AIAA Infotech@Aerospace 2010
20 - 22 April 2010, Atlanta, Georgia

AIAA 2010-3394

A Low-Cost Manipulator for Space Research and Undergraduate Engineering Education

Catharine L. R. McGhan[1] and Ella M. Atkins[2]
University of Michigan, Ann Arbor, Michigan, 48109

This paper describes a supervised undergraduate design-build-test effort in which young students supported by the University of Michigan's Undergraduate Research Opportunities Program (UROP) were able to design, build, test, and extend a robotic manipulator of dimensions comparable to those of a human arm. This project had two goals: to expose students to the design-build-test process in the context of robotic electrical, mechanical, and software systems, and to construct an artifact, the manipulator, for the subsequent support of graduate-level research in human-robot collaboration pursued by the first author. This manipulator system was low-cost, required little prior knowledge of machining, and was required to operate robustly and safely in an environment shared with its human collaborator. In this paper, we describe the design-build-test process from the pedagogical and system engineering perspectives. Ultimately, the students focused their efforts on research in sensing, software, and mathematical modeling of the manipulator, emphasizing the fact that hardware DBT is only the first step to a fully-operational robotic system. This was a key observation not made by many Aerospace students, even those exposed to DBT activities that typically highlight physical vehicle design while "black-boxing" sensors and software. This paper describes the process by which this group was educated in the context of the manipulator DBT process. We briefly discuss subsequent manipulator use and extension both for graduate research and ongoing undergraduate DBT exposure.

I. Introduction

As small, low-cost electromechanical parts components become widespread, Aerospace robotics education has grown to include more practical, hands-on design-built-test activities. This has put educators in the unique position of being able to utilize physical systems to demonstrate math, physics, and practical engineering concepts in full system embodiments, creating a more interactive and enriching experience for students in and out of the classroom. Generally, educational robotics platforms for Aerospace have focused on unmanned aircraft systems (UAS) and small payloads launched by weather balloon. Mobile robotic (rover) and manipulator systems are used more broadly, with connection to Aerospace applications such as extra-terrestrial planetary surface exploration. Currently, robotics in engineering courses introduce kinematics and dynamics mathematical models and in some cases include group projects accomplished in a few weeks near the end of the course.[1,2,3,4,5,6] A number of engineering programs support design-build-test (DBT) experiences through student teams. Courses such as "introduction to engineering" at the freshman level[7] or alternatives to more traditional capstone design courses have also been developed. With large class sizes and limited student contact hours, entry-level courses focus on specific pedagogical objectives with components and final product properties mostly specified in advance. Senior-level design/build-test courses and team-based projects can be more ambitious, often requiring up to an academic year to complete, but in Aerospace such efforts have largely focused on mechanical, structural, and aerodynamics subjects, using COTS electronics and software to the extent possible.[3,6,8,9,10] Entry-level kits extensible to the classroom are low-cost, in the range of $70 to $200 with more sophisticated systems at higher cost.[1,8,10] Challenging senior-level projects typically have student teams ranging in size from 2-3 students for small assignments to 20+ students in large teams for longer-term projects. Even at the graduate level, students may have had little experience with DBT thus may benefit by hands-on experiments both to personally gain exposure to practical systems and in some cases to help validate their research. Undergraduate or graduate students that have substantial experience with DBT can

[1] PhD Candidate, Aerospace Engineering Dept., University of Michigan, Ann Arbor, MI 48109, Student Member.
[2] Associate Professor, Aerospace Engineering Dept., University of Michigan, Ann Arbor, MI 48109, Associate Fellow.

Copyright © 2010 by Catharine L. R. McGhan and Ella M. Atkins. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

also mentor students with less experience, enabling the mentor to gain valuable experience in teaching and collaborating as well as reinforcing their own DBT skills. Ideally, the less experienced students can assist the graduate student in tasks or in constructing a platform relevant to his/her research; such was a goal of the work presented in this paper.

A. Motivation for a "Safe Manipulator"

Robotic manipulation requires a combination of mechanical, electrical, and software systems. As summarized below, the authors had an interest in both human-robot collaboration and in mentoring young students in a DBT project that might also benefit the research effort. Given the majority of engineering DBT projects that emphasize hardware design and achieve functionality through black-boxed electronics and embedded software, the authors focused on devising a project that would highlight and emphasize the electronics and software engineering side of DBT but in a robotics system context. The authors had previously used mobile robots to navigate in an environment shared with humans; much less attention had been paid to shared environments in which humans and robots were moving their arms to manipulate objects in their environment. This project therefore evolved to focus on a manipulator system conceived to address these educational and research concerns.

Most commercial off-the-shelf (COTS) manipulator systems can apply too much torque to be safe given that command errors can sometimes occur and that COTS systems necessarily are designed to grasp and manipulate parts of nontrivial mass. Most manipulator systems also do not move in ways easily predictable by a human, a requirement for our research. Thus, a prerequisite to our planned human subject testing was to design, build, and validate a robotic hardware platform – a manipulator system – capable of operating in a shared environment. We hypothesized that a manipulator with human-like dimensions and motion ranges would interact most naturally with human subjects. Thus, to support human-robot collaboration testing, primary requirements for the manipulator system were that it must be a safe reliable system for physically-proximal interaction, it must have approximately the same sizing and motion characteristics of a human arm, and it must support research into the optimization of human-robot mission performance. Secondary requirements were that it must support or at least allow undergraduate learning in the DBT phase, and support future research and educational opportunities in the long-term, requirements that are shared with most research projects in academia. Some examples of this would be extensions to the physical system (e.g., addition of a manipulator gripper), sensor system (direct perception or motion tracking of humans), control (force feedback, path planning, etc.), and processing/networking upgrades. Such diverse use of the platform maximizes versatility for research and education without necessarily requiring higher development cost.

B. Research: Human-Robot Collaboration

The authors are pursuing research in the modeling of human intent in the context of physical human-robot collaboration. Human subject experiments are required to qualitatively and quantitatively assess and validate the true safety, efficiency, and workload associated with physically-proximal activities. Task sharing and interruption are key factors to investigate, and ideally both the cognitive and physical processes of human subjects would be engaged during testing. In a closed workspace, interaction through manipulation enables the definition of tests common to human and robot. Our primary graduate research objective has been to optimize human-robot mission performance such that the direct supervision of the robot's abilities by a human collaborator or operator is unnecessary.[11] This ultimately requires that the robot sense and identify the human's action trajectories, match them to tasks, and predict intent – translated to the human's most likely future actions and anticipated schedule of their execution – to minimize the likelihood of physically interfering with the human's arm movements or task-focused visual gaze. This also necessitates that the robot be able to sense and react safely and efficiently to the human in cases where the human does not move as anticipated. The manipulator system was designed based on requirements motivated by this human-collaboration research effort.

C. Education: Design, Build, Test (DBT)

With dual objectives – to support both research and education – we conducted a full design, build, test cycle for a "safe manipulator" with dimensions and movements comparable to those of a human arm. A graduate student, the first author of this paper, was responsible for specifying the design requirements, then teaching and directing the undergraduate DBT team. The graduate student provided guidance and feedback throughout the project to present technical background as appropriate, guide activities, and ensure the undergraduates followed a reasonable path to their own work and to the collaborative (integration) project tasks. The undergraduate DBT team was identified from the pool of freshmen and sophomore engineering undergraduates accepted into the University of Michigan's campus-wide Undergraduate Research Opportunities Program (UROP) (http://www.lsa.umich.edu/urop/).

D. Project goals

Prior to inception, goals were defined for the graduate and undergraduate students involved with this UROP-supported DBT project. The graduate student's goals were twofold: to design, build, and test a manipulator system useful for future research, and to learn about the process associated with educating a team of young undergraduates through a full design, build, test cycle. The graduate student was responsible for supervising the students throughout the project, while also collaborating with the students and assuming additional responsibilities as needed to ensure the project was a success. The goals for the undergraduate students were also dual: to acquire knowledge and skill associated with manipulator DBT, including hardware identification and manufacturing, computing and power system integration, and software DBT. As a fully-integrated system, the students would also be exposed to the complex interactions between all manipulator systems, although each student would focus on a particular sub-task in latter project stages as will be described below. The team's faculty advisor, the second author of this paper, met weekly with the graduate student and less often with the full team, providing high-level input and feedback on DBT decisions made primarily by the students. The advising goal was to ensure the students remained focused on feasible and appropriate goals and made steady progress toward these goals but to foster the creative learning process through independence to the extent possible.

As will be described below, the manipulator electromechanical hardware was comprised solely of easily-manufactured or COTS parts to allow students to quickly progress to the integration and subsequent kinematics modeling, software, and sensing DBT tasks. The project was constrained to be completed within a two-semester academic year, the limit of a normal UROP project. During the first semester, the graduate student supervisor was to remain on-hand for direct day-to-day supervision early in the project but during the second semester was to transition to lighter supervision to allow the undergraduates more independence in their work as their capabilities matured. In the selection of students, it was determined that the chosen UROP students should be early in their academic careers but with at least exposure to the full sequence of calculus and physics. We therefore selected sophomore undergraduates who displayed substantial enthusiasm for all aspects of a robotic system. Given the interdisciplinary nature of the project, we selected students with interest in hardware and software; previous experience was not required but was viewed as beneficial.

