A Remote Laboratory for learning embedded systems & control

Stephen J. Weddell, Philip J. Bones and Nicholas M. Wareing
Department of Electrical & Computer Engineering, University of Canterbury, Christchurch, New Zealand
Corresponding Author's Email: steve.weddell@canterbury.ac.nz

Structured Abstract

BACKGROUND
Provision of a laboratory environment that allows students fair and convenient access to a complex electro-mechanical apparatus, such as a model helicopter, which can be programmed, "flown" using virtual controls, and assessed entirely over the Internet, was our primary motivation. To achieve this our system was designed to be sufficiently robust to withstand a wide range of control parameters, while providing a high degree of protection against mechanical and electrical failure. These goals were met through a series of projects that have been conducted in cooperation with both technical and academic staff, project students, and in close collaboration with another university. Our ultimate goal is to provide an interesting and challenging remote laboratory that can be shared by other academic institutions throughout Australasia, final year high-school students, and open-day demonstrations.

PURPOSE
A key question we posed when preparing our third year computer engineering course was, "can we improve and motivate student learning by providing a capstone embedded system project that would be suitable for students from several degree streams, base laboratory resources entirely on student demand, and challenge students to learn the complexities of real-time embedded systems and control through interaction over a safe and highly flexible environment"?

DESIGN/METHOD
A model helicopter with a pair of DC motors formed the basis of this project, where height and yaw sensors provided closed-loop feedback to control stable assent, maneuvering and descent, thus defining a repeatable "flight path" over a web interface. Students gain experience by using the C programming language to read sensor data into an ARM microprocessor and drive both main and tail rotors using a proportional, integral, and differential (PID) algorithm to maintain stability. A separate (to student requirements) power supply and monitoring system protects each helicopter rig (heli-rig) from over-use. For example, temperature sensors on both the main and tail rotor motors provide a thermal cut-out mechanism, and feedback to students is provided through a series of LEDs viewed over an internet browser during each pre-booked, remote-lab session. Morse coded messages communicate system status on each of the three helicopter rigs (heli-rigs) that comprise our remote laboratory.

RESULTS
Student appraisal has been collated through surveys and group discussions, and this has been positive. This remote laboratory has allowed us to satisfy several learning outcomes. Firstly, students should enhance their design skills through team development of an interesting and reasonably complex project. Secondly, students incorporated a control algorithm that was robust for deployment on any of the three rigs provided. Thirdly, close cooperation and teamwork were required to complete the project. Lastly, several project milestones were introduced. This helped students manage their time by spreading the assessment evenly over the duration of the course.

CONCLUSIONS
The initial results of this capstone project have been compelling. As a result, we will continue to develop our heli-rig project by enhancing features using a remote laboratory interface. Student groups work within design constraints built into each heli-rig. It is anticipated that this work will extend to other engineering departments as they introduce experiments based on this platform.

KEYWORDS
Remote laboratories; embedded systems.

Introduction

The central premise of remote labs is that universities around the world can install and host physical experiments, which can be accessed by students over the Internet (Aktan et al., 1996; Jona et al.; 2011; Riman et al.; 2011; Ursutiu et al., 2013). The idea holds great promise because it can enhance collaboration between universities (sharing of expensive lab resources between universities) Ordula et al., (2013), and moreover, allow students to work from wherever they are and at a convenient time of day. This can help reduce demand for expensive lab resources and software (SAHARA, 2012) and is able to ensure save use of equipment during development, testing, and demonstration phases.

In 2012 the Department of Electrical and Computer Engineering at the University of Canterbury offered an embedded systems course, ENCE361. One of the projects, "Fun with Avionics", required students to write an embedded program to control the flight of a small model helicopter. The helicopter was fixed in a custom built stand, which allowed it to rise, fall, and turn through two degrees of freedom. Students purchased a Texas Instrument's Stellaris development kit that comprised a Cortex-M3 microcontroller board, a development platform for compiling and programming, and an integrated debugger. Sensors on the heli-rig output signals provided yaw and altitude data. Students were required to control both main and tail rotors using pulse width modulation (PWM) signals.

This assignment proved to be both challenging and interesting for students. It provides a platform to apply embedded programming concepts such as interrupts, implementing proportional, integral, derivative (PID) control, and to develop a rudimentary real-time operating system (RTOS). When introduced in 2012, not all students were able to gain equal access to the laboratory equipment. However, recent work by Wareing *et al*. (2013) provided the basis to build a suite of three helicopter rigs (heli-rigs), and this project was reintroduced into our 2014 course. This paper outlines the challenges associated with building and maintaining a heli-rig suite, as shown in Figure 1, and also highlights the pedagogical benefits resulting from this work.

