ME218c Smart Product Design

ME 218c Spring 2016 Project
Helping Understand New Grass Roots Yammering (HUNGRY): High Intensity Procurers for Political Organizations and Special-interests

Project Presentations on May 25, 2016 in the Atrium 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:

After the presidential election of last quarter, the time has come to ignore those campaign promises and pass entirely different bills starting with a revamping of the current voting system. Going forward, voting by the Congregation Of Non-committal Gentry Reliant on External Stimulation Spending (CONGRESS) will be conducted by lawmakers passing through one of two revolving doors. Lobbyists have a great deal of influence on the lawmakers and can induce them to walk through one door or the other. Special interests use PACs to bribe lobbyists to ensure that the vote goes in favor of their cause.

The Task:

Design and build a tele-operated Loyal Only By BriberY Influence Spreading Technology (LOBBYIST) vehicle and a companion Political Artillery Controller (PAC). Groups of LOBBYISTs will operate in the Political Arena at the end of the Atrium in Building 550. During rounds of the game the two groups of PACs, through their LOBBYISTs, will attempt to herd the lawmakers through the appropriate door and thereby pass the most important legislation of our day.

Specifications

General:

- ☐ Each team will construct a LOBBYIST and a PAC.
- ☐ The LOBBYISTs are devices capable of navigating in political morass while herding lawmakers into their revolving doors.
- ☐ The PACs are the wireless remote controllers for the LOBBYISTs.

Basic Game Play:

- ☐ A game round will be a competition between two special interest groups (Red & Blue) of 3 PACs each, and four free agent LOBBYISTs.
- ☐ The goal of the game is to convince members of CONGRESS to vote for your special interest by entering one of the two marked revolving doors.
- ☐ The game will continue until 218 seconds have passed or all members of CONGRESS have voted (German Style, by passing through one of the revolving doors).

The Political Arena:

- ☐ The Political Arena is comprised of a region at the West end of the building 550 Atrium, measuring approximately 16 ft. by 24 ft (see Figure 1).
- ☐ The boundaries of the Arena are formed by a plastic wall 1.5" high.
- ☐ At the beginning of each game the participating LOBBYISTs will be placed in one of the four corners of the Arena as shown in Figure 1.
- ☐ Each end of the Arena will be bounded by a revolving door into either the Red or Blue Camp.

- ☐ At 90 degrees to the revolving doors will be two Reversers. By physically bumping into the Reversers with sufficient force a LOBBYIST may temporarily (for 10 seconds) reverse the direction of the associated revolving door.
- ☐ As with all applications of physical force, activating a Reverser will have political RAMifications. Upon contacting the Reverser with sufficient force to activate it the LOBBYIST will be forcibly injected back into the Political Arena with a velocity and rotation that varies at each activation.