Below, we first present the system design requirements, with the first decision being to name the manipulator the Michigan Manipulator Arm (MM-Arm). We next overview the design-build cycle over hardware, software, and model development. Finally, we review results and discuss post-project manipulator use for research and education. The project was conducted in three phases, each of which went through a limited design-build cycle: an undergraduate education phase in which a single joint was constructed and tested, an augmentation phase where the full manipulator was constructed, and sensor and software development phases in which each UROP student pursued a distinct sensing/software research objective that ultimately was of benefit to the integrated system thus the full team.

II. MM-Arm Design Requirements

The first step of the project, completed prior to DBT team selection, was to determine the hardware and computing requirements, followed by development of a preliminary design including selection of major components. The team would then have a clear baseline on which to build rather than starting from scratch with little prior experience on which to build. A long-term objective was to support both fixed-base and mobile manipulation, in which the manipulator might be mounted on a mobile robot base. This required that the control computer be embedded rather than a desktop or laptop machine. From an educational perspective, the design had to account for the anticipated skill levels of engineering sophomores. It is a recognized and notable concern that "robotic systems capable of serving professional researchers often have quality or design requirements that push the limits of an undergraduate student team's knowledge and skill."[2] However, with careful supervision and a baseline design, we hypothesized the students could succeed in a full manipulator DBT effort. We also had cost constraints that constrained component choices, as well as the requirement for extensibility and use in a safe human-robot collaboration environment. The manipulator hardware layout was initially sketched by the graduate student, as were candidate choices for servos and computer architecture. The final hardware design was completed with the students, although this occurred early in the project with the undergraduates as apprentices to the graduate student rather than as independent designers.

A. Initial Requirements

The baseline desired research outcome of the project was that, by the end of the experience, there would exist a fully-operational, calibrated, four degree-of-freedom (4-DOF) robotic manipulator system capable of receiving joint-space (forward kinematic) and end effector (inverse kinematic) commands and executing them in real-time with constant specified joint velocities.

For safety purposes, the manipulator parts must be lightweight and sufficient structurally and mechanically sound for long-term use. Use of COTS parts was also preferable to facilitate initial assembly and subsequent repairs with minimal overhead. Stiffness of the linkages was important, and designed-in failure points were necessary, in our case using low-cost plastic gears that would strip when over-torqued before servos or arm linkages would break. Also, the speed and torque application capabilities were specified to ensure the arm could lift itself and a lightweight end effector (not required for this project) throughout its workspace. Safety for a proximal human test subject was also required. Because of this, it was determined that low torque joints for which power limits could easily be set should be incorporated, thus battery-powered COTS digital servos were chosen that could accurately and reliably follow position commands but that would not reach speeds or torques that would endanger a companion. It was also determined that joint-space motion limits would need to be designed into the control software to a high-speed impact in addition to the upper angular rotation limits already programmed into the digital servos.

From human-subject experiment considerations, the faculty supervisor and graduate student determined that the manipulator must have size and speed similar to a human arm; to do so, our arm was to approximately correspond to NASA's Robonaut arm in kinematics properties.[12] This choice also constrained the platform to a size small enough to easily manage, but large enough to facilitate the manufacture and assembly of all parts. To support potential mobility on a moving base, the computer system must be embedded. We determined more computational power might be required that would be available on a low-speed microprocessor such as a PIC or Atmel, targeting two potential processing architectures, PC/104 and Gumstix, with substantial processing capability, wifi compatibility, and for which institutional expertise was available.

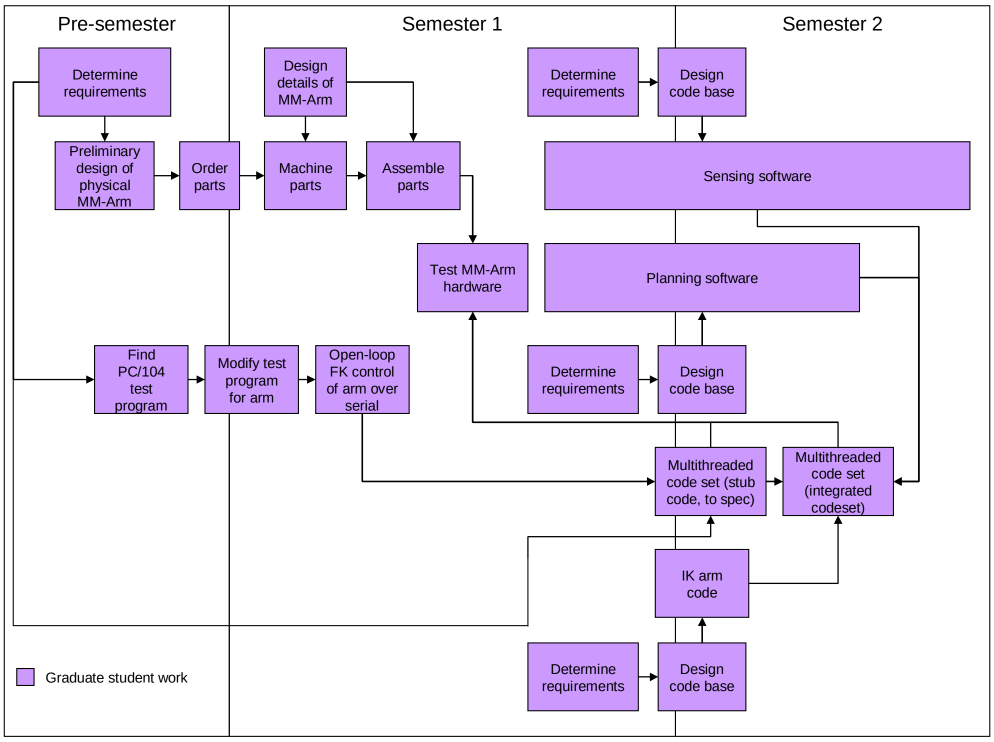


Figure 1: Initial Design-Build-Test Process.

B. Consideration of the student team

The primary education goal was for the undergraduate students to gain experience with all phases of DBT on a hardware project – a full cyberphysical system, and time permitting to pursue an appropriately-scoped research track that would contribute to the project but allow each student to achieve an individual research goal. From a system level, students would first learn about the complex dependencies between subsystems, after which each would obtain specialized knowledge in a particular aspect of the software/sensing/control that interested them. It was expected that the UROP team could complete this work in an academic year with realistic goals and sufficient mentorship. The secondary education outcome was that the graduate student would learn how to design such a system from concept to deployment while gaining experience in managing a team of undergraduate students.

The following characteristics were presumed of the undergraduate team:

- Undergraduates tend to have little experience with manufacturing[1]
- Students grasp fundamental concepts better than system-level concepts, especially those with little previous experience in the field. This suggests a project that combines instruction and initially offers small problems to learn; associated skills can later be combined once each subject/concept has been learned.
- Excitement/motivation levels need to be kept high throughout the project
- The possibility of a flawed design needs to be caught early; students need to learn from their mistakes but not to the extent that they become demotivated

To support the above assumptions about undergraduate students as well as the basic MM-Arm properties, a set of design principles were determined:

- COTS parts were to be used to the extent possible
- Parts with no to straightforward manufacturing to be used as needed
- Modular but straightforward software, with architecture specified by the graduate student and faculty mentor, would allow students to focus on a relevant subset of the code with little overhead in understanding the remaining code or its design
- Completion of manipulator part manufacturing and construction should occur in phases to allow the students multiple experiences with each phase of the DBT cycle
- Phased construction also allows early exposure to a partial hardware platform to built understanding of the test phase early in the project
- Feedback on feasibility of design details must be frequent initially but should decrease in frequency as students gain experience to ensure they are able to creatively devise solutions rather than following instructions.

For education purposes, it was further required that the computer system must support a full operating system (O/S) instead of a microprocessor-only design. This was for two reasons. First, the sole use of microprocessors for diverse projects limits students to details of bits and bytes rather than algorithms and networks. Further, use of the O/S allows students to build on their entry-level programming experiences rather than teaching what can appear to be a separate skill set. When writing code for a microprocessor, a specialized command set needs to be learned for that architecture, and students don't often make the connection with their "freshman introduction" to programming on a desktop/laptop. Additionally, memorization of specialized commands (e.g., registers) itself is most useful in the context of a system with complex logic, networking, and memory management – otherwise embedded code becomes "tricks" necessary to work with hardware only. Weight, power, and cost considerations often drive the choice of a microprocessor nonetheless, but in this case, processors with an O/S were available and weight/power were not significant concerns given the size of the rover to which the manipulator might ultimately be attached. By choosing a system with a full O/S, we were able to direct the students toward higher-level algorithm development and implementation and expose students to system-level behaviors. Choosing a platform for flexibility and performance as well as the "dual" education in algorithms and hardware interfaces was critical to the success of this projects.

The skills necessary for this project were to be taught or learned in phases, provided the students had good mathematics skills and some software background. The object-oriented approach (modular structure) shown in Figure 2 was to be used not only for the creation and testing of the control code of the manipulator system, but, in a sense, every aspect of this UROP project. The education-oriented project schedule was roughly as follows: for the first semester, the students were to be closely supervised, following tracks determined by the supervisor and advisor. During this first semester, the MM-Arm manipulator would be fully constructed and the most critical low-level software items coded, tested, and debugged. Once initial functionality of the arm was achieved, the students would then help define and pursue individual research tasks of relevance to MM-Arm, targeting completion near the end of the second semester in time to integrate their work with the arm. At the end of the second semester, the students would present a poster for the UROP forum, and also contribute to a "user's manual" for the MM-Arm that would detail their work, the MM-Arm design, its software, and its operation.

