Rubric for

Find Keepon – DRAFT

Capstone Python Project

CSSE 120, Introduction to Software Development – Robotics
Spring term, 2011-2012

- The features (functionality of the program) are listed on the following pages. **Highlighted features are required.** You earn points for non-highlighted features *only* if you complete *all* the highlighted features successfully.
    - Generally speaking, "successful" means at least 3 of 5, but note that most highlighted features are designed to be all-or-nothing.
- **You score on a feature can range from 0 (*not done*) to 5 (*exemplary*).** For the more ambiguous features (e.g. follow an *arbitrary* line …), *exemplary* means that you can do that feature at a very high level, while a 1 (*needs improvement*) would be the score for the simplest version of that feature.
- **Available points are:**
    - **Features implemented: 2,690 points available, but capped at 800**
        - More points increase your fame but not your total score
    - **Quality of code: 150** points available, but capped at **100**,
      and as much as ***minus 150*** for poor quality.
    - **Team deliverables (other than code, but including documentation among other items):**
      150 points available, but capped at **100**, and as much as ***minus 150*** for poor deliverables.
    - **Individual deliverables (other than code, but including documentation among other items):**
      150 points available, but capped at **100**, and as much as ***minus 150*** for poor deliverables.
- **Your score (out of 1,000, but with a maximum possible of 1,100) is computed by:**
    - Sum the points you earned in the above four categories
    - Multiply that sum by your Contribution Multiplier – your instructor's judgment of the degree to which you contributed to your team (100% for appropriate contributions, less for less than appropriate contributions)

  Since the score for Features is capped at 800 and each of the other three categories is capped at 100 points, **the maximum possible score is 1,100 of 1,000 (i.e., 110%).**

  Important: Note that if your team scores 1,400 on Features (a HUGE score), but the other three items are terrible, your score is a (800 – 150 – 150 – 150) = 350. Since that is of 1000, it would be an F (35%, failing).

For any feature, if you have any doubts about what that feature requires, just ask. For those interested, the last page of this document provides a breakdown of the points by subcategory.

Start, Interrupt and Quit: *The user can, via the graphical user interface (GUI:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
1. Enter the port number to which to connect	**1**							
2. Connect to her robot via the current port number	**1**							
3. Disconnect (without closing the program)	**1**							
4. Switch between modes (passive, safe & full)	**1**							
5. Run in the simulator	**1**							
6. Apply a "*kill switch*" that stops the operation that is currently running. Exemplary requires this ability for ANY operation. Stopping during teleoperation is not relevant to this feature.	**5**							
7. Ditto, but *pause/resume* instead of *stopping* the operation that is currently running	**5**							

Total possible for *Start, Interrupt and Quit*: 25 + 50 = 75 points (required/additional features)

Display Project Information: *The user can, via the graphical user interface (GUI), easily see:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
8. The name of this term's project (***Keepon Dancing***), the course name and current year and term	**0** **but required**							
9. A brief description of the project (you can quote from any description we provide if you wish)	**1**							
10. For each team member, her name and a short (fictitious if you like) bio	**1**							
11. For each team member, a short paragraph describing the main features for which that team member was the technical lead	**1**							
12. For each team member, and also for the team as a whole, the total person-hours spent on the project during Sprint 1†	**1**							
13. Ditto for Sprint 2 (report cumulative hours as well as hours during the Sprint)†	**1**							
14. Ditto for Sprint 3 (report cumulative hours as well as hours during the Sprint)†	**1**							
15. Anything else notable about the project	**1**							

Total possible for *Display Project Information*: 30 + 5 = 35 points (required/additional features)

† Include in-class *and* after-class hours. ***To earn this feature, it must be demonstrated shortly after* THIS *Sprint ends – NOT LATER.***

