.7.1 Challenges in Acceleration Integration

Despite its conceptual simplicity, acceleration integration for position estimation is susceptible to various sources of error. Sensor noise, biases, and drift in accelerometer measurements can introduce errors into the integration process, leading to inaccuracies in velocity and position estimates over time. Additionally, integration errors accumulate over time, resulting in drift in position estimates, particularly in long-term measurements.

Acceleration data is directly obtained from the sensor, typically acceleration is represented in three dimensions: x, y, and z. In the case of Movesense sensor, the z-axis generally experiences a force of 9.81 m/s^2, corresponding to the acceleration due to gravity.

Determining the velocity of the sensor, the acceleration data is integrated over time. The integration process involves summing up the acceleration values over each time interval, typically represented as deltaTime, to obtain the change in velocity. The integration equation, velocity = velocity0 + linearAcceleration * deltaTime, where velocity represents the current velocity, velocity0 represents the previous velocity, linearAcceleration represents the current acceleration experienced by the sensor, and deltaTime represents the time between consecutive acceleration samples.

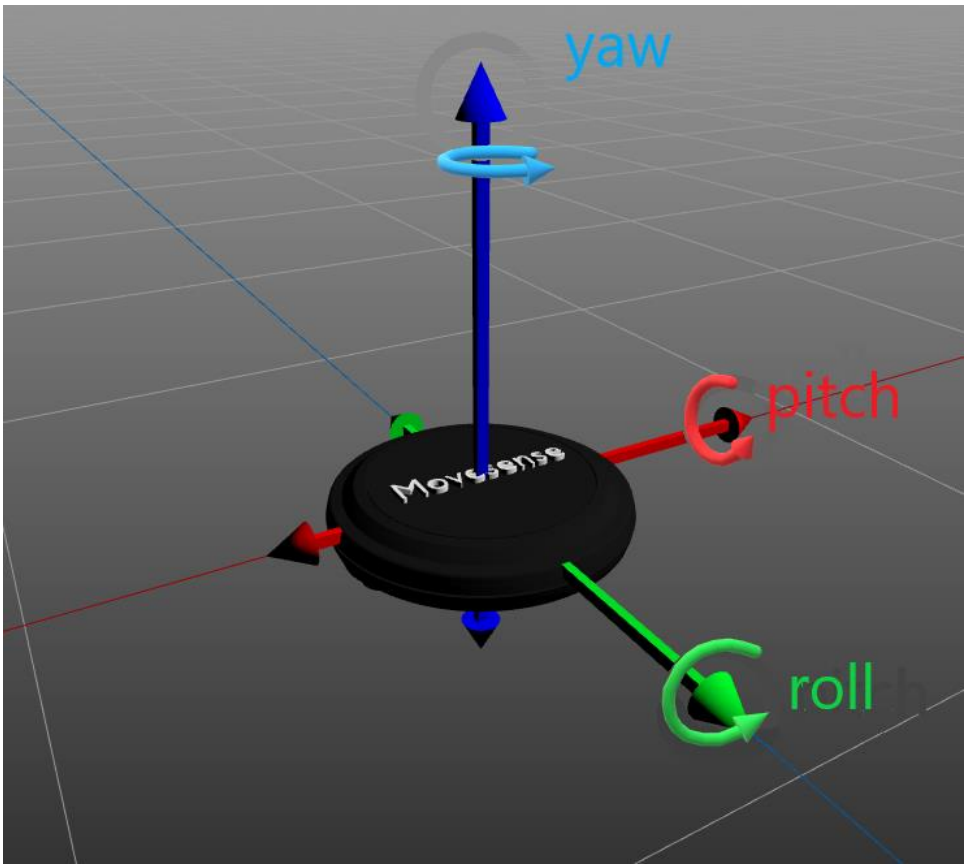
Given a sampling frequency of 13 Hz, meaning a new sample is obtained approximately every 0.08 seconds, deltaTime is typically set to 0.08 seconds for simplicity. This value can also be calculated precisely by dividing 1 by the measurement frequency ($1 / 13 \approx 0.0769$).