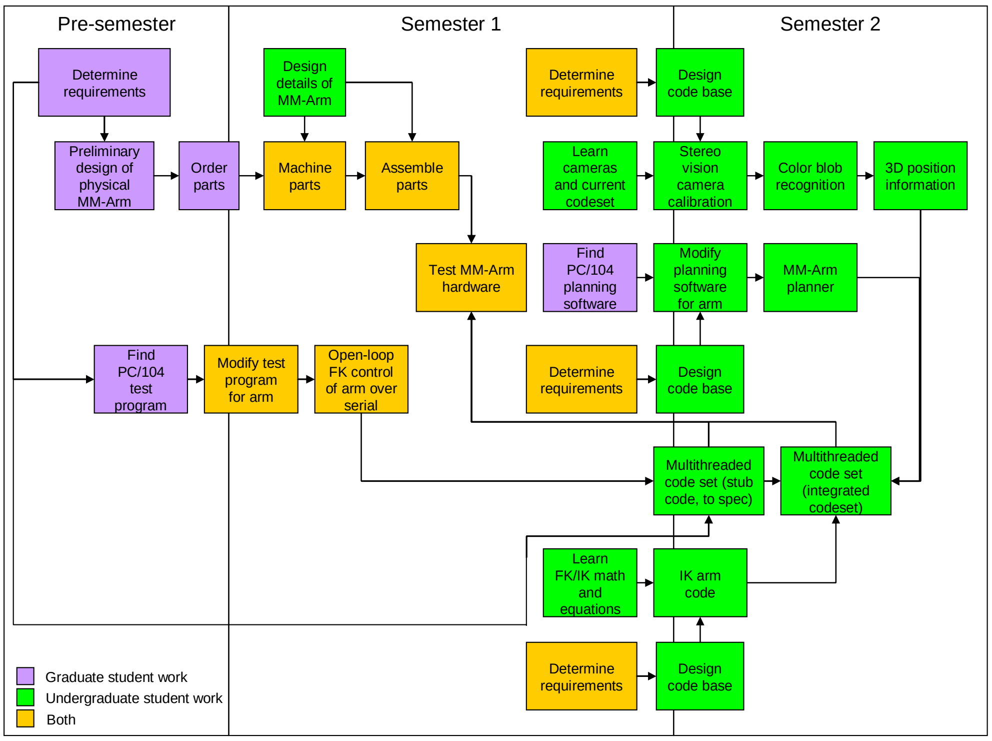

Pre-semester
Semester 1
Semester 2
Determine requirements
Preliminary design of physical MM-Arm
Order parts
Design details of MM-Arm
Machine parts
Assemble parts
Test MM-Arm hardware
Determine requirements
Design code base
Learn cameras and current codeset
Stereo vision camera calibration
Color blob recognition
3D position information
Find PC/104 planning software
Modify planning software for arm
MM-Arm planner
Find PC/104 test program
Modify test program for arm
Open-loop FK control of arm over serial
Determine requirements
Design code base
Multithreaded code set (stub code, to spec)
Multithreaded code set (integrated codeset)
Learn FK/IK math and equations
IK arm code
Determine requirements
Design code base
Graduate student work
Undergraduate student work
Both

Figure 2: Design-Build-Test Process with Undergraduate Student Involvement Considered.

As it was felt that all three students should gain a basic understanding of hardware and electronics construction, all three students were scheduled to participate in part manufacturing, electronics and wiring harness building, and final assembly. Given student backgrounds and interests, the work distribution would be split into three specialties of main responsibility: construction of the physical system and mechanisms and construction of additional electronics, design, implementation, and test of the software receiving, translating, and communicating motion commands to the arm, and lower-level signal processing. In terms of research, the work was also split. One student learned the fundamentals of robot kinematics and inverse kinematics to translate desired end effector positions to joint-space servo commands, and implemented algorithms to simulate the manipulator in Matlab. Concurrently, this student made measurements to verify arm Denavit-Hartenberg (D-H) parameters for forward and inverse kinematic development. Sensor systems were also a focus area, specifically obtaining feedback of joint angles and stereo vision processing to perform object recognition of uniquely-colored targets. The third student focused on software engineering, developing and testing multithreaded control, communication, and data acquisition code for the manipulator as will be detailed below.

In quickly testing the feasibility of a servo-controlled manipulator arm and keeping student motivation levels high, the three students were tasked to start their work with a COTS “elbow” R/C servo pan-tilt unit connected to an existing control system (the PC/104+ stack and serial servo controllers on the Autonomous Exploration Rover[13]) to do simple testing of the system. Subsequently, servo control were migrated to a lower-cost Gumstix-based computer system. This incremental approach to MM-Arm build and test allowed the students to start with a simpler, less breakable, and more approachable system while they improved their skill levels and expertise, allowing them to grow with the system. MM-Arm also was designed to help with motivation, as they could start working on hardware

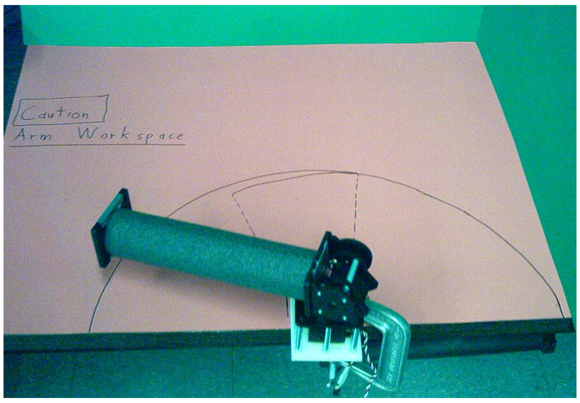
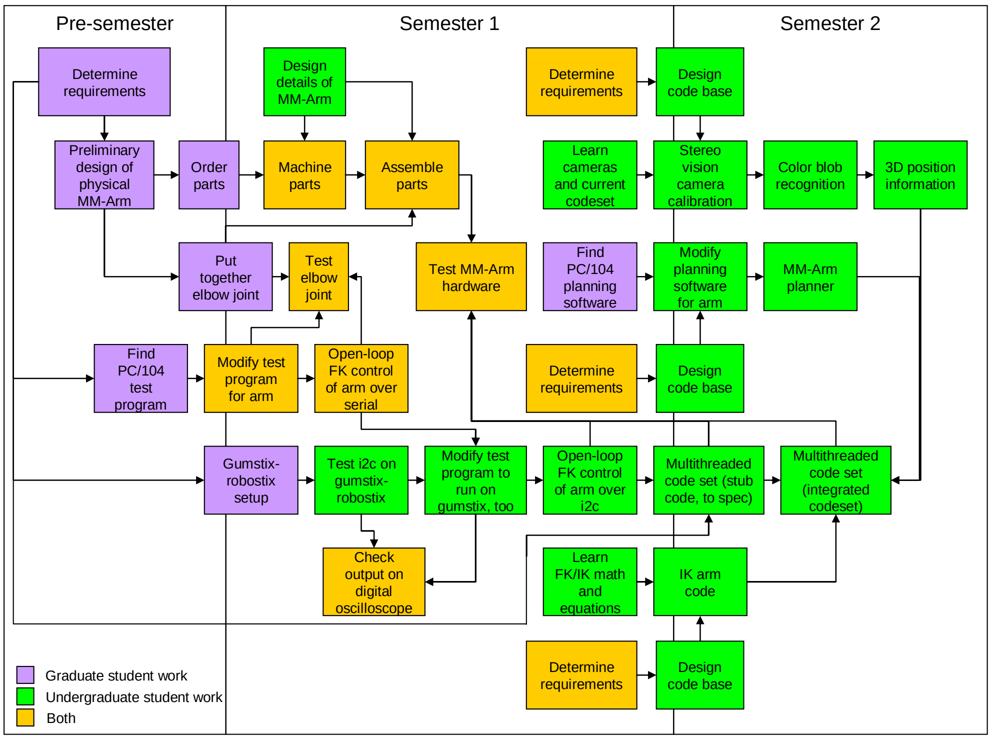


Figure 3: Simple Mounted “Elbow” R/C servo pan-tilt unit.

immediately, rather than following a longer more tedious process prior to prototype testing. Finally, use of initial components similar to the final system allowed students to perform component tests of hardware and code, identifying problems which otherwise might not be discovered until much later in the process.



Pre-semester
Semester 1
Semester 2
Determine requirements
Preliminary design of physical MM-Arm
Order parts
Design details of MM-Arm
Machine parts
Assemble parts
Put together elbow joint
Test elbow joint
Test MM-Arm hardware
Find PC/104 test program
Modify test program for arm
Open-loop FK control of arm over serial
Gumstix-robostix setup
Test i2c on gumstix-robostix
Modify test program to run on gumstix, too
Check output on digital oscilloscope
Determine requirements
Design code base
Learn cameras and current codeset
Stereo vision camera calibration
Color blob recognition
3D position information
Find PC/104 planning software
Modify planning software for arm
MM-Arm planner
Determine requirements
Design code base
Open-loop FK control of arm over i2c
Multithreaded code set (stub code, to spec)
Multithreaded code set (integrated codeset)
Learn FK/IK math and equations
IK arm code
Determine requirements
Design code base
Graduate student work
Undergraduate student work
Both

Figure 4: DBT Process with Undergraduate Student Involvement Considered and Phased Development.

