Vasa: An Autonomous Underwater Vehicle

Carl Ahlberg, Lars Asplund, Gabriel Campeanu,
Fredrik Ekstrand, Mikael Ekström, Juraj Feljan,
Andreas Gustavsson, Luka Lednicki, Ivan Švogor

***Abstract*—Vasa is a custom made autonomous underwater vehicle developed at Mälardalen University, Sweden. It is built in a modular fashion, including its mechanics, electronics and software. After participation at the RoboSub competition in 2012 and 2013, this year we are building on the gained experience and investing our effort into two aspects that we deemed crucial: developing a more robust vision system and performing more pool tests compared to the past years. In this paper we give an overview of how Vasa is built, both from the hardware and software points of view.**

I. Introduction

In the last decade, a lot of research has focused on developing vehicles which operate without direct control by a human agent. Autonomous underwater vehicles (AUVs) are a representative example. They offer various functional possibilities in different practical fields, such as naval exploration and mapping, search and rescue missions, waste recovery, navigation etc. Vasa [1], depicted in Figure 1, is an AUV developed at Mälardalen University, Västerås, Sweden. The robot was initially developed as part of a project course in 2011 and started to compete in the RoboSub competition [2] in 2012, where it won the prize for best craftsmanship. For the competition in 2013 most of the team changed and the focus was put mostly on the software of the robot. This year, a largely unchanged team is building on the experience from last year and, having learned from the mistakes made, we are investing most of our efforts into what we consider we lacked the most: making the vision system more robust and performing more testing (increasing the pool time drastically). Currently it is software that gives the most potential for improving on our previous results at the competition, which is why most of our efforts this year were focused there. Regarding the hardware, compared to last year we changed the inertial measurement unit and did miscellaneous minor tweaks, while more substantial changes are planned for the future.

There are nine team members working on the development of Vasa (seven PhD students and two seniors). We have a software and a hardware sub-team, with several team members working on both parts, and thus acting as a bridge between the sub-teams. The hardware sub-team is in charge of the mechanics and electronics of the robot, while the software sub-team develops the low-level software in charge of the robot movement and the high-level software that encompasses the decision center (the "brains" of the robot) and the vision system.

The rest of the paper is structured as follows. In Section II we describe the hardware

Fig. 1: A render of Vasa

Part	Description
A	Front hull
B	Rear hull
C	Battery
D	Power board
E	Backbone
F	High level CPU
G	Thruster
H	Camera
I	Markers and bottom camera
J	Gripper
K	Torpedoes
L	Connector

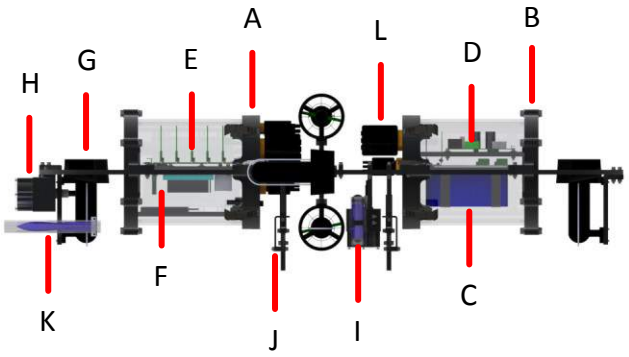

Fig. 2: Parts of Vasa

of the robot (the mechanic, electronics and sensors and actuators). Section III presents a general view of the software architecture, with its higher and lower software layers. Section IV presents our development strategy and the current status of the development. Finally, Section V discusses the future development of the robot.

II. Hardware

In this section we describe the mechanics, the electronics and the sensors and actuators of the robot, in their respective subsections. The robot was designed to be modular, since a modular design simplifies and optimizes the construction process as parallel development is possible. Furthermore, when removing and adding parts with ease, upgrading and servicing becomes simpler. The main defining feature of the robot is the outer frame which allows changing the placement of components in order to optimize the weight-distribution of the robot, even with last-minute additions and changes. The separation of the power and the system electronics into two glass hulls also simplifies the handling.

