eBug - An Open Robotics Platform for Teaching and Research

Nicholas D’Ademo     Wen Lik Dennis Lui     Wai Ho Li
Y. Ahmet Şekercioğlu     Tom Drummond

\{nick.dademo|dennis.lui|wai.ho.li|ahmet.sekercioglu|tom.drummond\}@monash.edu

Monash University, Australia

Abstract

The eBug is a low-cost and open robotics platform designed for undergraduate teaching and academic research in areas such as multimedia smart sensor networks, distributed control, mobile wireless communication algorithms and swarm robotics. The platform is easy to use, modular and extensible. Thus far, it has been used in a variety of applications, including an eight-hour demonstration managed by undergraduates at the Monash University Open Day. This paper describes both the eBug hardware and software which is in the process of being open-sourced online. The paper also highlights several research applications of the eBug, including its use in a wireless control testbed and as the eyeBug; an eBug that senses the world in real-time using the Microsoft Kinect.

1 Introduction

The eBug was first conceived in 2008 as a platform to involve undergraduates in a research project on wireless control of multiple robots. Since then, the eBug has undergone several iterations, resulting in an extensible platform that has been used in a variety of research applications as detailed in Section 3. The latest version is shown in Figure 1 which highlights the different layers of the eBug. The design is modular, modifiable and extensible in that it allows layers to be both replaced or added depending on the required application. The eBug is intended to be used in several research projects in the near future which are detailed alongside relevant references in Section 4. The hardware design and software sources are available online at the Monash Wireless Sensor and Robot Networks Laboratory Website\(^1\).

\(^1\)http://wsrlab.ecse.monash.edu.au/

Figure 1: e-Bug exploded view
1.1 Similar Platforms

When compared with several other robotic platforms of similar size and capabilities (Table 1), the eBug differentiates itself by firstly being a completely open platform. Here, open refers to the complete design (both hardware and software) being available online to the public for both improvement as well as customization. Furthermore, the relatively simple chassis design constructed of commonly available parts and materials allows the eBug to be replicated much more easily than other robotic platforms. Thus, the eBug is suited to be constructed “in-house” by students as a learning experience at educational institutions wishing to utilize the robot. For this reason, the cost shown for the eBug in Table 1 excludes labour. By way of example, the first author (a final year electrical engineering undergraduate at the time of writing) was able to construct an eBug in approximately 12 hours. This included the soldering of components on both the logic and power layers as well as the construction of the mechanical layer.

Out of the platforms listed in Table 1, the Robomote first produced by the University of Southern California in 2002 is the most similar to the eBug in terms of cost and availability of design files. One main difference is that the eBug uses more modern hardware. It also has the following advantages:

- More advanced communications capabilities (2.4 GHz 250 Kbps XBee vs. 916.5 MHz 19.2 Kbps OOK).
- Designed to traverse a variety of surfaces, including carpet.
- Features on-board lithium-polymer battery charger and supervisor/management circuitry.
- Able to carry additional sensors/peripherals (i.e. expandable).

1.2 Previous eBug Designs

Since 2008, the eBug has undergone several major design iterations (Figure 2):

- **Design 1**: (2008) Used DC motors and Ni-MH batteries.
- **Design 4**: (2010-) Near-complete redesign. Increased number of features. Improved, more efficient, LiPo-based power supply (Figure 1).

2 Design

As shown in Figure 1, the eBug features a modular design which separates the robot into distinct layers (from bottom to top): Mechanical, Power, Logic, and Expansion Board. A modular design such as this has the following advantages:

- If any modifications or improvements need to be made to the Power Layer for example, the Logic Layer does not also need to be changed or redesigned - that is, the Logic Layer does not care how the Power Layer is implemented but only cares that power is provided appropriately.
- Debugging is simplified by the separation of the power and logic sections of the eBug. For example, known-good layers can easily be swapped between robots, thus greatly speeding up the debugging process.
- In the case of hardware damage/faults, repair time is reduced due to the fact that only the faulty layer (as opposed to the entire unit) needs to be replaced.
- If the eBug Logic board is not suitable for a particular application, a custom board with a similar shape can instead be used by appropriately interfacing with the Power Layer.
- Additional layers can be created and added to the eBug to provide extra functionality (see Section 3.2).
2.1 Overview

The following are the major design goals set out prior to the design of the eBug and the corresponding results:

- **Long battery life for long-duration experiments**: The eBug has an approximate running time of 3 hrs during heavy use in a typical application such as described in Section 3.1. It also has a 14-hr idle running time (using a 2500 mAh 3-cell LiPo battery).