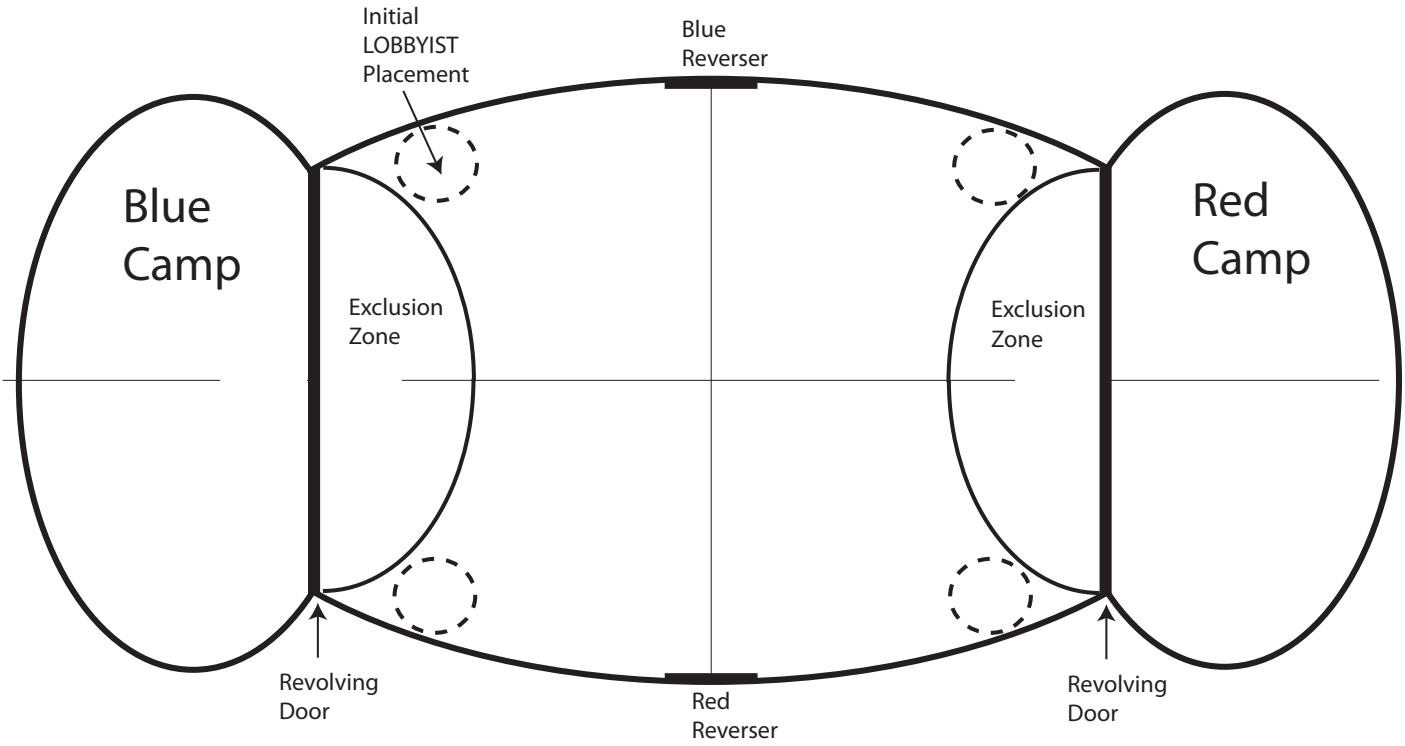

Figure 1: The Political Arena

The LOBBYISTs:

- ☐ Each LOBBYIST must be capable of moving under its own power within the Political Arena
- ☐ The LOBBYIST must hover above the surface of the Arena, and the only source of lift allowed is the lift fan provided by the SPDL.
- ☐ LOBBYISTs 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 of a pair of series connected NiMh or NiCd 7.2V batteries.
- ☐ Propulsion and steering systems are free with the caveat that they may not directly apply force to the Arena floor. An exception to this rule allows for dragging an element against the Arena to brake or steer. This will be acceptable as long as the contact mechanism would, if deployed while the LOBBYIST was not moving, create approximately equal drag in all directions and is not capable of supporting the vehicle.
- ☐ Control of LOBBYIST functions must be achieved via a PAC using the provided RF hardware (XBee24 modules)
- ☐ Each LOBBYIST must carry a display system meeting the requirements shown in the Display of Memory and Commitment section below.
- ☐ LOBBYISTs must incorporate an easily accessible switch that disables all moving systems.

- ☐ The perimeter of the largest normal projection of the LOBBYIST into the plane of the Arena must not exceed 72". The entire vehicle, projected vertically onto the field, must lie within the perimeter. Height is not restricted.
- ☐ LOBBYISTs must incorporate a class standard foam bumper around their perimeter, and must be tolerant of moderate bumping from other LOBBYISTs. The bottom edge of the foam bumper must be at a height between 0" (the Arena surface) and 1.25" above the Arena surface. The bumper must follow the same perimeter that is measured to fit the 72" requirement.
- ☐ Every LOBBYIST must be controllable through any of the PACs via the class-wide protocol (See Radio Communications in a later section).
- ☐ The LOBBYIST may issue messages to the PAC at a rate no greater than 5 Hz.
- ☐ If the LOBBYIST 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.
- ☐ During a game, each LOBBYIST will carry an SPDL supplied credential badge indicating which of the LOBBYISTs (1-4) it currently represents. The credential badge will include a 2-pin electrical connector (Molex KK256) to a resistor (1-4k) that can be used by the LOBBYIST to determine which badge it is currently carrying.
- ☐ LOBBYISTs, having very short memories and an intense fear of commitment, may not be controlled by a single PAC for more than 45 sec. Once the 45 seconds has passed, the LOBBYIST must be bribed by another PAC before the prior PAC may bribe it again. Software on the LOBBYIST is required to enforce this rule.

