ME218c
Smart
Product
Design

ME 218c Spring 2015 Project
Revenge of the DrEd

Project Preview on May 21, 2015 6-10 PM in SPDL.
Grading Session on May 26, 2015 3 PM-6 PM in DrEd Swamp.
Project Presentations on May 27, 2015 in DrEd Swamp starting at 6:00 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.

Background:

When we last saw DrEd he was in the basement of his lair, right before it exploded. Everyone thought that he was dead and ME218c would be an easy A. Not so!

He escaped though an underground series of tunnels where he has been tirelessly scheming to develop his Ultimate Weapon. Reliable sources place DrEd in a remote swamp, though no one who has gone to investigate has returned. Rumor has it that he has now developed the ability to take control of someone's mind with a single injection of a solution of carbonated water, carmel color, aspartame, phosphoric acid, potassium benzoate (to protect taste), natural flavor, citric acid and caffeine.

Due to toxic leakage from DrEd's facility, the swamp is impassible to all conventional vehicles. The only forays that have penetrated the perimeter of the swamp have been conducted by hovercraft which are able to remain above the industrial sludge but below the radar systems that DrEd uses to scan for intruders. DrEd is thought to have his own fleet of hovercraft that have been captured from prior missions and have been converted by DrEd to Mind-Controlled vehicles.

Queue branch has commissioned teams of intrepid engineers to develop a fleet of hovercraft to pierce DrEd's security cordon and vanquish DrEd once and for all.

The Task:

Design and build a tele-operated Swamp Penetrating Electronically Controlled Tactical Reconnaissance Explorer (SPECTRE) and a companion Mechatronic Integrated Interactive Intelligent Infrastructure Intrusion Interface (MI6). Groups of SPECTREs will operate in ~~Terman~~ DrEd's ~~Pond~~ Swamp. During rounds of the game the SPECTREs will attempt to reach DrEds mothership (the DrEdStar) and disable it. They will be opposed by DrEd's flotilla of Mind-Controlled SPECTREs.

Specifications

General:

- ☐ Each team will construct a SPECTRE and an MI6.
- ☐ The SPECTREs are devices capable of navigating in DrEd's Swamp while popping balloons carried on other SPECTREs and on the DrEdStar.
- ☐ The MI6s are the wireless remote controllers for the SPECTREs.

Basic Game Play:

- ☐ A game round will be a competition between two fleets, each composed of a number of SPECTRE/MI6 pairs. The makeup of the fleets (Free vs. Mind-Controlled) will vary for each round.
- ☐ The goal of the game for the free SPECTREs is to disable the DrEdStar while the Mind-Controlled SPECTREs will attempt to take control of the free SPECTREs.
- ☐ The game will continue until either all of the SPECTREs are Mind-Controlled (DrEd wins) or the DrEdStar has been destroyed (DrEd loses).

The Field:

- ☐ The Field is comprised of a region at the West end of DrEd Swamp, 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.
- ☐ The absolute boundaries of the Field are formed by the cement border to the pond.
- ☐ At the beginning of each game all participating SPECTREs will be placed in a physical arrangement at the whim of the teaching staff. One possible arrangement is shown in Figure 1.