- **Ability to travel on various surfaces**: Due to the use of hard rubber wheels with small ground-contact area, the eBug is able to travel on a variety of surfaces.

- **Ability to carry extra weight of additional expansion layers**: The eBug is able to carry additional weight while still retaining mobility. Section 3.2 details an example of a Microsoft Kinect (approximately 500 g) mounted on an eBug.

2.2 Hardware

**Microcontroller**

The eBug utilizes the Atmel AVR XMEGA 8/16-bit microcontroller which is capable of up to 32 MIPS at 3.3 V. Note that the XMEGA 100-pin variants, as used in the eBug, are all pin-to-pin compatible. Several reasons for choosing this particular microcontroller include:

- The XMEGA offers high-performance (typically 1 MIPS/MHz).

- Features a range of useful peripherals (e.g. DMA, Event System, 12-bit ADC/DAC, on-board USB controller) not usually found on typical 8/16-bit microcontrollers such as the AVR MEGA series (as used in the popular Arduino platform).

- Low power consumption - the XMEGA has a maximum current draw of only 20 mA.

- As will be discussed later in this paper, the eBug has not been designed to carry out complex, CPU-intensive tasks. Thus, a higher-performance microcontroller (i.e. 32-bit ARM Cortex-M series) is not required and would unnecessarily increase power consumption.

- Open-source software development tools for the Atmel AVR series of microcontrollers are freely-available (e.g. WinAVR). Furthermore, Atmel also provides a free Integrated Development Environment software package (AVR Studio).

**Power**

As shown in Figure 4, the eBug is powered by a 3-cell (11.1 V) lithium-polymer battery and also features an on-board battery charging circuit which allows the robot to be charged with an external DC power adaptor. Furthermore, protection circuits constantly monitor the lithium-polymer pack to

- Measure and maintain an accurate record of available charge which can also be displayed visually by a set of five on-board LEDs.

- Provide protection against the following events:
  - Cell over and under-voltage
  - Charge and discharge over-current
  - Charge and discharge over-temperature
  - Short-circuit

The majority of the on-board peripherals operate at supply voltages of 5 V and 3.3 V which are provided by a buck switching regulator and an LDO regulator respectively. The eBug can also be completely powered by an external DC power adaptor allowing long periods of use during development and debugging. Another feature of this design is that the unregulated battery supply voltage can also be directly accessed at the Logic Layer. This is useful for add-on boards which may require a higher supply voltage or an unregulated supply (e.g. DC motors, actuators).

**Communication**

Wireless communication capabilities are provided by a low-cost, low-power Digi XBee RF module operating within the ISM 2.4 GHz frequency band. The module has a maximum indoor range of approximately 40 m (line-of-sight) at a transmit power of 2 mW with current consumption ranging from 15 mA to 40 mA depending on whether the device is in the idle or transmit/receive state. The module features two modes of operation:

**Transparent Operation**: Module acts as a serial line replacement.
**API Operation:** Allows the full functionality of the module to be used (e.g. ZigBee addressing support, remote configuration, remote I/O pin sampling, mesh networking).

The eBug utilizes the XBee module in the API mode of operation in order to allow the use of the above-mentioned advanced features. API mode also offers the higher performance of the two modes - a maximum data throughput of 35 Kbps. Furthermore, the XBee module can also be put into sleep mode to lower power consumption when not needed.

**Sensors and Peripherals**
The eBug features a range of sensors and peripherals as shown in Figure 5:

- **Analog temperature sensor:** Enables measurement of temperatures from 2°C to 150°C with up to 0.5°C accuracy.
- **Analog light sensor:** Enables measurement of illuminances up to 100,000 lux via an adjustable software gain setting.
- **Speaker:** Driven by a 1 W mono BTL audio power amplifier. The amplifier can be shutdown when not needed to reduce power consumption.
- **Liquid crystal display:** 2x16 character LCD (with software-adjustable intensity backlight) outputs various run-time information such as CPU usage, test status, battery/supply voltage(s), remaining battery capacity etc.
- **RGB LEDs:** 16 RGB LEDs driven by a two cascaded 24-channel 12-bit PWM LED drivers allow the generation of almost any colour for visual feedback purposes. These LEDs are located in equispaced positions on the edge of the eBug.
- **Infrared (IR) receivers/proximity sensors:** IR receivers or alternatively, proximity sensors (or both) can be installed in 8 equispaced positions on the edge of the eBug depending on the required application.
- **Infrared emitters:** 16 IR emitters also equispaced on the edge of the eBug (but at intervals between the aforementioned IR receivers/proximity sensors) can be used for communication (via modulation) or basic obstacle detection when used in combination with the infrared receivers or proximity sensors, respectively.
- **Stepper motors:** Two 10 V 1.8°/step 35 mm hybrid bipolar stepper motors provide the eBug with mobility. The on-board stepper motor drivers are capable of full, as well as 1/2, 1/4, 1/8, and 1/16 (micro-)step modes, thus allowing the eBug to move smoothly even at low speeds.