The PACs:

- ☐ Each team will design and construct an PAC that will relay commands from a human operator to a LOBBYIST, and receive and display connection status with the LOBBYIST.
- ☐ The PAC must be capable of displaying to the operator an indication of active communication with its associated LOBBYIST.
- ☐ The PAC must provide a method for the operator to use to indicate which LOBBYIST (1-4) the operator would like to control.
- ☐ PACs 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.
- ☐ PACs must be un-tethered and portable by one person.
- ☐ Input to the PAC 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 PAC to issue commands to the LOBBYIST should be inventive and interesting for the audience to watch. Use of actions that make the operator look and feel foolish is encouraged.
- ☐ The PAC may issue commands to a LOBBYIST at a rate no greater than 5 Hz.
- ☐ PACs 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 Display of Memory and Commitment:

- ☐ Each LOBBYIST must carry a federally mandated Display of Memory and Commitment (DMC).
- ☐ The DMC will consist of a PIC12F752 programmed in assembly language that receives instructions from the main processor of the LOBBYIST to indicate when the display states should change.
- ☐ The DMC must indicate to the audience the currently controlling special interest (Red/Blue)

- ☐ The DMC must display to the audience an indication of the amount of time remaining before the bribe from the current PAC expires.
- ☐ The DMC must contain an electromechanical display indicating that communication with a PAC is currently active.

Game Details:

- ☐ Upon power-up or in the event of a loss of communication with its current PAC, the LOBBYIST will activate its electromechanical indication that it is searching for a controlling interest, deactivate its lift fan and wait for a request for control from a PAC.
- ☐ The operator of a PAC that wishes to control a particular LOBBYIST must select that LOBBYIST (1-4) using the PAC and make a unique control action to initiate taking control of the LOBBYIST. This action will result in the PAC sending a message to the LOBBYIST requesting control of the LOBBYIST. The details of the request process will be defined in the class-wide communications protocol.
- ☐ The LOBBYIST will respond to the first received request for control by sending a message back to the requesting PAC confirming receipt. At this time, the LOBBYIST will also de-activate its electromechanical indication of searching for a controller. After completing this process, the LOBBYIST is bound to that PAC until communication is lost or the bribe expires (after 45 sec.).
- ☐ If a LOBBYIST receives a request for control while it is already under control, it will silently ignore the request.
- ☐ Within 100ms of the bribe expiring, the LOBBYIST must activate its electromechanical indication that it is searching for a controlling interest and deactivate its lift fan.
- ☐ During a game, a LOBBYIST may not remain in the Exclusion Zone (See Figure 1) for more than 15 seconds at a time.
- ☐ Lobbying begins with the ringing of a bell and ends with a strike of a gavel.

Radio Communications:

- ☐ Communications between the LOBBYISTs, and PACs will take place over an SPDL-supplied 802.15.4 radio (Xbee24) using the Non-Beacon API mode of operation.
- ☐ To prevent eavesdropping on the conversations between the PACs and their LOBBYISTs command communications between them will be encrypted using the algorithm described in Appendix A of this document.
- ☐ Any PAC should be capable of controlling any LOBBYIST.
- ☐ Once a game begins, communication will take the form of bi-directional communications between a LOBBYIST and its bound PAC.
- ☐ Each LOBBYIST and PAC 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 LOBBYIST, and implement communications on the PAC.
- ☐ The class communications protocol must include a procedure for validation of communication between the LOBBYIST and PAC. The LOBBYISTs must, using the DMC, provide a visual indication of when a functioning communications link between the LOBBYIST and PAC exists.

