ME218c Smart Product Design

ME 218c Spring 2012 Project
Zombies on the High Seas

Project Preview on May 24, 2012 6-10 PM in SPDL.
Grading Session on May 30, 2012 3 PM-6 PM in Terman Pond.
Project Presentations on May 31, 2012 in Terman Pond starting at 4:00 PM.

Goal:

The goal of this project is to provide a framework in which you can apply your knowledge of microcontrollers and multi-processor communications to a task that will provide an enjoyable experience for the users and the observers.

Purpose:

The underlying purpose of this project is to provide you with an opportunity to gain experience in integrating all that you have learned in the ME218 course sequence, with an emphasis on the new material in ME218c.

The Task:

Design and build a tele-operated Human/Zombie Vehicle (HZV) and a companion Living/Undead Controller (LUC). Groups of HZVs will operate in Terman Pond. During rounds of the game the Zombie vehicles will attempt to convert the Human vehicles into Zombies and the Human vehicles will attempt to capture supplies from floating islands.

Specifications

General:

- ☐ Each team will construct an HZV and an LUC.
- ☐ The HZVs are devices capable of navigating in Terman Pond while it is filled with water while popping balloons carried on other HZVs and on the Supply Depots.
- ☐ The LUCs are the wireless remote controllers for the HZVs.

Basic Game Play:

- ☐ A game round will be a competition between two fleets, each composed of a number of HZV/LUC pairs. The makeup of the fleets (Human vs. Zombie) will vary for each round.
- ☐ The goal of the game for the Zombies is to convert the Human fleet to Zombies while the goal of the game for the Human fleet is to "gather supplies" from the Supply Depot(s).
- ☐ The game will continue until either all of the Humans have been converted to Zombies (Zombies win) or all of the supplies have been retrieved from the Depot(s) (Humans win).

The Field:

- ☐ The Field is comprised of a region at the East end of Terman Pond, initially measuring approximately 26 ft. wide by 30 ft. long (see Figure 1). One fountain spout will be located at the approximate center of the Field. Every effort will be made to ensure that the fountain will be off during grading sessions and public presentations.
- ☐ The absolute boundaries of the Field are formed by the cement border to the pond.
- ☐ At the beginning of each game all participating HZVs will be placed in a physical arrangement at the whim of the teaching staff. One possible arrangement is shown in Figure 1.