A. Mechanics

The mechanical design of Vasa (Figure 2) was performed with modularity and flexibility in mind. Two easily removable hulls are attached to the oval frame defining the robot. The hulls contain the bulk of the electronics, which are mounted using a support system allowing easy maintenance and upgrades. Thrusters and equipment are directly mounted to the frame (Figure 3).

Frame: The frame is cut using waterjet from 10mm EN-AW 5754 aluminium. It defines the outer edge of the robot and acts as the mounting structure for all parts of the robot. The frame is perforated for easy mounting of equipment using custom CNC-milled mounts.

Main hulls: The two main hulls contain most of the electronics and each one consists of five major parts; two flanges, two lids and one plastic tube (Figure 4). The flanges and lids are made of EN-AW 5754 aluminium and are initially cut with waterjet and then CNC-machined. Both flanges are attached to the ends of the acrylic glass tube using an ms-polymer based adhesive (Tekton MSPolymer) that ensures a waterproof connection while still being flexible. Since aluminium and plastic

react differently to changes in pressure and temperature, a 1mm separation between the flanges and the tube allows the connection to flex. The flanges are connected to the lids with screws and bolts on one side and compression spring latches on the other side. NBR o-rings are used as seal between the lids and flanges.

Electronics support system: The electronics support system (ESS) (Figure 5) is CNC-milled from polyoxymethylene (POM-H), also known as Delrin. One ESS is mounted to the inner lid of each hull, and is responsible for keeping all the electronics and inner components in place. The ESS consists of two circular plates separated by rods. The rods are mounted to grooves in the circular plates and act both as a supporting structure and as mounting points. The grooves in the circular plates span over 83% of the face of the plates which allows a lot of flexibility as the rods can be shifted along the grooves for optimal positioning. Smaller aluminium mounting plates are mounted on top of the rods, enabling the mounted components to be moved in a direction perpendicular to the rods. Additional rods can be added easily should more mounting points or additional components be required. Two tabs on the outside of the ESS mate with the outer lids, keeping the ESS in place during operation. At the outer side of each ESS, a cooling fan supplies the electronics with cooled air from the cooling fins on the inside of the lids. The fans are permanently mounted to the ESS.

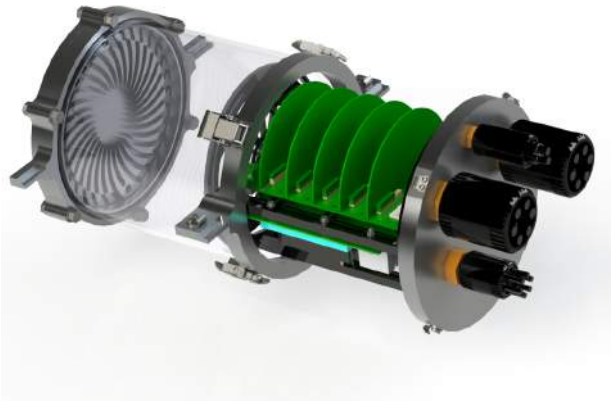

Fig. 4: Hull assembly

Fig. 5: Electronics support system

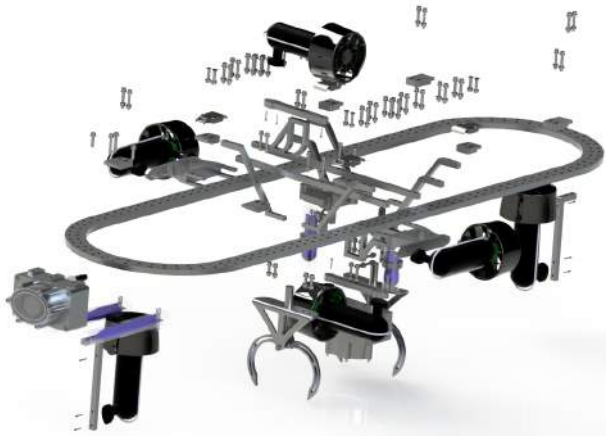

Fig. 3: Frame

Markers: Vasa has two markers (Figure 6) for dropping into specific bins during the competition. The markers are made of plastic and have four fins to keep them stable. Weights have been added to the front part of the markers to keep them negatively buoyant and to make them sink straight down. The release of the markers is controlled by solenoids and the markers are mounted inside acrylic glass tubes close to the downward facing camera for ease in aiming.