Display the Robot's State: *The user can, via the graphical user interface (GUI), easily see:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
16. Current values of the sensors: bump, wheel-drop, wall, cliff, IR, LEDs, voltage, current, overcurrent, charging state, song playing, etc. (about 1 point for each 4 sensors whose value is shown)	**3**							
17. Current port number	**1**							
18. Current mode (off, passive, safe or full)	**1**							
19. Operation(s) currently running (e.g., direction & speed while moving, number of the demo playing (if any), how many Keepons are currently visible, indicator for whether your robot is touching a Keepon,, indicators for whether conversation/dancing/singing/light shows are underway, etc)	**2**							
20. Position (x, y) & direction of your robot, relative to the robot's initial position and direction, after teleoperation (exemplary requires this feature to work after *any* movements, not just teleoperation)	**3**							
21. Number of Keepons visible	**2**							
22. Directions (angles) to the visible Keepons	**2**							
23. Distances to the visible Keepons	**2**							
24. Information about current or past odometry error	**3**							
25. Other interesting historical data (from earlier in this run) or cumulative or statistical data	**3**							
26. Other interesting state information	**3**							
27. Some or all of the above update automatically in some reasonable way (exemplary requires that the updates occur at time intervals that are appropriate to the thing being displayed)	**3**							

Total possible for *Display the Robot's State*: 0 + 140 = 140 points (required/additional features)

Highlighted features are required.

Teleoperation: *The user can, via the graphical user interface (GUI), make her robot:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
28. Move from its current position to any reasonably accessible position to which the instructor points (via ***teleoperation*** – you control the robot's motion)	**8**							
29. Include concurrent linear and angular motion during teleoperation (so that the robot moves in an arc that you control)	**3**							
30. Change speeds *in between* teleoperation or other actions (exemplary requires a full range of speeds and both directions)	**5**							
31. Change speeds *concurrent with* tele-operation, i.e., without stopping the action underway (exemplary requires a full range of speeds and both directions, as well as the ability to change speed/direction while using *any* teleoperation controls)	**4**							
32. Make the robot go up stair steps (and/or *gently* down)	**10**							
33. Do other interesting teleoperation (score depends on difficulty and creativity)	**5**							

Total possible for *Teleoperation*: 65 + 110 = 175 points (required/additional features)

Semi-Autonomous Motion: *The user can, via the graphical user interface (GUI), make her robot:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
34. Move linearly (at the current speed) a user-supplied *distance* (and then stop)	**2**							
35. Spin (at the current speed) a user-supplied angular *distance* (and then stop)	**2**							
36. As in the two previous items, but with linear and angular motions concurrently (stopping each when its distance limit is reached)	**2**							
37. Move linearly a user-supplied *time* (and then stop)	**2**							
38. Spin a user-supplied *time* (and then stop)	**2**							
39. As in the two previous items, but with linear and angular motions concurrently (stopping each when its time limit is reached)	**2**							
40. Modify the requested *distance/time* the user asks the robot to travel, *while* the robot is moving	**4**							
41. Move to a user-specified (x, y) position and direction relative to the robot's *current* position and direction	**2**							
42. Ditto, but relative to the robot's *initial* position and direction	**7**							
43. Move (at the current linear and angular speeds) until either bump sensor is activated	**3**							
44. Move (at the current linear and angular speeds) until any cliff sensor is activated	**3**							
45. Do, via a single control, any of several interesting user-specified movements, e.g. drive in a square (exemplary requires a large selection of choices available)	**3**							
46. Choose a demo and start that demo running, or stop the current demo running	**2**							
47. Turn on or off a stop-when-bumped-while-moving option (while doing *any* of the above types of motion, as well as teleoperation; exemplary requires this ability while doing the fully autonomous motions as well)	**4**							
48. Ditto, but a stop-when-cliff-sensor is activated option (while doing *any* of the above types of motion, as well as teleoperation; exemplary requires this ability while doing the fully autonomous motions as well)	**4**							

Total possible for *Semi-Autonomous Motion*: 55 + 165 = 220 points (required/additional features)

Find Keepon (autonomously) and Make Keepon Dance:

For all these features, your robot starts in a standard position – a marked place near the center of an open space, facing a wall. There won't be any obstacles except for Keepons, unless the feature below explicitly indicates the presence of obstacles. ***Your robot makes Keepon dance by bumping into Keepon (or the robot on which she/he/it resides), stopping, and sending a special IR signal.***

These features require autonomous motion – the user initiates the motion, but it is hands-off from there on (so no teleoperation, for example).