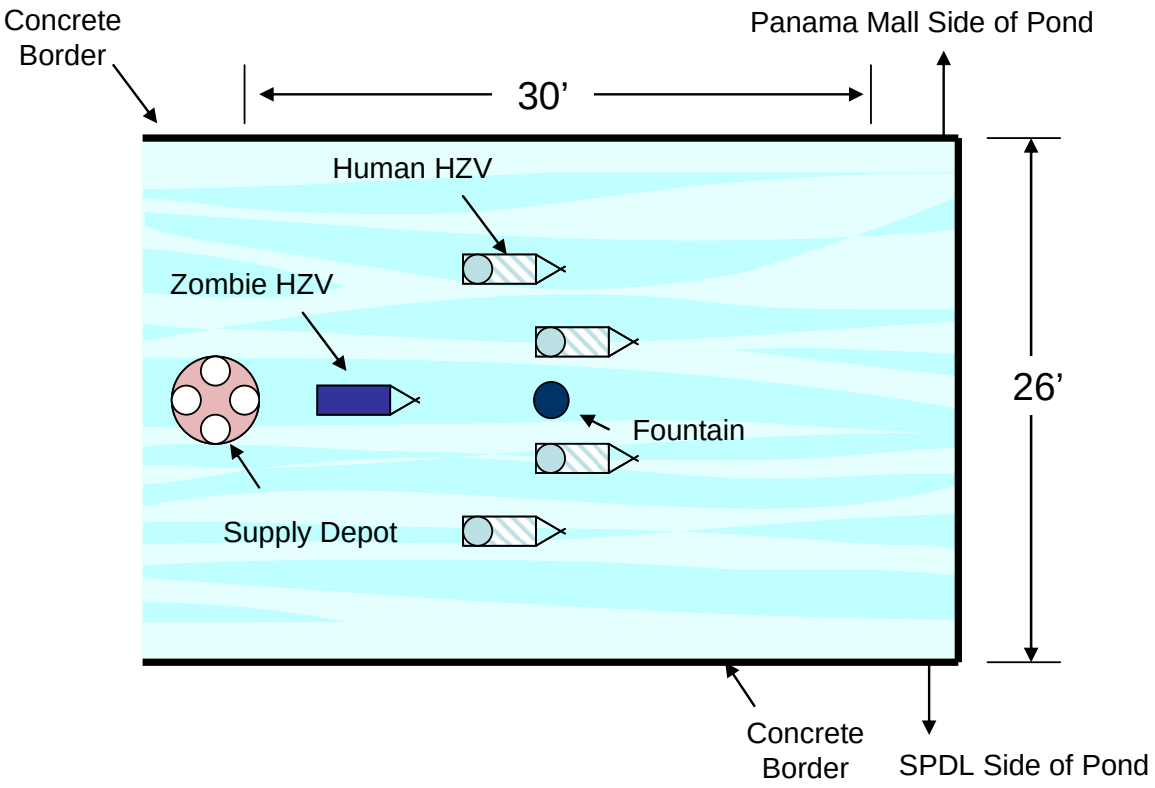

Figure 1: Field layout and dimensions with sample initial configuration

The Supply Depots:

- ☐ Each Supply Depot will consist of an approximately 16" diameter foam disk on which are mounted multiple inflated balloons. Each balloon represents a cache of supplies for the Human vehicles.
- ☐ An HZV (in Human form) can retrieve a cache of supplies from a Supply Depot by popping a balloon.
- ☐ The Supply Depot(s) will be floating freely in the pond, one possible configuration is shown in Figure 1.
- ☐ Being undead, the Zombies have no need of the supplies and may not disturb the Supply Depot contents in any way.

The HZVs:

- ☐ Each HZV must be capable of moving under its own power within the field (described above). Terman Pond will be filled with water (its normal state) at the time of the grading and the public presentation. Every effort will be made to ensure that the fountains are **OFF** during the events.
- ☐ HZVs must be battery powered and operate without a tether.
- ☐ Control of HZV functions must be achieved via an LUC using the provided RF hardware (XBee24 modules).
- ☐ Each HZV must carry a highly visible electro-mechanical indicator of its control status (under control or seeking a controller). This indicator must be under software control and clearly visible in sunlight at a distance of 20'.
- ☐ HZVs must incorporate an easily accessible switch that disables all moving systems.
- ☐ Each HZV must carry an SPDL standard balloon holder for the SPDL supplied balloons. This holder must be mounted at the aft end of the vehicle and be positioned such that at a height of 3" above the water line the balloon surface is at or beyond the outer perimeter of the bumper (see below).

- ☐ The perimeter of the largest normal projection of the HZV into the plane of the water surface must not exceed 72". Height is not restricted.
- ☐ HZVs must be stable in the presence of a 30mph wind.
- ☐ The only thing (other than the balloon) allowed to extend beyond the perimeter is an LUC controlled balloon popping mechanism. This mechanism may extend **once**, no more than 3" beyond the perimeter, and may be deployed for no more than 2 seconds at a time, in response to a discrete action by the LUC's operator.
- ☐ HZVs must incorporate a class standard foam bumper around their perimeter, and must be tolerant of moderate bumping from other HZVs. The bottom edge of the foam bumper must be at a height between 0" (the water line) and 1.5" above the water line of the HZV. The bumper must follow the same perimeter that is measured to fit the 72" requirement.
- ☐ Each HZV must carry a highly visible number that will be assigned to each team
- ☐ The HZV must enforce Zombie performance degradation (See Necromancer & Game Details sections).
- ☐ The HZV may issue messages at a rate no greater than 5 Hz.
- ☐ If the HZV fails to receive a message from its controller for 1 second, it will assume that there is a problem and revert to the controller search process described under Game Details.

