Luleå University of Technology

Dept. of Computer Science, Electrical and Space Engineering

D7039E – Project in Industrial Computer Systems

Project SailorAid

Authors
Axelsson Oskar,
Brolin, Daniel,
Eriksson, Kenny
Lundberg, Josef,
Grape, Elias
Sjölund, Johannes
Theolin, Henrik

Supervisor
van Deventer, Jan

November 9, 2017

Abstract

Some abstract description of the project... Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

testing...

Contents

1 Introduction

WHO??? (NO ONE)

The art of sailing has been around for millenia. For much of human history it has been an absolutely vital part of civilization, providing efficient means of transporting goods all around the world. Today sailing has become a leisure activity enjoyed by millions of people around the world. Modern sailboats come in a large span of sizes, from large ships with crews of dozens down to small single-man dinghies. While slipping across the waves out at sea with only the wind to drive you is a calming experience, it is not a simple thing to do. When you are alone on the water, you have to be in control of the tension of the sail, the attitude of the boat, the forces on the centerboard and more while deciding how to respond to all of these. The goal of Project SailorAid is to offload the decision-making from the sailor onto a compact, portable and simple system that will analyze these parameters and provide clear directions to the sailor.

1.1 Goals

The primary functional goals are as follows:

. Boat attitude

- Implementing an appropriate sensor array:
  * Accelerometer
  * Gyroscope
  * Magnetometer
- Fusing the sensor output to get an accurate estimate of boat attitude

. Position tracking and velocity

- Implementing a GPS system
- Fusing the GPS output with the accelerometer output for more accurate positioning and velocity

. Design a force measurement circuit for the centerboard

- Design an appropriate sensor mount off the centerboard
- Implement an appropriate sensor
- Implement a centerboard-depth sensor

2 The Physics of Sailing

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium

quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

3 Product Application

4 Hardware Design

5 Sensors

Lundberg, Josef (no one)

In order to sail properly and make the most out of the wind that's is supplied by the nature itself some data acquisition is needed. The sailing is all about this harnessing all the forces of the nature and the wind that it pushing towards you. Since there has not been any other extensive projects and measurements in this particular area the measurements have to be done in new ways.

5.1 Force sensors

The goal here is to have a system that can measure the forces that pushes on the centerboard by the water it goes through. The implementation: By looking at some different solutions there is not any other solutions that might be as clean looking and prominent as this approach. Important to know is that every solution is mandatory to be waterproof and sealed properly from the harsh environment that this system has as its home turf. The solutions that required the sensors to be mounted on the outside or in parts that would be in danger if a crash might occur was scratched. The board itself will not be disassembled in any major part of way. Meaning that this approach doesn't need any modifications to the board itself. This has been the goal and the choosen approach. Modifications in the mounting plate is the way to go, the other solutions is either way more difficult to apply and mount or more complex.

5.2 The prototype

To implement the gauges, a prototype is designed to show how the measurements will be made. The prototype is a bit bigger than the intended solution for this project but it's good to see how it would be constructed. The function is easy to understand. The board goes on the outside and can easily slide up and down past this ball. The ball itself is kept inside this small area where it can move in and out.

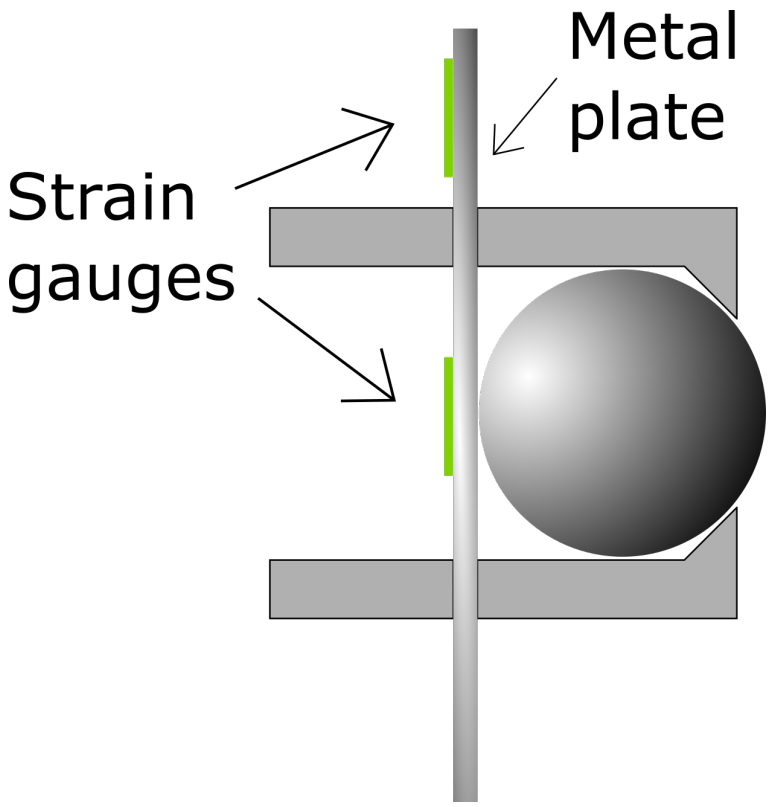

Figure 1: Function of first prototype

This way of implementing strain gauges was the first idea. The main case for this strategy was that in the start of this project these gauges were supplied to us, as a leftover from the last group. With this implementation, we could already start working on a prototype and get a small head start in to the project. But as some research shows, it is a more difficult way to solve this problem and it would take bit more work and some sensitive circuits to measure the force. The gauges also need to be stuck in place using some specific glue and can easily be done incorrectly and therefore prevent good measurements.

A model of the pressure sensor was constructed in the CAD program fusion 3D. This model was created in order to clearly show the function of this sensor and to help the thought process involved in the improving of this design.

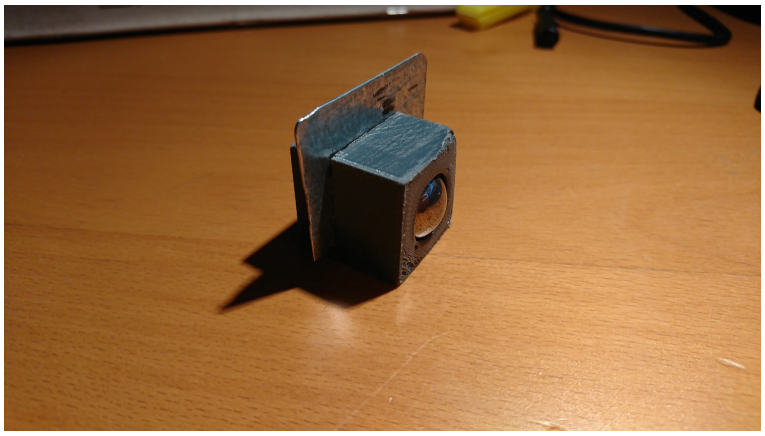

Figure 2: Function of first prototype

The force is then measured at the back where there will be a plate. The deflection of this plate which will be the origin to the strain will be measured through strain gauges. The gauge itself will measure a small difference in resistance. This small difference is going to be difficult to measure without any amplifying circuit connected. With a such small signal the system might have issues with noise. Another problem is the signal might drift, and therefore make different measurements as the circuit is running. And finally, with the measured values getting amplified with a big amount the resulting signal may be off by a large amount.