C. DBT team selection & task assignment

In determining the undergraduate participants, there was sufficient interest that the faculty advisor and graduate student were able to interview students and downselect. Of note is that we made some assumptions and choices on the educational level of UROP student based on the known engineering curriculum at the University of Michigan. We were able to select students from a large pool of primarily freshmen and sophomores applying to the University's UROP program. It should be noted that during their first year at the university, all engineering students are required to take a basic programming course in C++ and Matlab called ENGR101, Introduction to Programming. Engineering students also do not declare their engineering majors until the beginning of their third semester, so first-year students do not usually take substantial degree-specific coursework until their second year. This influenced our decision in two ways: (1) we were notably biased towards sophomores, who we knew would have a baseline level of expertise in math, physics, and programming, and (2) we weighed each student's past experience (e.g., student project work) and grades into our decision, since this would be more indicative of their skill sets and capabilities than their recently-declared majors.

Three students were chosen after interviews. Student A was a sophomore in electrical engineering who had "work[ed] on hardware control designs for car assembly plant robot end effectors" the previous summer and became interested in robotics. Student B was a sophomore in computer engineering who had "served as a member of one of the University of Michigan's CanSat Competition teams" (http://www.cansatcompetition.com/Main.html) as a freshman and over the previous summer, during which he gained some experience in communicating with sensors through microcontrollers. Student C was a sophomore in engineering physics who had worked at a local base, US Army TARDEC that summer, during which he "created pedestrian detection and tracking algorithms in C++ (by matching stereo-vision images) in order to integrate real-time human tracking;" he was "intrigued by designing, building, and programming a system from the ground up." We believed these three would be compatible and capable, and their skills and interests fit nicely with the suite of identified manipulator DBT tasks. Because the graduate student had a vested interest in the project outcome given plans for MM-Arm use in her research, her motivation was high, and she had a good understanding of project requirements. This allowed for flexibility in day-to-day work, as critical decisions could be made on the spot without a need to wait and seek further approval.

Explicit commitment to the project was required by all participants, and participants needed to be full-time students in good standing with the university. All students involved had full course loads or equivalent effort for both semesters of the project. The graduate student devoted a part of her research effort towards the project plus enrolled in two courses the first semester and one course the second semester. Through the UROP program, the undergraduate students took a three-credit work-study course as part of their full-time course schedule both semesters, which translated to 10-12 hours of work per week. Student work was always scheduled to be done in the lab since the project was very hands-on and this also enabled the team to more easily collaborate with each other and their graduate student mentor. Additionally, the project was very hands-on, typically requiring students to work with the MM-Arm platform or at least its embedded processors or sensors.

During the first semester, three meetings were held weekly, with length as required to resolve collaboration issues, etc. One weekly meeting involved the three students plus the graduate student to discuss detailed progress, results, and the next steps. One meeting was simply a time set aside for all students to co-locate in the lab for informal discussion, but to focus on individual work if coordination was not needed. A third more formal meeting was scheduled with all four students and the faculty advisor to report progress, discuss any significant changes in project direction or schedule, and discuss any challenges the team had not independently solved. The graduate student was in the lab at nearly all times that the students were slated to work, and the students work schedules were scheduled around their other classes so that there was as much overlap with other students as possible. The graduate student closely supervised the students to speed along the initial education process, during which it would be crucial for the students to stay on task, follow the design closely, and coordinate their work efforts. It was also necessary to minimize duplication of effort, and to keep everyone on schedule.

The meeting schedule for the second semester during the individual project phase included a weekly meeting between all students as before. The graduate student was, again, in the lab at nearly all times that the students were working, but as the students were working individually, with the full MM-Arm platform in a working state to be used by each student as needed, the graduate student served in a loose supervisory role, only helping when asked, to allow the students more freedom in their chosen research focus areas. The research and development tasks the students chose required a lengthier timeframe to complete, rather than the relatively rapid design/build process of constructing the manipulator with close supervision in the first semester.

III. MM-Arm Design & Build Process

A. MM-Arm Design Specification and Development Process

The MM-Arm was designed and assembled over three phases of development. In Phase I, a two degree-of-freedom elbow joint was assembled and tested. The immobile shoulder was mounted directly to a table edge, connected to and controlled by the serial servo controllers on the PC/104+ stack on an existing autonomous rover that might ultimately host the manipulator. The software used to control it was initially derived from a similar servo control program written for the autonomous rover. Phase II was a move from the PC/104+ architecture to the gumstix architecture (with robostix, an Atmel-based expansion board). This transition required the team to configure the computer boards, build appropriate connectors, and extend existing code to generate PWM control signals for the servos. A breadboard was built rather than a PCB to expedite servo control; a PCB could be manufactured in future work but was not required for this project given that the manipulator remained fixed-base throughout. A 6V lead-acid battery powered the system given that weight was not a concern and 6V batteries plus backups were available in the lab. During Phase III, the full MM-Arm physical hardware was completed, including a wooden base mount + shoulder joint + linkages + elbow joint + linkages and padding. The gumstix-robostix computer stack was the main computer architecture, but the PC/104 stack was also considered an option thus software was made cross-compileable to support either platform. The software itself expanded to allow control of four servos, not two, and was revamped to allow for later additions. A line driver board was designed and constructed for the rover electronics to boost its serial servo controller signal current and decouple the servo power line from the servo signal line. This was necessary given that the manipulator's four high-torque servos had a higher maximum power requirement than that required by the rover or the Phase I/II elbow alone.

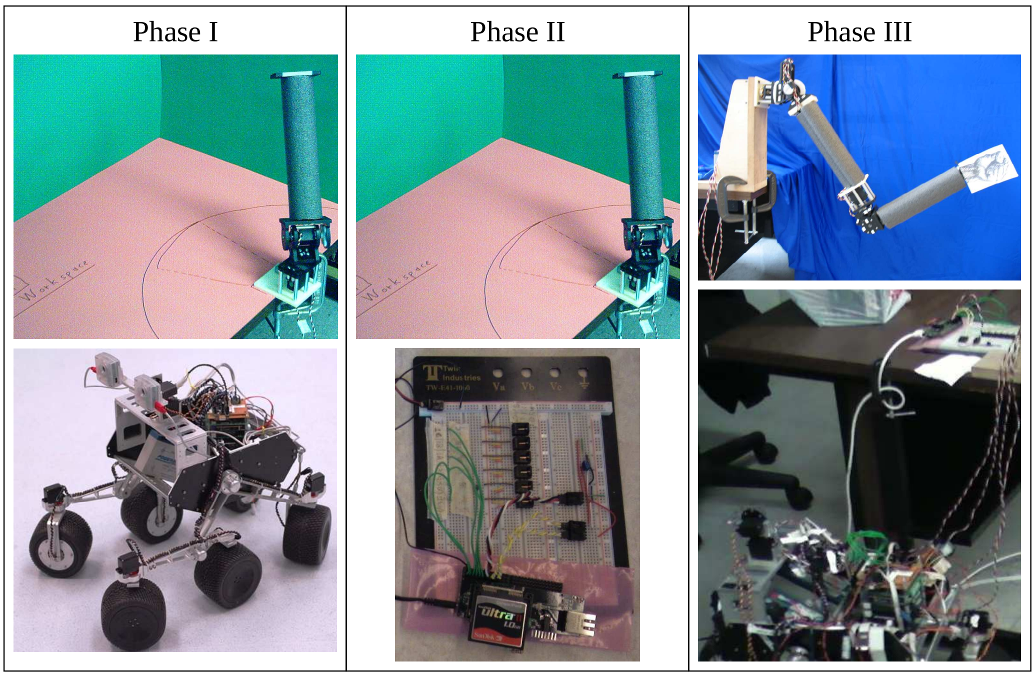
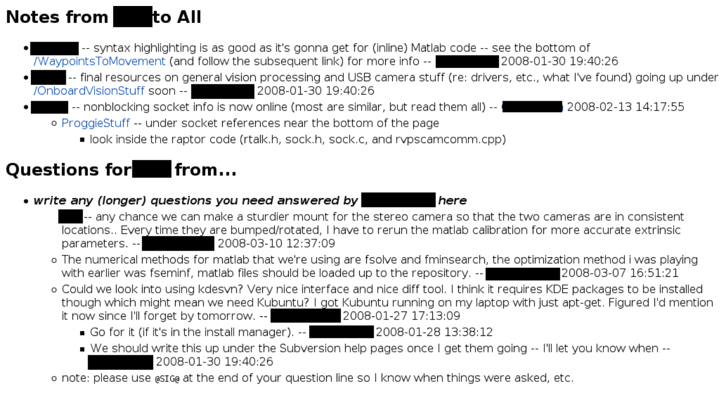


Figure 5: Phases I, II, and III of MM-Arm Development Process.

The physical construction of MM-Arm was a shared responsibility. Every student contributed to each phase of construction, gaining experience with machining, soldering, and assembly of the full MM-Arm platform. Testing of the MM-Arm was an incremental process. Each phase was fully completed before moving on to the next, and as such, each of the critical components for that phase had to be built and assembled before testing, replacement or repair or debugging, and approval. Most testing was software-based and, whenever possible, was set up to not require the MM-Arm to be in-circuit with servos powered-on for preliminary testing. The hardware did not require substantial debugging, as it was usually visually apparent whether a component was broken or defective.

Later software development, once the critical path was completed, was designed and "built" (coded) on alternate computer systems generally without a need for validation on the MM-Arm platform until after the eventual integration with the main code base (the "multithreaded program" with code stubs) created in Phase III. Some coding projects could be separately tested on a desktop system (such as the stereovision color sensing), without a need to run locally on the gumstix-robostix or PC/104 platforms.



