Inertial navigation system*

Nils Brynedal Ignell
nbl08001@student.mdh.se

Aseem Rastogi
aar10001@student.mdh.se

ABSTRACT

For most AUVs, an accurate estimation of the AUV's position and orientation is important for mission success. In the Vasa Project, an Inertial Measurement Unit (IMU) was the only sensor used to obtain position and orientation data. This solution was found to be insufficient, particularly regarding the accuracy of the yaw (heading). Lessons learned from the Vasa Project were used in the design of the Naiad AUV.
The Naiad AUV is equipped with two inertial sensors: One VN-100 Inertial measurement unit (IMU) and a SAAB Fiber optic gyroscope (FOG). The VN-100 is a significantly better IMU than the one used in the Vasa Project. The fiber optic gyroscope is used as a second sensor for yaw readings.
The IMU and FOG are connected to a circuit board known as the *INS board*. This circuit serves as an electrical interface to a Generic CAN controller that reads the sensors and outputs the readings on the CAN bus.
The *Sensor fusion* board will in turn receive these readings from the CAN bus and calculate an estimation of the AUV's orientation and position.
The IMU, the FOG, the INS board and the Generic CAN controller connected to the INS board constitute the *Inertial navigation system*.

1. INTRODUCTION

Any Autonomous Underwater Vehicle (AUV) such as the Naiad AUV is highly dependent on accurate information about its orientation and position in order to perform its missions. The Inertial Measurement Unit (abbreviated "IMU" in this report) used in Project Vasa [9] did not give sufficiently accurate orientation data. The main problem was the yaw angle (also known as the heading).
The pitch and roll angles rarely deviate more 30 degrees during the course of a normal AUV mission (for either the Vasa or Naiad AUVs). The yaw angle however, can turn several full rotations during the course of a mission. This causes the yaw angle to drift significantly more than the pitch and roll angles. Moreover, the pitch and roll angles can easily be corrected using the gravity vector as a reference. This cannot be done with the yaw angle since the yaw angle moves in a plane perpendicular to the gravity vector.
For this reason many IMUs, including that used on the Vasa AUV, rely on a magnetometer that measures the Earth's magnetic field to correct the yaw angle. This may work well, but in some cases the magnetometer can be subject to interference from other electromagnetic fields created by equipment such as electric motors (including those on the AUV itself), generators, power lines, etc. For this reason it was decided that the Naiad AUV was to be equipped with a Fiber Optic Gyroscope (abbreviated "FOG" in this report) in order to get a separate sensor for the yaw angle.

All the electric circuitry needed to interface with the IMU and the FOG was put on one circuit board known as the *INS board*. This board is "stacked" on top of a Generic CAN controller. The *Generic CAN controller* is a small (approx. 76 by 40 millimetres) electronic board that has an AT90CAN128 microcontroller [3], an MCP2551 CAN transceiver [8] and all the peripheral circuitry needed for these as well as its own power supply. The Generic CAN controller is used throughout the project for several tasks. It can be connected to the CAN bus and has two UART buses as well as an SPI bus.
The INS board is powered by the Generic CAN controller and provides it with the SPI output from the FOG and a UART interface with the IMU. The Generic CAN controller sends the readings of the IMU and FOG over the CAN bus to the Sensor fusion board.

2. IMPLEMENTATION

The sensors used were the VN-100 Inertial Measurement Unit from VectorNav Technologies [13] and the 8088 000-112 Fiber optic gyro from SAAB [11]. The VN-100 (refereed to as the "IMU") is a digital sensor outputs its measurements on a serial interface. The IMU's maximum output rate of 200 samples per second is used. The IMU can output angular rates of rotation, orientation, acceleration as well as the output from its magnetometer. The orientation can be given in several different formats including quaternion, directional cosine orientation matrix as well as yaw, pitch and roll readings. Yaw, pitch and roll readings were chosen since this is the most compact way of describing the orientation.

*This report was written during the fall of 2013 in an advanced level project course at Mälardalen University, Sweden.

The Fiber optic gyro (refereed to as the "FOG") is an analog sensor and gives an analog voltage output proportional to the angular speed of rotation. The analog output is filtered and sampled by an analog-to-digital converter (ADC) that interfaces with the AT90CAN128 on the Generic CAN controller via SPI.

The software running on the Generic CAN controller is called AT90CAN_Ins_Controller. It handles communication with the IMU and the ADC that is connected to the FOG as well as outputs the measurements from the sensors onto the CAN bus.