New idea: A better solution is to make some research into load cells, which is a sensor which also utilizes strain gauges to measuring forces. The difference is that the gauges are already implemented in the sensor. The difference in the prototype is instead of having a metal plate, it can be built with a piece of plastic or rubber which can deform so the force is distributed directly to the sensor. By implementing this sensor, a lot of time was saved in troubleshooting. And by having a sensor unit, the modified mounting plate will be easier to produce.

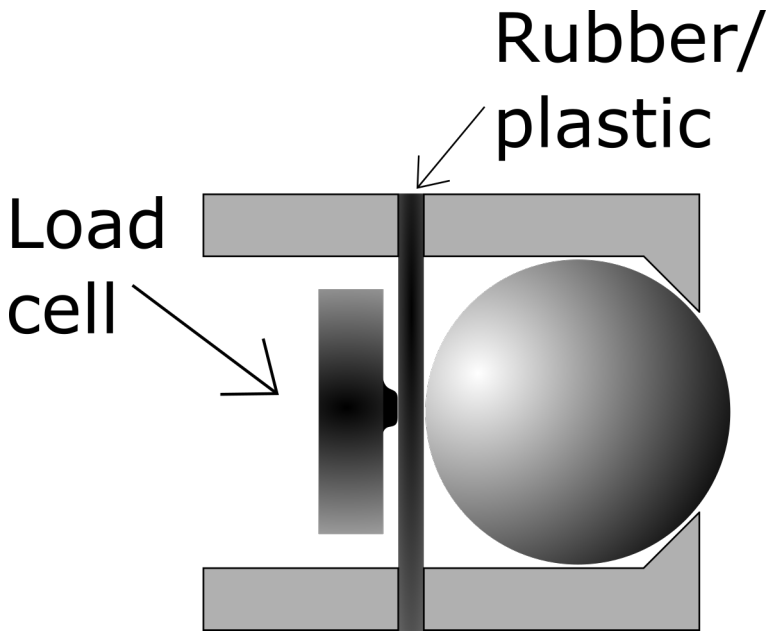

Figure 3: Function of second prototype

5.3 Choice of component

The force from the board onto the mounting plate will be a considerable amount. The actual force is something that's not known for sure. The initial assumption was that a load cell with a 90.75 kg force range should be enough. In the case the sensor will be maxed out the cell it's rated for a 150% overload without causing some damage to the sensor. The choosen sensor for this application is selected to this part, the compression load cell called FX1901. From the datasheet the voltage readings of this piece could be calculated. With a maximum voltage reading of around $36mV/V$

Figure 4: Load cell, FX1901

5.4 Amplifier

In order for the microcontroller to make some good measurements from the load cell an amplification for the That's a small signal and needs to be amplified to get some good measurements. A good measurement signal to the MCU should be in the order of in between 0 - 5 volts. This is achieved by an amplification gain of around 20.

A suitable amplifier needs to be choosen from the vast ocean of different models. Inspiration is taken from The university of Chicago**??** in an experiment where they uses this exact load cell togheter with an instrumental amplifier called INA125. This amplifier is somewhat more complicated and have some more feautures that other amplifiers. In the same family of instrument amplifiers a model called INA126 is selectet as a less complicated and more power efficiant solution.

The choice fell on this little fellow, the INA126.

Figure 5: Amplifier for the load cell signal

Which not look so interesting but has the benefit of having a smaller power consumption then many others by being a bit simpler than many others. But sufficient for our purpose.

It seems like a smart choice because when the system is battery operated, like in this case, every watt counts. The gain on this piece is easily calculated with this function. Gain is 5 plus 80k ohm divided by our chosen resistor Rg.

If our desired 20 gain might not work or the voltages calculated is off, the gain is easily redone with this expression.

$$Gain = 5 + \frac{80k\Omega}{R_G} \tag{1}$$

Now that we know how to get the force measurements, we are going to talk about how to measure the frequency of waves at sea.

5.5 Height of daggerboard sensor

The main issues might be that the Height of daggerboard: One of the best implementations of a height sensor would be the use of a linear wire distance sensor. This particular sensor measures how far a wire is pulled, which gives a very accurate measurement. This solution can be totally watertight and concealed in the main centerplate.

Figure 6: Linear draw wire sensor, Micro epsilon MK30

We have found some sensors that might work for us, this is the smallest we found. It's 3 cm wide and about 5 cm high. As we have some tight space constraints this can probably fit inside or just stick out a little bit. This particular sensor is in the range of 2000 kr, which feel like a lot. But if no other solution works this might be considered again.

We have also looked into some light sensors. This is implemented with the use of a plate placed ion top of the dagger-board and with the light being sent up to this panel, the height can be calculated. First we looked into some ir-sensors. They will probably send the signal in a wide spread which will make the distance measurement troublesome as this signal has just a small plate to bounce of.

But with the use of a LIDAR system, we can point our light signal at an exact spot and then get an exact measurement of the height. Many of the lidar systems we found was too big for our project and also very expensive. A suitable sensor that we found was a small "micro lidar" circuit from adafruit:

5.5.1 Component

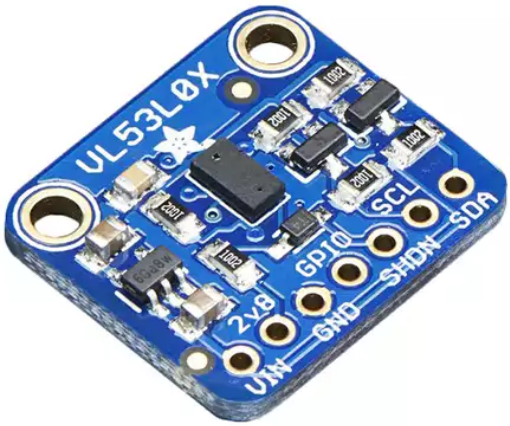

Figure 7: Time of flight distance sensor, Adafruit VL53L0X

Problems:

As there will always be water around and on the centerplate the light that is sent might get directed "wrong" if there is only some small water droplets in between the sensor and the panel that we want to reflect our light on.

By the fact that the senor have to be water proof the signal has to go through a medium. The medium can be some type of plastic or even glass. If the signal can be read correctly through this medium or if the signal will get corrupted, we don't know yet. That is something that we need to test later when our parts has come in.

6 Software Design

Sjölund, Johannes (no one)

The software has been divided into two parts, the firmware for the ARM MCU with associated sensors, and an Android application which can display sensor data. These two parts utilize a Bluetooth connection to communicate their current states. For example, when the IMU calculates a new orientation, this data should be processed by the firmware, and the resulting calculations sent to the Android application over Bluetooth to be displayed to the user.