General Requirements:

- ☐ At a minimum, either the PAC or the LOBBYIST 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 log 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 reviewed 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.
- ☐ PACs or LOBBYISTs based substantially on purchased 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 LOBBYISTs and the PACs should be safe, both to the user and the spectators.
- ☐ Intentionally disabling or damaging other LOBBYISTs 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).
- ☐ No part of the LOBBYIST may become ballistic.
- ☐ 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 **05/02/16** 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 LOBBYIST and the PAC. 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. 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 **05/04/16**. Ed will meet with the communications committee on the evening of **05/05/16** to provide feedback on the specification.

Communications Standard:

Due by 5:00 pm on **05/06/16**. This is the working draft of the communications standard.

First Check-Point:

On **05/10/16**, 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 **05/16/16**, 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 **05/19/16**, 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' LOBBYISTs and PACs.

Grading Session:

During the Grading Session on **05/24/16**, each team will be required to demonstrate the ability to successfully participate in a game. This will include

1) Establishing communications between your LOBBYIST and PAC and between your LOBBYIST and the PAC from another team.

2) Navigating a LOBBYIST from the initial position and successfully inducing at least one lawmaker to pass through a specified revolving door.

3) Displaying on both the PAC and the LOBBYIST 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 **05/25/16** starting at 6:00 pm in the Atrium of Building 550. At this event, members of the public will be encouraged to act as operators of the PACs.

Report:

Draft due on **05/30/16** by 4:00 pm. The final version (with revisions incorporated) is due by 5:00 pm on **06/03/16**.

Celebration:

A celebration of the past 3 quarters of ME218 will take place at the Alpine Inn on **06/02/16** 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 LOBBYIST and PAC during the first & second check points and project preview. Members of the teaching team will operate the LOBBYIST via the PAC 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 05/10/16**.**
- ☐ **Second Check Point (10%)** Based on the results of the performance demonstrated on 05/16/16**.**
- ☐ **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.

Appendix A

Encryption Algorithm:

All commands from the PAC to a LOBBYIST will be encrypted using a rotating XOR Cipher (described in detail below). Since the communications channel is essentially one-way (with the exception of acknowledgements), only one encryption key is necessary. This 32 byte randomly generated key will be provided by the PAC to the LOBBYIST as part of the process of pairing between the PAC and the LOBBYIST. All subsequent commands from the PAC to the LOBBYIST through the end of the period of control (max. 45 sec) will be encrypted using this key.

As part of the pairing process, the details of which will be defined by the class-wide communications protocol, the PAC will provide to the LOBBYIST the current key (`uint8_t EncryptKey[32]`). For this application, all messages prior to pairing as well as the key may be transmitted in plaintext.

Once the receipt of the key is acknowledged, both PAC and LOBBYIST will reset internal counters (`uint8_t i`) to zero. This counter will then count modulo 32 (so that it will rotate through the bytes of the encryption key) with every new byte encrypted and decrypted.

To transmit a message, `XmitMsg = [a,b,c]`, the PAC would construct an encrypted version using the encryption key: `Encrypt =[ a^Key[++i], b^Key[++i], c^Key[++i]` where `++i` is understood to be an increment by 1 modulo 32. The LOBBYIST would decrypt this received message using the stored encryption key: `Decrypt = RecvMsg[0]^StoredKey[++i], RecvMsg[1]^ StoredKey[++i], RecvMsg[2]^ StoredKey[++i]` where, once again, `++i` is understood to be an increment by 1 modulo 32.

At this point, the LOBBYIST would interpret the Decrypted message according to the protocol defined by the class-wide communications committee.

Note: because the encryption key is rotating it is critical that the transmitter and receiver stay in sync. Dropped or lost packets must be re-tried until success in order to remain in sync. It will be the responsibility of the communications protocol to provide for this robustness.

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.