Similarly, to determine the position of the sensor, the velocity data is integrated over time using a similar equation: position = position0 + velocity * deltaTime. Here, position represents the current position, position0 represents the previous position, velocity represents the integrated velocity from the previous step, and deltaTime remains the same as before.

2.8 Orientation

In Picture 3, the Movesense sensor is depicted, showcasing the orientation of its axes. The coordinates indicate the direction of each axis, elucidating the rotations occurring along the pitch, roll, and yaw axes within the Movesense body frame. All three axes are each represented by a different color according to the right-hand rule convention for describing rotations:

- The x-axis is represented by red.
- The y-axis is represented by green.
- The z-axis is represented by blue.

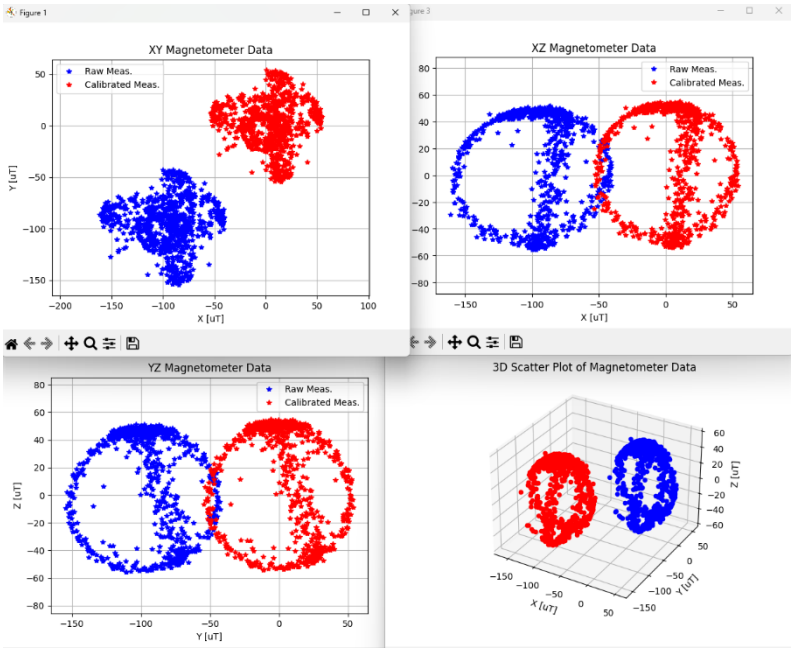


Picture 3. Visualization of Movesense sensor axes.

2.9 Calibration

Calibration involves the adjustment of sensor outputs to match known reference values, minimizing errors and discrepancies that may arise due to manufacturing imperfections, environmental factors, or sensor drift over time.

Accurate calibration is paramount in IMUs because even minor inaccuracies can lead to significant deviations in orientation, velocity, and position calculations. When precision is paramount, improperly calibrated sensors can result in erroneous navigation or unreliable motion tracking.