6.1 ARM firmware

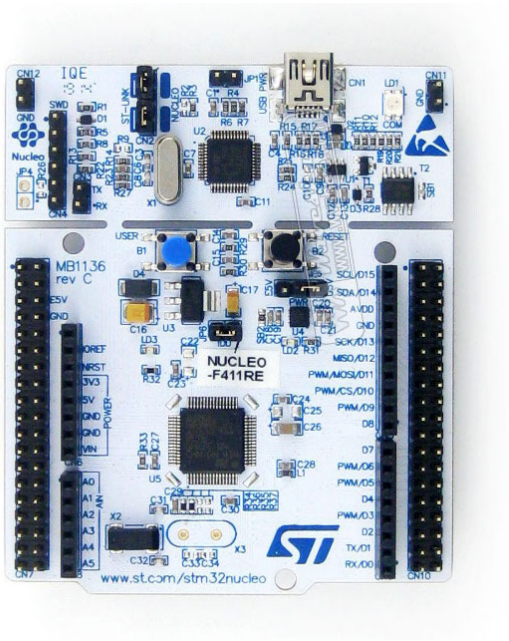

Figure 8: NUCLEO-F411RE development board for the ARM Cortex-M microcontroller STM32F411RE.

For rapid prototyping and firmware development purposes, the NUCLEO-F411RE development board seen in figure 8 was used. This board contains break-out pins for the ARM Cortex-M microcontroller STM32F411RE, a UART to USB bridging circuit and general purpose LEDs and buttons. It is compatible with various Arudino shields as well as expansion boards developed by ST.

In order to speed up firmware development, the STM32CubeMX[1] initialization code generator was used to set up a basic working system. This application, developed by ST, can generate C language code for setting up MCU clocks, peripherals, interrupts and similar. It is controlled by a graphical interface for setting MCU options and controlling the previously mentioned code generation.

The main challenge in working with this type of code generation is integrating it with external software libraries directly not built for it. If the library interferes with generated code by overriding settings and register values, the

software may enter an undefined state and stop working. Care therefor had to be taken to only use the parts of the libraries which did not interfere. Frequent testing of any newly added functionality had to be done in order to find interfering parts.

Three libraries produced by ST were used, one for the Bluetooth module, the IMU chips and the range sensor.

6.1.1 Bluetooth

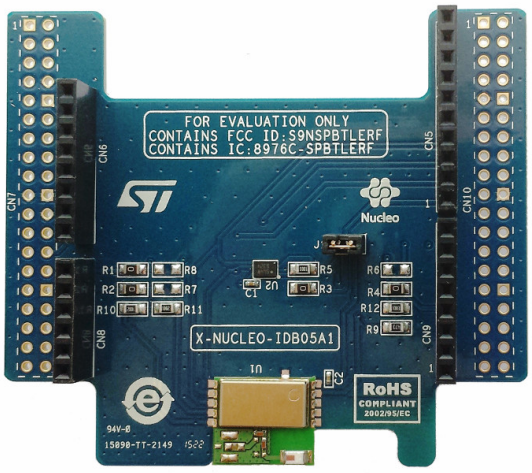

Figure 9: X-NUCLEO-IDB05A1 Bluetooth Low Energy evaluation board for the STM32 Nucleo

For prototyping, the Bluetooth evaluation board X-NUCLEO-IDB05A1[5] seen in figure 9 was used, which could be stacked on top of the Nucleo board. The pins on the evaluation board connected it to an SPI port on the MCU.

To avoid having to implement the Bluetooth stack from scratch, the firmware package called X-CUBE-BLE1[2] developed by ST was used. It consisted of several parts – MCU and Bluetooth evaluation board device definitions such as named pins and ports, functions for manipulating them, a Bluetooth GATT server implementation, as well as several demo applications showing usage examples. Additionally an Android demo application for displaying sensor data from Bluetooth was available from the Google Play platform, called BlueNRG[4]. The library code was integrated into the code generated by STM32CubeMX, added as an external library and statically linked.

While ST included example code for communicating with the Bluetooth module over SPI through interrupt based DMA transfer, this code was quite difficult to get working. Instead it was decided that blocking SPI communication were to be used, since this was much simpler to get working. The reasoning was that since the module supported a baud rate of up to 10 Mbit/s, this would be fast enough to cause miminal interference with other parts of the firmware code.

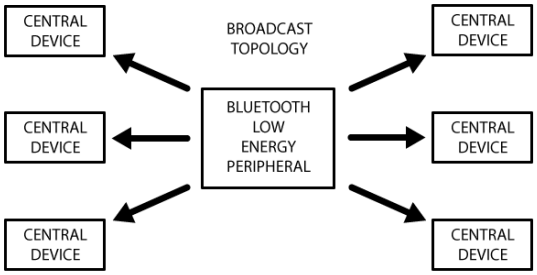

Figure 10: Bluetooth GAP topology.

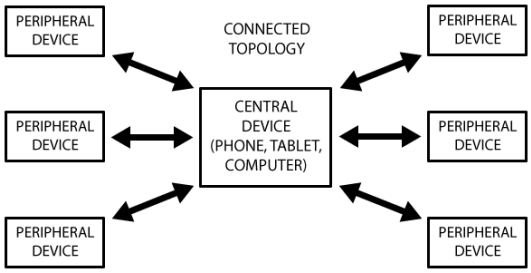

Figure 11: Bluetooth GATT topology.

As mentioned previously, the library implemented the Bluetooth GATT protocol. This protocol supports bidirectional communication from a single central device, in this case an Android cell phone, to several peripheral devices, such as the embedded system in this project. The library also supported the Bluetooth GAP protocol, which is a unidirectional communication protocol allowing one peripheral device to broadcast to multiple central devices. Figures 11 and 10 illustrates the topological differences between these protocols.

For this project, the GATT protocol was chosen. The reasoning was that enabling the Android app to send commands to the embedded system could be useful for controlling functionality. This meant that only a single phone could be connected to the system at any time, as opposed to the GAP protocol, which would allow multiple phones to listen to the Bluetooth broadcasts. Since the embedded system is designed to be used on a small dinghy with space for a maximum of two people, this seemed like a reasonable trade-off. If the system was to be used on a larger sail boat, the GAP protocol might be more useful, since it would allow multiple passengers to listen to broadcasted sensor data.

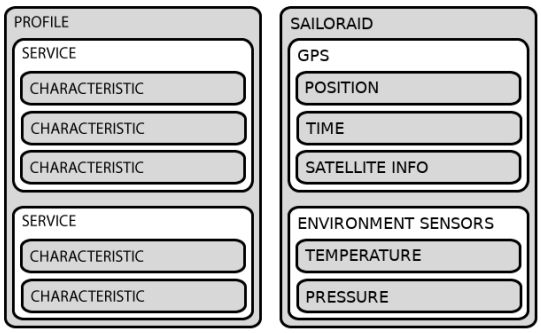

Figure 12: Bluetooth GATT transaction profile with usage example.

The GATT protocol performs transactions by nested structures called Profiles, Services and Characteristics. An example of this structure can be seen in figure 12. These structures were already implemented in the X-CUBE-BLE1 and updated by simple function calls. When new sensor data was received from e.g. the GPS or IMU devices, these functions were called at regular intervals which pushed the data to the Android app. Each profile were given a unique identifier which allowed the app to recognize which type of data was received.

