Mobile Robotics Scoring Addendum

Team Ranking

Teams will be given a total score based on the Engineering Notebook (Page 1 of the Design Rubric), the Design Interview (Page 2 of the Design Rubric), a separate Programming Interview where the team explains the functionality of their code to the judges, the team's highest Programming Skills Score and the team's highest Driving Skills Score. Teams are ranked by the sum of their scores in these five (5) categories.

All teams will be given the same number of Robot Skills Matches to be determined by the Competition Organizer.

In the case of ties, the tie will be broken by looking at the following in order.

1. Engineering Notebook Score
2. Team's highest Programming Skills Score
3. Team's highest Driving Skills Score
4. Team's next highest Programming Skills Score
5. Team's next highest Driving Skills Score
6. Repeating 4 and 5 until all scores are exhausted
7. Design Interview score
8. Programming Interview score

In the very unlikely event that the scores are still tied, the judges will deliberate one last time to determine a winner.

Point Value Calculations

Engineering Notebook Score on Page 1 of the Design Award Rubric is recorded then multiplied by 10 and submitted on the Judges Score Card (Maximum 270 points)

Design Interview Score on Page 2 of the Design Award Rubric is recorded then multiplied by 10 and submitted on the Judges Score Card. (Maximum 150 points)

Programming Interview Score on the Programming Interview Process is submitted on the Judges Score Card (Maximum 100 points)

Highest Programming Score is multiplied by 10 and submitted on the Judges Score Card (Maximum 190 points)

Highest Driving Score is multiplied by 10 and submitted on the Judges Score Card (Maximum 290 points)

SkillsUSA

Team Number _____

Mobile Robotics Judges Score Card

Engineering Notebook (maximum 270 points) _____

Design Interview (maximum 150 points) _____

Programming Interview (maximum 100 points) _____

Highest Programming Skills Score (maximum 190 points*) _____

Highest Driving Skills Score (maximum 290 points*) _____

Total Score (maximum 1000 points*) _____

Used for tiebreaking purposes only:

_____ Engineering Notebook Score

_____ Team's highest Programming Skills Score

_____ Team's highest Driving Skills Score

_____ Team's next highest Programming Skills Score

_____ Team's next highest Driving Skills Score

_____ Team's next highest Programming Skills Score

_____ Team's next highest Driving Skills Score

_____ Team's next highest Programming Skills Score

_____ Team's next highest Driving Skills Score

_____ Team's next highest Programming Skills Score

_____ Team's next highest Driving Skills Score

_____ Team's Design Interview Score

_____ Team's Programming Interview Score

Note: If a team scores higher than the maximum, the team will be given the maximum points.

Mobile Robotics Programming Interview Process

This interview is comprised of 4 sections. For each section please read all instructions and questions before assessing the team.

Section 1: General Programming Information (Maximum 15 pts)

For this section you will be asking the team general information about their program. This section will make sure teams have come prepared for their interview.

Did the team bring a laptop with their code?

No (0 pts)		Yes (5 pts)	

Did the team bring their robot?

No (0 pts)		Yes (5 pts)	

Ask the team, what programming software are they using. Does it match the code that was brought to the interview?

No (0 pts)		Yes (5 pts)	

Section 2: Program Design and Fluency (Maximum 60 pts)

In this section you will ask the team to walk you through their code. Ask the team to start at the very beginning and explain the program until the robot stops. Read all questions beforehand because you will need to assess the program after the walk through is complete. The following questions are for the judge and should not be asked to the team.

Did the program include comments?

1 pt	2 pt	5 pt	9 pt	10 pt
Program did not contain comments.	Program contained comments but lacked in depth. The comments were only useful for the programmer.			Program contained in depth comments for their entire code base. Comments were articulate and meaningful.

Did the program use variables instead of hard coding numbers? (eg. when they set the speed of the motor, is it a number or a variable)?

1 pt	2 pt	5 pt	9 pt	10 pt
Program did not include any variables.	Program contained a mix of variables and hard coded values. Variable may not be organized.			The program used variables for all or most opportunities. Variables were organized and named in a meaningful way.

Did the program contain advanced programing structures like loops and if else statements?

1 pt	2 pt	5 pt	9 pt	10 pt
Program did not contain any loops or if else statements.	The program only had a few loops or if/else structure. Some parts of the code were reused in loops but others were programed linearly.			The program contained many loops and if/else structures.

Did the program contain functions that were used throughout their code?

1 pt	2 pt	5 pt	9 pt	10 pt
Program did not contain any functions.	The program used some functions but missed opportunities to make a function.			The program had multiple functions and was used to reuse code wherever possible in their program.

Is the code formatted in an organized manner?

1 pt	2 pt	5 pt	9 pt	10 pt
Program did not follow any kind of format. Code was not properly indented or spaced in a neat fashion.	Most or some of the code was formatted. There are areas where code could have been formatted a little better.			The entire code base is formatted and spaced.

How did the team conduct the walkthrough of their code?

1 pt	2 pt	5 pt	9 pt	10 pt
The team showed zero or minimal knowledge of their program. They were not able to articulate what their program does or where it starts.	Team was able to walk you through the program. Students read the comments verbatim and were not able to explain more than what was already written in the program. The team was unsure about how some of the code worked in some sections.			The team was able to explain all parts of their program. The team used proper terminology when talking about their program. The team was able to explain their code without having to read the comments verbatim.

Section 3: Smart Programming (Maximum 15 pts)

In this section you will be asking the team specific questions about their program. The judge will assess the team on how well they answer each question.

Ask the team how many sensors are on their robot that they programed.

1 pt	2 pt	3 pt	4 pt	5 pt
Team uses one or less sensors on their robot.	The team uses a moderate amount of sensors (2 - 3).			Team used a large amount of sensors (4+).

Find a sensor on the team's robot, or one they mentioned in the question above. An example could be a Gyro Sensor. Ask the team to show you where in their code that they use this sensor. Is the team able to explain and show you how they used the sensor?

1 pt	2 pt	5 pt	9 pt	10 pt
Team did not use any sensors or could not find how they used the sensor in their code.	The team struggled to find where they used the sensor in their code, and/or was only able to explain how they used the sensor by reading comments in that section. The team did not fully understand what data was being collected by the sensor and how it was used by the program.			Teams were able to quickly find the sensor in their program. They were able to explain in great detail how the program uses the data from the sensor.

Section 4: Compilation (Maximum 10 pts)

In this section you will be asking the team to compile their program and download it to their robot. After the compilation is completed, the judge will exam the team's program. If the program fails, allow some time for students to debug and fix the issue.

Did the program compile and download to the robots with no errors or warnings?

1pt	2pt	5pt	9pt	10pt
The program did not compile and had multiple errors. Students were not able to fix their program in due time.	The program did not compile correctly the first time, but students were able to quickly identify the cause and fix the issue in due time. Code eventually compiles correctly but has warnings. Students code compiles and downloads correctly the first time but shows many warnings. Known warnings that do not affect the program are undocumented in their program.			Students code compiles and downloads to the robot the first time. Code shows zero errors and zero warnings. Known warnings that do not affect the program are documented.