Notes from [redacted] to All

- [redacted] -- syntax highlighting is as good as it's gonna get for (inline) Matlab code -- see the bottom of /WaypointsToMovement (and follow the subsequent link) for more info -- [redacted] 2008-01-30 19:40:26
- [redacted] -- final resources on general vision processing and USB camera stuff (re: drivers, etc., what I've found) going up under /OnboardVisionStuff soon -- [redacted] 2008-01-30 19:40:26
- [redacted] -- nonblocking socket info is now online (most are similar, but read them all) -- [redacted] 2008-02-13 14:17:55
  - ProggieStuff -- under socket references near the bottom of the page
    - look inside the raptor code (rtalk.h, sock.h, sock.c, and rvpscamcomm.cpp)

Questions for [redacted] from...

- ***write any (longer) questions you need answered by [redacted] here***
  [redacted] -- any chance we can make a sturdier mount for the stereo camera so that the two cameras are in consistent locations.. Every time they are bumped/rotated, I have to rerun the matlab calibration for more accurate extrinsic parameters. -- [redacted] 2008-03-10 12:37:09
  - The numerical methods for matlab that we're using are fsolve and fminsearch, the optimization method i was playing with earlier was fseminf, matlab files should be loaded up to the repository. -- [redacted] 2008-03-07 16:51:21
  - Could we look into using kdesvn? Very nice interface and nice diff tool. I think it requires KDE packages to be installed though which might mean we need Kubuntu? I got Kubuntu running on my laptop with just apt-get. Figured I'd mention it now since I'll forget by tomorrow. -- [redacted] 2008-01-27 17:13:09
    - Go for it (if it's in the install manager). -- [redacted] 2008-01-28 13:38:12
    - We should write this up under the Subversion help pages once I get them going -- I'll let you know when -- [redacted] 2008-01-30 19:40:26
  - note: please use @SIG@ at the end of your question line so I know when things were asked, etc.

Figure 6: Sample Communication from the Project Wiki.

B. Collaboration Tools

Since the students did not have fully-overlapping work schedules in the first semester, and even less overlap during the second, collaboration tools were considered necessary to help keep everyone in-the-loop during major development and task transition periods, as well as various code integration steps. Multiple tools were used to help facilitate the team's work. The graduate student configured, demonstrated, and insisted on use of a version-controlled code repository, subversion, to help keep track of changes made in the code, and for general development purposes: commenting code; keeping track of stable codesets; branching and merging code as functionality was created, tested, and folded back into the main branches. This helped resolve library dependencies and the sharing of codesets between students, and gave a fine level of control over the automated archival and retrieval of any versions of the code. This also took pressure off the students to commit only fully-function code versions, as it was much easier to track down the differences between previously-working and newly-broken code when trying to add new functionality.

The graduate student also configured and encouraged the use of a MoinMoin wiki for keeping track of each individual's daily and long-term tasks as well as informal and formal documentation generated throughout the project. The wiki was used as a central source of information on how everyone progressed with their projects at any given time – completed/resolved, ongoing, and future tasks – including a list of bugs and errors that could cause issues, both in the software and hardware. This turned out to be very useful for the daily tasks. Having an up-to-date location for this information made it easier to fix known problems between sessions, as all of the participants would check the wiki for status updates at the beginning of their work sessions before working with the MM-Arm hardware, so that they could either help fix things themselves – and document the problem and the fix, or the results they learned from troubleshooting to help the process along – or modify their own daily tasks so that they would not need to use anything malfunctioning or broken until it had been fixed. In the case of hardware testing, this also facilitated the collaboration process in that everyone could assume that if there were no problems documented on the wiki, and no major changes listed without associated testing, that the MM-Arm was operational. This eliminated time carefully verifying MM-Arm configuration before starting each work session. Thus, the MM-Arm could be used with some level of security, while watching for new problems. The extensive use of the wiki also allowed it to be used as an archival source of information on the work done by the group during this time period, critical to longevity of the system given that all young students tend to disappear once their project term is complete.

C. Design Specifics

The materials used for construction of the MM-Arm were obtained in as close to finished form as possible. Many parts were chosen due to availability – they were spare supplies already available in the lab. This helped to keep the cost of new expenditures low. It was also very useful in that there were usually knowledgeable people working in the lab who could be approached as resources for information in their application. Other supplies, notably the carbon fiber arrows, were obtained as donations to the project from local businesses whenever possible. It was happily noted that many businesses were willing to give a plethora of free samples, information, and advice when it was explained that the project was a undergraduate learning experience at a major university. As a result, the overall expenditures for the system totaled ~$450. If we had had to obtain all parts at-cost for the MM-Arm gumstix-robostix platform, it would have cost ~$1136, or ~$1819 for the MM-Arm PC/104+ platform version.

As noted above in the requirements, the MM-Arm needed to be sized similarly to a human arm, and was to use COTS servos. Thus, in order to satisfy human-analogue Denavit-Hartenberg (D-H) parameters as closely as possible,[14] weights of the parts needed to be determined, torques on the structure calculated, and the specific pan-tilt units and servos were sized for the available manipulator torque capability we desired, according to their relative placement. A list of sizes, shapes, and mounting points was determined from the available pan-tilt units, and torque output was tabulated from the available digital servos. A safety factor of 2 was chosen and calculations of the necessary torque at maximum extension to hold a small additional weight (~100 grams) were made to determine the lower limit on torque requirements needed for sizing the servos. Once an appropriate servo selection was accomplished, the

Table 1: Component Listings and Approximate Pricing

Components	Expenditure Amount	Cost to Purchase
Gearbox pan system + servo (SPG400 + HSR-5995TG)	$185.00	$185.00
Gearbox tilt system + servo (SPT400 + HSC-5955TG)	$205.00	$205.00
Pan and tilt system + servos (2) (SPT200 + HS-5645MG)	{existing hardware, free!}	$160.00
Top plates, 2 (for interfacing)	$8.00	$8.00
ABS plastic bar (for interfacing)	$7.00	$7.00
Carbon fiber arrows (3, 12" ea)	{donated, free!}	$120 / 12 arrows, 12" ea
Threaded arrow inserts (12)	{donated, free!}	$7 / 12 pack
Bohning Power Bond	{donated, free!}	$12.00
Miscellaneous hardware (washers, shoulder screws, spacers)	{some donated} $45.00	$60.00
Miscellaneous electronic hardware (connectors, wire)	{existing hardware, free!}	$20.00
6V 7Ah lead-acid batteries (2) (PS-640)	{existing hardware, free!}	$33.00
6V charger (PSC-61000A-C)	{existing hardware, free!}	$34.00
CompactFlash card (2 GB)	{existing hardware, free!}	$15.00
Prototype board + TTL buffer chip	{existing hardware, free!}	$30.00
PONTECH SV203 servo motor controller boards (2)	{existing hardware, free!}	$120.00
Advantech PCM-3370F PC/104+ CPU + cable kit	{existing hardware, free!}	$750.00
12V 7Ah lead-acid battery (PS-1270)	{existing hardware, free!}	$20.00
12V charger (PSC-12800A-C)	{existing hardware, free!}	$33.00
Breadboard + jumper wires	{existing hardware, free!}	$30.00
Gumstix connex (400 MHz)	{existing hardware, free!}	$130.00
Robostix (ATMega128)	{existing hardware, free!}	$50.00
netCF board	{existing hardware, free!}	$60.00

offsets of the pan-tilt units, lengths of carbon linkage shafts, and details of the mounting were finalized, the servo torque requirements checked one final time, and all parts not already available for use were ordered.

The resulting design utilized COTS high-torque (low power) R/C digital servos (in essence, low torque motors) and hard plastic pan-tilt mechanisms (stiff enclosures) for the shoulder and elbow joints. The servos and pan-tilt units were the lightest low-cost packages that could be found that would also supply enough torque to move the manipulator at a useful speed, determined from the calculation procedure described above. Servos are robust and reliable, able to handle a range of voltages, and somewhat forgiving to power spikes, bad commands, and other general misuse.

The rigid linkages between the joints were composed of multiple carbon fiber tubes (made from arrow shafts donated by a local vendor). Since the weight of a set of chosen servos is not customizable, weight had to be reduced elsewhere, and carbon fiber linkages were the best readily available lightweight material capable of retaining enough stiffness under the load weight to allow the manipulator to conform to traditional rigid body assumptions between joints. The joint linkages are comprised of groups of three small-diameter arrow shafts, screwed into holes on the termination plates attached to the joint pan-tilt units with traditional fasteners.

The servos chosen operate over a range of DC power, from 5.4V to 7.2V, although at lower voltages the torque output is lower. We decided to supply 6V of battery power to the servos via two 7 Amp-hour lead-acid batteries in parallel. Batteries were chosen as the preferred power source for the servos to put a limit on the current draw of the servo motors. The drain over time would also make the manipulator safer in a sense because the hardware would be able to supply less torque as power diminished, and thus any possible impact would have less effect. Decoupling the manipulator's servo motor power from the control signal power also allowed the system to be easily depowered during testing, and allow system tests where the control code could run 'offline' – without the manipulator having to move – which was a particularly useful feature for the students working in parallel on the code and hardware. Also, designing for the use of battery power more easily allows for future platform mobility.

Also for safety purposes, foam padding was used to wrap the MM-Arm linkages, in order to add an extra level of safety to the human-proximal operations. Should any contact occur between the MM-Arm and a human, between the imposed power limits, low inertia, and low speed of the manipulator, damage would be close to negligible as a whole, and the impact would be mitigated even further by the soft compressible material.