The LUCs:

- ☐ Each team will design and construct an LUC that will relay commands from a human operator to an HZV, and receive and display status information from the HZV.
- ☐ The LUC must be capable of displaying to the operator an indication of active communication with its associated HZV.
- ☐ The LUC must provide a method for the operator to use to indicate which HZV the operator would like to control.
- ☐ Each command to the HZV to deploy its balloon popper must be initiated by a user action at the LUC.
- ☐ LUCs must be battery powered, and shall have sufficient battery capacity for at least 8 hours of continuous operation. The report should show documentation and calculations to support meeting this requirement.
- ☐ LUCs must be untethered and portable by one person.
- ☐ Input to the LUC should involve at least 3 sensing modalities (e.g. position, force, audio, acceleration, etc.). Use of unusual interface methods is encouraged.
- ☐ The actions required by the user of the LUC to issue commands to the HZV should be inventive and interesting for the audience to watch. Use of actions that make the operator look and feel foolish is encouraged.
- ☐ The LUC may issue commands to an HZV at a rate no greater than 5 Hz.
- ☐ LUCs should be intuitive to operate, and/or have sufficient visual instructions that a typical spectator (even a non-engineer) would be able to learn its controls within the time span of a single game round.

The Balloon Monitor:

- ☐ The Balloon Monitor will report the status of the balloon (intact or popped) when queried by your HZV. The connections to the Balloon Monitor are as shown below:

  Pin 1: Pwr, Pin 2: Gnd, Pin 3: Data, Pin 4: Query

- ☐ A new status report will be made each time the query line is pulled from high to low and held low for at least 1mS.

- ☐ The status message will be a TTL level (0-5V) asynchronous serial data stream in 96008N1 format. The data frame will be an 8-byte frame of the form:

  0x02 AA BB CC DD EE CK 0x03

  Where, for an intact balloon status, AA = 0x48, BB = 0x75, CC = 0x6d, DD = 0x61, EE = 0x6E and CK = 0x5F and, for the popped status, AA = 0x5A, BB = 0x6f, CC = 0x6d, DD = 0x62, EE = 0x69 and CK = 0x53. For both cases the CK value is a checksum that is calculated according to AA $\oplus$ BB $\oplus$ CC $\oplus$ DD $\oplus$ EE. In an uncorrupted message, the XOR of the 5 data bytes and the checksum will produce a result of 0.

- ☐ Testing the validity of the checksum to detect a corrupted message will be required.

The Necromancer:

- ☐ The Necromancer will be an SPDL supplied source of radio messages directed to individual members of the Zombie fleet.
- ☐ These messages from the Necromancer will be used to modify the performance of a Zombie HZV.
- ☐ The default degraded state of max 50% speed may be changed at any time to enhance performance (up to max 75% speed), reduce turning performance (limit turn rate in one direction to 50% of the un-degraded turn rate) or confuse the operator (swap left and right and/or forward and backward controls).
- ☐ Should a message from the Necromancer be received by a Human HZV, it should be ignored.

Game Details:

- ☐ Upon power-up or in the event of a loss of communication with its controller or conversion to a Zombie, the HZV will activate its electromechanical indication that it is searching for a controller and wait for a request for control from an LUC.
- ☐ If, upon power-up, the Balloon Monitor indicates a popped balloon, the HZV will begin the game as a Zombie with degraded performance.
- ☐ The operator of an LUC that wishes to control a particular HZV must select that HZV using the LUC and make a unique control action to initiate taking control of the HZV. This action will result in the LUC sending a directed message to the HZV requesting control of the HZV.
- ☐ The HZV will respond to the first received request for control by sending a message back to the LUC confirming control. At this time, the HZV will also de-activate its electromechanical indication of searching for a controller. At this point, the HZV is bound to that LUC until communication is lost or the HZV is converted into a Zombie.
- ☐ If an HZV receives a request for control while it is already under control, it will silently ignore the request.
- ☐ Within 100mS of when an HZV is converted into a Zombie by having its balloon popped, it must activate its electromechanical indication that it is searching for a controller.
- ☐ The newly formed Zombie will then respond to requests for control using the process described above.
- ☐ When an HZV is converted into a Zombie by the popping of its balloon, it may not immediately re-bind with the LUC that was controlling it at the time of conversion.
- ☐ When an HZV is converted into a Zombie by popping its balloon, it will have its performance degraded by moving at only 50% of its speed as a Human. Changes in the degraded performance may come in messages from the Necromancer.
- ☐ Once communication is established between HZVs & LUCs, the game will be opened and closed with a blast from an air horn.
- ☐ Prior to the start of a game, an on-deck collection of LUCs will be assembled. The operators of these LUCs will take control of the Zombies as they are created.