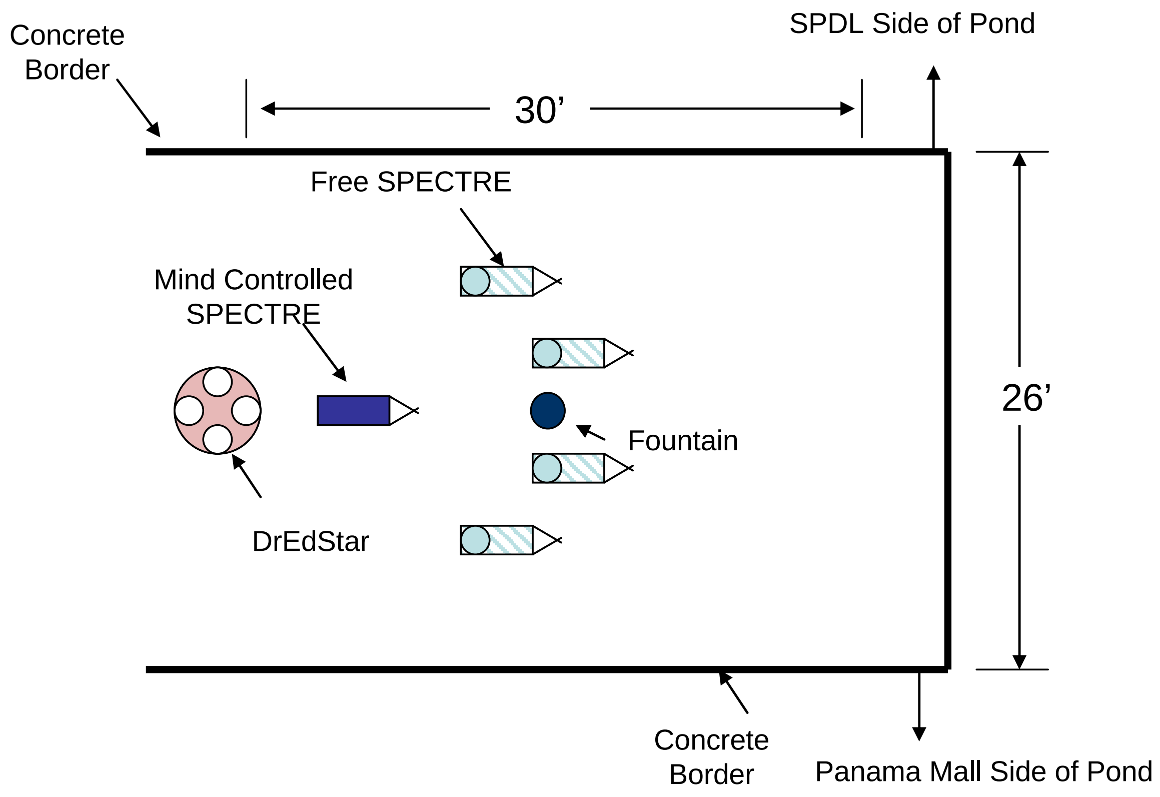

Figure 1: Field layout and dimensions with sample initial configuration

The DrEdStar:

- ☐ The DrEdStar will consist of an approximately 16" diameter hovercraft on which are mounted multiple inflated balloons. Each balloon represents a layer of DrEds defenses.
- ☐ A free SPECTRE can disarm a layer of defenses by popping a balloon.
- ☐ The DrEdStar will be floating freely in the swamp; one possible configuration is shown in Figure 1.
- ☐ The Mind-Controlled SPECTRESs, being minions of DrEd, will not want to damage the DrEdStar and should not disturb the DrEdStar defenses in any way.

The SPECTREs:

- ☐ Each SPECTRE must be capable of moving under its own power within the field (described above). DrEd Swamp will not be filled with water (its normal state given the drought) 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.
- ☐ The SPECTRE must hover above the surface of the swamp, and the only source of lift allowed is the lift fan provided by the SPDL.

- ☐ SPECTREs must be battery powered and operate without a tether. NiCd or NiMH batteries are the only approved power sources.
- ☐ The lift fan must only be powered through the series connected pair of SPDL supplied diodes. **Do Not connect the lift fan directly to batteries or the power supply.** The lift fan is not rated for the battery voltage.
- ☐ Propulsion and steering systems are free with the caveat that they may not directly apply force to the swamp. An exception to this rule allows for dragging an element against the swamp to brake or steer. This will be acceptable as long as the contact mechanism would, if deployed while the SPECTRE was not moving, create approximately equal drag in all directions and is not capable of supporting the vehicle.
- ☐ Control of SPECTRE functions must be achieved via an MI6 using the provided RF hardware (XBee24 modules)
- ☐ The perimeter of the SPECTRE must be largely convex. Whether or not a perimeter is largely convex will be determined by moving a straight-edge around the perimeter at the height of the bumper. If the contact between the straight-edge and the perimeter resolves to 2 or more points (rather than a single point), then the maximum distance between the straight-edge and the perimeter of the vehicle must not exceed 1/(distance between contact points).
- ☐ Each SPECTRE 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'.
- ☐ The SPECTRE must be capable of navigating across a Context Engineering Cord Protector (sample provided in SPDL).
- ☐ SPECTREs must incorporate an easily accessible switch that disables all moving systems.
- ☐ Each SPECTRE 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 swamp the balloon surface is at or beyond the outer perimeter of the bumper (see below).
- ☐ The perimeter of the largest normal projection of the SPECTRE into the plane of the swamp must not exceed 72". Height is not restricted.
- ☐ SPECTREs must reliable in the presence of fog (which often occurs in swamps).
- ☐ The only thing (other than the balloon) allowed to extend beyond the perimeter is an MI6 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 MI6's operator.
- ☐ SPECTREs must incorporate a class standard foam bumper around their perimeter, and must be tolerant of moderate bumping from other SPECTREs. The bottom edge of the foam bumper must be at a height between 0" (the swamp line) and 1.5" above the swamp surface. The bumper must follow the same perimeter that is measured to fit the 72" requirement.
- ☐ Each SPECTRE must carry a highly visible number that will be assigned to each team
- ☐ Every SPECTRE must be controllable through any of the MI6s via the class-wide protocol (See Radio Communications in a later section).
- ☐ The SPECTRE may issue messages at a rate no greater than 5 Hz.
- ☐ If the SPECTRE 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.
- ☐ Since DrEd's swamp is villainously gloomy, it is suggested that teams incorporate running lights to aid operation of their SPECTREs in varying lighting conditions.