A design note is that points of failure were purposefully designed into the structure. All the high-torque servos have very strong internal metal gears, which do not fail easily. Externally, plastic gears were chosen for the connection points on and to the metal shafts. Thus, when the inevitable bad commands and other hardware problems occurred, the lower-cost, easily- and quickly-replaceable external plastic gears failed, not the more costly internal servos gears or their housings or pan-tilt units (which were made of much harder and more durable plastic). This made errors less costly in general, and had the added benefit of allowing the students to be able to both fix and learn from their own mistakes – in this education environment, it was better for students to be comfortable enough in making mistakes that they wouldn't be afraid of the hardware or off-put by reasonable accidents, but not so comfortable that they would break things with impunity or through carelessness.

In our project, the graduate student, with support from the faculty advisor, specified the C++ software architectures show in Figures 7 and 8, and assigned the students modules to implement and test. We started off using the PC/104+ stack for MM-Arm control. Two PONTECH SV203 servo motor controller boards received command signals sent via an RS/232 serial port connected to an Advantec PCM-3370F PC/104+ CPU board running Linux Fedora Core 3 and the legacy servo control software, written in C++. When it became apparent that use of the PC/104+ rover system was a bit complex and overpowered for the current DBT needs, we decided to move to a simpler alternate platform that would more easily allow a mix of both low-level and high-level programming. We had knowledgeable people in the lab who had used multiple types of these sorts of systems before and could help with some of the setup of the devices, and a smaller simpler platform was alluring for an undergraduate project. Because of this, we chose as our new primary platform a gumstix-based architecture, a good compromise of high computing performance and lightweight, low-power packaging. This hardware, specifically a gumstix connex (400 MHz) with robostix Atmel support and a CompactFlash card, can boot a specialized version of a Linux operating system and run C++ programs specially compiled on a development system. Through an i2c bus, programs run on the gumstix board can access registers on the robostix board, which include the real-time sending of PWM signals to control devices such as servos, and optionally the reading of some A/D channels for sensor polling. Our robostix provided some exposure to programming efficiently at a low level, as well: for our project, one of the students interested in lower-level programming was directed to learn about the robostix chipset (ATMega128), and expand the base supplied codeset for single-ended A/D input conversion to include support for differential A/D inputs.

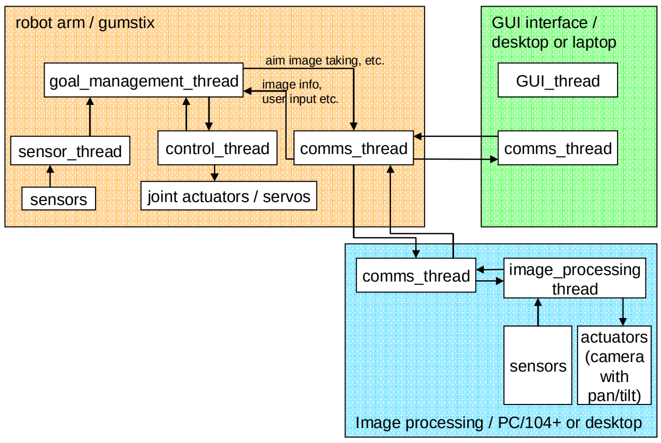


Figure 7: MM-Arm Computer Architecture Design.

An old Dell laptop was used for the gumstix-robostix development system. A preexisting preconfigured desktop system, with a Firewire card and associated libraries to allow use of two Firewire cameras, was used for the image processing development system, as it had been used in the past. A third desktop system, which had a standard g++ compiler and Matlab installed, was used for the programming of the skeleton code for the multithreaded program structure and the prototype and streamlined FK/IK codesets. Splitting development across three separate computer systems allowed the students to each have their own workspace and machine on which to program in parallel.

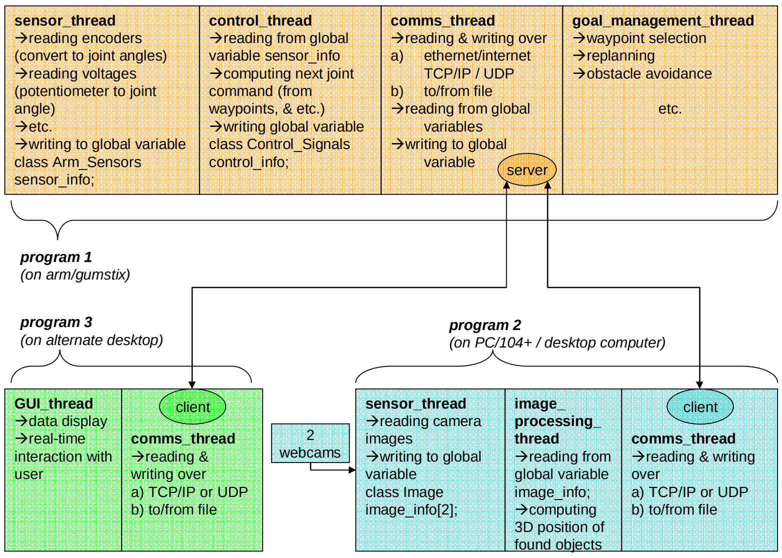

sensor_thread
→reading encoders (convert to joint angles)
→reading voltages (potentiometer to joint angle)
→etc.
→writing to global variable class Arm_Sensors sensor_info;
control_thread
→reading from global variable sensor_info
→computing next joint command (from waypoints, & etc.)
→writing global variable class Control_Signals control_info;
comms_thread
→reading & writing over
a) ethernet/internet TCP/IP / UDP
b) to/from file
→reading from global variables
→writing to global variable
server
goal_management_thread
→waypoint selection
→replanning
→obstacle avoidance
etc.
program 1
(on arm/gumstix)
program 3
(on alternate desktop)
program 2
(on PC/104+ / desktop computer)
GUI_thread
→data display
→real-time interaction with user
client
comms_thread
→reading & writing over
a) TCP/IP or UDP
b) to/from file
2 webcams
sensor_thread
→reading camera images
→writing to global variable class Image image_info[2];
image_processing_thread
→reading from global variable image_info;
→computing 3D position of found objects
client
comms_thread
→reading & writing over
a) TCP/IP or UDP
b) to/from file

Figure 8: Multithreaded Code Design.

The software development went through two major revisions on two fronts. The first was expansion of the preexisting rover demo/test program generally used for manually debugging the rover servos. This program had additional functionality added to it to allow for easier control and testing of the calibration of the servos. Next, the program was modified to allow similar testing on the gumstix-robostix platform, which involved the addition of code that could send the proper commands over the i2c interface, and the revision of an available codeset for the robostix Atmel chipset that would allow easy access to the i2c bus with solely changes to a program running on the gumstix. From this start, the main multithreaded codeset created borrowed from the servo control code perfected in the test program for its control_thread, and included other code stubs for interfacing that would allow for future functionality in the research to be added in more easily. Once the primary (gumstix) multithreaded codeset was created, the other two programs were to be created by copying the original skeleton code and then modifying each version for use by including the students' second semester independent work.

IV. Project Results

A. Manipulator capabilities & performance

At the end of this stage of development, the MM-Arm test platform had baseline functionality in place and tested. It was fully operational using the "test program" on the gumstix, and partially-operational (two-joints at a time) on the PC/104+ system. The low-level control of the MM-Arm was working beautifully, and the tie-ins for the path planning code that was to come was already in place. The FK/IK code implementation was complete, and waiting to be added to the main codeset where necessary. The color object recognition was functional (calibrated and giving results), but would need more work to be implemented fully into the system. All the critical goals for this project were met.

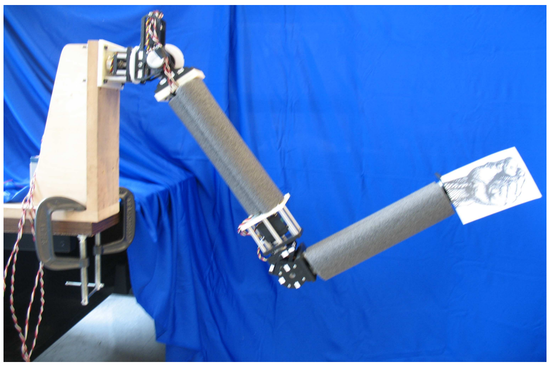


Figure 9: MichiganMan(ipulator) Arm after DBT I

B. Student experiences and future paths

The graduate student gained experience in designing a robotic system, in the teaching and supervision of students, as well as with general managerial skills required in planning and directing a team and, by the end, a viable platform for their research. The undergraduate students gained new knowledge and skill sets associated with the building and testing of all the individual aspects of the manipulator system (hardware, electronics, computing, power, and software), including an appreciation of the complex interactions between the systems within the full system-under-test, as well as teamwork and communication skills. As part of this, during the build phase, each of the three students in turn was tapped to cut and sand carbon fiber tube linkages, drill holes in the plastic plates, solder servo cord extensions, and perform an assortment of other minor tasks associated with the assembly of parts, learning basic construction techniques and skills. In the test phase, they also had hands-on experience in debugging and repairing problems with and on the hardware platform, both its structural components and its electronics and computer systems. They all gained further experience in C++ programming and the process by which to combine and transfer codesets, including dependencies, along with a lesson in the importance of commenting code, in a medium-sized coding project. Each student had also gained skills according to their chosen specialization of interest for their second semester independent work, including the original creation during the design phase. It is noteworthy that no student's coursework suffered as a result of working on this project (participation did not have a negative effect on the students involved).