**Expansion Header**
Additional functionality can be easily added to the eBug via the Expansion Header which provides power (5 V and 3.3 V) and communication interfaces (such as UART and SPI).
Figure 7: Positions of the major peripherals on the eBug Logic Layer

3.3 V) as well as direct access to the XMEGA and XBee I/O pins. Figure 1 shows an example of an expansion board created to provide USB connectivity for debugging purposes. In a similar way, expansion boards can be created to allow the use of sensors/peripherals not already present on the eBug. Entire boards can even be created which add an extra layer to the eBug and provide space for more advanced hardware such as image sensors, high-power microcontrollers etc. - the eyeBug in Figure 13 is an example of this.

2.3 Software

The embedded software (firmware) has been designed to allow the eBug to act as a “slave” device which receives basic commands and then carries out the corresponding function. These basic commands include those which, for example, turn on a particular LED, read the temperature sensor, reset the device etc. Consequently, the firmware uses an event-driven architecture consisting of high-level drivers for each of the on-board peripherals in combination with a real-time embedded operating system (RTOS). The use of a real-time operating system allowed the major functions of the eBug to be divided into separate tasks which simplified the software design. Furthermore, the use of a RTOS also allows the eBug to react both quickly and predictably to received commands. The structure of the eBug firmware is shown in Figure 8.

Real-Time Operating System

The eBug uses Micrium’s $\mu$C/OS-II real-time preemptive embedded operating system [Labrosse, 2011] which can be distributed freely without a license in non-commercial (educational) applications. Major factors in choosing this operating system over other alternatives included:

- $\mu$C/OS-II is used as a real-time systems teaching tool in undergraduate studies at Monash University and thus many students are already familiar with the operating system.
- $\mu$C/OS-II is also used widely in other educational institutes around the world.
- Detailed documentation concerning both the operation and usage of $\mu$C/OS-II is readily available.

Application Programming Interface (API)

The entire functionality of the eBug can be accessed via a simply-to-use API which allows rapid application design and implementation. In other words, the eBug can be controlled simply by sending data (in a particular format) to either the XBee RF module or Expansion Header - no modification or reprogramming of the eBug firmware is necessary. This is one of the most powerful features of the eBug in that it allows non-roboticists (or those unfamiliar with the AVR architecture or embedded software in general) to access the full capabilities of.
the robot by simply using this high-level API. Figure 10 shows an example usage of the eBug API to wirelessly read the value of the temperature sensor connected to the ADC.

3 Applications

3.1 Wireless Control Testbed

The first application of the eBug was to create a testbed which not only demonstrated control of multiple robots over a wireless link (using a central computer, i.e. centralized control), but also served as a platform to test various formation control algorithms using real hardware. While tracking robots using augmented-reality (AR) markers has been demonstrated before [Fiala, 2004], the main focus of our application is to create a reliable testbed. As shown graphically in Figure 11, the testbed functions as a closed-loop control system as follows:

1. Frame is grabbed from an overhead USB camera.
2. Software running on a nearby PC uses ARToolKit-Plus \(^4\) to firstly search the frame for any BCH-coded markers (which are located on top of each eBug). The position and orientation of each eBug in the frame is then calculated.
3. The appropriate control command (in the form of an eBug API packet) is then calculated for each eBug depending on the desired control behaviour.

4. This command is then finally sent wirelessly (using an XBee module connected to the PC) to each eBug. Process repeats from Step 1.

A simple control algorithm was successfully implemented on the testbed which directs each eBug toward a random target position without collisions with other robots. The testbed captures frames at 640x480 resolution at 60 FPS (frames per second) with control commands sent to each eBug at 100 ms intervals. Furthermore, as the control software is modular by design, more complex control algorithms can easily be run on the testbed. A video demonstrating the testbed is available as a multimedia attachment to the paper. It can also be viewed online\(^1\).