Radio Communications:

- ☐ Communications between the HZVs, LUCs, and Necromancer will take place over an SPDL-supplied 802.15.4 radio (Xbee24) using the Non-Beacon API mode of operation.
- ☐ Any LUC should be capable of controlling any HZV.
- ☐ Once a game begins, communication will take the form of bi-directional communications between an HZV and its bound LUC.
- ☐ Each HZV and LUC will be assigned a unique ID in the form of the source address of each SPDL-supplied radio.
- ☐ The details of the communications protocol will be defined and specified by a Communications Committee, which will consist of one member from each project team. The specification must be in a written form and with sufficient detail that someone sufficiently skilled in ME218 material could implement it.
- ☐ In order to better balance the workload and learning among team members, each of the following tasks must be completed by a different member of the team: serve on the communications committee, implement communications on the HZV, and implement communications on the LUC.
- ☐ The class communications protocol must include a procedure for validation of communication between the HZV and LUC. The LUCs must provide a visual indication of when a functioning communications link between the HZV and LUC exists.
- ☐ The class communications protocol must include provisions for 5-byte data messages from the Necromancer to a Zombie HZV.

General Requirements:

- ☐ At a minimum, either the LUC or the HZV must contain two actively communicating processors. There is no class imposed upper limit on the number of processors employed. The only processors permitted on the HZV are PIC microcontrollers.
- ☐ The microcontroller that interfaces with the Balloon Monitor must be programmed entirely in assembly language.
- ☐ You are limited to an expenditure of **$150.00/ team** for all materials and parts used in the construction of your project. Materials from the lab kit or the Cabinet Of Freedom do not count against the limit. All other items count at their fair market value.
- ☐ A project logbook must be maintained for each group. An on-line blog is appropriate to meet this requirement as long as it is made available to the teaching staff for review. This book should reflect the current state of the project, planning for the future, results of meetings, designs as they evolve etc. The project logbook will be collected at irregular intervals for evaluation.
- ☐ A report describing the technical details of the system will be required. The report should be of sufficient detail that a person skilled at the level of ME218c could understand, reproduce, and modify the design. The report must be in website format, and be suitable for posting on the SPDL site.
- ☐ HZVs based substantially on purchased vehicle platforms are not allowed.
- ☐ All projects must respect the spirit of the rules. If your team is considering something that **may** violate the spirit of the rules, you must consult a member of the teaching staff.

Safety:

- ☐ Both the HZVs and the LUCs should be safe, both to the user and the spectators. The balloon popping mechanism must be designed in such a way at to prevent injury from accidental contact with body parts.
- ☐ Intentionally disabling or damaging other HZVs is not allowed. Prohibited actions include, but are not limited to, the following: ramming at excessive speed (as determined solely at the discretion of the teaching staff), swamping and/or sinking.

- ☐ No part of the HZV may become ballistic.
- ☐ HZVs will be exposed frequently to lots of water. Electronics, actuators and energy storage devices (e.g. batteries) do not typically fare well in the presence of water. Plan on it. Design accordingly.
- ☐ All portable electronic devices must remain off and properly stowed during taxi, take-off, and landing.
- ☐ The teaching staff reserves the right to disqualify any device considered unsafe.

Check-Points

Design Review:

On **05/08/12** between 9am & 4:30pm we will conduct a design review, one team at a time. Each team should prepare a few sheets of paper showing your proposed designs for both the HZV and the LUC. You will have 5 minutes to walk us through your ideas. The focus should be on system level concepts, **not detailed hardware or software**. Proposed popping methods must be approved for safety at this time. We will spend the balance of the time-slot giving feedback and asking questions. At this time, initial calculations are required for estimating the mass of your proposed HZV as well as any water displacement calculations relevant to your design. You will present these in 550-122 or 550-126 (depending on the time-slot chosen).

First Draft of Communications Standard:

Due by 5:00 pm on **05/09/12**. Ed will meet with the communications committee on the evening of **05/10/12** to provide feedback on the specification.