At the end of the second semester, the students did create and present their poster at the UROP forum, and won a prize for their work. They also contributed to a "user's manual" for the MM-Arm, a document that detailed the majority of their work over the year and persists as a reference document to this day (with some updates as later changes have been made). All of the in-progress wiki documentation also still persists on the work computer's webserver, as does the subversion repository. All three students also went through a 'debriefing' meeting with the graduate student, in order to transfer any last-minute breakthroughs in work and give a broad overview of the fine details of the codesets they created and modified.

1. Graduate Student

One major observation made by the graduate student was that, coding the software for the system for this and other efforts is typically much more difficult and time-consuming for the UROP students than constructing the hardware, particularly given challenges with embedded software testing and validation.

Another observation was that, in the first semester, when a great deal of time was spent by the graduate student actively interacting with the team, to the point of micro-management, a great deal of progress was made. However, in the second semester when the graduate student more or less allowed the students to work unsupervised – passively available in the same room, but not actively asking after the daily work – from time to time, the undergraduate students would get stuck for nontrivial periods and yet not ask for help. At first glance this seemed very wasteful, but upon reflection with the advisor, the graduate student came to realize that allowing the undergraduates to flounder and realize when they should ask for help, when they would come to need it, was also a useful skill for them to learn. From this, the graduate student concluded that it is possible that a great deal of oversight is necessary in order to insure that the critical parts of an undergraduate-built project are finished on-time and within spec, but that there are merits in trusting smart undergraduate students to work on their own, as well.

It should also be noted that, due to the micro-management during the first semester, the graduate student was able to keep abreast of most developments between the undergraduate students, and thus the informal weekly student meeting time necessary for this was minimal. Because of this, and due to popular demand by the undergraduate students, the rest of the informal meeting time between students began to be used as a tutorial learning experience, when the graduate student could give a quick-and-dirty information session on whatever topic(s) that the students either needed to know for the week, or had otherwise expressed interest in that were related to the manipulator development. Some of the topics discussed included: a explanation of servo sizing from torque requirements, to explain the early MM-Arm design choices; binary math and computer memory; PWM servo control; TCP/IP sockets; the gumstix-robostix platform; PC/104 platforms, their form factor and interfacing; image processing and the physics of light and stereo vision; and the forward and inverse kinematics of manipulator systems. Having a single meeting to do this best served all the students involved, as the graduate student would only have to discuss the material once, and the undergraduate students could benefit from a brainstorming-like session where all three of them could ask questions at once, and build off of one another. These tutorials were an immense help in educating the students on topics that were necessary to their work in a timely fashion, though it was not trivial work to prepare for these weekly, as it usually took several hours to compile the information for a session.

2. *Student A*

This student was tasked primarily with managing the construction of the physical system and mechanisms, and construction of additional electronics, such as the layout and soldering of a line driver board to boost the servo signals that controlled the joint motion.

During the second semester, this student took on a set of computational tasks related to the mechanisms were the creation of matrix and vector math libraries useful to robot manipulation, the use of (taken) measurements of D-H parameters of the arm for forward kinematic development (equations and code), and inverse kinematic development (equations and code). There was an option of two task sets for this path, and the one not picked involved characterization of the MM-Arm's physical movement capabilities and FK trajectory creation and following.

This student worked quietly during the second semester, and it wasn't until the student had struggled for weeks and the graduate student finally forcibly sat down and asked to see the status of progress that it was realized that the student was having great difficulty with the FK and IK math. At that point, the graduate student decided that this was a minimal part of the whole and simply took on this portion of the task, completing the initial calculations and giving them to student A. Once this was done, student A was wholly able to progress rather quickly and completely independently, coding the associated work in Matlab and then C++.

Upon exodus, student A went on to work on a different project with the lab – the TableSat system – and was able to work over the course of the Summer 2008 semester virtually unsupervised on collaborative research that eventually resulted in a paper.[15]

3. *Student B*

This student was tasked primarily with managing the creation of code controlling lower-level signal processing and communcations, such as: gumstix-robostix communications, interboard I2C, and troubleshooting for the initial control of the MM-Arm.

During the second semester, this student took on a set of later computational tasks that included: maintenance and expansion of the multithreaded codeset; and planner/scheduler modification from the AER code to manipulator use, which was an expansion on both the original program structure and communications tasks. This involved some work with student C, where student B gained experience in how to learn how to take over development of a codeset one did not create oneself.

As part of this transfer, this student transferred this codework so that it would operate on the gumstix-robostix platform. The student ran into difficulties in running the multithreaded codeset with socket communications, and spent a great deal of time characterizing and subsequently tracking down the issues causing this. The student discovered that the gumstix operating system simply did not support multithreaded operations well, so the code was streamlined and became serial in operation. The socket communication issues were an odd time delay problem, which the student was able to track down as what seemed to be a problem with the Ethernet driver for the gumstix itself. Both of these problems, once characterized this fully by the student, helped prompt the decision to move control of the MM-Arm back to the PC/104+ computer stack for later MM-Arm research.

Upon exodus, student B worked on some "AI-level" coding and computer vision in the Fall 2009 semester for the A2Sys Lab for a bit. He had taken extended courses in AI in the following semesters; this project had helped expand their interest a bit.

4. *Student C*

This student was tasked primarily with managing the software coding of the global program structure and baseline mid-level control algorithms, such as open-loop real-time forward kinematic control of the arm – a program for debugging the arm – and multithreaded code development for the clients/servers. At the end of this, there was some discussion with student B, where student C gained experience in how to transfer knowledge when giving over control of a codeset.

During the second semester, this student took on a set of advanced computational tasks related to vision processing, stereo vision calibration, and color object recognition. There was an option of two task sets for this path, and the one not picked would have involved the addition of sensors to the MM-Arm joints, giving feedback of the MM-Arm's actual joint space position, and use of this feedback information in the software.

This student needed some help in finding resources to explain the math involved in finding the 3D coordinates of objects recognized in an image, as well as resources on the calibration of stereo cameras and possible libraries that could be useful in these calculations. Once the student had this information and was able to connect it back to previous knowledge from prior work, the student was happily able to work independently with not much need oversight. At the end of this, the student had resolved most major issues and was at a stopping point where a future student would be able to easily pick up where this work was left off – stereo images captured, a calibration procedure set up and working, colored balls able to be recognized, with image coordinates as an output result. What would need to follow next would be transformation of the camera frame coordinates to whatever coordinate system is needed, and then the compilation and use of this information by the MM-Arm system.

Upon exodus, this student took a summer internship position in industry.

V. Subsequent MM-Arm Use

A. MM-Arm Research: Summer 2008 – Winter 2009 semesters

After the Generation I UROP group had completed their work, the graduate student then took over the continued development of the code, and continued the research for which the MM-Arm was designed as a test bed platform to experimentally explore. Code implementing the path planning and cognitive layers was written, with helpful insight and direction by the faculty advisor. In the process of this, the code was finally converted to work completely on the PC/104+ computer stack with the serial servo controller boards, and the underlying code that controls the operations was both cleaned up and streamlined as part of this. The multithreaded functionality was removed temporarily, and the gumstix+robostix dual-compilation functionality was completely removed. This period demonstrated that the student team did in fact generate an artifact with long-term utility for research activities.

B. Undergraduate Education and MM-Arm Research: Summer 2009 semester, Winter 2010 semester

The baseline desired research outcome of the Generation II UROP Program was that, by the end of summer, general human subject testing would be approved for the MM-Arm platform, a full test scenario and procedures would be in place, and the first main battery of tests over a statistically significant number of test subjects would have taken place and the associated data collected.

The baseline desired education outcome was that a part-time UROP student, an undergraduate sophomore in aerospace engineering designated beforehand by the advisor, would have had a chance to work with and learn about the manipulator system, learning a bit from the experience, while helping with the human subject testing. The MM-Arm system was basically in its near-final form at this stage, so not much new design work had to be done on the manipulator system. However, for the human subject testing, the graduate student still needed to design the workspace setup itself and the tests to be conducted needed to be designed, with input from the faculty advisor.

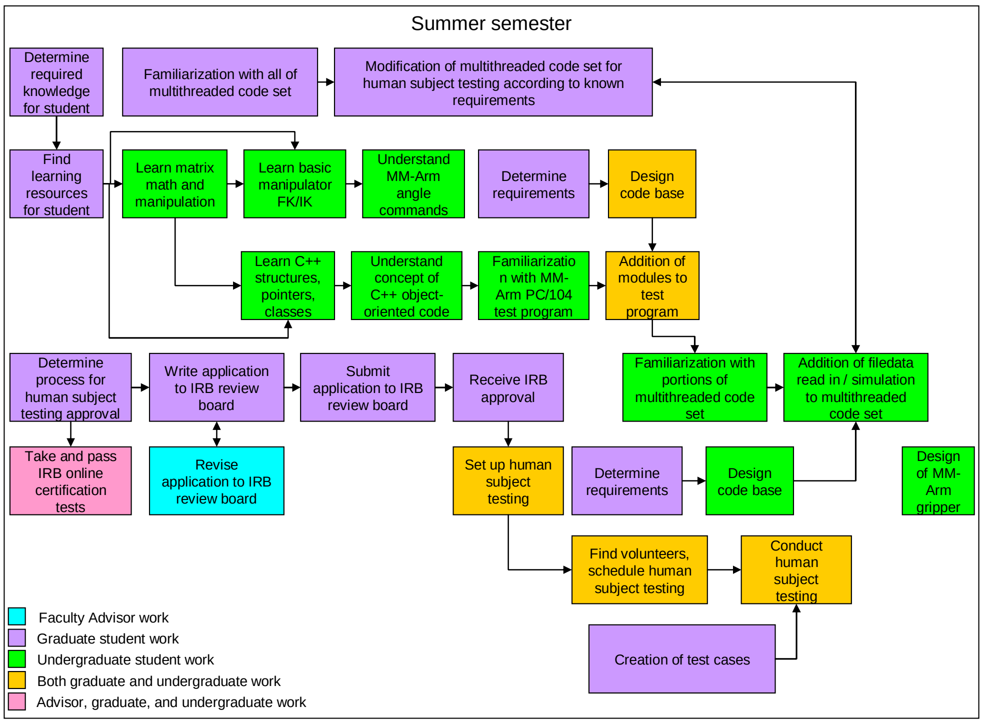
1. MM-Arm project / system design

The undergraduate and graduate student work paths are shown below in Figure 10.

