Milton: An open hardware underwater autonomous vehicle

Robert Codd-Downey, Michael Jenkin and Katherine Allison
Department of Electrical Engineering and Computer Science
York University
Toronto, Ontario, M3J 1P3, Canada
{*robert, jenkin*}*@cse.yorku.ca, katie.allison@mail.utoronto.ca*

***Abstract*— Although there are a large number of autonomous ground contact and flying robot platforms, this has not been the case for underwater robots This has recently changed with the development of inexpensive thrusters and other underwater components. Leveraging these components and design principles learned from more expensive remotely operated vehicles this paper describes Milton, an inexpensive open hardware design for a traditional thruster-based underwater robot. Utilizing commercial off-the-shelf hardware and a ROS infrastructure, Milton, and Milton-inspired designs provide an inexpensive platform for autonomous underwater vehicle research.**

***Index Terms*— embedded computing, mobile robotics, underwater robotics.**

I. Introduction

The field of underwater robotics is relatively small in comparison with other sub-fields of autonomous systems. To see this, contrast the number of available underwater autonomous platforms with those available for ground contact and flying robots. For both ground contact and flying robot researchers there exist a large number of experimental and application-specific platforms to support research in the lab. Unfortunately the same cannot be said for underwater robotic systems where very few such platforms exist. Interestingly, the lack of entry-level/research ROV's is not due to the lack of applications for such devices. Tele-operated underwater robots or ROVs are used extensively in the exploration of the deep sea. ROV's find application in location-specific tasks (e.g., servicing underwater structures in the oil and gas industry) as well as in-long duration and long-distance operations including inspection of underwater cables and other structures.

Entry into this research domain is limited by the lack of inexpensive, extensible infrastructure to support research efforts. Here we discus the design and development of an open hardware robotics research platform nicknamed Milton[1]. Milton is designed to be extensible and to provide guidance on a number of design decisions for researchers who are interested in developing their own underwater research platform.

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the NSERC Canadian Field Robotics Network (NCFRN).

[1]Milton is named after the toast making photocopier robot from the TV show Archer

From a design point of view, Milton is intended to be power, data, and computation independent (it requires no tether to a surface-based operator). It is also designed to have an operational period of approximately and hour, and a maximum depth that exceeds 40m. As such, it is designed to operate within the same envelope as recreational divers. A key issue for underwater robotics research where vehicle failure may require recovery by a human operator.

II. Existing Systems

Recent open hardware designs developed by companies such as OpenROV and BlueRobotics have helped to introduce underwater robotics to the hobbyist and research markets. The design of these robots are adequate for simple exploration and surveying tasks, however they lack either the maneuverability, computational power, sensors or autonomous capabilities necessary for use as a research platform. From a research point of view a critical issue for many existing platforms is the lack of sufficient on-board computational power to run standard robotics middleware (e.g., ROS) and support for generic (research) sensors and actuators. A key design goal of Milton was the development of a platform that provides access to standard software tools (ROS) and easily extensible support for a range of sensors and computational units. Although this leads to a more bulky vehicle than might be possible the generality of the design is intentional.

There does exist a small number of commercially available ROV systems in the under $5K space that support underwater robotics research. For example, the OpenROV [1] is an inexpensive entry level vehicle capable of simple maneuvering whilst recording high definition video to a remote computer. Unfortunately due to the 4DOF motion capabilities of the robot and the lack of a standard robotic middleware it is not a viable candidate for robotics research. BlueRobotics' BlueROV [2] and BlueROV2 [3] utilize high efficiency thrusters specifically designed to operate in both fresh and saltwater as well as enough thrusters to provide 6DOF motion capabilities. Unfortunately, these two designs lack the computational power and software to perform effectively without their tethers.

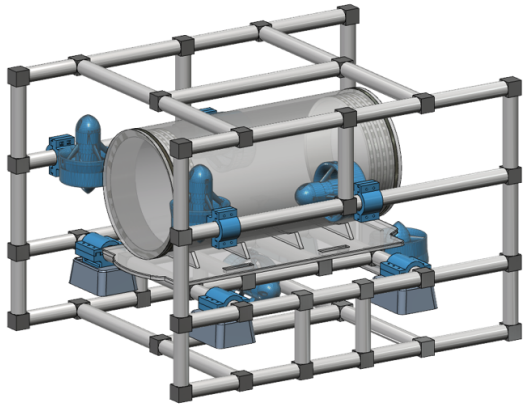
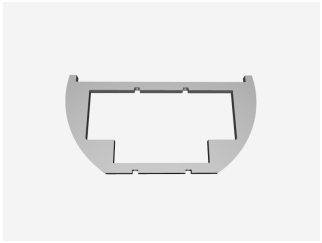
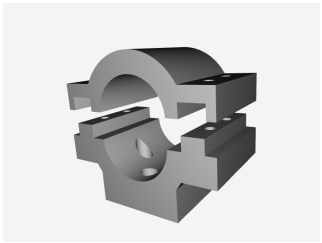