6.1.2 IMU

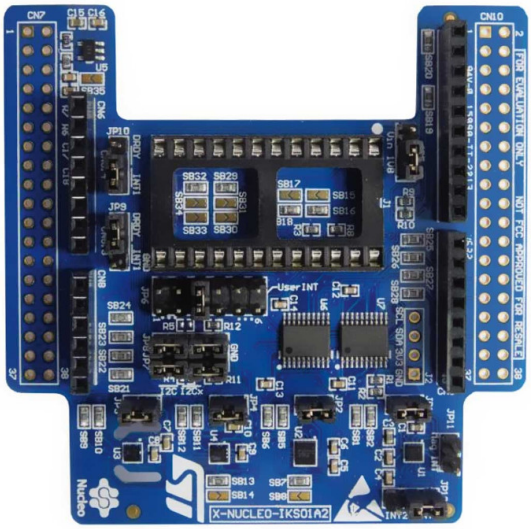

Figure 13: X-NUCLEO-IKS01A2 motion MEMS and environmental sensor expansion board for the STM32 Nucleo

In order to measure the various time dependent spatial features such as orientation, acceleration and velocity, an IMU device was used. More specifically, the X-NUCLEO-IKS01A2[6] evaluation board (figure 13) developed by ST was chosen for rapid prototyping purposes. This board included the LSM6DSL 3D accelerometer/gyroscope, the LSM303AGR 3D accelerometer/magnetometer,

the HTS221 humidity and temperature sensor as well as the LPS22HB pressure sensor.

To interface the firmware with the board, the X-CUBE-MEMS1 [3] library developed by ST was used. This library implemented the I^2C communication protocol used by the previously mentioned IMU devices in the form of simple function calls, which saved a lot of development time. It was quite simple to integrate with the code generation from STM32CubeMX, only a few source definitions had to be modified. Like with the Bluetooth library (section 6.1.1) blocking communication was chosen to simplify the code, even though the MCU supported interrupt based DMA transfers. The I^2C operated in fast mode at 400 kHz which was thought to cause minimal interference with the rest of the system in blocking transfer mode.

An important use case for the IMU was to determine the current orientation of the dinghy. To accomplish this, a type of sensor fusion algorithm called Madgwick AHRS (section 6.1.3) was used.

6.1.3 Madgwick AHRS

Madgwick AHRS [7] is a type of sensor fusion algorithm which calculates the current orientation in space from three dimensional vectors of acceleration, angular velocity and magnetic field strength. It was developed in 2010 by Sebastian O.H. Madgwick as a more performant alternative to the Kalman filter approach. It basically integrates the angular velocity from the gyroscope, while using the accelerometer and magnetometer to create a reference to the horizontal plane. Earth's magnetic poles provides a horizontal vector which lies on the plane, while the gravitational acceleration is the plane's normal vector. This is then used by the algorithm to compensate for drift in gyro integration. The algorithm stores orientation in quaternions (rotation vectors with four elements), but can convert it to Euler angles, which can be more easily used.

The mathematical background of this algorithm is quite complicated and outside the scope of this report, see the official report [8] for more details.

6.1.4 USB UART

Since this project involved analyzing sensor data for developing sensor fusion algorithms, for example combining GPS and accelerometer for accurate positioning (section 7) and measuring water wave properties, it was important to be able to log data at a reasonably high frequency. Transferring serial commands between a computer and the MCU also helped in debugging the code. To this end, a hardware UART-over-USB chip was used, the ST-LINK/V2-1 on the Nucleo board, and FTDI FT232R on the custom project board.

At a relatively low baud rate of 115200 bps, it was determined that send and receive should both be interrupt based using DMA transfers to minimize the impact on system resources. Reception of data like key presses from a computer was handled one character at a time. The characters were appended as a string until the enter key was detected. At this point the string was matched against a list of valid commands, and the appropriate task performed – such as sending current sensor values. Sending was implemented as a simple circular buffer which could be transferred to the UART peripheral registers using DMA.

In order to log sensor values for later analysis a simple MATLAB script was developed for listening to sensor data over the UART serial port. By inputting a serial command, the embedded system starts sending live sensor data at at constant rate. The script listens to this and logs it to a table.

6.1.5 GPS

Figure 14: Maestro A2235-H GPS module with built-in antenna

The GPS module used in this project, A2235-H by Maestro[9] could communicate with the MCU through either I^2C or UART. Both protocols require only two pins to operate, but UART communication was determined to be easier to implement in code. The UART baud rate of the GPS module was set to 4800 Hz by default. While this could be changed by software there was no reason to do so. The low baud rate did however mean that blocking transmissions might cause problematic interruptions in the firmware code. To prevent this, interrupt based communication through DMA was implemented, using the same type of queuing system as the USB UART (section 6.1.4).

Data from the GPS was formatted according to the NMEA message standard. It is used by nearly all GPS devices internally, but is quite hard for a human to read. For example,

```
GPGSA,A,3,03,22,31,23,01,06,09,11,,,,,,1.9,1.2,1.5*33
GPRMC,152053.000,A,6537.0389,N,02208.0160,E,0.17,264.54,240917,,,A*6A
GPGGA,152054.000,6537.0389,N,02208.0160,E,1,08,1.2,14.0,M,25.0,M,,0000*68
```

contains three so called sentences. GPGSA contains data about the number of active satellites and positional accuracy. GPRMC and GPGGA both encode longitude, latitude, current time and date, as well as other data.

Several NMEA parsing libraries are freely available on the web. The one chosen for this project was called Libnmea[10] and allowed the sentences to be automatically recognized, parsed and stored into easy to use C structures.

6.1.6 Range sensor

In order to communicate with the VL53L0X range finder sensor, another software library called X-CUBE-53L0A1[11] developed by ST was used. This software was written for another Nucleo expansion board called X-NUCLEO-53L0A1 which integrated three range sensor units to perform gesture recognition. It was however possible to modify the code to work with the single sensor used in this project. The library implemented a leaky integrator algorithm (basically a low-pass filter) to reduce measurement noise and improve accuracy.

Since the sensor measures the time-of-flight of a laser beam, it was obvious that having the firmware code block while waiting for measurement data was not a realistic option. Instead, the code repeatedly alternated between instructing the sensor to initiate a measurement, and reading back the data from the last measurement. Since the module was meant only to measure the height of the dinghy center board a low measurement frequency of 1 Hz was used.

6.2 Android application

THEOLIN, HENRIK (NO ONE)

The main reason for this project is to give a sailor qualitative feedback and help in clearing the mind of the techniques required to achieve a smooth sailing experience so that the sailor can focus on the joy of sailing. It is therefore crucial that the data is displayed in a manner that is easy to interpret and provide the help that is called for. To increase the flexibility of the design it was decided to implement several different user interfaces that the user could switch between while running the application. This was determined to be a good way of increasing the chances that the user would find a interface to it's liking. It was also determined that not only visual representaion would be the best way to go since the sailor needs to be in constant motion to counteract the forces applied on the ship by the wind and current. This would make watching a screen to retreive information somewhat difficult. Other ways of representing data was implemented using text-to-speech where the sailor would get important information by sound aswell as text. Vibration was also used with different vibration sequences depending on diffenrent sates of the ship.