The student spent much of the first half of their summer work in learning about the MM-Arm platform itself and the general kinematics associated with manipulator motion. Once through this, the student was given reading assignments and tasks that expanded their knowledge of the C++ programming language. Once accomplished, the student was exposed to the test program codeset and given some simple useful tasks. Once proficiency and understanding in that codeset was obtained, the student was given another task which involved exposure to the larger codeset that the graduate student had been modifying – the child of the original "multithreaded" codeset. These code projects were not on the critical path for the baseline system functionality, but were helpful modules supplying necessary functionality for later research.

During this time, the graduate student helped teach these concepts and also supervised the student's work, while writing and submitting the application for human subject testing to the university IRB for approval with the faculty advisor's help. Everyone involved in the project then took the online tests necessary to allow them to conduct the human subject tests. Once approval was given, the graduate and undergraduate student took turns conducting the human subject tests that the graduate student scheduled.

Towards the end of the summer, the undergraduate student was given the freedom to pick a next topic of work. After discussing with him the topics and things he found interesting, he was given two choices – working on a Qt GUI for later human testing, or creation of an actuated wrist and hand to be mounted at the wrist of the MM-Arm. He chose the latter, and worked mainly independently to research several manipulator gripper designs, design his own, and then create CAD drawings in SolidWorks of the parts he was ordering and machining, learning the program as he was going using the help files and online tutorials.



Summer semester
Determine required knowledge for student
Familiarization with all of multithreaded code set
Modification of multithreaded code set for human subject testing according to known requirements
Find learning resources for student
Learn matrix math and manipulation
Learn basic manipulator FK/IK
Understand MM-Arm angle commands
Determine requirements
Design code base
Learn C++ structures, pointers, classes
Understand concept of C++ object-oriented code
Familiarization with MM-Arm PC/104 test program
Addition of modules to test program
Determine process for human subject testing approval
Write application to IRB review board
Submit application to IRB review board
Receive IRB approval
Familiarization with portions of multithreaded code set
Addition of filedata read in / simulation to multithreaded code set
Take and pass IRB online certification tests
Revise application to IRB review board
Set up human subject testing
Determine requirements
Design code base
Design of MM-Arm gripper
Find volunteers, schedule human subject testing
Conduct human subject testing
Creation of test cases
Faculty Advisor work
Graduate student work
Undergraduate student work
Both graduate and undergraduate work
Advisor, graduate, and undergraduate work

Figure 10: Research and Education Work Paths.

2. *Collaboration tools*

Along with the original tools used in the UROP program, use of the Trac tool was very helpful as the undergraduate and graduate student found bugs in the test program code and other aspects of the software, and set out to keep a running log of those problems found and fixed.

3. *Project Results*

At the end of the summer, the student created a poster for the MM-Arm project, and presented it to the UROP staff. The graduate student received a full set of data for their further research.

With the changes to the program, we were able to bring a normal student, working alone, up to speed on the existing functional system and have that student contributing to the project within one month of part-time work.

4. *Student Experiences and Future Paths*

Currently, this student has gone on to work as a part-time undergraduate researcher for the lab, continuing his manipulator gripper work – designing and building a removable end effector for the manipulator.

VI. Conclusions & Future Work

We have described a design, build, test project based on a robotic manipulator with human-arm scale and kinematics. In this project, an undergraduate student team gained experience with practical DBT processes from conception through implementation and validation. The hardware test bed, though created and used for graduate research, is also invaluable as an educational tool, providing a platform that supports many avenues of research and education for students with a wide variety of levels of experience, both undergraduate and graduate. A constrained hardware design helped the team quickly build the system then focus their attention on mathematical, sensing, and software research goals. Subsequent use of this platform for human subject experiments and for an undergraduate to learn about end effector DBT has demonstrated value of the project beyond the initial education described in this paper. We believe this model of graduate student mentorship on projects of use in their research can provide a valuable form of mentorship for both graduate and undergraduate students. Such projects are valuable to the mentor and mentees - we encourage their adoption as an alternative or supplement to organized student teams for DBT experience beyond the classroom.

Acknowledgements

The authors would like to thank the UROP students discussed in this paper – Jeremy Green, Kevin Matzen, Ryan Wolcott – for their hard work and efforts in helping make this hardware test bed platform a functional, useful device for research. The authors would also like to thank and Gabriel Arroyo for his subsequent extensions of MM-Arm, Chris Chartier for machining the wooden MM-Arm base mount, and David McLean for sharing his robotic hardware practical expertise and for being a sounding board for design ideas.

References

[1]Hirose, S., "Creative Education at Tokyo Institute of Technology," *International Journal of Engineering Education* [online journal], Vol. 17, No. 6, 2001, pp. 512-517, URL: http://www.ijee.dit.ie/articles/Vol17-6/IJEE1215.pdf [cited 3 November 2009].

[2]Kitts, C., "An aggressive robotics development program for integrative undergraduate education," Santa Clara University, Santa Clara, CA, 2003 [online], URL: http://hubbard.engr.scu.edu/docs/pubs/2003/03-An_Aggressive_Robotics.pdf [cited 3 November 2009].

[3]Fiorini, P., "LEGO kits in the lab [robotics education]," *IEEE Robotics & Automation Magazine* [online], Vol.12, No.4, Dec. 2005, pp. 5, URL: http://ieeexplore.ieee.org.proxy.lib.umich.edu/stamp/stamp.jsp?arnumber=1577016&isnumber=33338 [cited 3 November 2009].

[4]Nagai, K., "Learning while doing: practical robotics education," *IEEE Robotics & Automation Magazine* [online], Vol. 8, No. 2, June 2001, pp. 39-43, URL: http://ieeexplore.ieee.org.proxy.lib.umich.edu/stamp/stamp.jsp?arnumber=932756&isnumber=20189 [cited 3 November 2009].

[5]Bruder, S. and Wedeward, K., "An Interactive Online Robotics Course," Intelligent Automation and Soft Computing [online], Vol. 13, No. 1, 2007, pp. 105-116, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.5890&rep=rep1&type=pdf [cited 3 November 2009].

[6]Sutherland, K. T., "Undergraduate robotics on a shoestring," *IEEE Intelligent Systems and Their Applications* [online], Vol. 15, Issue 6, Nov.-Dec. 2000, pp. 28-31, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00895855 [cited 3 November 2009].

[7]Washabaugh, P. D., Olsen, L. A., and Kadish, J. M., "An Experiential Introduction to Aerospace Engineering," 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 8-11 2007, (AIAA-2007-296).

[8]Lentz, J., Maulucci, R. A., and Eckhouse, R. H., "Real-Time Devices at Practical Prices: Low Cost Laboratory Projects," Third IEEE Real-Time Systems Education Workshop, 1998 [online], URL: http://www.cs.umb.edu/~eckhouse/RTpaper.pdf [cited 3 November 2009].

[9]Greenwald, L., and Kopena, J., "Mobile robot labs," *IEEE Robotics & Automation Magazine* [online], Vol.10, No.2, June 2003, pp. 25-32, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1213613&isnumber=27286 [cited 3 November 2009].

[10]Maxwell, B. A., Meeden, L. A., "Integrating Robotics Research with Undergraduate Education," IEEE Intelligent Systems [online], Vol. 15, No. 6, Nov/Dec 2000, pp. 22-27, URL: http://www.cs.swarthmore.edu/~meeden/papers/IEEErobot-education.pdf [cited 3 November 2009].

[11]McGhan, C. L. R., and Atkins, E. M., "Physically-Proximal Human-Robot Collaboration: Enhancing Safety and Efficiency Through Intent Prediction," in Proc. Infotech@Aerospace Conference, Seattle, WA, Apr. 2009.

[12]Nickels, K., "Hand-Eye Calibration of Robonaut," *San Antonio Chapter IEEE Computer Society Past Meeting Presentations September 16, 2004* [online archive], URL: http://www.ieee-cs-cts.org/past_meetings.htm [cited 3 November 2009].

[13]Ransan, M., and Atkins, E., "A Collaborative Model for Astronaut-Rover Exploration Teams," AAAI-2006 Spring Symposium on Human-Robot Teams, Palo Alto, CA, March 2006.

[14]Craig, John J., *Introduction to Robotics: Mechanics and Control, 3rd ed.*, Pearson Education, Inc., Upper Saddle River, NJ, 2005, pp. 28, 35-36, 68-69.

[15]E. Atkins, E., Green, J., Yi, J., Wu, H., Browne, J., Mok, A., and Xie, F., "The TableSat Platform and its Verifiable Control Software," Infotech@Aerospace Conference, AIAA, Seattle, WA, Apr. 6-9, 2009 (AIAA-2009-2021).