Communications Standard:

Due by 5:00 pm on **05/11/12**. This is the working draft of the communications standard.

First Check-Point:

On **05/15/12**, you must demonstrate

1) The ability of the HZV to receive and correctly decode and respond to commands from the LUC (simulated inputs are acceptable at this time).

2) That your HZV platform has been built, and is capable of bearing the approximate weight of all the necessary components it will carry when complete. It is encouraged, but not required, to demonstrate working propulsion and steering subsystems.

3) The ability to communicate with the Balloon Monitor and report the balloon status.

The final working version of the communications standard is due. No further changes are allowed to the standard. This protocol will be evaluated with respect to its completeness and suitability for the proposed system. **Note:** this is a functional evaluation only. The focus should be on demonstrating **functional** hardware and software.

Second Check-Point:

On **05/22/12**, you must demonstrate the ability to communicate all required functionality between your HZV and LUC. This will include commands from the LUC to the HZV and all status messages from the HZV to the LUC.

Project Preview:

At the Project Preview on **05/24/12**, each team must demonstrate (in addition to the 1st & 2nd check-points' functionality)

1) The ability to successfully send and execute the drive and steering commands (including the actuation of all electromechanical outputs) from an operator of the LUC to the HZV.

Grading Session:

During the Grading Session on **05/30/12**, each team will be required to demonstrate the ability to successfully participate in a game. This will include

1) Establishing communications between your HZV and LUC and between your HZV and the LUC from another team.

2) Navigating an HZV from the initial position and successfully popping at least two balloons on a Supply Depot.

2) Displaying on both the LUC and the HZV the correct status of communications.

Public Presentation:

This will take place on **05/31/12** starting at 4:00 pm in Terman Pond. At this event, members of the public will be allowed to act as operators of the LUCs.

Report:

Draft due on **06/4/12** by 4:00 pm. The final version (with revisions incorporated) is due by 5:00 pm on **06/08/12**.

Evaluation

Performance Testing Procedures:

One or more of the team members will demonstrate the HZV and LUC during the first & second check points and project preview. Members of the teaching team will operate the HZV and LUC during the grading session.

Grading Criteria:

- ☐ **Concept (15%)** This will be based on the technical merit of the design and coding for the machine. Included in this grade will be evaluation of the appropriateness of the solution, as well as innovative hardware, software and use of physical principles in the solution.
- ☐ **Implementation (15%)** This will be based on the prototype displayed at the evaluation session. Included in this grade will be evaluation of the physical appearance of the prototype and quality of construction. We will not presume to judge true aesthetics, but will concentrate on craftsmanship and finished appearance.
- ☐ **First Check Point (10%)** Based on the results of the performance demonstrated on 05/15/12**.**
- ☐ **Second Check Point (10%)** Based on the results of the performance demonstrated on 05/22/12**.**
- ☐ **Preliminary Performance (10%)** Based on the results of the performance demonstrated during the Project Preview.
- ☐ **Performance (15%)** Based on the results of the performance testing during the Grading Session.
- ☐ **Report (10%)** This will be based on an evaluation of the report. It will be judged on clarity of explanations, completeness and appropriateness of the documentation.
- ☐ **Report Review (5%)** These points will be awarded based on the thoroughness of your review of your partner team's report. Read the explanations, do they make sense? Review the circuits, do they look like they should work?
- ☐ **Log Book (5%)** This will be evaluated by the evidence of consistent maintenance as well as the quality and relevance of the material in the log book.
- ☐ **Housekeeping (5%)** Based on the timely return of SPDL components, cleanliness of group workstations as well as the overall cleanliness of the lab. No grades will be recorded for teams who have not returned all loaned materials.

Gems of Wisdom from Prior Generations

- Get the radio working with the 'E128 first.
- Do not continue working until the wee hours of the morning unless you absolutely have to because errors propagate when tired. A fresh look at things in the morning will save you a lot of pain at night. Sleep is not a crutch, it is a necessity.
- Put some time into your first prototype. You might be surprised how many things you throw together for testing purposes make it into your final project.
- Label or color-code your connectors so that it's easy to plug them into the right place. Connectors that can only be hooked up one way (such as Molex) prevent undesirable incidents like reversing the voltage and ground connections and frying components in the process.
- When building networks, add nodes one at a time to better track down "bad nodes".
- Debugging LEDs are useful for getting feedback on the operational state of PICs.
- A "power central" board is a good thing to have, particularly if you're dealing with multiple supply voltages. This makes the circuitry cleaner, and can save you from supplying your PIC with 37 volts.