6.2.1 Visual Feedback

It was determined that the visual feedback given to the sailor was to include very little text information and would consist mainly of images that changes position based on sensor data to give a good representation of what was going on with the ship. But because of different personal preferences the ability to switch between different layouts based on what the sailor felt was most important at any given time was implemented. To switch between different layouts a simple swipe on the screen toggled the view to the next layout. All layouts consisted of a subset of views from the complete set including

15 Incline displayed a ships relative incline against and artificial horizont

16 Pressure moved a pin along a colored bar to represent high of low pressure applied on the centerboard

17 Bearing rotated a compass to show the ship relative bearing against true north

18 Map displayed current locaion of the ship

19 Drift was represented with a colored bar with two arrows that moved to the relative drift direction to show the sailor if the ship was holding it's set navigaitional reference.

Feedback a text that changed values based on the current state of the boat

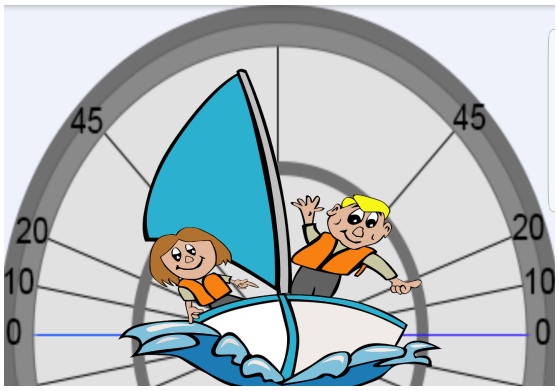

Figure 15: Incline feedback view

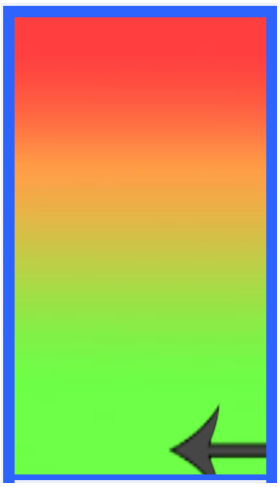

Figure 16: Pressure feedback view

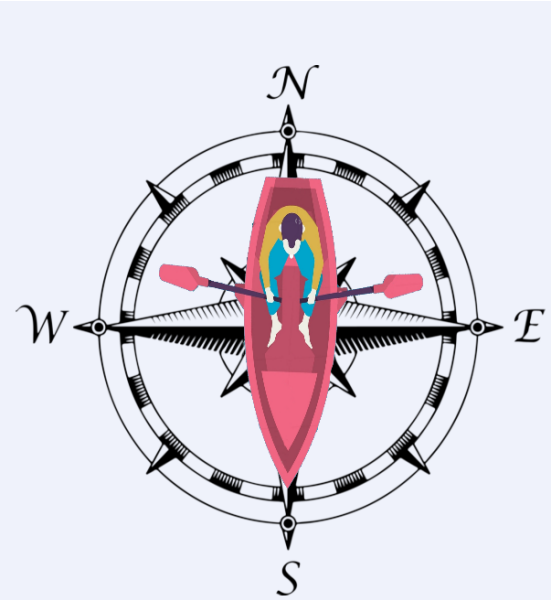

Figure 17: Bearing feedback view

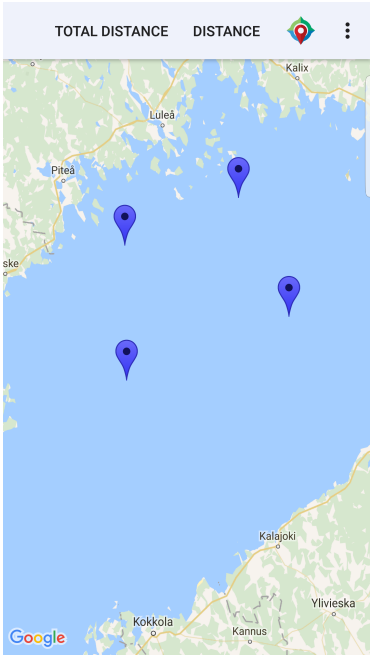

Figure 18: Map feedback view

Figure 19: Drift feedback view

6.2.2 Audio feedback

Based on values from the sensors different states were implemented and for each state a text was read out to the sailor. A priority for each state to prevent less important events from interrupting the more important ones. The feedback was implemented such that the audio feedback would continue even if the device was put in sleep mode.

6.2.3 Haptic feedback

Based on the same states as the audio feedback vibration was also implemented. The frequency and length of the vibrations were implemented in such a way the each state had a unique signature that could be recognised by the sailor after some practise with the system.

6.2.4 Log

The ability to analyse the sailtrip a log was implemented and stored on the device internal storage. These files could then be read or deleted at a later date. Storing a log was implemented such that the sailor would need to start a logging session after bluetooth connection had been established to the ship. After the log was started it would create a file on the device internal storage and write sensor data to the file at the same frequency as the data was transmitted by the mainboard. While logging was active the device would stay active even if the screen was put in sleep mode so that the sailor could choose to only log data and not view the information displayed on the screen. After the sailtrip was finished the sailor could read the saved log file and receive a summary of the trip 20. More information from the log could be analysed by reading graphs 21 where the sensor data gathered was shown over time.

Avg Incline: -0.32980242
Max Incline: 36.411453
Max Drift: 1.4229604
Total Drift: 103.87611
Avg SOG: 3.5639582
Top SOG: 4.92632
Max pressure: 1032.4
Avg Pressure: 1032.2876

Figure 20: Log data summary

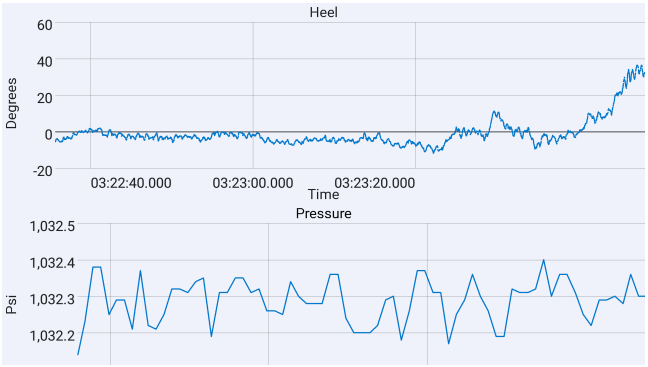

Figure 21: Log graphs

6.2.5 Map

For the sailor to get more information about location a map was implemented. The sailor then had the ability to see a path over the trip and the total distance traveled. Waypoint could be placed if the sailor wanted to decide in advance a certain trip. The distance of all the waypoints was displayed and the distance currently traveled to give the sailor information on how long the trip was and how much of the trip was left to be sailed. A path to the nearest waypoint was showed to the sailor to help navigating, when the sailor way close to the waypoint the path to the next waypoint was shown.