Grippers: The purpose of the grippers (Figure 7) is to grab objects in the competition.

There are two identical gripper mechanisms, each controlled by a solenoid. The solenoids are single-action and recover their normal open state by spring power.

Camera housings: The camera housings (Figure 8) are designed around the Point Grey Dragonfly2 [3] board-level camera with optics. The housing consists of three CNC-milled parts. The front and back parts are made of EN-AW5754 aluminium, and the lens is made of 3mm acrylic glass with extra fine thickness tolerance. To waterproof the camera housing, two standard dimension NBR o-rings are used. There is also enough space to fit the camera board mounting with spacers, so that the camera angle can be adjusted. The front camera housing is mounted directly on the frame in front of the forward hull, and the downward facing camera is mounted between the two hulls.

Connectors: The external sensors and actuators require communication links into the hulls. For reliability and ease of service, removable interconnects with fixed hull connectors were chosen. All external connectors are made by Seacon[4] and the cables are spliced together with solder, heat shrink tube, silicon and vulcanizing tape. The number of connectors is larger than the current needs to enable future expansion.

B. Electronics

The design concept of the electronics is to be modular with a minimum of cable interconnects. The electronics are divided into a number of circuit boards each dedicated to handling specific tasks. These circuit boards are connected to a Mini-ITX computer which runs the bulk of the robots software (see Section III). The Mini-ITX computer and the circuit boards communicate via CAN messages.

Fig. 6: Marker

Fig. 7: Gripper

Batteries: The system is powered by a 18,5V (19-21V operational) Lithium polymer battery. The basic configuration is to use one 5S1P 8Ah battery pack, but a 5S2P 16Ah pack may be used to double the run time.

Backbone: The backbone PCB doubles as a communications bus as well as a power supply. Its slot-based (header) design provides a modular system without the use of cables. The board supplies each slot with 3.3V, 5V, -5V, and unregulated battery power (19-21V), together with a CAN (Controller Area Network) bus interface.

Router board: The router board is designed to translate between the CAN bus and the USB interface. The board is equipped with

Fig. 8: Camera housings

an AT90CAN microcontroller by Atmel. The USB serial communication is handled by an FTDI232R chip which transforms serial signals into UART, which is in turn passed to the microcontroller. On the CAN side of the board, a TJA 1040 transceiver serves as a bridge between the microcontroller and the CAN bus.

GPIO board: The GPIO board is designed for high amount of input/output functionality. The board is mainly used for connecting external sensors with the AT90CAN. Currently, an inertial measurement unit (IMU) is connected to the board providing the robot with heading and inclination data necessary for the PD controller functionality. Additionally, it is fitted with an Omni Instruments Series 2600 Submersible Depth Sensor for depth readings to a maximum depth of 15 meters.

Solenoid board: The solenoids are controlled by a dedicated board capable of controlling six solenoids. The AT90CAN microcontroller connects through the CAN bus and the solenoid drivers are of model DRV104 by Texas Instruments.

Power board: The power board's purpose is to regulate the power and to protect the electronics from overcurrent and transient voltages. It is fitted with an AT90CAN controller connected to an LCD screen providing user feedback. The board is equipped with relays designed to cut the power to the motors or to the rest of the system in the event of power failure or kill switch activation.

Motor controller board: The thrusters are controlled by six Pololu VNH5019 H-bridges. They are connected to an AT90CAN for logical processing and CAN bus access. In order to maximize heat dissipation, the H-bridges are mounted to an aluminium plate and fitted with cooling heat sinks (Figure 9).