To earn these features, the user can make Keepon dance given that:

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
Camera features. ***For each of the following, the feature must be implemented by using the camera in a meaningful way.*** ***Keepon will be on the floor or on a Create (you must deal with both possibilities), unless specified otherwise below.***								
49. ***Keepon is visible*** when the camera is pointing the same direction your robot is pointing	**5**							
50. ***Keepon would be visible*** if the camera were pointing a different direction	**5**							
51. One or more Keepons are within reasonable range of your robot, but they are obscured	**5**							
52. There are multiple Keepons visible (either directly or by spinning), and you must make all of them dance (partial credit if you make some of them dance)	**5**							
53. Keepin begins visible straight ahead, but moves (you can assume that your robot is physically capable of catching Keepon, but exemplary requires that you catch Keepon in a reasonable time under a wide variety of escape antics that Keepon might try)	**5**							
54. Ditto, but now there are multiple Keepons (at the start or during the chase) and you must move toward the closest Keepon at every point of the chase	**5**							
55. Keepon is suspended in the air (your robot must stop approximately beneath Keepon)	**5**							
56. Ditto, but now Keepon flies in the air	**5**							
57. Keepon is at the end of a curvy, colored line. (Keepon will *not* be visible, so you must line-follow, but you can color-calibrate before the run.)	**5**							
58. Your robot and Keepon play tag (your code runs on your robot and the Keepon robot)	**7**							
59. Your robot and Keepons play Marco-Polo (your code runs on your robot and the Keepon robot – see your instructor for what "blindfold" means here)	**7**							
60. Other interesting situations that require use of the camera (score depends on difficulty and creativity)	**7**							
61. Keepon is somewhere *(that is, anywhere your instructor chooses!)* on campus	***Priceless***							

(continued from the previous page)

To earn these features, the user can make Keepon dance given that:

Non-camera features.

For each of the following, the feature must be implemented without using the camera. Also, the motion must be "reasonable" – no completely random wandering – given that you KNOW that Keepon is placed as described.

62. Keepon is within 15 feet straight ahead (Keepon may be placed *after* your robot starts moving, *behind* your robot)	**2**							
63. Keepon is on the circumference of a square whose sides are each 10 feet (your robot starts in the center facing directly toward one of the sides)	**3**							
64. Keepon is on the circumference of a circle whose radius is N feet, where the user specifies N at run time (your robot starts in the center of the circle)	**3**							
65. Keepon is somewhere along a path that the user supplies by giving waypoints† given that there are no obstacles on that path (exemplary requires a complicated path)	**5**							
66. As in the previous item, but with obstacles on the path (exemplary requires several obstacles)	**5**							
67. Keepon is at a waypoint, but where the waypoint changes as your robot is moving toward the previous value of the waypoint†	**5**							
68. Keepon can be found by other interesting dead reckoning (score depends on difficulty & creativity)	**4**							
69. Keepon is somewhere along a wall (you must do wall-following to earn this feature; Keepon will *not* be on the first wall you follow)	**2**							
70. Keepon is somewhere along a known oval-shaped black line‡	**5**							
71. Keepon is somewhere along a known more-curvy black line‡	**3**							
72. Keepon is somewhere along a curvy line whose darkness requires light calibration‡ (exemplary requires semi-automated calibration)	**3**							
73. Keepon is somewhere along a more difficult line to follow that has no intersections‡	**5**							
74. As above, but the line has intersections‡	**3**							
75. Keepon can be found by other interesting line and/or wall following (score depends on difficulty and creativity)	**5**							

Total possible for *Making Keepon Dance*: 100 + 495 = 595 points (required/additional features)

† ***Waypoints*** are (x, y) coordinates relative to your robot's initial position and direction. For the feature in which the way-point is changing, the simplest implementation would have the user change the waypoint but the best implementation might acquire waypoints from an arbitrary stream, perhaps even a network stream!

‡ ***You must do line-following to earn this feature.*** For the most difficult line-following features, light-calibration will be required. For the most difficult lines, you will *not* have the line in advance, but you can assume that the line is dark on a light surface (but the darkness/lightness may change smoothly during the run), there are no impossibly sharp turns, and the line is between 1 and 12 inches wide (but its width may vary during the run). You may be able to earn partial points if your robot can deal with some but not all of these challenges.