3.2 eyeBug: eBug with RGB-D Sensing

Figure 13 shows the eyeBug, a real-world example of how more powerful capabilities and features can be easily added to the eBug by utilizing its expandable design. The eyeBug is an eBug extended with a Microsoft Kinect\(^5\) sensor and a BeagleBoard-xM\(^6\):

- The Microsoft Kinect is a low-cost RGB-D sensor that provides colour and depth images at 30Hz through a USB interface. It has rapidly gained a growing user base in the robotics community as it is both cheap (AUS$150) and has a large community of open-source software developers. Example images taken by the eyeBug’s Kinect are shown in Figure 14.
- The BeagleBoard-xM is a low-cost ARM Cortex-A8 development board (AUS$150) which includes USB 2.0 as well as many other commonly-used interfaces (e.g UART, SPI, I2C, Ethernet).

The Kinect communicates with the BeagleBoard-xM via USB and a custom PCB designed to interface the BeagleBoard’s UART to the eBug allows direct control of the robot. This Kinect Layer is placed above the eBug Logic Layer and includes boost converter circuitry to supply

\(^4\)http://handheldar.icg.tugraz.at/

\(^5\)http://www.microsoft.com/en-us/kinectforwindows

\(^6\)http://beagleboard.org
the 10 V needed to power the Kinect. Furthermore, the board also features a dedicated area for prototyping which provides access to 3.3 V-tolerant BeagleBoard-xM GPIO lines.

Preliminary results show that the eyeBug is able to perform a pseudo-random walk around an arena while avoiding obstacles. The Kinect is used to sense obstacles so that the robot always turns away from the nearest-sensed obstacle. The algorithm running on the BeagleBoard-xM (with an embedded version of Ubuntu 11.04) is implemented in C++ and makes use of OpenCV\(^8\) and the libfreenect\(^9\) drivers. The unoptimised implementation of our algorithm runs in real-time at 12 FPS on the eyeBug. Additional technical details can be accessed online\(^1\).

A video of the eyeBug is available as a multimedia attachment to the paper. It can also be viewed online\(^3\).

### 3.3 Performance of “Follow the Leader” Formation Control Algorithms

The eBugs were also used in a project which tested the performance of decentralized algorithms for formation control under noisy localization information and constrained communication conditions. Decentralized control involves the robots performing tasks without an external agent coordinating or directing the individual robots. This project’s focus was to control formations of eBugs with two competing objectives:

1. Move all eBugs from their initial position to a final destination through a path of travel (an arbitrarily defined trajectory).
2. Maintain the relative distance between the eBugs such that a set formation is retained during their travel [Lawton \textit{et al.}, 2003].

The technique of reconciling those two competing objectives defines different formation control algorithms.

In 2003, Lawton and his team published a seminal paper [Lawton \textit{et al.}, 2003] and proposed three control laws for decentralized formation control under progressively more rigorous constraints. Using three prototype eBugs in our research project, we began implementing these control strategies to quantify their robustness against localization estimation inaccuracies and the effects of robot-to-robot communication channel delays and bandwidth limits.

Early results obtained for one-leader, two-follower formations were published in a technical report [Siripala, 2010]. We are currently bringing more eBugs online for more comprehensive experiments with larger formations.
involving more complex communication and sensing scenarios.

4 Conclusions and Future Work

This paper presents the design details of a low-cost open robotics platform, the eBug, which is modular, modifiable and extensible. Moreover, applications of the platform have clearly demonstrated its ease of use; even for non-experts. We are currently expanding our number of eBugs to 30 robots. This will allow us to tackle a variety of research projects by validating theoretical models with real world implementations. Planned research projects include networked control systems [Abdallah and Tanner, 2007; Antsaklis and Baillieul, 2007; Sinopoli et al., 2003; Stubbs et al., 2006] over wireless networks [Kumar, 2001] and networked robotics [Kim et al., 2009; Pohjola et al., 2009; Jung et al., 2010], communication algorithms in mobile wireless sensor networks [Ekiç et al., 2006], data harvesting from sensor fields by using mobile robots [Tekdas et al., 2009; Gu et al., 2006], and formation control for robotic swarms [Anderson et al., 2008; Lawton et al., 2003; Tanner et al., 2004; Ren et al., 2007].

Acknowledgments

The authors would like to acknowledge the following people who have contributed in making the eBug possible:

- David McKechnie who provided the initial eBug concept and design in 2008.
- Aidan Galt and Rory Paltridge who both designed and produced early eBug iterations in 2008 and 2009 respectively.
- Alexandre Proust for all his work with the Kinect-related software development.
- Tony Brosinsky for his assistance in the production of the eBug mechanical layer.
- Ray Cooper, Ian Reynolds, Geoff Binns, and Raymond Chapman for their constant technical and logistical support.
- The Department of Electrical and Computer Systems Engineering for their financial support.

References