Sonar and hydrophone board: In order to localize the pinger and detect the range to objects, a special active sonar and hydrophone board was constructed. The board monitors three passive omnidirectional hydrophones. It amplifies and filters their signals in order to filter out noise and non-pinger generated sounds. The pinger heading calculations and the CAN bus communication is handled by an AT90CAN microcontroller. The board also connects the microcontroller to an active sonar, which outputs an analogue value directly proportional to the distance of any object in front of the sonar.

Fig. 9: Motor controller with cooling plate and heat sinks

C. Sensors and actuators

Vasa is equipped with an IMU, a pressure (depth) sensor, two cameras and a sonar (active, as well as a passive one). Its actuators consist of two grippers, torpedo and marker release mechanisms (described above) and six thrusters.

IMU: The robot is fitted with a VectorNav VN-100 Rugged IMU. The sensor contains a three axis gyroscope, a three axis accelerometer, and a three axis magnetometer. The IMU connects to the sensor card through a UART interface.

Pressure Sensor: The robot utilizes a pressure sensor, Series 2600 submersible depth sensor, from Omni Instruments. As it is submersible, it requires no additional casing, so the analog feedback signal goes straight from the sensor into the front hull.

Cameras: Vasa is equipped with two Dragonfly 2 CCD cameras by Point Gray[3]. These are board-level cameras using the IEEE 1394 (Firewire) communication standard.

Sonar: The active sensor module is a Senix TSPC-30S1-232. This sensor has configurable outputs between analog signal and RS-232 serial communication. The passive sonar system uses H2c hydrophones from Aquarian audio products. These are designed to pick up signals from 20Hz to 100Khz.

Thrusters: Six off-the-shelf Seabotix BTD150 [5] brushed DC thrusters (Figure 10) are used for propulsion. The thruster positions enable motion with six degrees of freedom. The three linear motions are:

- Heave — Vertical (up/down) motion
- Sway — Lateral (side-to-side) motion
- Surge — Longitudinal (font/back) motion

The three rotational motions are:

- Yaw — Rotation around vertical (up/down) axis
- Pitch — Rotation around lateral (side-to-side) axis
- Roll — Rotation around longitudinal (font/back) axis

With this configuration, the robot has the ability to move in the most advantageous way to its target, irrespective of its current position. The thruster configuration (Figure 10) chosen is a variation of the cubic formation that enables the robot to maintain its modular design together with the two main hull configuration and optimal sensor, gripper and marker positions. All the thruster pairs have one single intersection point which is positioned slightly in front of the center of the frame. This enables fitting the downward facing camera and grippers in their near-optimal positions.