2.1 Inertial measurement unit

The IMU is powered by the +5V feed from the Generic CAN controller. It provides a serial interface at both 3.3 Volt TTL levels and RS-232 levels, neither of which can be directly connected to Generic CAN controller since the UART buses on the AT90CAN128 microcontroller work at +5 Volt TTL levels. A level change was consequently needed. The MAX232 [7] driver/receiver was chosen to convert between RS-232 levels and +5 Volt TTL levels since it needs only a 5 Volt supply.
The IMU has one output pin called *SYNC_OUT* that can be configured to output a synchronization pulse whenever the IMU has completed a measurement. The SYNC_OUT pin is connected to Port D, pin 2 on the AT90CAN128. This pin activates the *external interrupt 2*.
The above configuration enables an interrupt service routine to be activated every time the IMU has made a measurement. [1]

2.1.1 Software interface

As mentioned, the IMU communicates over a serial interface and is connected to the UART0 bus of the AT90CAN128. The AT90CAN_Ins_Controller software starts by initiating the UART to run at a baud-rate of 115200 baud.
The default mode of the IMU outputs data at an undesired rate and format. Hence a set of commands, shown in chronological order in Table 1, need to be sent to configure the IMU. The default baud-rate of the IMU is 115200 baud which is too slow for the highest output rate. For this reason a command to switch the baud-rate is sent.
The commands sent to the IMU start with a dollarsign "$", followed by "VNRRG" that indicates that one wants to write to a register, a comma, a two digit number indicating the ID of the register, a comma, register specific parameters, an asterisk "*", a checksum and finally a newline character. The first message sent turns off the use of checksums since checksums were not considered to be needed.
After the messages are sent, the UART bus is reinitialized to run at 230.4 kBaud. The receive buffer is flushed to remove any old irrelevant data.

As mentioned, an external interrupt service routine (ISR) is activated each time the IMU finishes a measurement. Since the process of receiving and interpreting a response from the IMU is too computationally demanding to be done in an ISR, the ISR will only set a boolean flag.

[1] Note that this pin also is the receive pin on the UART1 bus. Hence UART1 is therefore inactivated.

The main loop calls a function called *Update* that reads the interrupt flag. If the flag is set the function will start to read the UART receive buffer. Each message from the IMU will have the following format:

$VNYBA,+123.456,+789.123,+456.789,+123.456,
+789.123,+456.789,+123.456,+789.123,+456.789*

The *Update* function begins by calling a function called *Start_Message* that reads characters one at a time, searching for the dollar sign. Once the dollar sign is found the function ensures that the text "VNYBA," follows, if not the message found in the buffer is not desired so the function searches for the next dollar sign and the process repeats.
Once the correct message has been found, its contents needs to be decoded. The message consists of 9 float values: yaw, pitch and roll angles, acceleration in x-, y- and z-axes and magnetic strength in x-, y- and z-axes, each value separated by commas.
Characters are read one by one and put into a temporary buffer until a comma is found. This way this buffer contains a string on the "+123.456" format. Since the Ada toolchain used for the AT90CAN128 in this project did not include any implementation of string-to-float conversion (such as Float'Value), a function fStr2Float (put in the Str2Float package) was implemented.

Values for yaw, pitch and roll angles as well as acceleration in x-, y- and z-axes are read. These values are put into CAN-messages using functions in the *Can_Float_Conversions* project. The CAN-messages are sent on the CAN bus to be read by the Sensor fusion.
The magnetic readings are not read and are consequently left in the receive buffer. This is another reason for having the *Start_Message* function mentioned above.

2.2 Fiber optic gyroscope

Since the Fiber optic gyroscope is a high precision analog sensor great care has to be taken when designing its interface circuit in order to attain full precision.
The accuracy of the FOG is highly dependent on the quality of the power supply, any noise on the power supply will decrease accuracy. The same need for accuracy is present in the analog to digital conversion. Fortunately, the manufacturer provided the Naiad AUV project with a schematic for the circuit that was to be used for the analog to digital conversion. The analog to digital converter (ADC) used was the ADS1255 [2] extremely low-noise, 24-bit analog to digital converter which has an SPI interface.