7 Kalman filter

Axelsson, Oskar (no one)

8 Introduction

Sensor theory

Sensor fusion can be observed everywhere e.g., living animals uses all of its senses to survive daily, an animal cannot hunt using its eyes only, it has to combine its sense of smell, eyes and hearing to hunt the pray[18]. Sensor fusion theory is not only found in the living species it is found in cars, planes, computers and so on and this to enhance performance and accuracy[18]. In this project sensor fusion will be used to enhance the accuracy of the dinghy's position and velocity. The fusion will be between a global positioning system, GPS and an inertial navigation system, INS. The INS build on a low price inertial measurement unit.
The GPS's accuracy is not uniform since it might be buildings reflections, atmospherics delays or clock bias errors[13]. Using only information provided by a INS is not sufficient either since the sensors in the IMU will drift after time, but using the information provided by the sensors in the IMU only for short time will give accurate readings.

8.1 Kalman Filter

A popular filter to use when applying sensor fusion is to use a Kalman filter, (KF). The Kalman Filter is a recursive filtering method for discrete data, the algorithm was developed by an Hungarian mathematician Rudolf (Rudi) Emil Kalman in 1960[13]. Its popular to use due to its efficiency when calculating predictions.[15]

Different Frames

Navigation algorithm involves various coordinate frames and therefore transformation between frame is a must.

The inertial frame denoted i-frame for future notation is defined such that its origin is at the center of Earth and its axes X_i, Y_i and Z_i is non-rotating with respect to the stars. Z_i is coincident with the Earth's polar axis, i.e. North.

The Earth navigation frame denoted e-frame for future notation is fixed with respect to Earth and has its origin at the center of the Earth. The frame is defined as X_e, Y_e and Z_e, with Z_e along Earth's polar axis. Axis X_e lies along the intersection of the plane of the Greenwich meridian with the Earth's equatorial plane. The e-frame rotates at a constant rate with respect to the i-frame and is denoted ω_e.

The Navigation frame denoted n-frame for future notation is a local frame and has its origin located in the navigation system, in this case, point P, see Fig. 22, and its axes aligned with the directions of north, east and down, denoted NED. The turn rate of the navigation frame with respect to Earth's fixed frame, ω^n_{en}, is directed by the motion of point P with respect to the Earth and is referred to the transport rate.

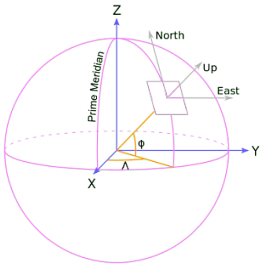

Figure 22: Earth and navigation frame.

The Body frame denoted b-frame is the sensitive axes of the IMU's sensors are made to coincide with the axes of the moving platform in which the sensors are mounted. The body, in this case, is referred to the dinghy, from Fig. HÄR SKA EN BILD IN

- Yaw (ψ): is the is the deviation of the vehicle's forward y-axis from north, measured clockwise in the E-N plane.
- Pitch (θ): This is the angle that the forward y-axis of the b-frame makes with the E-N plane (i.e. local horizontal) owing to a rotation around its transversal x-axis.
- Roll (ϕ): This is the rotation of the b-frame about its forward y-axis, so the forward axis is also called the roll axis and the roll angle i

Transformation between Earth frame and Navigation frame

Referring to Fig. 22 it can be observed that its possible to align the n-frame with the e-frame, this is done by rotating the n-frame by $(\lambda - 90)$-degrees around its x-axis (east-direction) and $(-\phi - 90)$-degrees about its $z - axis$, (downward direction)[14]. Where λ and φ is the latitude and longitude, respectively. Then the transformation between the two frames can be done using the Direction Cosine Matrix, noted DCM for future notation, which is defined as[14].

$$C_n^e = R_z(-\lambda - 90)R_x(\varphi - 90) \tag{2}$$

Where C_n^e should be interpreted as moving from n-frame to e-frame, R_x and R_z are the rotation matrices around its axis, respectively. Expanding Eq. (2)

$$C_n^e = \begin{bmatrix} cos(-\lambda - 90) & sin(-\lambda - 90) & 0 \\ -sin(-\lambda - 90) & cos(-\lambda - 90) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi - 90) & sin(\varphi - 90) \\ 0 & -sin(\varphi - 90) & con(\varphi - 90) \end{bmatrix} \tag{3}$$

$$C_n^e = \begin{bmatrix} -sin(\lambda) & -cos(\lambda) & 0 \\ cos(\lambda) & -sin(\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & sin(\varphi) & -cos(\varphi) \\ 0 & sin(\varphi) & sin(\varphi) \end{bmatrix} \tag{4}$$

$$C_n^e = \begin{bmatrix} -sin(\lambda) & -sin(\varphi)cos(\lambda) & cos(\varphi)cos(\lambda) \\ cos(\lambda) & -sin(\varphi)sin(\lambda) & cos(\varphi)sin(\lambda) \\ 0 & cos(\varphi) & sin(\varphi) \end{bmatrix} \tag{5}$$

Exploring the orthogonality its possible to transform from e-frame to n-frame by taking the inverse of the equation above, i.e.

$$(C_n^e)^{-1} = C_e^n \tag{6}$$

Using Eq. (6) its now possible to move from e-frame to n-frame.

Transformation between Body frame and Navigation-frame

Transforming from n-frame to b-frame is done in the same way, i.e. using rotation matrices. The DCM, moving from b-frame to n-frame is given by[14]

$$C_b^n = R_z(-\psi)R_y(-\theta)R_x(-\phi). \tag{7}$$

Where

$$R_z(-\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$R_y(-\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{9}$$

$$R_x(-\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{10}$$

Expanding Eq. (7) with (8), (9) and (10)

$$C_b^n = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{11}$$

$$\tag{12}$$

$$C_b^n = \begin{bmatrix} cos(\theta)cos(\psi) & -cos(\phi)sin(\psi) + sin(\phi)sin(\theta)cos(\psi) & sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi) \\ cos(\theta)sin(\psi) & cos(\phi)cos(\psi) + sin(\phi)sin(\theta)sin(\psi) & -sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi) \\ -sin(\theta) & sin(\phi)cos(\theta) & cos(\phi)cos(\theta) \end{bmatrix}. \tag{13}$$

Where ψ, θ and ϕ are the Euler angles as described earlier.