- Think twice before planning to provide PWM for motors with PICs (at least the one with 20 pins). You will need to take care of output compares and timers also.
- You will need to leave some pins on PICs (especially those with only 20 pins) open for debugging.
- Don't hesitate to add another PIC and SPI communication. It's really easy.
- Using shift registers for debugging can also be a helpful trick to obtain more information, but it is not good for timing issues.
- Try working during the day (seriously!). Debugging is way easier with a clear head.

- The radio boards runs on 3.3V not 5V. The iButton reader runs on 5V not 3.3V. Design your circuits accordingly and be ready to convert between the two voltages.
- Jameco is NOT OPEN on weekends. Don't postpone your trip until Saturday – you will be sorely disappointed.
- Don't be dead set on a theme at the beginning of the project. Let the project theme develop as you move through the project. You'll be surprised how many great ideas pop up as you go along.
- Hot glue down all soldered wire connections. You'll lose a lot of time tracking down an error that may end up being a loose/broken wire.
- Write all functions as non-blocking code – no matter where they fit into the flow of the program.
- Just because two points on a circuit look like ground when probed doesn't mean they are connected.

- Test circuit as it will be implemented in final form, as well as fully integrated.
- Test in environment in which hardware will be used (radios outside, with appropriate distant in between).
- Testing our radio pair in the presence of other active radio pairs revealed problems that didn't exist when we test alone.
- It's easy to make a design with bad ergonomics which make it impossible for the user to perform the task. Prototype/try out the user scenario yourself as early as possible.
- Keep circuit diagrams up to date as you make them.
- Use lab notebook so that all information is at one place and teammates can have easy access to it.
- Take a lot of pictures as you go.
- Remember to HAVE FUN.

- If you are having intermittent problems (e.g. it works only some of the time) check your connections – especially those connecting your various circuits to a common ground.
- Modularize as much as possible – test all of the components separately before integrating
- Build and test all of your circuits and sensors on breadboard before you make them hard mounted on perfboard.
- When moving your circuits from breadboard to perfboard, rather than dismantling your breadboards, leave your working breadboards intact and buy new components and build entirely new circuits on the perfboard. That way, if something goes wrong once everything is built, you will always have a backup copy of your circuits on the breadboard that you know worked before you integrated everything.
- Isolate your circuits onto individual perf-boards (rather than having a giant perf-board with all of your circuits). Makes it much easier to take them out to debug them.
- A nice pair of wire snips (flush cutters) and wire strippers makes wirewrapping and circuit building in general much easier.
- Do your circuit calculations to make sure you have enough/not too much voltage/current/power
- Have plenty of spare parts ready to go in case something blows at the last minute
- Always take the time to test on the actual competition field at the actual competition location
- Make sure at least two people of the group understand or at least have an idea of each component – mechanical, electrical, software. Doesn't have to be the same two people, but it insures that if someone's missing, that the group isn't stuck.
- Be friendly with other teams – you never know when you're going to need help.

- Test your wireless communication outside and at range!
- Test your components for interoperability with everyone else's before game day.
- Source an off-the-shelf housing/controller and gut it. That way you can focus on the electronics and not the mechanical design.
- If you can avoid having to spend time building something by buying an equivalent part, do it!
- Remember your banksel commands and save yourself hours of debugging.
- Make things accessible (i.e. batteries, boards, DIP sockets, etc.) so you don't have to unscrew things when you need to test, power cycle, or reset things.
- Learn to use assembler macros. They clean up your code visually.

- Don't use macros when you can use a function-type subroutine!
- Size does matter. The bigger or larger the motions involved in your controller, the more entertaining it will be to watch.
- Do not bury your wireless antenna in a box. Try to keep it out in the open.
- Buy a good pair of wire strippers, preferably ones that can strip 30AWG wire. Your fingers will thank you.
- Try to avoid having to scramble together parts or code for a check-off. This means keeping on top of the project schedule. If you don't, you will end up throwing away a lot of hours on setups that will not make it to your final design.
- PICs are notoriously difficult to debug. Either source an MPLAB ICD2 (or equivalent), or finish your circuitry early, before writing the bulk of your code. You will find that changing even small things in assembler can cost you hours.