Fig. 1: **Milton CAD Model**
Detailed CAD model of Milton showing the design of its frame and position of its thrusters, ballast weights and waterproof housing.

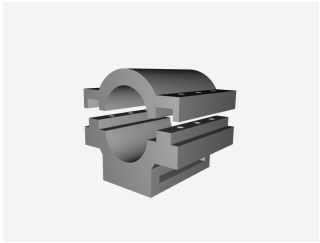


(a) battery holder tray rib

(b) standard tray rib



(c) motor bracket



(d) weight bracket

Fig. 2: **3D printed parts**
A series of tray ribs provide two mounting surfaces that run the length of the housing tube. *(a) battery tray ribs provide space in the undercarriage to hold five batteries. (b) standard tray rib provides the most possible space in the undercarriage. The mounting brackets were specifically designed to firmly grip 3/4 inch PVC pipping. (c) each motor bracket has four holes with relief to attach thruster. (d) each weight bracket holds a standard dive weight and has a channel wide enough for 4cm velcro straps. This is the standard width of a recreational dive belt, making it easier to source weights to properly ballast the vehicle.*

III. Milton

A. Hardware

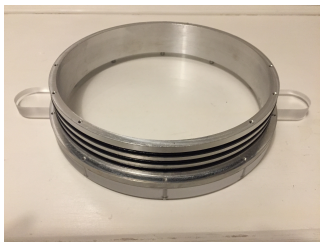
It probably goes without saying, but a key issue in the development of any underwater robot is the requirement to keep the water out. This is an issue that is not typically the key requirement in ground-based robotics, but this requirement leads to a number of critical design decisions in the development of an underwater platform. There are many ways of meeting this requirement for an underwater vehicle, but perhaps the easiest – and the one followed in the design of Milton – is to divide the vehicle into two major sections. The first is a pressure vessel that contains components that need to be kept dry. The second is the collection of components that can be exposed to the water. These two sections are connected by a structural lattice. For Milton, the pressure vessel is based on a cylindrical pipe structure with ends that are designed to provide a water tight seal to 40m.

Milton makes extensive use of 3D printed parts as well as PVC pipes and fittings in order to keep production costs low and to allow for experimentation with the vehicle. Customs parts that would otherwise need to be produced in a machine shop were designed for fabrication using a 3D printer (e.g MakerBot Replicator). PVC piping and connectors are used to create the frame to which all of the robotic components are mounted. One important issues when printing 3D parts for underwater robots is ensuring that the parts themselves contain no sealed voids that might break under pressure.

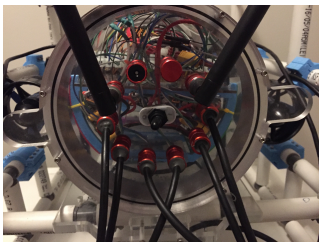
The basic structure of Milton is shown in Figure 4. The pressure vessel is housed in the centre of the device and attachment point on the robot's frame provide connectors for thrusters, external sensors and critically ballast that is used to (i) make the robot neutrally buoyant and (ii) trim the vehicle so that it remains upright and level in the water column in the absence of thruster forces.

The robot's frame and it's subsequent components were designed around the requirement that it can be disassembled, packed and shipped within standard 1560 pelican cases, which can be transported by most airlines. The robot's frame consists of two side walls and three adjacent supporting structures. PVC pipes and standard wedge fittings were used to create the frame. There is a noticeable lack of four-way pipe fittings in the design. Although it would certainly be possible to construct a vehicle with four-way connectors, such connectors are typically more fragile than their two- and three-way versions.

All sensitive electronics are compartmentalized in a single watertight housing, consisting of a 16 inch long cast acrylic cylinder (8.25 inch diameter, 0.25 thick), whose ends are secured using custom end plates. The clear end plates provide ports for visual sensors and displays. Each face plate is



(a) blank end cap



(b) penetration end cap

Fig. 3: **Housing end caps**
(a) blank end cap showing the O-ring structure. (b) end cap with cable, sensor and switch penetrators.