The MI6s:

- ☐ Each team will design and construct an MI6 that will relay commands from a human operator to a SPECTRE, and receive and display status information from the SPECTRE.
- ☐ The MI6 must be capable of displaying to the operator an indication of active communication with its associated SPECTRE.
- ☐ The MI6 must provide a method for the operator to use to indicate which SPECTRE the operator would like to control.
- ☐ Each command to the SPECTRE to deploy its balloon popper must be initiated by a user action at the MI6.
- ☐ MI6s 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.
- ☐ MI6s must be un-tethered and portable by one person.
- ☐ Input to the MI6 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 MI6 to issue commands to the SPECTRE should be inventive and interesting for the audience to watch. Use of actions that make the operator look and feel foolish is encouraged.
- ☐ The MI6 may issue commands to a SPECTRE at a rate no greater than 5 Hz.
- ☐ MI6s 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 SPECTRE. 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 9600 8N1 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 for the correct packet format and the validity of the checksum to detect a corrupted message is required.
- ☐ The microcontroller that interfaces with the Balloon Monitor must be programmed entirely in assembly language.

Game Details:

- ☐ Upon power-up or in the event of a loss of communication with its controller or conversion to Mind-Controlled, the SPECTRE will activate its electromechanical indication that it is searching for a controller, deactivate its lift fan and wait for a request for control from an MI6.

- ☐ If, upon power-up, the Balloon Monitor indicates a popped balloon, the SPECTRE will begin the game as a Mind-Controlled minion of DrEd.
- ☐ The operator of an MI6 that wishes to control a particular SPECTRE must select that SPECTRE using the MI6 and make a unique control action to initiate taking control of the SPECTRE. This action will result in the MI6 sending a directed message to the SPECTRE requesting control of the SPECTRE.
- ☐ The SPECTRE will respond to the first received request for control by sending a message back to the MI6 confirming control. At this time, the SPECTRE will also de-activate its electromechanical indication of searching for a controller. At this point, the SPECTRE is bound to that MI6 until communication is lost or the SPECTRE becomes Mind-Controlled.
- ☐ If a SPECTRE receives a request for control while it is already under control, it will silently ignore the request.
- ☐ Within 100ms of when a SPECTRE becomes Mind-Controlled by having its balloon popped, it must activate its electromechanical indication that it is searching for a controller and deactivate its lift fan.
- ☐ The newly controlled minion will then respond to requests for control using the process described above.
- ☐ When a SPECTRE is converted into a DrEd minion by the popping of its balloon, it may not immediately re-bind with the MI6 that was controlling it at the time of conversion.
- ☐ Once communication is established between SPECTREs & MI6s, 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 MI6s will be assembled. The operators of these MI6s will take control of the DrEd minions as they are created.

Radio Communications:

- ☐ Communications between the SPECTREs, and MI6s will take place over an SPDL-supplied 802.15.4 radio (Xbee24) using the Non-Beacon API mode of operation.
- ☐ Any MI6 should be capable of controlling any SPECTRE.
- ☐ Once a game begins, communication will take the form of bi-directional communications between a SPECTRE and its bound MI6.
- ☐ Each SPECTRE and MI6 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 SPECTRE, implement communications on the MI6, and implement the communications with the Balloon Monitor.
- ☐ The class communications protocol must include a procedure for validation of communication between the SPECTRE and MI6. The MI6s must provide a visual indication of when a functioning communications link between the SPECTRE and MI6 exists.