The angular velocity of the e-frame with respect to the i-frame projected onto the i-frame is given as[14]

$$\bar{\omega}_{ie}^e = \begin{bmatrix} 0 & 0 & \omega_e \end{bmatrix}^T. \tag{14}$$

Where ω_e is the angular velocity of the earth in the i-frame and has a constant value of $7.2921158 \times 10^{-5}\ rad/s$[14]. Using the projection matrix Eq. (2) its possible to project $\bar{\omega}_{ie}^i$ onto the n-frame

$$\bar{\omega}_{ie}^n = C_e^n \bar{\omega}_{ie}^e = \begin{bmatrix} \omega_e cos(\varphi) & 0 & -\omega_e sin(\varphi) \end{bmatrix}^T. \tag{15}$$

$\bar{\omega}^n_{en}$ represents the turn of the n-frame with respect to the e-frame its called the transport rate and may be expressed as the rate of change of latitude and longitude as follows

$$\bar{\omega}^n_{en} = \begin{bmatrix} \dot{\lambda}cos(L) & -\dot{L} & -\dot{\lambda}sin(L) \end{bmatrix}^T . \tag{16}$$

Where $\dot{\lambda} = v_e/(R_N + h)cos(L)$ and $\dot{L} = v_n/(R_E + h)$[14], here we assume the Earth to be an ellipsoid and that there is no variation in earth gravitation depending on where the user is on the ellipsoid. R_N is the meridian radius of curvature and defined as $R_N = R_0(1 - e^2)/(1 - e^2 sin^2 L)^{3/2}$[14]. R_E is the transverse radius of curvature and defined as $R_E = R_0/(1 - e^2 sin^2 L)^{1/2}$. h is the height above the surface of the earth. Rewriting Eq. (16) with $\dot{\lambda}$ and $\dot{L}$

$$\bar{\omega}^n_{en} = \begin{bmatrix} v_e/(R_E + h) & -v_n/(R_N + h) & -v_e tan(L)/(R_N + h) \end{bmatrix}^T \tag{17}$$

where v_e and v_n are the velocities in East and North direction, respectively. Now $\bar{\omega}^n_{in}$ can be obtained by adding Eq. (15) and (17).

$$\bar{\omega}^n_{in} = \begin{bmatrix} \omega_e cos(\varphi) + v_e/(R_E + h) & v_n/(R_N + h) & -\omega_e sin(\varphi) + v_e tan(L)/(R_N + h) \end{bmatrix}^T \tag{18}$$

Now Eq. (18) is a function dependent both on velocity and position.

Inertial Navigation equation

describing the position of the dinghy in the n-frame is done by[14].

$$\bar{r}^n = \begin{bmatrix} \varphi & \lambda & h \end{bmatrix}^T . \tag{19}$$

Since velocity is described as the rate of change of its position with respect to a frame of reference, and is a function of time, the velocities in North, East and down can be expressed as.

$$\begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix} = \begin{bmatrix} (R_E + h) & 0 & 0 \\ 0 & (R_N + h)cos(\varphi) & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} \tag{20}$$

Where $\dot{}$ symbolizes the first derivative with respect to time. Thus, $\dot{\varphi}$, $\dot{\lambda}$ and $\dot{h}$ can be found by rewriting Eq. (22) as

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{(R_E+h)} & 0 & 0 \\ 0 & \frac{1}{(R_N+h)cos(\varphi)} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix} \tag{21}$$

The velocity dynamics $\dot{v}$ can be described in its respective direction[16]

$$\dot{v}_N = f_n - v_E(2\omega_e + \dot{\lambda})sin(L) + v_d\dot{L} + g \tag{22}$$

$$\dot{v}_E = f_n + v_N(2\omega_e + \dot{\lambda})sin(L) + v_D(2\omega_e + \dot{\lambda})cos(L) - g \tag{23}$$

$$\dot{v}_D = f_n - v_E(2\omega_e + \dot{\lambda})cos(L) - v_N\dot{L} + g \tag{24}$$

The attitude dynamics are defined such [16]

$$\dot{C}_b^n = C_b^n \Omega_{ib}^b - \Omega_{in}^n C_b^n. \tag{25}$$

The first term, $C_b^n \Omega_{ib}^b$ is a function of the body rates, as sensed by the strapdown gyroscope, the second term, $-\Omega_{in}^n C_b^n$ is a function of lower navigation frame rates. Ω is a skew matrix formed from the elements of the turn rate

Integration of INS/GPS Kalman Filter

Linearizion of Non-Linear Equations

Non-linear differential equations of navigation must be linearized in order to be usable in estimation methods such as the Kalman Filtering [16]. Using perturbation method to linearize the non-linearized equation for example, the perturbation of the position, velocity and attitude DCM can be expressed as:

$$\hat{\bar{r}}^n = \bar{r}^n + \delta \bar{r^n} \tag{26}$$

$$\hat{\bar{v}}^n = \bar{v}^n + \delta \bar{v^n} \tag{27}$$

$$\hat{C}_b^n = (I - E^n) C_b^n \tag{28}$$

Where $\hat{.}$ is referred to the estimated value, δ is the perturbation errors. E^n is the skew-matrix of the attitude errors [16], i.e.

$$E^n = \begin{bmatrix} 0 & -\delta_D & -\delta_E \\ \delta_D & 0 & -\delta_N \\ -\delta_E & \delta_N & 0 \end{bmatrix} \tag{29}$$

Attitude, Velocity and Position errors

Position errors

Implementing the fusion Kalman Filter

Implementing a continuous Kalman filter using integration between Inertial Navigation System and a Global Positioning System.

$$\hat{\dot{\bar{x}}} = F\bar{x} + G\bar{u} \tag{30}$$

Where F is describing the dynamics of the system, $\bar{x}$ is the state vector, G is the design matrix, $\bar{u}$ is the input matrix, i.e. forces acting on the dinghy recorded by the IMU and $\hat{\bar{x}}$ is the estimated state vector.

Where F is a 9×9 matrix and $\bar{x}$ is a 9×1 is given by

$$F = \begin{bmatrix} F_{rr} & F_{rv} & 0 \\ F_{vr} & F_{vv} & (\bar{f}^n \times) \\ F_{er} & F_{ev} & -(\bar{\omega}_{in}^n \times) \end{bmatrix}, \qquad \bar{x} = \begin{bmatrix} \delta \bar{r}^n \\ \delta \bar{v}^n \\ \delta err^n \end{bmatrix} \tag{31}$$

G is a 6×6 matrix and $\bar{u}$ is a 6×1 vector and is defined as.

$$G = \begin{bmatrix} -C_b^n & 0 \\ 0 & C_b^n \end{bmatrix}, \qquad \bar{u} = \begin{bmatrix} \delta \bar{\omega}_{\dot{i}b}^b \\ \delta \bar{f}^b \end{bmatrix} \tag{32}$$

The elements of $\bar{u}$ is assumed to be white noise with zero mean, thus

$$Q = diag\begin{bmatrix}\sigma_{ax}^2 & \sigma_{ay}^2 & \sigma_{az}^2 & \sigma_{gx}^2 & \sigma_{gy}^2 & \sigma_{gz}^2\end{bmatrix}^T \tag{33}$$

where $\sigma_{a(x,y,z)}^2$ and $\sigma_{g(x,y,z)}^2$ is the variance for the accelerometer and gyroscope in every direction, respectively.