Fig. 4: **Milton**
shows the Milton robot fully assembled on the bench. The mounting points for the ballast weights and the 3d printed mounting points for the thrusters are shown.

attached to a flange using a series of screws along the its circumference, water ingress through this gap is prevented by a single O-ring within the circumference of the mounting screws. Each flange provides a groove for three redundant O-rings which prevent water ingress into the acrylic tube. The rear face plate has a number of drilled holes to allow external non-sensitive electronic components to receive power and data. Figure 3(a) shows a closeup of an end-cap of the pressure vessel. A single O-ring in the upper surface provides a seal between the end cap and the end-cap assembly. These two components are mated together by a series of screws that are on the wet side of the O-ring seal. As these end-caps are inserted/removed each time the robot is serviced, redundancy here is key. The end cap plastic has extra large extensions that assist when removing the end cap from the robot. Note that there is no latch holding the end cap to the robot. Rather the robot relies on the pressure differential between the pressure vessel and the external environment to keep the end caps on the robot.

Figure 3(b) shows a detail of the end cap of the robot and the cables that penetrate it. Each and every penetration of the pressure vessel is a potential entry point for water. Milton utilizes Blue Robotics Cable penetrators which provide a connector with an integrated o-ring to prevent water entry along the end-cap plate and utilizes solid cables within a plastic core to eliminate water entry along the cable itself. Two end-cap penetrators require specific attention. The first is a depth/temperature sensor (Blue Robotics Sensor MS5837-30BA) which enables the robot to estimate its depth in the water column. The second is a pressure release connector. This connector is opened when the end caps are inserted in order to facilitate sealing the vehicle. (If no pressure release was available, seating the end caps would be problematic given the sealed nature of the tube.) This opening is also used to test the seal of the device prior to deployment. By attaching a vacuum pump to the device it is possible to simulate the pressure differential that the vehicle will experience at depth, prior to deployment and to check for leaks in the vehicle's seals.

Electronic components within the waterproof housing are affixed to an electronics tray. This tray is constructed using a series of ribs which support two flat acrylic plates. There are two types of ribs (shown in Figure 2) the battery ribs are specifically designed to support five batteries firmly in place in the bottom section of the tray. The top tray is wide enough to hold the larger electronic components such as the Jetson TK1 and the ZED camera. The tray as a whole snuggly contours the interior of the cylindrical housing and is prevented from sliding around using foam pads along the edge of the ribs.

There are a number of components such as the thrusters and ballast weight that need to be rigidly attached to the robot. Instead of creating fixed mounting points for these components specialized brackets (shown in Figure 2) were designed that contour the PVC pipes that comprise the robot's frame. These brackets allow the thrusters and ballast weights to be mounted in a variety of places on the robot's frame.

B. Electronics

Electronics internal to the pressure housing has two primary functions. (i) to provide controlled power to the thrusters to move the vehicle, and (ii) conditioned power to the on-board sensors and computational units.

- **Jetson TK1** - high performance low power embedded computer.
- **3D Labs** - ZED high definition USB 3.0 stereo camera.
- **Samsung 1TB SSD** - V-NAND 850 EVO solid state drive.
- **Adafruit BNO055** - 9-DOF absolution orientation sensor fused IMU.
- **BlueRobotics Bar30** - High resolution 300m depth/pressure sensor.
- **Adafruit DS3231** - High precision real time clock.

- **Adafruit INA219** - High side I2C DC current and voltage sensor.
- **BlueRobotics BasicESC** - 30A PWM electronic speed controller.
- **Teensy 3.2** - arduino-compatible micro-controller.
- **RoboSavvy Powerboard** - hot swappable battery power board.
- **MultiStaar** - 14.8V 5400mAh 10C Lithium Polymer batteries.
- **BlueRobotics T200** - high efficiency underwater thrusters.
- **Wifi Antennas** - dual-band wireless antennas.

1) Sensing and computation: The Jetson TK1 [4] is a small embedded computer designed by NVIDIA. The TK1 has an NVIDIA Kepler GPU with 192 CUDA cores (upto 326 GFLOPS) and a 2.32 GHz quad-core ARM Cortex-A15 CPU with a typical power consumption between 1-5 watts. This is an ideal choice for an underwater robot because of its low power consumption vs high computational power. The TK1 also has an number of I/O expansion ports for connecting external sensors. There is one major flaw in the design of the Jetson TK1, when power is physically disconnected from the board a large delay is required before reconnection if the board is to initiate a boot sequence. This is problematic if a hard reset is required at depth. This was solved by removing the 0.014F capacitor at C6D4 [5].