General Requirements:

- ☐ At a minimum, either the MI6 or the SPECTRE must contain two actively communicating processors. There is no class imposed upper limit on the number of processors employed.
- ☐ You are limited to an expenditure of **$200.00/ team** for all materials and parts used in the construction of your project. Materials supplied to each team by SPDL, 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.
- ☐ SPECTREs 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 SPECTREs and the MI6s 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 SPECTREs 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 SPECTRE may become ballistic.
- ☐ SPECTREs will likely be exposed to fog in DrEd's Swamp. At times, this fog may be quite thick. Electronics, actuators and energy storage devices (e.g. batteries) do not typically fare well in the presence of water (which will condense from the fog). Plan on it. Design accordingly.
- ☐ Approved small portable electronic devices may now be used during taxi, take-off, and landing.
- ☐ The teaching staff reserves the right to disqualify any device considered unsafe.

Check-Points

Design Review:

On **05/05/15** between 9am & 4:30pm we will conduct a design review, one team at a time. Each team should prepare a few images showing your proposed designs for both the SPECTRE and the MI6. You will have 5 minutes to walk us through your ideas. The focus should be on system level concepts, **not detailed hardware or software**. We will spend the balance of the time-slot giving feedback and asking questions. Proposed popping methods must be approved for safety at this time. In addition to your concepts, at this time you must present, in printed form, your plan for the development, integration and testing steps that you will follow to complete the project. The plan must identify what functionality you will demonstrate at the two check-points and the project preview along with the test procedures that you will use to prove that your team has met the check-point. Check-point tests must follow an incremental integration strategy with each successive check-point demonstrating all of the functionality of the prior checkpoint(s) as well as the new functionality. This plan must be approved by the teaching staff. If we feel that it is seriously flawed, we will ask you to revise and resubmit the following day. The presentations and review will take place in 550-126 or 550-162 or the Grove (low ceilinged area beyond the Atrium), depending on the time-slot chosen.

First Draft of Communications Standard:

Due by 5:00 pm on **05/06/15**. Ed will meet with the communications committee on the evening of **05/07/15** to provide feedback on the specification.

Communications Standard:

Due by 5:00 pm on **05/08/15**. This is the working draft of the communications standard.

First Check-Point:

On **05/12/15**, you must demonstrate your approved 1st check-point functionality according to your defined testing procedure.

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. You may submit for approval a final revision of your check-point plan at this time.

Second Check-Point:

On **05/18/15**, you must demonstrate your approved 1st check-point and 2nd check-point functionality according to your defined testing procedure. The functionality demonstrated at this time must include full implementation of the communications protocol.

Project Preview:

At the Project Preview on **05/21/15**, each team must demonstrate (in addition to the 1st & 2nd check-points' functionality) your approved project preview functionality. The functionality demonstrated at this time must include a demonstration of interoperability between at least 2 teams' SPECTREs and MI6s.

Grading Session:

During the Grading Session on **05/26/15**, each team will be required to demonstrate the ability to successfully participate in a game. This will include

1) Establishing communications between your SPECTRE and MI6 and between your SPECTRE and the MI6 from another team.

2) Navigating a SPECTRE from the initial position and successfully popping at least two balloons on the DrEdStar.

3) Displaying on both the MI6 and the SPECTRE the correct status of communications.

A detailed grading check-off procedure will be published at a later date.

Public Presentation:

This will take place on **05/27/15** starting at 6:00 pm in DrEd's Swamp. At this event, members of the public will be encouraged to act as operators of the MI6s.

Report:

Draft due on **06/1/15** by 4:00 pm. The final version (with revisions incorporated) is due by 5:00 pm on **06/05/15**.

Celebration:

A celebration of the past 3 quarters of ME218 will take place at the Alpine Inn on **06/4/15** starting at 3:00 pm. Mark your calendars now and save the date.