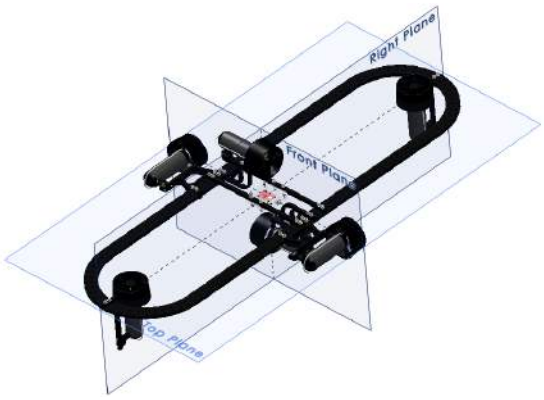

Fig. 10: Thruster configuration

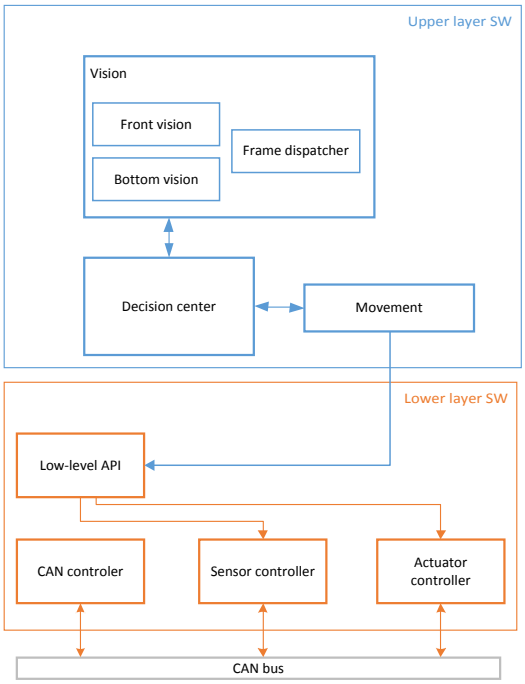

Fig. 11: Vasa's software architecture

III. Software

The underlying platform of the software of Vasa is the Debian Wheezy 7.0 operating system [6] running on the Mini-ITX computer mentioned in Section II. The main characteristic of the robot's software architecture are two component-oriented layers. The bottom layer, i.e. the lower layer software, provides the API for communication with the sensors and actuators. The upper layer software uses the API provided from the lower layer and defines the software components necessary for the complex tasks, such as object detection, object recognition, navigation and mission execution. The architecture is depicted in Figure 11 and is described in detail in the following subsections.

A. Lower-layer software

As stated in Section II-B, the electronics of the robot are composed from both of-the-shelf

components (Mini-ITX board, sensors, actuators) and custom developed electronic boards. The communication between the heterogeneous electronic boards is handled through the CAN bus. It is suitable for linking different types of hardware due to its broadcast characteristics — every component connected to the CAN bus can directly communicate with all other components. The orange part of Figure 11 shows the low-level software. This part of the architecture enables communication with the hardware, i.e. reading from the sensors and controlling the actuators. It is composed from three controller components and an API component.

The CAN controller component handles messaging through the common bus. This involves message sending, receiving, prioritization, synchronization etc. The sensor controller is used to handle the data sent from the sensors. It handles the sampling frequency and data interpretation (e.g. binary operations).

The actuator controller and sensor controller work in a fairly similar fashion, however with opposite communication flows. The controllers handle data coding into a format understandable to the hardware. The low-level API component provides access to sensor data and the actuator commands to the high-level software, in the form of function calls (e.g. `get_depth(depth)`, `move_fwd(speed)`).

The entire lower layer software is implemented in Ada [7], a programming language developed for the purpose of systems programming, concurrent and real-time system development. Ada was chosen because of its I/O standard libraries which simplified the CAN controller development (it also provides a fail safe system with a watchdog algorithm). Furthermore, it has a fairly simple way of interfacing with other languages, which is a benefit for this implementation since the upper layer software is partly written in C++. When starting the robot, the operating system first starts the lower layer software, which then hands over the control to the upper layer software. At run time the layers communicate frequently, as the upper layer handles the intelligence, while the lower layer handles system support.

B. Upper-layer software

The blue, upper part of Figure 11 shows the architecture of the high-level software. It consists of three main software components: decision center, vision and movement.

Decision center: The decision center is the main high-level component, it corresponds to the brains of the robot. It receives information from the vision component about objects being detected, makes high-level decisions about which actions the robot should perform, and controls the movement of the robot accordingly (through the movement component). It runs as a hierarchical state machine - there is a state machine describing the actions for each mission the robot has to perform, and there is a state machine that takes care of switching between the missions. The decision center is implemented in Ada.

Vision: The vision component consists of three sub-components: frame dispatcher, bottom vision and forward vision. Each component runs as a separate program and is implemented in C++.

The main reason for using three separate programs to implement the vision is that the two cameras can only be accessed from within one and the same program, the frame dispatcher, due to underlying architectural limitations. The frame dispatcher uses the FlyCapture SDK [8] to obtain frames from both cameras and then forwards the appropriate frames to the bottom and forward vision components. Already here initial frame manipulation is done by choosing the desired brightness, resolution etc.

Image frames are transferred from the frame dispatcher to the front and bottom vision programs via shared memory to achieve high performance. The front and bottom vision programs analyze the images to detect objects of interest and sends the resulting information to the decision center. The communication between the vision components and the decision

center (in both directions) is done via UDP sockets.