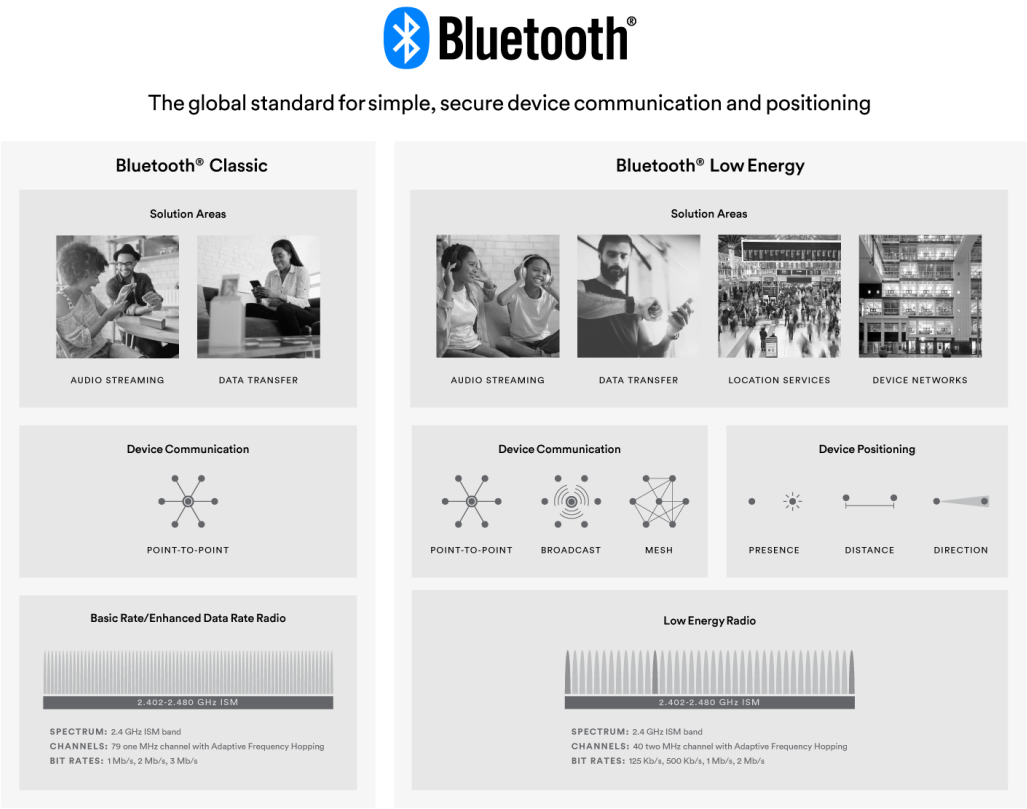
Picture 4. Effect of soft- and hard iron calibration. Picture taken in Magneto12 software [8]

In Picture 4, a comparison between raw magnetometer data and calibrated magnetometer data is presented, highlighting the efficacy of calibration techniques in enhancing data accuracy and reliability. The raw magnetometer data represents the unprocessed magnetic field measurements captured by the sensor. In its raw form, the data may exhibit distortions and biases caused by external magnetic interference, sensor imperfections, or environmental factors. As a result, the raw data often appears elongated or distorted, making accurate orientation estimation challenging.

2.10 Bluetooth Low Energy

Bluetooth Low Energy (LE) technology operates on a remarkably low power level, ideal for conserving energy. Operating across 40 channels within the 2.4GHz unlicensed ISM frequency band, it offers developers extensive flexibility in crafting products tailored to their specific connectivity needs. Picture 5 showcases the versatility of Bluetooth LE, supporting diverse communication topologies such as point-to-point, broadcast, and mesh networks. This empowers the establishment of resilient and scalable device networks. While initially renowned for its proficiency in device communication, Bluetooth LE has evolved into a dependable solution for delivering high-

precision indoor positioning services. Recent enhancements enable devices to discern the presence, distance, and direction of others, further expanding the technology's utility beyond simple connectivity. [9]



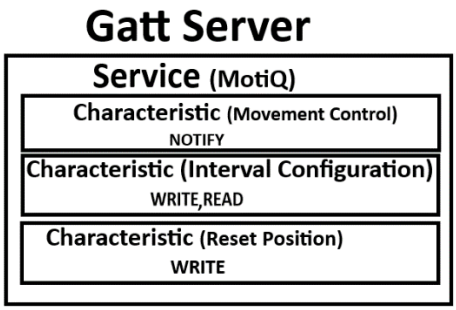
Picture 5. Showing the differences between Bluetooth Classic and -Low Energy [9]