2.2.1 Power supply

The FOG has three power lines, +12 Volts, -12 Volts and +5 Volts as well as a power ground and a signal ground.
According to the manual of the FOG, the +5 Volt line is not allowed to be present without the +12 Volts and -12 Volts lines. For this reason a PVG612 Photovoltaic Relay [10] is used to switch the +5 Volt line. The relay is activated by a digital output pin from the AT90CAN128 which is set high after a delay at start up, thus ensuring that the +5 Volt line is not present before the +12 and -12 Volt lines.
In order to stabilize the supply voltage, several capacitors

Table 1: Messages sent to initiate the IMU

Command name	**Command string**	**Meaning**
Communication Protocol Control	$VNWRG,30,0,0,0,0,0,0,1 *68	Turns of checksums, sets error messages to be returned by IMU if incorrect messages are received, and turns off some other functions.
Async Data Output Type Register	$VNWRG,06,0*	Turns off asynchronous output to avoid unwanted data.
Async Data Output Frequency Register	$VNWRG,07,200*	Sets asynchronous output rate to 200 Hz.
Synchronization Control	$VNWRG,32,3,0,1,0,3,1,1,500000,0*	Configures the synchronization pulse.
VPE Basic Control	$VNWRG,35,1,2,1,1*	Various settings for the attitude filtering algorithm.
Async Data Output Type Register	$VNWRG,06,16*	Sets asynchronous output to "Yaw, Pitch, Roll, Body True Acceleration, and Angular Rates". This turns on asynchronous output.
Serial Baud Rate Register	$VNWRG,05,230400*	Sets the baud-rate to 230.4 kBaud.

are connected between each supply line and ground. These capacitors are placed as close to the FOG connector as possible and have very low equivalent series resistance (ESR) in order to remove as much noise as possible.

Although the Generic CAN controller could supply the INS board with +12 Volts, this supply is not galvanically isolated from the supply voltage. For this reason only the 5 Volt supply, which is galvanically isolated, was used by the INS controller and consequently DC-to-DC conversion from 5 Volts to +12 and -12 Volts was needed.
The User manual of the FOG specifies that low noise switching DC-to-DC converters with a switching frequency of at least 400 kHz are to be used. The LT1931 inverting DC-to-DC converter [6] was used to supply -12 Volts. The LMR62014 step-up voltage regulator [5] was chosen to supply +12 Volts. Both of which use switching frequencies well over 1 MHz.

2.2.2 Analog to digital conversion

Before analog to digital conversion, the analog output from the FOG is filtered to remove high frequency components that could otherwise give rise to aliasing distortion. This filter is included in the schematic provided by the manufacturer of the FOG and was not modified during the project. The THS4131 differential amplifier [12] is the central component of the anti-aliasing filter.

The ADS1255 analog-to-digital converter runs at 100 samples per second and has a resolution of 24 bits, however not all 24 bits are accurate at this sample rate.
The ADS1255 is supplied by both 5 Volts and as well as 3.3 Volts. For this reason the LM1117MP-3.3 linear voltage regulator [4] was used to convert from 5 Volts to 3.3 Volts.

Both the ADS1255 and the THS4131 require a voltage reference of 2.5 Volts. This voltage level needs to be very stable and noise free, or accuracy of the whole circuit could be affected. The ADR421ARMZ ultraprecision, low noise voltage reference circuit [1] was used for this task.

2.2.3 Software interface

The Analog to Digital Converter (ADC) communicates with the AT90CAN128 using SPI. The ADC also has a Data Ready pin (DDRY) which goes low when the data is available for the SPI. The DDRY pin is connected to the external interrupt pin 3 of the AT90CAN128. The software also initializes Timer 1 of the AT90CAN128 and the resolution is set to 1 μs. This timer is required since the time delay between issuing a read command to the ADC and then reading the data on the SPI is less than 1 μs. The other clock in the AT90CAN128.CLOCK has a resolution of 1ms.

1. **Initialization:** The code starts with initializing the timer to 1 μs and starting it. Then the SPI is initialized with a clock of 1 MHz. The SPI mode is set to setup on rising and sample on falling edge (CPOL = 0, CPHA = 1). The AT90CAN128 is also configured as SPI master. It then selects the slave device, i.e. the ADS1255 ADC, by pulling the Slave Select (SS) pin low. Then the ADC is RESET by writing the RESET command on the SPI. The ADC is then configured by writing to the registers. It is configured to use 100 Samples per Second (SPS) for analog to digital conversion. After that an auto calibrate command is send to complete the initialization.

2. **Execution:** When start is called the SPI sends the Continuous Read Data Command (RDATAC) on the SPI and then waits for 1 μs before reading the data. Once the first data is read, the external interrupt is initialized. The external interrupt is configured to gener-

ate an interrupt on the falling edge of the Data Ready pin (DDRY). Whenever the interrupt occurs the SPI reads 3 bytes of data from the read buffer and then updates the yaw value to the current one. Whenever the *get_yaw* function is called it disables the interrupt and returns the latest yaw value before enabling the interrupt again.