Some positive effects from running the vision as three separate programs are that a clear distinction between different components is achieved, the available parallelism in the ITX processor is highly utilized, and the amount of necessary changes to the software once we introduce our custom camera boards (see Section V) is minimized; the required changes will be to remove the frame dispatcher program and within the front and bottom vision programs instead read the images from the location in memory where they are put by the FPGA component on our custom boards.

For object detection and recognition we use OpenCV [9]. It is an open source library developed by Intel, mainly aimed at real-time computer vision. Some of the main features of OpenCV used for Vasa are color space conversion, morphological transformations, Hough transformations, Canny edge detection, etc. For more information on our vision algorithms, see Section IV.

Movement: The movement component is implemented in Ada and provides movement primitives and complex navigational functions. However, these movement primitives are different from the ones in the lower layer, since they provide a more convenient usage (e.g. `move_fwd(speed)` from the lower layer becomes `MoveForward(distance, speed)` in the upper layer). An example of a more complex behavior is object alignment, where the movement component uses a data stream from the vision component with position information about the detected object and uses this data to change its position accordingly.

IV. Strategy and current status

In this section we briefly discuss our work strategy chosen for this year and report on the current status of the robot.

As we are now in the second year in a row of participation in RoboSub with a largely unchanged team and stable hardware, we have a much better understanding of what to expect at the competition. We can build on our experience from last year to tailor our development strategy. What this mostly means is that we can correct the mistakes we made in preparing for RoboSub 2013. There are two major aspects we lacked and which this year's strategy builds upon: more testing and a more robust vision system.

Testing

The only pool we had available for testing during 2013 was a city owned pool (Figure 12), where the first in-water test took place on April 1st. The issue with this pool is that we would typically only get short time slots. This meant the overhead of moving our gear to the pool and back was considerable and we could not always test when we wanted to. To tackle this problem, we purchased a 5 meter wide garden pool (Figure 13) which we placed at our campus. This means we now have more or less constant pool access and that the time necessary to move the gear from our lab to the pool is down to a couple of minutes. Of course, due to the size of the garden pool, we can only use it for limited kinds of tests (for instance basic movements and object recognition), and still have to use the city pool for running full missions.

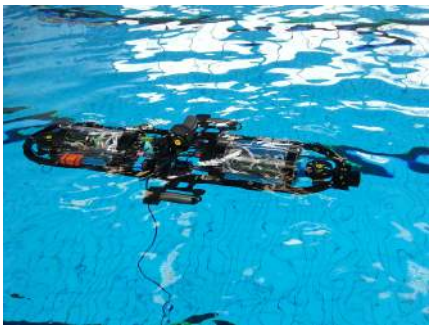

Fig. 12: Testing Vasa in the large indoor pool

To further increase testing time and increase cohesion within the team, we decided to spend one week away from our campus. Since all team members have obligations outside of RoboSub, working off-campus meant getting away from said obligations and being able to

Fig. 13: Testing Vasa in the small garden pool

invest all of our time on developing the robot. During this week we made considerable contributions to the software including improving the vision algorithms and improving the high-level movement algorithms (aligning to an object of interest).

Vision

The other main aspect of our strategy is to develop a more robust vision system compared to the one we had last year. Last year's vision algorithms started the image filtering chains with filtering out objects based on their color, and the interval defining a particular color had to be manually calibrated based on the current lighting conditions. This manual calibration was tedious and time consuming.

This year the image filtering chains start with filtering out objects based on their contours, and later when color is taken into account, it does not rely on manually defined intervals. Furthermore, at the same time we have several recognition algorithms active, so for every detected object there is a confidence parameter.

Finally, we take into account the history of object movement, meaning that the confidence parameter increases if the object has been detected in a similar place in consecutive frames. All this together contributes to the vision system being much less prone to noise than last year.