Jamming with Keepon (via music or lights or dance): *The user can, via the graphical user interface (GUI), make her robot:*

		Exemplary		**Satis-factory**	**Needs improvement**		**Not done**	**Weight × Score**
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
76. Play whatever **note** the user specifies for a user-specified duration (3 points for playing the note; 2 more point for allowing durations longer than 4 seconds)	**2**							
77. Play a **song** of your own choosing (1 point for any song; 1 more point if the song is more than 16 notes; 1 to 3 more points if the song is genuinely *interesting*)	**4**							
78. Play any of a *selection* of songs from which the user chooses	**3**							
79. *Compose* and play its own songs (exemplary requires *genuinely* interesting songs)	**6**							
80. Allow the user to choose between *blocking* and *non-blocking* modes when playing a note or song	**2**							
81. Set its **LEDs** to whatever state the user specifies	**2**							
82. Perform an **LED light-show** of your own choosing (1 point for any light show; 1 more point if the light show uses all three LEDs and a variety of intensities and red-green spectrums; 1 to 3 more points if the light-show is genuinely *interesting g*)	**3**							
83. Perform any of a *selection* of LED light-shows from which the user chooses	**2**							
84. *Compose* and perform its own LED light-shows (exemplary requires *genuinely* interesting light shows)	**4**							
85. Perform a **dance** of your own choosing (exemplary requires the dance to be *interesting*)	**3**							
86. Perform any of a *selection* of dances from which the user chooses	**2**							
87. *Compose* and perform its own dances (exemplary requires *genuinely* interesting dances)	**4**							
88. Perform a dance, light-show and song *concurrently*	**3**							
89. Perform a *choreographed, more-than-16-notes,* dance, light-show and song *concurrently* (exemplary requires *genuinely* interesting choreography and concurrency throughout the performance)	**7**							
90. Earn a spot for Keepon and yourself on *American Idol*	***Priceless***							

Total possible for *Jamming with Keepon*: 30 + 205 = 235 points (required/additional features)

Talking to Keepon: *The user can, via the graphical user interface (GUI), make her robot do the following with a Keepon that is within hearing distance (about 1 meter). All "talking" and "listening" is via the IR receiver and sender, per a standard set by your instructor.*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
91. ***Send*** and display-as-sent a single-byte message to Keepon, chosen by the user from the list of single-byte messages, and sent per the standard. For *exemplary*, you must encode messages per an encoding that your instructor supplies.	**3**							
92. Ditto, but a multi-byte message.	**5**							
93. ***Receive*** and display-as-received a single-byte message from Keepon, chosen by Keepon and sent per the standard. For *exemplary*, you must decode messages per an encoding that your instructor supplies.	**3**							
94. Ditto, but a multi-byte message	**5**							
95. Talk with Keepon via a standard that you design and implement, that meaningfully *augments* the instructor-supplied standard	**3**							
96. Talk more meaningfully with Keepon via a *protocol* that you design and implement, on both your robot and on Keepon. For *exemplary*, the protocol must consider efficiency and other networking issues.	**10**							
97. Be a Rogerian therapist for Keepon, ala Eliza – google for *Eliza therapist* (for exemplary, you must implement the therapist yourself, but there are easier – but still very challenging – approaches that score fewer points)	**20**							
98. Convince Keepon to follow you home	***Priceless***							

Total possible for *Talking to Keepon*: 30 + 265 = 295 points (required/additional features)

User interface: *The user interface:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
99. Implements the basic actions via a reasonable GUI	**4**							
100. Allows teleoperation in a particularly user-friendly way (easy to understand, easy to operate)†	**4**							
101. Is *visually attractive*†	**6**							
102. *Works nicely* – How *easy is it to use* the interface without instruction? How much *effort is required* to do actions? Is it easily *extended*? and so forth.)†	**8**							
103. Uses multiple screens *consecutively* in a meaningful way	**2**							
104. Uses multiple screens *concurrently* in a meaningful way	**2**							
105. Uses via the *keyboard* meaningfully (in addition to the mouse)	**3**							
106. Allows devices beyond the mouse and keyboard to control the robot (joystick, external touch screen, wii mote, …)	**6**							
107. Stores and retrieve interesting, relevant information in files that the user could edit (exemplary requires several sets of information, both storing and retrieving, and good file organization)	**6**							
108. Communicates over the network in a meaningful way (score depends on difficulty and creativity)	**10**							
109. Does actions with a nice, *interesting* user interface beyond buttons (LOTS of possibilities here – menu's, drop-down's, sliders, progress bars, …)‡	**10**							
110. Does other *interesting* things (not scored in the above) related to the user interface	**4**							

Total possible for *User Interface*: 90 + 235 = 325 points (required/additional features)

---

† This will be your instructor's subjective opinion. Be forewarned that exemplary will be hard to earn.