Since a computer does not use continuous time domain the equation has to be transformed into discrete domain. The state equation will then be expressed as

$$\hat{\bar{x}}_{k+1} = \Phi_k \bar{x}_k + \bar{w}_k \tag{34}$$

Here Φ_k is the state transition matrix in discrete time domain, in order to obtain a discrete state transition matrix the inverse Laplace transform is performed on the continuous state transition matrix, i.e.

$$\Phi_k = \mathcal{L}^{-1}\left[(SI - F)^{-1}\right] \tag{35}$$

Since the sample interval, Δt is very small in this case Eq. (35) can be approximated

$$\Phi_k = e^{F\Delta t} \approx I + F\Delta t. \tag{36}$$

From Eq. (34), $\hat{w}_k$ is the driven response t_{k+1} due to the input white noise. White noise is uncorrelated between sample periods, i.e the noise between t_k and t_{k+1} is uncorrelated [19]. Then the covariance matrix which is associated with $\bar{w}_k$ is [19]

$$\mathbb{E}[\bar{w}_i \bar{w}_j^T] = \begin{cases} Q_k & i = j \\ 0 & i \neq j \end{cases} \tag{37}$$

and Q_k can then be approximated using first order of the discrete transition matrix [20]

$$Q_k \approx \Phi_k G Q G^T \Phi_k^T. \tag{38}$$

If Eq. (33) is analyzed, by increase the norm of Q_k the Kalman filter trusts the measurements more than the system, which will make the output more noisy due to noise induced from the measurements, the advantages with a large norm is that the time lag will decrease. If the norm is small the measurements will be less induced with measurement noise but time lag will increase, which means that we don't trust the measurements. Determining Q_k can be done by testing several different settings and from that make an assumption, but a good assumption should be that the trajectory of the output should follow the GPS data.

The Kalman filter is a linear quadratic estimator which is recursive and the estimator is unbiased and has minimum variance. the algorithm starts from a random process model, i.e. Eq. (34) and the following observation matrices [20].

$$z_k = H_k \bar{x}_k + \bar{e}_k \tag{39}$$

where z_k is the measurement vector and e_k is random measurement noise, with following characteristics[19].

$$\mathbb{E}[\bar{e_i}\bar{e_j}^T] = \begin{cases} R_k & i = j \\ 0 & i \neq j \end{cases} \tag{40}$$

The Kalman Filter can be seen as a two step filter with a prediction update and a correction update. In the latter case the Kalman gain is first calculated by

$$K_k = P_k^- H^T (H P_k^- H^T + R_k)^{-1} \tag{41}$$

then the state vector is updated

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{42}$$

the last step is updating the covariance matrix

$$P_k = (I - K_k H)P_k^- \tag{43}$$

When correction update is done the algorithm makes a prediction update this is done by

$$\hat{x}_k^- = \hat{\hat{x}}_k \Phi_k \tag{44}$$

then updating its covariance

$$P_k^- = \Phi_k P_k \Phi_k^T + Q_k \tag{45}$$

Where $(.\bar{}_k)$ should be interpreted such as calculating the prediction at time k given time $k-1$.

The measurement vector z_k is containing the difference of velocity and position from the INS and GPS

$$z_k = \begin{bmatrix} \lambda_{INS} - \lambda_{GPS} \\ \varphi_{INS} - \varphi_{GPS} \\ h_{INS} - h_{GPS} \\ v_{nINS} - v_{nGPS} \\ v_{eINS} - v_{eGPS} \\ v_{hINS} - v_{hGPS} \end{bmatrix} \tag{46}$$

The measurement matrix H_k is defined such

$$H_k = \begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{47}$$

The measurement noise matrix R_k is defined such as

$$R_k = diag(\sigma_{r_n}^2 \quad \sigma_{r_e}^2 \quad \sigma_{r_d}^2 \quad \sigma_{v_n}^2 \quad \sigma_{v_e}^2 \quad \sigma_{v_d}^2) \tag{48}$$

where σ_r^2 and σ_v^2 is the variance of the position and velocity in all direction, respectively.

The Kalman Filter is invoked every time the GPS is updated, i.e. $1Hz$, but since the IMU and the GPS updates at different frequencies a problem arises.

The problem is such that at $t_{GPS}(k)$ there won't be a value to read from the IMU, since discrete time domain. To solve this problem a linear interpolation is done between $t_{imu}(k)$ and $t_{imu}(k+1)$, where $t_{imu}(k) \leq t_{GPS}(k) \geq t_{imu}(k+1)$. See Fig. 23.

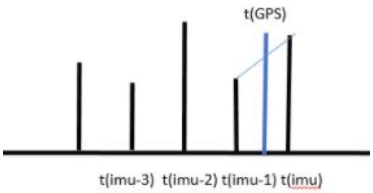

Figure 23: Different update sequences IMU and GPS

The following linear interpolation equation is used for calculating the positions at $t_{GPS}(k)$

$$r_n(t_k(GPS) = r_n(t_{imu}(k)) + \frac{r_n(t_{imu}(k+1)) - r_n(t_{imu}(k))}{t_{imu}(k+1) - t_{imu}(k)}(t_{GPS}(k) - t_{GPS}(k)) \tag{49}$$

and the equations for the velocities

$$v_n(t_k(GPS) = v_n(t_{imu}(k)) + \frac{v_n(t_{imu}(k+1)) - v_n(t_{imu}(k))}{t_{imu}(k+1) - t_{imu}(k)}(t_{GPS}(k) - t_{GPS}(k)). \tag{50}$$

References

[1] http://www.st.com/en/development-tools/stm32cubemx.html

[2] http://www.st.com/en/embedded-software/x-cube-ble1.html

[3] http://www.st.com/en/embedded-software/x-cube-mems1.html

[4] http://play.google.com/store/apps/details?id=com.st.blunrg

[5] http://www.st.com/en/ecosystems/x-nucleo-idb05a1.html

[6] http://www.st.com/en/ecosystems/x-nucleo-iks01a2.html

[7] http://x-io.co.uk/open-source-imu-and-ahrs-algorithms

[8] http://x-io.co.uk/res/doc/madgwick_internal_report.pdf

[9] http://update.maestro-wireless.com/GNSS/A2235-H/Maestro_GPS_Evaluation_Kit_EVA2235_H_User_Manual_V01.pdf

[10] http://github.com/jacketizer/libnmea

[11] http://www.st.com/content/st_com/en/products/ecosystems/stm32-open-development-environment/stm32-nucleo-expansion-boards/stm32-ode-sense-hw/x-nucleo-53l0a1.html

[12] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the*. IEEE, 85(1):6 –23, jan 1997

[13] BOKEN *BOKEN*.

[14] Dr. Oliver Nelles
nonlinear system identification.

[15] Duygun, M., Kutlu, L. & Sickles, R.C. J Prod Anal (2016) 46: 155. https://doi.org/10.1007/s11123-016-0477-z

[16] Noureldin ., Karamat T.B., Georgy J. (2013) Basic Navigational Mathematics, Reference Frames and the Earth's Geometry. In: Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration. Springer, Berlin, Heidelberg

[17] https://edg.uchicago.edu/tutorials/load_cell/

[18] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the*. IEEE, 85(1):6 –23, jan 1997

[19] Probability and Random Processes with Applications to Signal Processing, 4/E (2012). Henry Stark, John W Woods. Pearson Higher Education.

[20] Estimation, Control, and the Discrete Kalman Filter, 1989. Donald E. Catlin