Figure 1: The University of Canterbury's Heli-Rig suite.

Remote laboratories

Remote laboratories by definition do not allow students to physically interact with equipment or apparatus used for an experiment in a laboratory setting. Thus, they are more secure,

reducing problems associated with theft, vandalism and damage to equipment through inappropriate use. Student safety is also improved through physical isolation from hazardous equipment. These factors may have economic benefits, in the form of reduced occupational health and safety (OSH) compliance costs and reduced insurance premiums.

Until recently, the technological challenges involved in real-time Internet communication have limited the growth of remote laboratories. However, recent improvements and innovations in web technologies have rapidly begun to remove these impediments. The University of Technology Sydney (UTS) has had extensive experience implementing fully featured remote laboratories for project and course work.

An Avionics-Based Remote Laboratory

Adaptation of SAHARA for our heli-rig has enabled us to focus on the rig-client, i.e., the electromechanical assembly that comprises a model helicopter, sensors, power supply, and protection mechanisms. One of the difficulties students originally encountered when writing and testing programs in the lab, was not to damage the helicopter when the control system code was still in the testing phase. This often involved physically holding the helicopter to stop it swinging wildly from side to side; overcoming such a testing regime proved challenging for a remote lab environment however this was achieved using a “maintenance” embedded system which was available only to system developers and technicians.

Damage to the helicopter was limited to rotational components, such as motors and spindles. Since more of this damage occurred during sustained full-power sessions, a monitoring system that timed the period of motors running at 80% PWM duty cycle was incorporated. Additionally, the temperature of each motor was periodically sampled, and a combination of both parameters provided protection to extend the meantime failure of motors. Lastly, student groups could book only one session at a time, where a login and initialisation period allowed overheated components to cool (somewhat) before a new session. A signaling system was developed to provide students with feedback, as defined in their lab instructions, if certain limits were exceeded.

System Design

Several tools have been created to assist with the development of remote laboratories. MIT’s iLab (2014) and LAB2GO (Zutin, 2010) are two such examples. We have chosen the open source SAHARA labs framework (2012). This is a generic platform for designing customised remote laboratories and is developed and maintained by a team at the University of Technology Sydney. SAHARA consists of three separate components:

1. Web interface.
2. Scheduling server.
3. Rig client.

The interactions between these components, and with incorporation of our heli-rig client, are shown in Figure 2.

The Web Interface

The web interface is a platform comprising a series of webpages with controls and a video window that allow student groups to make a booking, and when available, interact with the experiment. Several enhancements to the original prototype developed by Wareing *et al.,* (2013) have been made. These include:

1. A set of yaw buttons has been added. This provides a second degree of freedom (DOF) by supporting a rotational component between +45 deg. and –45 deg.
2. A bright LED is visible within the video frame. Flashing sequences of Morse code provide students with feedback indicating the reason if power to the heli-rig is cut,

e.g., excessive over-driving, motor over-heading, etc.

3. A maintenance embedded system has been incorporated to reduce damage.
4. A support button to report operational irregularities directly to the support team.