Evaluation

Performance Testing Procedures:

One or more of the team members will demonstrate the SPECTRE and MI6 during the first & second check points and project preview. Members of the teaching team will operate the SPECTRE via the MI6 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 05/12/15**.**
- ☐ **Second Check Point (10%)** Based on the results of the performance demonstrated on 05/18/15**.**
- ☐ **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

Planning

- Plan and experiment early for mechanical structures.
- The initial planning phase is very important. Take the time to plan out the entire design before getting started in the lab.
- Design the software together as a team before coding anything. This will reduce the chances of having to redesign the software later in the project. Also, with more team members involved, the process should be quicker and everyone will be familiar with the software design.
- Put in long workdays before the last week of the project. The lab is less crowded early in the project, and work can be more productive during these times. Also, mistakes are more critical at the end of the project. Account for some mistakes.
- Make sure that your teammates can step in to work on any part of the project when needed. It is risky if there is only one expert for a given part of the project.
- Front load software development and try to hammer out comm bugs as early as possible; this will leave you time to get have fun and get creative with mechanical towards the end of the project.
- Think ahead of the physical layout, especially so you can access the batteries and important switches easily.
- You should have extra parts (just in case...).
- Try to reuse code and components from your 218b project. This saves time and money.
- Plan ahead the layout of the boards, and their placement on the bot.
- We must say it. Keep it simple.
- Don't underestimate how long decorations might take (and be careful if you're overtired).
- At the first design meeting, figure out what the simplest way to meet all the requirement is. Then add bits and pieces that you just think are cool or fit your awesome theme.
- On that note, pick an awesome theme and have some fun with it!
- The controller takes just as much time (if not more) than the thing that moves around. Don't neglect it.
- 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.
- 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.
- Allocate your pins and subsystems early. A spreadsheet that shows all of your pins is very handy.
- Size does matter. The bigger or larger the motions involved in your controller, the more entertaining it will be to watch.
- 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.
- Start by making a schedule for the project and include any outside events like vacations, graduations, etc. to avoid surprises later on.
- 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.

Mechanical

- Iterate on your hovercraft design early to figure out what works in terms of skirts, weight balance, and propulsion systems.
- Rapid prototyping is extremely beneficial for mechanical and electrical design.
- Prototype quickly to test the principle first.
- Don’t be afraid to improvise if the laser cutters are not available.
- Remember to spend time on prototyping new mechanical systems. It took us multiple days and over 5 prototypes to get a good hovering platform.
- Make sure all molex connections are secure and that the crimps are providing strain relief on the wire. Especially if you're doing your friend a favor and crimping a wire for him.
- Do not make the robot too heavy. Actual robot might be way different from the prototype
- Getting it to hover is easy, getting it to go where you want is near-impossible.
- 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.
- 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.
- 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.
- Modularize mechanical systems so that simpler parts can be made earlier and used from the beginning of development.

Electrical

- If a pin does not seem to be reading or writing and you have checked your code, make sure to check you cables as they can fail too.
- Be very careful when laying out circuits to solder. Test them incrementally.
- Update electronic schematics as you go along, so any teammate can readily understand what is happening in a certain board and pick up from there.
- Never underestimate the utility of a good connection or a good molex.
- Bypass caps, as I'm sure you learned in A and B.

- 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.
- 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.
- Do your circuit calculations to make sure you have enough/not too much voltage/current/power
- Buy a good pair of wire strippers, preferably ones that can strip 30AWG wire. Your fingers will thank you.
- 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.
- PICs are apparently not designed to be inserted backwards. We recommend against doing this.
- Keep circuit diagrams up to date as you make them.

Software

- Build the software in gradual levels of complexity. Spend time at each level to make sure everything works. Knowing the base is solid helps focus debugging where you need it.
- The time it takes to cycle through the framework loop varies widely depending on the number of events in the SM queues. Keep this in mind when deciding between polling for flags or using interrupts.
- Download and use the Pro compiler for the PIC16F690, this will DRAMATICALLY cut down your program space and even some of your data space.
- Be creative with timing when resources are limited; the PIC has only one output compare but this doesn't prevent you from using timer overflows for timing.
- Remember your banksel commands and save yourself hours of debugging.
- Don't over complicate things by making state machines for everything. Many things can be done in a simple service or even in an interrupt response routine (i.e. constructing packets)
- Think carefully about your state machines before you jump to the code. See what cases would break it (and edit accordingly, of course).
- Take the time to pseudo code.
- Comment your code such that anyone could follow what you implemented.
- If an apparent software bug does not make sense, stop for a few minutes. It may be electrical.
- Initialize all your variables to some value when you declare them.
- Watch out for index out of bounds errors on arrays. These happen silently and can completely trash your program, so it is worth while to have a lot of error checking for this.
- 218C has a lot more software than 218B. Get your communication bugs ironed out early.
- Using flags as a low-overhead alternative to state machines is great when you have something relatively simple to implement. It also might be necessary if you are using the ES framework on a PIC and can't post events within an interrupt.
- If possible, consolidate states. Don't create new states for every special case.
- Write all functions as non-blocking code – no matter where they fit into the flow of the program.
- 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.
- Develop a clear understanding of the communications protocol from the beginning of code development.
- 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.
- 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.
- Make sure all data tables are in the correct location.