- If a change causes things not to work the first thing you should check is if the code is in the correct bank. It is always a good idea to use a bank select command at the start of every routine rather than assume you'll know where it is.
- Build and test the code in small manageable pieces. If a lot of changes are made at once and the new program doesn't work, it is very hard to isolate the problem without a lot of work.
- Use the debugger. Running routines through the debugger to see what will happen will save lots of time and effort. Getting a routine to work in the debugger usually allows you to assume problems that come up in actual testing are hardware rather than software related.
- When you're tired and everything starts to fail don't forget to check the batteries.
- If you're tired and everything starts to fail and it's not the battery consider going home and looking at it again the next morning rather than changing a lot of code. Often it is some small little change you overlooked and are too tired to notice.
- Make sure all data tables are in the correct location.
- Make use of calls and macros whenever possible to keep the code clean. This also makes repetitive actions easier to code and change.
- Make use of #defines for labeling pins and value as much as possible. This makes it very easy to see what pins are connected to what and allows for the easiest changes. Rather than searching for a specific port and pin throughout the code you only have to change one #define value.

- Don't believe anyone that tells you that the 218C project is less time consuming than the 218b project. It's not.
- Pick your battles early. Learning to program the PIC's and the Zigbees is a lot of work on its own. Trying to add other challenges can be tough.
- Move to solder boards or wire wrap boards as soon as you can. If you are developing simple hardware that you understand well, don't be afraid to solder it on a board. Troubleshooting bad connections on a breadboard is a waste of your time.
- Allocate your pins and subsystems early. A spreadsheet that shows all of your pins is very handy.
- Practice on the course as soon as possible to test your operable range.

- PICs are apparently not designed to be inserted backwards. We recommend against doing this.
- Don't spend more than a few hours debugging SPI code before debugging all of the related hardware.
- Start by making a schedule for the project and include any outside events like vacations, graduations, etc. to avoid surprises later on.
- Black objects left in the sun tend to melt any hot glue that is exposed. This is detrimental to the project's structural integrity. It is therefore wise to a) avoid hotglue or more realistically, b) avoid leaving black hotglued objects in the full sun for extended periods of time.

- Although checkpoints are important, the key is to continue working on the final product, so at all times try to write code/build hardware that you will be able to use in the final product. Try to minimize writing special "check-off code" and building "check-off hardware" that you won't use later.
- Thinking very carefully about your electrical design/layout will save you lots of soldering time. By designing carefully, you'll optimize locations of every components, and you'll end up making a lot less solder joints/connectors/electrical boards, fewer corrections.
- Use Debugging Leds if using PICs. Reserve a few outputs so you can toggle the bits and see if you get into loops or states. This was really helpful when we were trying to figure out what was wrong with our code. Also, since we already used the SSP and Asynchronous communications outputs, we could not use printfs to the terminal.
- Check your #defines and labels. With PIC programming, you tend to have a lot of GOTOs and CALLs which means you need a lot of labels. Try to have a good system for labeling things and creating your constants and variables. We used CAP_UNDERSCORE for # defines and FirstCapitalLetter with no spaces for variables. Where we went wrong was creating "FORWARD" and "FORWARD_CMD" which we misinterpreted and messed us up for a long time.
- Talk to other people about the communication protocols and how they implement their code. It's hard to figure out the datasheets by yourself with no help from anyone.
- Debug code extensively prior to integration with other software/hardware elements.
- Utilize 7-segment display or LCD display for real-time debugging.

- Modularize mechanical systems so that simpler parts can be made earlier and used from the beginning of development.
- Develop a clear understanding of the communications protocol from the beginning of code development.
- Communicate well within your team so that some tasks are not overlooked, while others are duplicated.
- Learn some of the common problems with writing PIC code (such as dividing your long code into small sections using the "Maincode code" command).
- Bathe as frequently as possible; encourage others to do so as well.