The Jetson TK1 does not provide an onboard clock to keep time while not powered on. This presents a real problem because data collected by the robot still needs to be accurately timestamped. The DS3231 real time clock provides an extremely accurate clock that is maintained while the Jetson is powered off. This ensures that data sets separated by a battery recharging step are marked as such.

The Adafruit INA219 voltage and current sensor is included as a safety sensor to prevent the robot from unknowingly running out of power in the middle of a task. This allows the robot to safely halt the current mission and safely return to the surface in the event the voltage of the main battery drops to a critical level.

The 16GB of storage onboard the Jetson TK1 is enough to store the Ubuntu operating system and software packages needed to operate the robot. However this is woefully insufficient to store data produced by the robots sensors. A 1TB solid state drive was added for data storage.

The wireless antennas enable the robot to communicate via Wifi and Bluetooth while hovering at the surface. This allows the robot to receive commands pertaining to the current mission or to send relevant data collected from the most recent mission.

The robot's sensor suite is comprised of the 3D Labs ZED stereo camera, the Adafruit BNO055 sensor fused IMU and the BlueRobotics Bar30 depth and temperature sensor. These sensors provide the robot with both visual and inertial sensing capabilities. The ZED camera leverages the GPU on board the Jetson to produce depth maps and point clouds at 15 frames per second in VGA resolution.

2) Thruster control: The Teensy 3.2 [6] micro-controller is used to drive the PWM signals for all six of the BlueRobotics basic electronic speed controllers. The Jetson TK1 does have PWM capabilities on four GPIO pins, however this is insufficient to control the required six thrusters. The Teensy 3.2 micro-controller was chosen due to its small form factor and compatibility with the ARM version of the FTDI_SIO kernel module. The arduino nano has a similar form factor but suffers from a defect requiring it to be physically disconnected and subsequently reconnected in order to be recognized by udev. This is problematic as any reboot would require opening the underwater housing. The Teensy 3.2 also has the benefit of not overlapping PWM pins with I2C or SPI pins allowing further expansion of its utility.

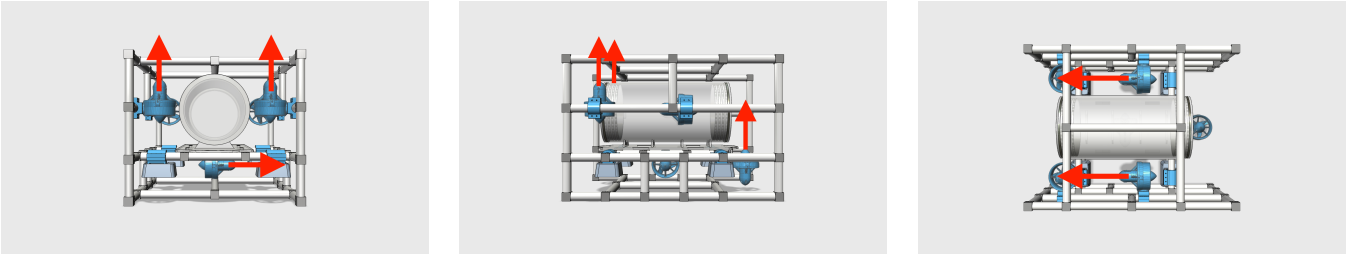
Milton uses six BlueRobotics T200 motors for thrust. These, motors are efficient underwater brushless DC motors capable of producing 15.1 Nm (Newton meters) of torque. The thrusters are configured to provided 6 degrees of freedom (DOF). The configuration and thrust pattern are detailed in Figure 5. Each degree of freedom is controlled by a combination of 2-3 motors surrounding the robot's center of gravity.

The two RobotSavvy power boards are used to provide power to the electronic speed controller used to drive the BlueRobotics T200 motors. The power boards allow the seamless transition from one battery power source to a second backup battery. The power boards also include output pins to detect which battery is currently being to drive the motors.