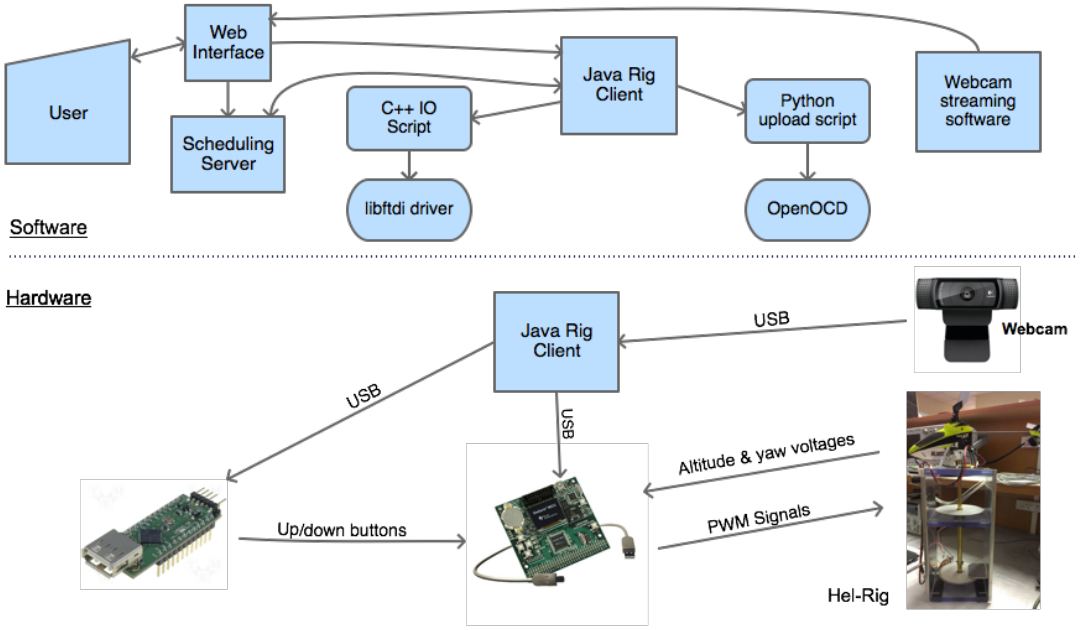

Figure 2: Interaction between key Sahara components with client (heli-rig) hardware adaptations.

The web interface screen for our current heli-rig is shown in Figure 3. Note that students have the choice of either using a demonstration program, or uploading their own program. Required maneuvers include *k* vertical steps using the “Up” and “Down” buttons, and at any vertical step, rotate *n* steps either clockwise (CW) or counter-clockwise (CCW), where $k = 1 \ldots 10$ and $-8 \le n \le 8$ representing 5 mm and 15 deg. increments (per button press), respectively. Continuous real-time video is provided through a selection of video formats, and an “Operations List” confirms each command.

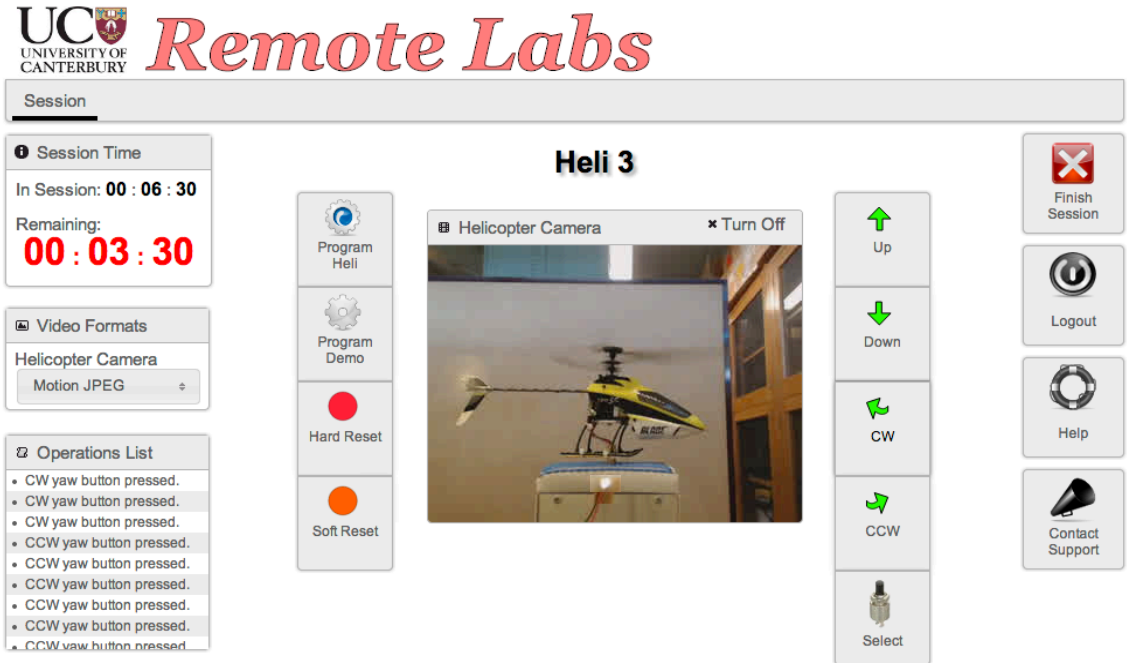

Figure 3: The web interface showing heli-rig client #3: helicopter, pedestal and feedback LED.