2.11 GATT

The Generic Attribute Profile (GATT) defines how BLE devices exchange data by employing two key concepts: Services and Characteristics. Services represent a collection of related functionalities offered by a device, while Characteristics describe specific data points within a service. [10]

For instance, consider a BLE device designed for motion tracking. This device provides a custom service as depicted in Picture 6, that encompasses three distinct characteristics related to Bluetooth communication. The first characteristic, named 'Movement Control,' operates as a NOTIFY characteristic, enabling the device to toggle data transmission to the application. The second characteristic, 'Interval Configuration,' supports WRITE and READ operations, allowing the user

to adjust the frequency of data transmission from the device to the application. Lastly, the third characteristic, 'Position Reset,' facilitates WRITE operations to reset the sensor's pose, effectively setting its orientation and position to 0,0,0.



Picture 6. Structure of GATT Server with Service Characteristics.

2.11.1 Services and characteristics

Services in BLE technology organize data into manageable units, called characteristics. Services can consist of multiple characteristics and are identified by a unique numeric ID known as a UUID, which can be either 16-bit or 128-bit. Officially adopted BLE services, such as the Heart Rate Service, have standardized UUIDs and may contain multiple characteristics, each serving a specific purpose, such as Heart Rate Measurement. [10]

Characteristics represent individual data points and are identified by predefined UUIDs. Similar to Services, Characteristics play a vital role in distinguishing data points and can be standardized by the Bluetooth Special Interest Group (SIG) or customized for specific applications. For instance, the Heart Rate Measurement characteristic is essential for the Heart Rate Service and utilizes a specific UUID. Characteristics facilitate interaction with BLE peripherals, enabling data transmission both to and from the peripheral. They are crucial for understanding and implementing communication protocols, such as creating a UART-type interface with custom characteristics for transmitting and receiving data. [10]

3 Movesense sensor as the measurement unit

Having gone through the fundamental concepts of motion tracking and the required sensors, it becomes crucial to explore the backbone of the data collected and presented within the application. The focus shifts towards comprehending the pivotal sensor driving this process.

First, an exploration is undertaken into why the Movesense HR2 sensor was chosen, with the reasons behind this decision discussed. Then, instruction is provided on how to build and install firmware on the sensor, setting the stage for a better understanding of its functionalities and operations.

3.1 Selection criteria for the sensor

The Movesense HR2 sensor, developed and manufactured in Finland, provides developers with an accessible and versatile platform for innovation. With its well-documented API and ease of programming in C++, supported by ample examples, it facilitates rapid prototyping and creative exploration. Despite its advanced features, the sensor remains affordable, priced at 269€ for the developer kit (excluding VAT). Picture 7 showcases the Movesense HR2 sensor next to a 2 euro coin and a Maxell CR2025 coin cell battery, emphasizes its compact size, further enhancing its appeal for various applications.

Picture 7. Size comparison between Movesense sensor, two Euro coin and CR2025 battery

3.1.1 Microcontroller unit

The Movesense sensor boasts powerful hardware, featuring the Nordic Semiconductor nRF52832 chipset. This advanced chipset houses a 32-bit ARM® Cortex®-M4 processor, providing high-performance computing capabilities. With 64kB of on-chip RAM and 512kB of on-chip FLASH memory, the sensor offers ample storage and memory for running complex applications and storing data. Additionally, the chipset includes an integrated BLE radio operating at 2.4GHz, enabling seamless wireless connectivity with other devices.

3.1.2 Accelerometer & Gyroscope

Movesense incorporates an always-on 3-axis accelerometer and 3-axis gyroscope, offering unparalleled performance and versatility for a wide range of applications.

The accelerometer features an impressive range of ±2/±4/±8/±16 g full scale, catering to diverse needs with high precision. It supports flexible sampling frequencies ranging from 12.5Hz to 1666Hz, enabling optimal data capture in various scenarios.