C. Batteries

A key problem in driving any autonomous system relates to the choice of batteries. A common issue in deploying an ROV is actually transporting it to its deployment. Unfortunately international transportation places strict limits on the nature of batteries that can be transported by air, and it turns out that many island countries now lack non-air transportation. Although more bulky and powerful batteries exist, Milton relies on three Multistar 14.8V 5.2Ah Lithium Polymer batteries. Individually these batteries meet international shipping requirements, and can be easily sourced and abandoned should this be necessary[2]. The Jetson TK1 is powered from a single batteries routed through a current sensor and a buck/boost converter to condition the power to 12V regulated. Each of the other two batteries are used to provide power to three of the six thruster via the RoboSavvy power boards. These power boards provide the option of including two additional batteries as backup that can be source when the primary battery drops below a configured voltage.

[2]batteries meet international commercial air shipping requirements at the time of writing.



(a) front view *(b) side view* *(c) top view*

Fig. 5: **Thruster Configuration**

Shows the direction of positive thrust(red arrow) for each of the motor's on the robot. Each robot is also capable of producing thrust in the opposite direction. (a) shows the motors used to roll or sway the robot. (b) shows the motors used to pitch or heave the robot. (c) shows the motors used to yaw or surge the robot.

D. Software

The Robot Operating System (ROS [7]) is a robotic middleware that provides a standard software infrastructure as well as a collection of tools and libraries aimed at simplifying the software development process. ROS has become the defacto standard for research in the robotics community. Thus it is imperative that any new robotic platform geared towards research provided at least basic support for ROS.

On an embedded computer such as the Jetson TK1 computational resources are a premium commodity. Using a naive software architecture it is especially easy to consume these resource when dealing with high bandwidth sensors such as the ZED stereo camera. To help manage these resources two software stacks are defined; a control stack and a vision stack. The control stack follows the software architecture defined detailed in the ros_control [8] package. Software on the control stack is written in the form of controller plugins. These plugins can be dynamically loaded and unloaded onto the control stack using a Controller Manager. Controller plugins interact with the robot's hardware via a set of hardware resource interfaces which define the capabilities of each of the robot's sensors and actuators. These interfaces are serviced within a generic control loop feedback mechanism within the robot hardware interface layer. Hardware resource interfaces provided by control include the Adafruit BNO055, Adafruit INA219, BlueRobotics Bar30 and the six BlueRobotics T200.

The vision stack utilizes the ROS nodelet architecture [9] which allows a number of tasks to operate within a single process, without incurring copy costs when passing intraprocess messages. This architecture allows the vision stack to process a large amount of data in the form of images and point clouds without saturating network bandwidth. Much like the control stack, nodelets in the vision stack operate as plugins and can be dynamically loaded and unloaded on to the stack via a Nodelet Manager.

IV. Future Work

The 3D printed motor brackets enable the robot's thrusters to be mounted in a number of different configurations resulting in a variety of thrust patterns. This flexibility comes at a price as any slight adjustment to the motors placement alters its dynamics. Future work on Milton will focus on the development of an automatic calibration procedure that utilizes both visual and inertial sensors to determine the effect of each thruster on the robot's motion.

References

[1] OpenROV. (2015) Openrov — underwater exploration robots — openrov — underwater exploration robots. Accessed 30-April-2017. [Online]. Available: https://www.openrov.com

[2] BlueRobotics. (2015) Bluerov (retired) - bluerobotics. Accessed 30-April-2017. [Online]. Available: https://www.bluerobotics.com/store/retired/bluerov-r1/

[3] BlueRobotics. (2015) Bluerov2 - bluerobotics. Accessed 30-April-2017. [Online]. Available: https://www.bluerobotics.com/store/rov/bluerov2/

[4] NVIDIA. (2015) Jetson TK1 embedded development kit. Accessed 30-April-2017. [Online]. Available: http://www.nvidia.ca/object/jetson-tk1-embedded-dev-kit.html

[5] NVIDIA. (2015) Jetson TK1 design diagram. Accessed 30-April-2017. [Online]. Available: http://developer.download.nvidia.com/embedded/jetson/TK1/docs/242-7R375-1000-D00.Searchable.PDF.of.Assembly.Drawing.pdf

[6] PJRC. (2015) Pjrc store. Accessed 30-April-2017. [Online]. Available: https://www.pjrc.com/store/teensy32.html

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[8] W. Meeussen. (2014) ros_control - ros wiki. Accessed 1-May-2017. [Online]. Available: http://wiki.ros.org/ros_control

[9] T. Foote and R. Rusu. (2014) nodelet - ros wiki. Accessed 1-May-2017. [Online]. Available: http://wiki.ros.org/nodelet