Debugging

- Strive to meet all the checkpoints, but do not lose sight of the end goal. Often the team should aim to achieve more than that necessary to check off.
- When stuck, consult other teams. Often, your classmates will have had the same problem as you. This can save a lot of time.
- Be willing to help others as well.
- Have an extra set of charged batteries, so you don’t have to wait to test.
- Integrate incrementally. Test incrementally.
- LEDs can be very helpful! Use them to your advantage, especially when a PIC isn't telling you much.

- You are going to love the debugger and the logic analyzer. Take advantage! But remember that's not a good idea to debug interrupts!!
- Always check if things are powered correctly, especially if you're tired.
- 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.
- Use the PICKit - it makes debugging quite easy!
- The hardware debugger reserves a few variables and adds a little bit to your program space. This means if you are at the limits of your PIC's memory, then the debugger can push you past the limit and you won't be able to compile to debug. Try using the Pro compiler to cut down the code size.
- The logic analyzer is your new best friend.
- Have you checked power and ground?
- 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.
- Debugging LEDs are useful for getting feedback on the operational state of PICs.
- Try working during the day (seriously!). Debugging is way easier with a clear head.
- Using shift registers for debugging can also be a helpful trick to obtain more information, but it is not good for timing issues inless you use SPI hardware.
- Jameco is NOT OPEN on weekends. Don't postpone your trip until Saturday – you will be sorely disappointed.
- Just because two points on a circuit look like ground when probed doesn't mean they are connected.
- 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
- When you're tired and everything starts to fail don't forget to check the batteries.
- Debug code extensively prior to integration with other software/hardware elements.
- Utilize 7-segment display or LCD display for real-time debugging.
- 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.
- You will need to leave some pins on PICs (especially those with only 20 pins) open for debugging.

Big Picture

- Don't forget that the ultimate goal is to learn mechatronics. Have fun!
- As with all 218 projects, KEEP IT SIMPLE! If you can get away with making something as simple as possible, DO IT!
- Practice having the wisdom of knowing when to stop and sleep or take a break.
- Remember to have fun!
- Take a lot of pictures as you go.
- Use lab notebook so that all information is at one place and teammates can have easy access to it.
- Be friendly with other teams – you never know when you're going to need help.
- Bathe as frequently as possible; encourage others to do so as well.

Communications

- Start early for testing communication code
- Be sure to use the logic analyze for debugging communication. It will help you find errors that would be virtually impossibly to find otherwise
- Comm is cool.
- Get communication up and running first. Once it's working everything else in software follows quite easily.
- When debugging communication, the Logic Analyzer is your best friend.
- When debugging communication, use the lab's Zigbee with a GUI as a debugging tool as well
- Get the radios correctly sending and receiving messages as quickly as possible.
- Sending and receiving asynchronous messages was easier with an interrupt response rather than going through lots of states. Some may disagree, but think about it.
- Figure out early on how many PICs you will need so you can design the communication network early.
- Watch out for timing issues in your wireless communications - try to ensure that incoming and outgoing messages won't interrupt the previous transmission through proper use of waiting states.
- Use interrupts, not event checkers, for communication with the XBEE. The timing is a lot tighter this way and leads to less problems with communication collision.
- Get the radio working with the 'E128 first.
- When building networks, add nodes one at a time to better track down "bad nodes".
- Don't hesitate to add another PIC and SPI communication. It's really easy.
- 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.
- Test your wireless communication outside and at range!
- Test your components for interoperability with everyone else's before game day.
- Do not bury your wireless antenna in a box. Try to keep it out in the open.
- Practice on the course as soon as possible to test your operable range.

- 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.
- Don't spend more than a few hours debugging SPI code before debugging all of the related hardware.

Teamwork

- Your team should be in constant communication-- open and clear about what is happening and what the goals are.
- Everyone should be on board with important decisions.
- First define as a team what you will need to do, and then assign responsibilities to people to do them.
- Have other teammates verify what you are doing. Divide work such that every person is learning what she/he wants and could easily substitute for another teammate.
- Establish clear expectations and engage in regular status checkups.
- 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.
- Communicate well within your team so that some tasks are not overlooked, while others are duplicated.