Other improvements

Last year the robot was damaged in transport to San Diego, which meant we had to invest the first two days of the competition on repairing it rather than doing further testing. This year we obtained better containers to minimize the risk of this repeating. They showed to be another good investment, as the transport to and from the aforementioned testing camp went flawlessly.

We also made miscellaneous hardware improvements including making it possible for the robot to be able to run in the pool on AC power, thus removing the need for often occurring battery changes. The most considerable hardware change was getting a better quality IMU, as the previous one was affected by the vibrations of the thrusters, which caused the robot to drift unpredictably. Pool tests showed that with the new IMU this is no longer an issue.

Current status

With roughly one month left until the competition, the vision algorithms are more or less finished. These will probably see further improvements, but now our focus shifts to the movement center and the decision center. The former is more or less straightforward, we simply need to finish the implementation, while the latter is more challenging and requires more high-level design decisions (for instance, we need to decide on a strategy if the robot gets lost, then if the robot temporarily looses sight to an object it is aligning to and so on). Compared to the same time period last year we improved considerably, but lots of work remain to be done.

V. Future work

Being constructed in a modular system, Vasa is continuously maintained and enhanced. In this section we present what we envision to have in place for next year's competition.

The most significant change planned is to enhance the vision system by introducing stereo vision, and to replace the current ITX computer that currently runs the software. Both changes are covered by a custom electronics

board called GIMME2 (General Image Multi-view Manipulation Engine). GIMME2 (Figure 14) is a stand-alone mobile processing platform developed at Mälardalen University. It has two OmniVision OV10810 10-Megapixel image sensors capable of delivering 30 fps (4320x2432) or 60 fps@1080p. It combines FPGA and a dual core ARM Cortex-A9 on a single chip, making it a platform powerful enough to run the complete software of the robot. It also gives us the flexibility of choosing the optimal software deployment onto the processing units (CPU and FPGA). The plan is to have one GIMME2 board for the front camera system, another board for the bottom camera system and finally a third board which can be used for the decision center.

Fig. 14: GIMME2 platform

The current structure of the software makes the planned hardware changes possible with minimal required changes to the software. Also, due to the fact that the vision algorithms are both more robust and generally perform better than last year, we expect that any changes to the software will be of an evolutionary nature and less drastic than this year's.

ACKNOWLEDGMENTS

We would like to thank the following people who were in various ways involved in the development of Vasa: Jan Carlson, Federico Ciccozzi, Séverine Sentilles, Emil Segerblad, Micael Östberg, Rickard Linder, Anton Widenius, Ammar Ismail, Johnny Holmström, Athar Ahmed, Ejaz Ui Haq, Rafat Ghanim, Lars Lagerholm, Peter Wåhlin, Mingli Wu, Martin Ekström, Giacomo Spampinato, Bengt Erik Gustavsson, Henrik Lekryd, Göran Svensson.

We would also like to thank the following companies for their support: ÅF, Würth Elektronik, Stainless Steel Welding HB, Preciform AB and Gullbergs Marina AB.

REFERENCES

[1] The Vasa Project, http://www.mrtc.mdh.se/projects/ralf3/robosub/, [Accessed: 2014-06-01].
[2] The RoboSub Competition, http://www.auvsifoundation.org/Competitions/RoboSub, [Accessed: 2014-06-01].
[3] Dragonfly2, http://ww2.ptgrey.com/firewire/dragonfly-2, [Accessed: 2014-06-01].
[4] Seacon, http://seaconworldwide.com/, [Accessed: 2014-06-01].
[5] Seabotics, http://www.seabotix.com/, [Accessed: 2014-06-01].
[6] Debian, https://www.debian.org/releases/wheezy/debian-installer/, [Accessed: 2014-06-01].
[7] Ada, http://www.adacore.com/adaanswers/about/ada/, [Accessed: 2014-06-01].
[8] Point Grey Research, Inc., http://ww2.ptgrey.com/sdk/flycap, [Accessed: 2014-06-19].
[9] OpenCV, http://opencv.org/, [Accessed: 2014-06-01].