3. RESULT

The majority of the inertial navigation system (INS) was implemented during the project. However, there are some bugs that cause the AT90CAN_Ins_Controller software to crash. At the time of writing, it is believed that this is caused by some form of error that occurs when the micro controller is programmed. Unfortunately, there has not been enough time to solve these problems. Documentation has been prioritized.

The FOG was never connected to the INS board. Since the FOG is an expensive and sensitive piece of equipment it was decided only to connect it once all the hardware and firmware was done and meaningful tests could be performed. This point was not reached during the project.

The IMU was tested during the project and it works very well. The results indicated that the IMU performs very well with very little drift and that it is almost immune to magnetic disturbances.
However, only a few simple experiments were performed in order to test the drift of the orientation readings of the IMU. These tests were only performed during short durations (minutes). In the future it would be wise to subject the IMU to sustained testing over a greater period to ensure sustained reliability.

Throughout all the testing that has been done during the project, no cases of data corruption (bit flips etc.) of commands sent to the IMU or data received from the IMU have been experienced. Hence, the assumption that the checksums were not needed seems to be correct.

4. CONCLUSION AND FUTURE WORK

As stated in Section 3, the majority of the INS system is implemented. Once the bugs are fixed, at least outputting readings from the IMU to the CAN bus should work.
To get readings from the FOG there is some more implementation to do, but this should not take much time.
Future work should primarily focus on getting the INS system and Sensor Fusion to work together only using the IMU data (but without data from the FOG). Once this is done, focus should be to determine the accuracy of this system and how much drift there is. In terms of orientation there should only be a problem of drift in the yaw-axis (see Section 1), which is what the FOG is meant to correct.
Once this is done the accuracy of the FOG should be tested. It is likely that it will need some calibration before the desired accuracy is achieved.

There are some possible ways to implement the INS system differently:

- Connecting the INS board (the interface circuit) directly to the Sensor Fusion board directly instead of sending INS data over the CAN bus. This would reduce the traffic on the CAN bus significantly and decrease the time it takes to send a measured value from the IMU or the FOG to the Sensor Fusion board. This would however go against the Project's goal of having a modular design.
- It could be possible (assuming one does *not* do the above) to implement some form of filtering (Kalman filters etc.) in the AT90CAN_Ins_Controller software in order to reduce noise in the acceleration data. This is not something that has been researched during this project, but it might be worth looking into in the future.

5. REFERENCES

[1] Adr421 datasheet. http://www.analog.com/static/imported-files/data_sheets/ADR420_421_423_425.pdf. Accessed 2013-12-28.
[2] Ads1255 datasheet. http://www.ti.com/lit/ds/symlink/ads1255.pdf. Accessed 2013-12-28.
[3] At90can32/64/128 datasheet. http://www.atmel.com/Images/doc7682.pdf. Accessed 2013-12-28.
[4] Lm1117 datasheet. http://www.ti.com/lit/ds/symlink/lm1117-n.pdf. Accessed 2013-12-28.
[5] Lmr62014 datasheet. http://www.ti.com/lit/ds/symlink/lmr62014.pdf. Accessed 2013-12-28.
[6] Lt1931 datasheet. http://cds.linear.com/docs/en/datasheet/1931fa.pdf. Accessed 2013-12-28.
[7] Max232 datasheet. http://www.ti.com/lit/ds/symlink/max232.pdf. Accessed 2013-12-28.
[8] Mcp2551 datasheet. https://www.sparkfun.com/datasheets/DevTools/Arduino/MCP2551.pdf. Accessed 2013-12-28.
[9] M. Östberg et al. Project vasa system overview. 2012.
[10] Pvg612a datasheet. http://www.irf.com/product-info/datasheets/data/pvg612a.pdf. Accessed 2013-12-28.
[11] The fiber optic gyro (fog) product sheet. http://www.saabgroup.com/Global/Documents%20and%20Images/Land/Weapon%20Systems/FOG/000-112_RevB_web.pdf. Accessed 2013-12-28.
[12] Ths4131 datasheet. http://www.ti.com/lit/ds/slos318h/slos318h.pdf. Accessed 2013-12-28.
[13] Vn-100 user manual. http://www.vectornav.com/Downloads/Support/UM001.pdf. Accessed 2013-12-28.