For exemplary you must (and for other levels you should) justify your user interface design explicitly by scoring it via principles like:

- Ben Schneiderman's Eight Golden Rules of User Interface Design at:

  http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html

- Apple's principles of IOS development at:

  http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Principles/Principles.html

‡ Your score will depend on how nice and how *interesting* your user interface is, in your instructor's subjective opinion. Generally speaking, each new TYPE of user interface widget earns you points, while repetitions of widgets of the same type do not. Also, the features listed explicitly in lines above this one do not count toward this feature (no double-counting).

Highlighted features are required.

Hardware: *Your robot makes use of hardware that you ADD to the ROBOT:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
111. Your robot can successfully tap or poke Keepon when Keepon is on another robot (with an arm that you add, or something like that)	**10**							
112. Your robot can press the Keepon's buttons for responding to music and touch	**10**							
113. ***Camera:*** Your user interface shows what the camera sees (for exemplary, this is updated automatically at a reasonable rate)	**5**							
114. ***Camera:*** Your user interface shows a processed form of what it sees (e.g. blobs) (for exemplary, this is updated automatically at a reasonable rate)	**5**							
115. ***Camera:*** Your user interface allows teleoperation of the camera's pan and tilt	**5**							
116. ***Camera:*** Other interesting features that do not appear elsewhere in this feature list (score depends on difficulty and creativity)	**10**							
117. ***Other sensor(s) that you add yourself (not built-in):*** Your user interface can display its reading	**6**							
118. ***Other sensor(s) that you add yourself (not built-in):*** Your robot uses the sensor(s) to do something interesting. Exemplary requires that the interesting thing has something to do with Keepon.	**10**							
119. ***Other sensors:*** Other interesting features that do not appear elsewhere in this feature list (score depends on difficulty and creativity)	**6**							
120. ***Motor(s) that you add yourself (not built-in):*** Your user interface can make it act (operate)	**6**							
121. ***Motor(s) that you add yourself (not built-in):*** Your robot uses it to do something interesting. Exemplary requires that the interesting thing has something to do with Keepon. The top two features in this section are examples of this (no double-counting).	**10**							
122. ***Motors:*** Other interesting features that do not appear elsewhere in this feature list (score depends on difficulty and creativity)	**6**							
123. ***Volume control*** (of the Create)	**10**							
124. Make good use of ***multiple cores*** in your laptop (laptop hardware, see instructor for details)	**10**							
125. Other hardware:	**10**							

Total possible for *Hardware*: 0 + 595 = 595 points (required/additional features)

Your Ideas: *Suggest your ideas to us, we'll tell you whether or not they earn points:*

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Features (functionality)	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
126.	**?**							
127.	**?**							
128.	**?**							
129.	**?**							
130.	**?**							

Total possible for Features: 425 points on required features, plus **2,265** points on additional features = **2,690** points (***but limited to a maximum of 800 for Features – more is great, but won't increase your score***)

Quality of code:

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Deliverable	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
131. Final version of code: is decomposed into functions in a reasonable way	**5**							
132. Final version of code: re-uses functions where practical, does not contain repetitive code	**5**							
133. Final version of code: functions are reasonably sized (generally 5 to 20 lines of code, but exceptions are possible)	**3**							
134. Final version of code: no magic numbers	**2**							
135. Final version of code: well-chosen variable and function names	**2**							
136. Final version of code: Every module (file) has a comment at the top that lists the author(s) and briefly describes that module	**1**							
137. Final version of code: All code meet the standards imposed by *Source ~ Format Code* (control-shift-F) in Eclipse	**1**							
138. Final version of code: white space is used appropriately (in particular, there is one line – no more and no less – between each function definition)	**1**							
139. Final version of code: meets the other coding standards that we have demonstrated throughout	**1**							
140. Code after Sprint 1: is of good quality (per the above characteristics, summarized)†	**4**							
141. Code after Sprint 2: is of good quality (per the above characteristics, summarized)†	**5**							

Total possible for *Quality of Code*: 150 points (but capped at **100 points**)

† ***To earn this feature, it must be demonstrated shortly after* THIS *Sprint ends – NOT LATER.***

Required *team* deliverables:

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Deliverable	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
142. Screen Layout (can be very rough and does not need to be kept up-to-date, must be demonstrated during or shortly after Sprint 1)	**1**							
143. Release Plan for Sprint 1†	**4**							
144. Release Plan for Sprint 2†	**5**							
145. Release Plan for Sprint 3†	**5**							
146. Structure Chart or other document that organizes the code for Sprint 1†	**1**							
147. Structure Chart or other document that organizes the code for Sprint 2†	**1**							
148. Structure Chart or other document that organizes the code for Sprint 3†	**1**							
149. Code after Sprint 1: Every function has a doc-comment that specifies the function – what it does (not how!), the parameters expected, and the value(s) returned (partial credit even if your comments are brief – don't bog down in documentation, use it to help your work)‡	**4**							
150. Ditto, for Sprint 2‡	**4**							
151. Ditto, for Sprint 3‡	**4**							

Total possible for *Team Deliverables*: 150 points (but capped at **100 points**)

† ***To earn this feature, it must be demonstrated shortly after* THIS *Sprint begins – NOT LATER – and then updated and demonstrated again shortly after* THIS *Sprint ends – NOT LATER.***

‡ ***To earn this feature, it must be demonstrated shortly after* THIS *Sprint ends – NOT LATER.***

Required *individual* deliverables:

For ______________________________

		Exemplary		Satis-factory	Needs improvement		Not done	Weight × Score
Deliverable	**Weight**	**5**	**4**	**3**	**2**	**1**	**0**	**Points**
152. Task List for Sprint 1†	**3**							
153. Task List for Sprint 2†	**4**							
154. Task List for Sprint 3†	**4**							
155. Code in your module(s) after Sprint 1: is of good quality (per the characteristics listed in the Quality of Code section, summarized)‡	**1**							
156. Ditto, after Sprint 2‡	**2**							
157. Ditto, after Sprint 3‡	**2**							
158. Code in your module(s) after Sprint 1: Every function has a doc-comment that specifies the function – what it does (not how!), the parameters expected, and the value(s) returned (partial credit even if your comments are brief – don't bog down in documentation, use it to help your work)‡	**1**							
159. Ditto, after Sprint 2‡	**2**							
160. Ditto, after Sprint 3‡	**2**							
161. Peer evaluation for Sprint 1 is honest, thoughtful and perceptive‡	**2**							
162. Ditto, for Sprint 2‡	**2**							
163. Ditto, for Sprint 3‡	**5**							

Total possible for *Individual Deliverables*: 150 points (but capped at **100 points**)

Instructor's judgment of the degree to which you contributed to your team (your points are multiplied by this number):

Appropriate contributions: 100% Not as well as we expect: ______________________

Your score = (Team Total + Individual Total) × Contribution Multiplier

= (____________ + ______________) × _____________

= _______________ of 1,000 (but the maximum possible score is 1,100, i.e., 110%)

† *To earn this feature, it must be demonstrated shortly after THIS Sprint ends – NOT LATER – and may be checked DURING the Sprint. You should update your Task List at every class session.*

‡ *To earn this feature, it must be demonstrated shortly after THIS Sprint ends – NOT LATER.*

For those interested, here is a breakdown of the points:

	Points available from:		
Category	**Highlighted features** (team must complete these features to earn other points)	**Additional features**	**Total points available from features**
Start, Interrupt and Quit	25	50	75
Display Project Information	30	5	35
Display the Robot's State	0	140	140
Teleoperation	65	110	175
Semi-Autonomous Motion	55	165	220
Finding Keepon (autonomous)	100	495	595
Jamming with Keepon	30	205	235
Talking to Keepon	30	265	295
User Interface	90	235	325
Hardware	0	595	595
Total from Features (capped at 800):	425	2,265	2,690 (but limited to a maximum of **800**)

Process points:	**Available**	**Capped at**
Quality of Code	150 available (but up to -150 for very poor work)	100
Team Deliverables	150 available (but up to -150 for very poor work)	100
Individual Deliverables	150 available (but up to -150 for very poor work)	100
Total from Process (capped at 300):	450	**300**
So the total available is 3,140, but capped at 1,100 = a score of 110%		

So:

- A team that is *perfect* on all the high-lighted features, process and code, but does nothing else, scores 725 points, i.e., a low C.
    - High-lighted features must be completed (not necessarily to perfection) before any additional points can be earned from Features.
    - It is not hard to earn 300 points of 300 for process – it just requires steady effort.
- Each letter grade increase (C to B, B to A, A to A+) requires about 10 additional "easy" features or 4 medium-hard additional features. Note: "easy" is relative to the others, which range from hard to impossible.
- Earning 110% requires maxing out on both Features (800) *and* Process (300).