The Scheduling Server

The scheduling server forms the core of SAHARA. It is written in Java and needs to connect to a MySQL database. The scheduling server provides a web interface that lab technicians and lecturers can use to allocate access to specific rigs or entire classes of rigs. Our heli-lab required customization that allowed students to: i) log into SAHARA using our LDAP server, ii) select one of three heli-rigs, iii) book a session for a specific day and time, iv) join a queue of waiting users. SAHARA provides users with feedback in the form of a graphic, showing blocks of previously booked sessions, and available booking time-slots on a rig-by-rig basis. Both students and staff quickly realized that queuing (iv), is immediately displaced in preference to a booking (iii). Therefore, given the benefit of hindsight, queuing will not be made available in our next project offering.

The Rig Client

Three Rig Client control types are provided by SAHARA (2012):

1. Peripheral Control: A rig has a controller that is 'outside' of SAHARA.
2. Primitive Control: The rig is controlled directly by the Rig Client.
3. Batch Control: The rig operates using batch instructions uploaded by the user. No other intervention is required.

For our purposes, the primitive control rig is most suitable, because ongoing user interaction is required in the form of button presses. Conveniently, this is the same control type used by a programmable logic rig, already developed by UTS. Both rigs required a student to upload compiled code to a remote target device.

Additionally, this allows us to use the SAHARA web interface, rather than a Remote Desktop Session. Such a user-friendly solution offers additional security because no further ports (other than Port 80) need to be opened on the lab server.

The Rig Client is coded in Java, and utilises several third party libraries. In order to design a customised rig, a developer creates a new Java class that extends the *AbstractControlledRig* class. The Rig Client program loads in this class at runtime (after specifying its presence in a configuration file). The *HeliRig* class has four main functions:

1. A facility to take a compiled student program from its temporary upload directory (or the demo program) and flash a Stellaris microcontroller residing on each rig.
2. A facility to channel virtual button presses to specific pins of a USB to serial device. A second USB port is used for this and an FTDI (2013) chip is used to interconnect to Stellaris IO ports that are used for actual button inputs.
3. Reset functions which the Rig Client runs after a student finishes using the rig; this ensures the rig is prepared for the next user. For example, students develop and program using a microcontroller board that they purchase. Their compiled "bit" file is uploaded via SAHARA, essentially replacing a previously uploaded program on an identical microcontroller board on a rig. The rig microcontroller resides in a low power state whilst not in use.
4. Test functions run periodically by the Rig Client to determine if the rig remains in a working order.

Discussion

Latency is one issue that had the potential to cause problems with this project. We have tested the rig (hosted at the University of Canterbury in Christchurch, New Zealand) from Canberra, Australia, and have found the delay in the video to be acceptable, at approximately one second.

Currently, a visual webcam stream is the only means by which students can assess performance of their program while running on a heli-rig. Cardoso *et al.,* (2012) highlighted a number of issues with this. Firstly, no debugging information, in terms of program execution, is available to students. Secondly, if random or poorly chosen PID control parameters are used, the helicopter may run for an extended periods at both height and yaw stops; worse still, the helicopter can oscillate between each stop. Despite system safeguards that have already been discussed, the mean-time failure of key components increases. Therefore, feedback, at least in the form of yaw and altitude data, is required.

We have attempted to image the Stellaris microcontroller display within the field-of-few of the camera. However, considering the size of the display with respect to the wide-field requirements to view helicopter “flight”, this was not possible. An alternative method is to use the second serial port (alternate to the debug port) of the Stellaris device to send sensor data back via the web interface in a text stream.

Pedagogical Considerations

A set of clearly defined learning outcomes is essential for an course and provides students with: i) an insight into what they will learn, ii) a means to assess how well topics and concepts were both taught and absorbed, and iii) a list of goals that provide a clear indication of required expectations and competency level, for which assessment can be based.

There are several learning outcomes defined for our third year embedded systems course, ENCE361. There include:

1. Build on a first course in microcontrollers to design, test, and debug an embedded system from a project specification.
2. Enhance design skills by utilising internal microprocessor peripherals, such as timers, serial interfaces, and analogue-to-digital converters to build an embedded system.
3. Learn to implement an algorithm in the C programming language.
4. Understand what is meant by the hardware and software interface and the constraints of a real-time embedded application.
5. Write well structured code for the development of software modules to run on an advanced microprocessor.
6. Demonstrate competency in utilising an advanced commercial tool-chain to develop an embedded application within a team environment.
7. Learn the basics of scheduling theory and apply this to an embedded system with real-time constraints.

