P10216 Detailed Design Review Action Items and Feedback to Action Items

**Hardware:**

- Issue: Positioning of ultrasonic distance sensors. Distance between low and high-range sensors is too large.
  Steps to take: Move sensors to a more mid-level position on the front of the robot.

- Issue: Possibility for missing objects low to the ground.
  Steps to take: Mount proximity sensors around the "skirt" of the chassis.

- Issue: DC-DC Converter not necessary/inefficient for 6V -> 5V buck regulation.
  Steps to take: Select a linear regulator rated for the anticipated load.

- Issue: Wiring scheme complete, but wiring diagram needed to realize physical design.
  Steps to take: Using a vector board as a platform, make a plan for physical wiring of all components, using appropriate connectors, wiring.

- Issue: Output power for speakers may not be sufficient for an outdoor environment.
  Steps to take: Purchase sample quantities and test suitability outdoors.

- Issue: GPS interface doesn't fit in well with current navigation sensor interface scheme.
  Steps to take: Arbitrate GPS control/interface directly to the SBC over a UART.

- Issue: Avoid custom navigation algorithms (i.e. very situational specific and non-reusable code) for various situational scenarios.

- Issue: Current scheme for heading correction not well-defined.

**Both P10215 and P10216 Interfaces**

Software algorithms are still very high level or missing in some cases. One or two scenarios may help define the interfaces. I would like to see two or three “top-to-bottom” and “bottom-to-top” situational scenarios (translations from commands to physical motor control (%PWM, Encoder) and vice versa)) for odometry related scenarios. As an example, Marcus reviewed the Navigation Algorithm State Machine and reviewed change in state from Motion Halted State to Turning 30 degrees (and once 30 degrees is completed Moving Forward) states. I would like to see Pseudo Code or flowchart for each level
of software with data and motor control to ensure everyone agrees to needed handshaking and interpretation of commands and data. As an example of this scenario, ensure the Robot does not interfere with its surroundings while turning, the robot turns on its axis. Will this translate in the left motor turning counterclockwise while the right motor turns clockwise XX encoder pulses?..... or left motor turns clockwise at 20% PWM and right motor turns clockwise at PWM is 95%? Since the motors face one another what does clockwise and counter clockwise for each motor mean in terms of direction? ..... Using the “Serial Communications protocol Interface” document, depict or list the series of communicated commands WITH data (fill up 8/16 signed integer/empty command). At the MCU and Controller the translation of data to commands. At the MCU how does encoder data get translated to positive and negative integers. The gear ratios are known and the encoder resolution is known so this should be easy to do. On the MCU, how are Velocity, Rotation, and Encoder values calculated, as an example? What defines or controls the persistence of these parameter values listed in “Serial Communications protocol Interface”? If GPS and/ or compass are involved, add these.

Issue: Wiring scheme complete, but wiring diagram needed to realize physical design. Using Cadence Capture draw a schematic showing all connectors, pins. 0.1 headers are good and allows for easy discounts for signal connections. Show or note harnessing assumptions. Label ribbon cable, discrete wiring, color coding (helps during debug), gauge of wire and so on. If sensors need extensions, add a connector, harness, .... I would highly recommend adding pins for test points. Easy to do now and can save hours in debug.

Firmware:

- Issue: Current "WHO_AM_I" register implementation is fine, but more sophisticated schemes could be utilized for checking communication link status.
  Steps to take: Implement a periodic counter in the "WHO_AM_I" register. Slaves (Embedded Controllers) updates their "WHO_AM_I" register periodically. The Master (SBC) checks the registers periodically to determine "alive" status of slaves (Embedded Controllers).

- Issue: Lack of logging crash reports, reset reports may cause a system to cycle in reset at a given point.
  Steps to take: Implement logging of reset causes in non-volatile storage for future checking to diagnose recurrent issues.

Software:

- Issue: Current scheme for heading correction not well-defined.
Steps to take: Use a weighted average scheme for heading from the compass, GPS and odometry. Compass holds highest weight (most accurate).

- Issue: Current scheme for obstacle avoidance may not route robot around objects efficiently, cause zig-zagging.
  Steps to take: Investigate potential-field navigation algorithm or define a reasonable time that the robot continues moving away from an object before trying to re-approach destination.

- Issue: Destination being defined by a point (i.e. long/lat coordinate) and a heading may constrain navigation.
  Steps to take: Since robot only cares about finding sunlight, shade, ideal condition for plant, specify destination as a heading rather than combination of a heading an point.
  A point destination is only relevant when robot wanders outside of its bound.

- Issue: Avoid custom navigation algorithms (i.e. very situational specific and non-reusable code) for various situational scenarios.
  Steps to take: Reuse the wander algorithm already developed in many of the other implemented situations.

- Issue: Current wireless client-server model uses serial Java objects which will not allow for wirelessly downloadable code to be executed.
  Steps to take: Speak with SE students to develop a sufficient scheme. For now continue with a temporary scheme for testing.