Similarly, the gyroscope delivers exceptional performance with a range of ±125/±245/±500/±1000/±2000 degrees per second full scale. Alongside sampling frequencies ranging from 12.5Hz to 1666Hz, it provides precise measurements for accurate motion tracking and orientation sensing.

3.1.3 Magnetometer

Movesense also incorporates a digital output magnetic sensor, known for its ultralow-power consumption and high-performance capabilities, serving as a 3-axis magnetometer. It offers precise measurements spanning a range of ±4/±8/±12/±16 gauss full scale, ensuring accuracy in detecting magnetic fields.

3.2 Firmware requirements

Docker, CMake and Ninja are essential tools required for this project. Docker provides a containerized environment, ensuring consistency across different systems and simplifying the setup process. Meanwhile, CMake is a versatile build system generator that streamlines the compilation and configuration of software projects.

In this project, Docker is utilized to run a container containing the Movesense build environment, simplifying setup and enabling efficient management of dependencies and configurations. Within the Docker environment, CMake generates the necessary 'build.ninja' files, laying the foundation for the subsequent build process. Once the build files are generated, Ninja command ‘ninja && ninja dfupkgs’ can be run to create the Movesense_dfu.zip packet, enabling seamless firmware flashing to the sensor.

3.3 Firmware building

To build sensor firmware, pull the docker container.

```
docker pull movesense/sensor-build-env:2.2
```

To run the docker container:

```
docker run -it --rm -v c:/My/Project/Folder/movesense-device-lib:/movesense:de
legated movesense/sensor-build-env:2.2
```

Navigate to Movesense folder with (in docker):

```
cd /movesense

mkdir build

cd build
```

Create ninja files:

```
cmake -G Ninja -DMOVESENSE_TOOLCHAIN_FILE=../movesense-device-lib/MovesenseCor
eLib/toolchain/gcc-nrf52.cmake -DCMAKE_CORE_FILE=../movesense-device-lib/Moves
enseCoreLib/ ../ -DCMAKE_BUILD_TYPE=Debug ../
```

Now you can create DFU and hex files that can be used on to make the sensor work:

```
ninja && ninja dfupkgs
```

Or download the preconfigured files from the repository.

3.4 Firmware deploying

To initiate the firmware update process, begin by accessing the build folder on your computer. Then, transfer the Movesense_dfu.zip file to your mobile device.

Next, open the Movesense sample application on your mobile device, which you can obtain from the following link. [11]

Navigate to the Device Firmware Update (DFU) section within the application. Locate and select the Movesense_dfu.zip file that you transferred to your device earlier.

Choose the Movesense sensor for which the firmware deployment is intended. Confirm your selection by pressing the 'Proceed' button, and then affirmatively respond to the subsequent confirmation prompt.

By following these steps, you can seamlessly update the firmware on your Movesense sensor using the Movesense sample application on your mobile device.

4 Qt-based Mobile App

As previously noted, a key aspect of this thesis is the development of an application using the Qt application framework. In this section, the intricacies of the application user interface (UI) and Bluetooth pairing will be explored. This involves the process of establishing a connection between devices using Bluetooth technology, and exploring how the functionality is implemented within the application.

4.1 User interface

The user interface of the application is designed to be simple yet visually appealing. Picture 9 showcases the main UI, displaying information such as the Movesense model, orientation, and sensor position. Additionally, Picture 9 also features a button to initiate the search for BLE devices, providing users with convenient access to device discovery functionality.