There is at least anecdotal evidence that our heli-rig remote laboratory reinforced each of these learning outcomes, especially given that students had the convenience to complete this work wherever and whenever they so desired. It is important to note that scheduled laboratory time was still available to students, where qualified teaching staff and teaching assists were on-hand. However, these labs were more like “drop-in” sessions, where students groups could meet, discuss their project, and then complete work each had agreed to do before the next session. Furthermore, code repositories were created for each group so commitments could be made individually and merged when groups resumed their work as a team.

In addition to incremental feedback from students in the laboratory, comments were posted to the project forum on the course webpage and interviews were conducted with class reps before key milestones was assessed. Students were asked to complete a questionnaire about the project and rate from 0 to 5 six specific statements concerning this project, and provide an optional, general comment. Of the 106 students enrolled in ENCE361, 54

students responded. The results of this survey are presented in Figure 4.

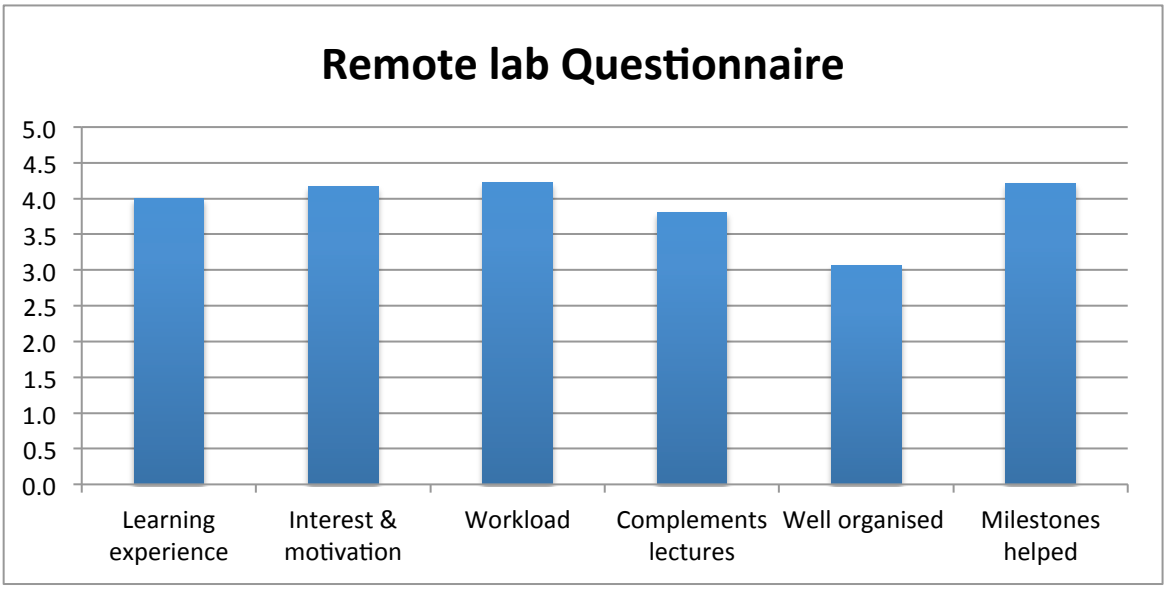

Figure 4: Student feedback to our remote laboratory project.

In evaluating Figure 4, scores of over 4.0 in our institution are considered very acceptable. This suggests most students were interested and motivated by this project. However, the workload was high. We intend to address this by replacing a second data acquisition project with a set of lab exercises related to the remote lab, thus extending the project work from 6 weeks to 9 weeks of the 12-week course.

In addition, students acknowledged that the setting of several milestones, which included simple tests in the lab, helped them in managing the completion of this project. In fact, all 53 groups of 2 students completed this project. Also, students appreciated the learning experience and agreed this project complemented formal lecture material. This requirement helped ensure the successful application of their PID algorithm. The lower score for the "Well organised" category can be explained by the teething problems experienced in configuring the booking system. On several occasions, patches in software and organizational changes had to be made in order to compensate for mechanical wear and server issues. Groups were provided with additional access time to help compensate for this, however these disruptions did not go unnoticed. Refinements are being implemented for future years and are discussed in the concluding section of this paper.

