Capstone Python Project – ***Features (draft)***

CSSE 120, Introduction to Software Development

General instructions:

The following assumes a 3-person team. If you are a 2-person or 4-person team, see your instructor for how to deal with that.

General Project Requirements:

- All features MUST be implemented in a ***Graphical User Interface (GUI)***. This means that you ***should NOT be using the Console for any of your input or output.***
  - All team members ***must*** contribute to the GUI.
- ***To earn a passing grade, each team member*** must complete one ***green*** feature, one ***blue*** feature, and one ***light yellow*** feature.
  - This is a ***BARE MINIMUM*** and will result in a C- for the project if you have excellent process and high-quality code, else a lower grade.
  - Green and blue features are simpler, yellow are more sophisticated.
  - Uncolored features are open-ended and provide room for creativity.
- ***To earn a high grade, each team member must*** complete the ***minimum requirements*** above ***and*** complete one of the ***dark yellow*** features ***and*** complete ***uncolored*** features, including some that are genuinely challenging. There is no set number for this; do the best your team can (and ask your instructor for guidance as needed).
- Use as many ***different kinds of GUI widgets*** as you can.
- Use a strong software development ***process*** throughout.

The best projects will take care to re-use each other's GUI, functions and data wherever practical.

Grading and demos:

The grading is based on the ***success of the team as a whole*** as well as ***your own individual contributions*** to the team, including but not limited to:

- Did you implement your required features, with correct and complete code?
- To what extent did you go beyond that, both in quantity and in level of challenge?
- Did you use a strong software development process, including using Trello throughout to track your work, keeping track of your hours, using meaningful commit messages, and using iterative enhancement?
- Is your code high-quality?
- Is your code documented appropriately?

On Friday of 10th week (or possibly during the weekend that follows it), your instructor will require that each team give a demo of all of the features that were implemented.

Due date:

The final project code is due at the start of class on Friday of 10th week.

Features (brief version – longer version coming soon):

Each of the following features will have a longer, more complete, description. The following descriptions convey the basic idea of the feature, but not its details. ***THIS IS A DRAFT – details may change.***

1. [Team-coded, with your instructor] The user can ***connect to the robot,*** as well as ***shut it down cleanly***, with a way to specify the robot's IP address.
2. The ***GUI indicates,*** for ***each Sprint*** and ***each team member***:
    - The ***total hours*** that the team member worked during that Sprint.
    - The ***cumulative hours*** that the team member has worked up to and including that Sprint.
3. ***The robot can GO STRAIGHT*** in a specified ***direction*** (***forward or backward***) for a specified ***distance (in inches or centimeters)*** at a specified ***speed***, at some reasonable degree of accuracy.

    Additionally, the code contains ***functions that the entire team can use*** whenever the robot needs to go a specified distance.
4. ***The robot can SPIN (in place)*** in a specified ***direction*** (***left/right***) for a specified ***number of degrees*** at a specified ***speed***, at some reasonable degree of accuracy.

    Additionally, the code contains ***functions that the entire team can use*** whenever the robot needs to spin a specified number of degrees.
5. The robot can be ***tele-operated*** (i.e., remote-controlled, like a remote-control car) with **keyboard keys**.
6. The robot can ***spin and move forward/backward to track an object*** using its ***camera*** (for determining what spinning to do) and its ***front proximity sensor*** (for determining what forward/backward motion to make).
7. The robot can ***search for an object by moving its head left/right and up/down***, and then ***move so that its head is pointing straight at the object***.
8. [Team-coded, with your instructor] ***Long-running loops can be interrupted.***
9. ***The robot can use its reflectance sensors to follow a curvy black line,*** using ***Bang-Bang control*** as well as using ***PID control***.
10. ***The robot can use its encoders to go straight,*** using ***Bang-Bang control*** as well as using ***PID control***.
11. ***The robot can use its camera and front proximity sensor to follow an object,*** using ***Bang-Bang control*** as well as using ***PID control***.
12. Given a list of x/y coordinates (***waypoints***), the robot can ***move to each***, pausing briefly at each waypoint, using Manhatten movement (i.e., all movement is along the x-axis or y-axis).
13. The robot can ***"talk" to another*** robot using head movements and other forms of communication.
14. The robot can ***parse files with songs and play the songs*** using its buzzer.
15. The robot can move quasi-randomly, using its sensors to ***avoid objects, stop at lines***, and more.
16. The robot can ***display emotion***.
17. The robot can ***compose*** music, and then play its compositions.

18. The robot can ***watch a conductor and play music accordingly***.
19. The robot can ***compose a fictitious bio*** for itself and/or for you.
20. The robot can do ***sophisticated movements***, e.g. trace a regular polygon, parallel park, and more.
21. The robot can ***tweet***!
22. Use a ***Leap Motion device*** (and accompanying Python software) to control the robot with hand movements.
23. Do ***interesting things*** (beyond those already listed) with the robot's ***camera***.
24. Do ***interesting things*** (beyond those already listed) with the robot's **standard sensors**.
25. Do interesting things with ***additional motors, servos and/or sensors.***
26. Use ***swarm techniques*** and/or distributed algorithms to accomplish interesting things.
27. Use ***parallel algorithms*** (in processes and/or threads, in a single processor or across cores) to accomplish interesting things.
28. Use ***internet communication*** and/or ***files*** to do interesting things.
29. ***Interact with a different kind of robot***, e.g. a quadcopter or BERO robot.
30. Do something interesting… ***[You suggest what!]***