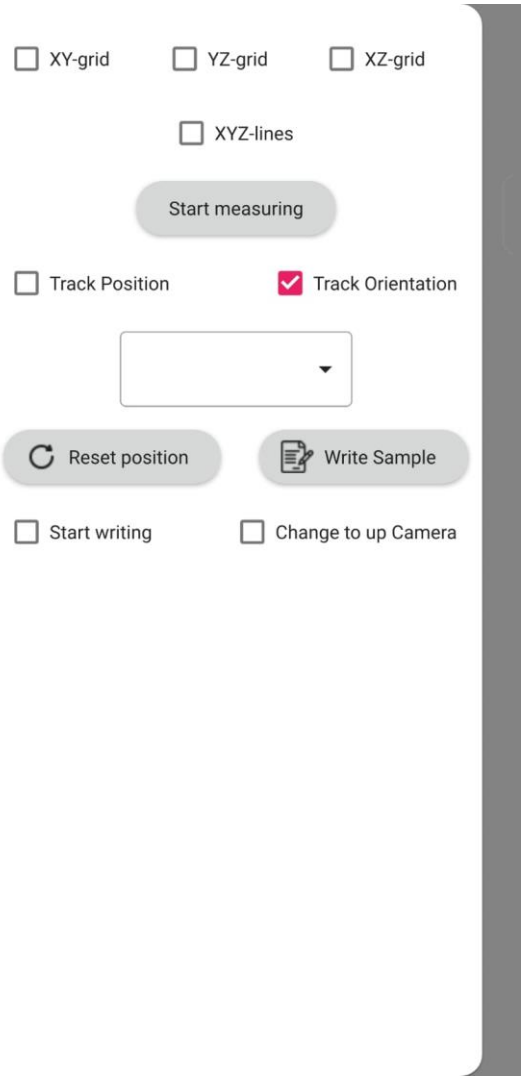
In Picture 10, there is the drawer feature, accessible by sliding a finger from left to right. Inside the drawer, users have access to additional functionalities. These include options to write samples for calibration, initiate writing sensor data to a file, switch camera views between front and upward perspectives, and adjust the sample rate via a combo box.

Furthermore, the drawer houses two checkboxes for enabling or disabling orientation and position tracking. These features provide users with greater control over the application's behavior.

The application consists of both C++ and QML components, each serving distinct roles in its architecture. In the backend, C++ is responsible for tasks such as data processing and interfacing with external devices and APIs. Here, classes and methods are designed to deliver robust functionality, effectively encapsulating complex operations.

On the frontend, QML takes charge of defining the visual elements of the user interface, encompassing buttons, text fields, and graphics. QML files are used to describe the structure and behavior of UI elements, making it easy to create dynamically responsive interfaces.

Picture 9. Main UI of the Qt app.



Picture 10. Showcasing the content of the drawer.

4.2 Bluetooth pairing

The connection will be established via Bluetooth, which is activated by tapping the sensors HR connectors labeled L and R, as depicted in Picture 8. When the user initiates the device search process on the application by pressing the "Search" button, the application will begin scanning for Movesense-named devices using Qt's Bluetooth module. It will then automatically connect to the first device detected and automatically start displaying the data coming from the sensor.

Picture 11. Movesense HR connectors L and R.

5 Result

The completion of the application and sensor software marks the culmination of the project's endeavors. While the outcome may differ from the original goal, this journey has been a valuable learning experience, imparting crucial lessons in efficiency and adaptability to the author.

Throughout the project, numerous use cases for inertial measurement data were explored, highlighting the inherent value in experimentation and exploration. Despite encountering challenges along the way, each obstacle served as a learning opportunity, contributing to the author's growth and development.

The thesis involved the development of both a Qt application and sensor firmware. The application was designed for the client, while the firmware focused on data gathering and connecting to the application. Comprehensive comments were included in both the application and firmware files to aid in future development efforts. The work meets the specified criteria, and the client expressed satisfaction with the results.

The project not only provided an opportunity to gain insights into the Qt framework and its proper utilization, but also served as a platform to explore various Qt modules extensively. In addition to delving into sensor development, which introduced new and exciting challenges such as flashing software over Bluetooth, the project involved utilizing key Qt modules such as QtBase, QtQuick, QtQuick3D, QtQuickControls, and QtBluetooth.