Lastly, slight differences in the mechanical alignment of sensors and wear-and-tear of components specific to each rig meant that students were faced with a “real-world” question: is it possible to ensure consistent avionic maneuvers, irrespective of rig? Even though students felt this was a frustrating and unacceptable consequence of their “perfect” code, having to derive parameters for their PID algorithm to satisfy this requirement as part of their final demonstration, proved to be an invaluable, and undocumented, learning outcome.

Conclusions & Future Work

Our purpose for developing “Fun with Avionics” as a remote laboratory project was to reinforce the learning outcomes defined for this course. Given the positive feedback from the majority of students interviewed and how many took part in our survey, we believe this was a highly successful outcome.

We have several proposals for future work on this project. Firstly, we will continue to improve on the robustness of our heli-rigs. This is particularly important since ENCE361 is

only offered in one semester; our 'off-season' used for repair and refinement will be minimised as other institutions "come on line", and routine maintenance must be kept to a minimum. Secondly, some student groups admitted to "beating the system", in terms of extending group allocation time, and by various means. This will be addressed in future by refinements to the scheduler. Thirdly, as outlined in our discussion, providing students with sensor data through their web interface is a priority. Lastly, we plan to build another 6 heli-rigs, supporting and maintaining at least 6 in service and provide students access on a 24/7 basis.

References

Books

Kazmierkowski, M.P., "Using Remote Labs in Education (Zubia, J.G. and Alves, G.R.; 2011) [Book News]," Industrial Electronics Magazine, IEEE , vol.7, no.1, pp.67,68, March 2013.

Journal article

Aktan, B., Bohus, C., Crowl, L., and Shor, M., (1996) "Distance learning applied to control engineering laboratories," *Education, IEEE Transactions on*, Vol. 39, No. 3, pp. 320–326.

Cardoso, A., Vieira, M., and Gil, P. (2012), "A remote and virtual lab with experiments for secondary education, engineering and lifelong learning courses," *International Journal of Online Engineering (iJOE)*, vol. 8, no. S2, pp. 49–54.

Jona, K., Roque, R., Skolnik, J., Uttal, D., and Rapp, D., (2011) "Are remote labs worth the cost? Insights from a study of student perceptions of remote labs", *International Journal of Online Engineering (iJOE)*, Vol. 7, No. 2, pp. 48–53.

Ordua, P., Rodriguez-Gil, L., Lpez-de Ipia, D., and Garcia-Zubia, J. (2013), "Sharing remote labs: A case study." *International Journal of Online Engineering*, vol. 9, pp. 26 – 27.

Riman, C.F., El Hajj, A., and Mougharbel, I., (2011) "A remote lab experiments improved model," *International Journal of Online Engineering (iJOE)*, vol. 7, no. 1, pp. 37–39.

Conference proceedings

Hanson, B., Culmer, P., Gallagher, J., Page, K., Read, E., Weightman, A., and Levesley, M. (2008). "Remote Laboratories in the Curriculum." Measurement , 614, 056-160.

Ursutiu, D., Samoila, C., and Dabacan, M., (2013) "Cross platform methods in digital electronics engineering education," in *Remote Engineering and Virtual Instrumentation (REV), 2013*, *10th International Conference on*.

Wareing, N.M., Bones, P.J., Weddell, S.J. (2013), "A remote lab implementation using SAHARA", Electronics Conference New Zealand (ENZCon), 5-6 Sep 2013. In ENZCon Conference Proceedings, 51-56.

Zutin, D., Auer, M., Maier, C., and Niederstatter, M., (2010), \Lab2go: A repository to locate educational online laboratories," in *Education Engineering*, IEEE.

Online source

Sahara (2012), UTS Remote Labs Installation Guide Rev. 3, Retrieved March 20, 2012, from http://sourceforge.net.projects/labshare-sahara/

iLabs (2014), Massachusetts Institute of Technology, Retrieved June 22, 2014, from http://icampus.mit.edu/projects/ilabs/

FTDI (2013), "FT245R USB FIFO IC datasheet version 2.12", Retrieved February 10, 2012, from http://www.ftdichip.com/Products/ICs/FT245R.htm.

Acknowledgements

We would like to thank David van Leeuwen, Philipp Hof, and David Healy for their work in building the heli-rigs used for this project and for making them (mostly) "student-proof". The authors would also like to acknowledge Michael Diponio from the University of Technology Sydney for his assistance in clarifying some of more subtle features supported by SAHARA. Their encouragement and helpful hints in tracking down faults were invaluable.

Copyright statement

The following copyright statement should be included at the end of your paper. Substitute authors‟ names in final (camera ready) version only.