The utilization of these Qt modules played a crucial role in enhancing the application's functionality and user experience. For instance, QtQuick and QtQuick.Controls were instrumental in designing responsive and scalable UI layouts, thereby ensuring optimal usability across different devices, screen sizes and device orientations. QtQuick3D, on the other hand, facilitated the rendering of a 3D view to display the sensor model and enabled dynamic manipulation of its position and rotation within a 3D space.

Furthermore, the integration of the QtBluetooth module enabled seamless communication between the application and the sensor device, thereby facilitating data exchange and interaction. Additionally, the project involved gaining valuable knowledge in creating a QAbstractListModel in Qt and dynamically adding objects to a 3D view using QtQuick3D during runtime, further expanding the author's skill set and understanding of Qt development principles.

Lastly, the process of calibrating IMU sensors added another layer of understanding to the author's skill set, underscoring the interdisciplinary nature of the project and its contribution to professional growth.

6 Conclusions

Position tracking with an IMU presents a plausible solution, albeit with challenges in ensuring data accuracy and precision. Simple integration of accelerometer data alone often results in significant drift, even with correct calibration. Conversely, orientation tracking utilizing IMU concurrently offers a more reasonable and reliable approach, yielding more accurate results.

Visualization of tracked data can be achieved by plotting sensor data points in a 3D chart view or by animating or moving an object within the application's 3D view. User experience considerations should be paramount in the application planning process.

To enhance position calculation accuracy, the incorporation of a high-pass filter into the firmware post-velocity and position integration is proposed. This addition aims to mitigate drift and improve data precision. Additionally, a toggle switch will be introduced to enable switching between real-time data measurement and movement-based measurement display, providing users with flexibility in data presentation.

Furthermore, the inclusion of a rotation matrix in the system aims to align acceleration data relative to the rotation of Earth, thereby eliminating gravitational force interference and improving overall accuracy. These enhancements contribute to refining the performance and usability of the position tracking system.

The following tools have been used in the work:

ChatGPT. (2023). OpenAI. GPT-3.5, Used to check and improve grammar and text, 02/2024. https://chat.openai.com

Sources

1 The Qt Company. About Qt [Internet]. [cited 22.12.2023]. Available from: https://wiki.qt.io/About_Qt

2 Moment of inertia. Britannica [Internet]. [cited 06.02.2024]. Available from: https://www.britannica.com/science/moment-of-inertia

3 Inkinen T. Momentti 1 Insinöörifysiikka. Helsinki: Otava; 2024. p. 198. Picture 8.7.

4 MEMS Exchange. What is MEMS? [Internet]. [cited 06.02.2024]. Available from: https://www.mems-exchange.org/MEMS/what-is.html

5 VectorNav Technologies. Inertial Navigation Primer: Theory of AHRS. [Internet]. [cited 06.02.2024]. Available from: https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-ahrs

6 Lambrecht S, Nogueira SL, Bortole M, Siqueira AAG, Terra MH, Rocon E, Pons JL. Inertial Sensor Error Reduction through Calibration and Sensor Fusion. Sensors. [Internet]. 2016;16(2):235. [cited 25.1.2024] Available from: https://doi.org/10.3390/s16020235

7 Biswaindu P. Sensor Fusion: The Ultimate Guide to Combining Data for Enhanced Perception and Decision-Making. 17.05.2023 [cited 25.1.2024] Available from: https://www.wevolver.com/article/what-is-sensor-fusion-everything-you-need-to-know

8 Merlin B. 27.10.2021 [cited 13.02.2024] Available from: https://sites.google.com/view/sailboatinstruments1

9 Bluetooth® Wireless Technology. Bluetooth [Internet]. [cited 12.01.2024] Available from: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

10 Kevin T. Adafruit Industries. Introduction to Bluetooth Low Energy. Connected Network Topology. [Internet]. [cited 13.02.2024]. Available from: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt

11 Bitbucket. Downloads. [Internet] [cited 16.02.2023]. Available from: https://bitbucket.org/movesense/movesense-mobile-lib/downloads/