

Capstone Python Project – **Features (draft) for Mutchler's sections**

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 generally 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 **grey** 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):

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 ***disconnect from it cleanly***, with a way to specify the robot's port (for wired connections) and IP address (for wireless connections).
2. **The GUI displays the contents of the hours-X.txt file for each team member.**
 - See Feature #4 below for a continuation of this item.
3. The GUI provides a way for the user to set:
 - **The speeds of the wheels** (with some way to indicate forwards vs. backwards), in some reasonable units.
 - **The time in seconds that the robot should move.**

Additionally, there are **functions that the entire team can use to get the current values** of the above.

Additionally, there is a **button or other mechanism to make the robot move** at the specified wheel speeds for the specified number of seconds (and then stop).

4. **The robot can GO STRAIGHT for a specified distance (in inches or centimeters)** in a specified **direction (forward or backward)** at a specified **speed**, at some reasonable degree of accuracy. Here, "reasonable" means "reasonably straight" and with accuracy that is generally within a few inches. See Feature #10 for an extension of this Feature that strives for better accuracy.

Additionally, the code contains **functions that the entire team can use** whenever the robot needs to go a specified distance.

The implementation of this feature MUST use relevant aspects of the previous feature – both GUI widgets and functions.

IMPORTANT: All subsequent features should use the **GUI and functions** provided by Feature #3 and Feature #4 wherever appropriate.

5. **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.

This feature is an extension of Feature #2. Keep the code for Feature #2 unchanged (so that the student can get credit for it). ADD code for this feature. If you wish, you may replace the display from Feature #2 with the display from this Feature.

6. **The robot can be tele-operated** (i.e., remote-controlled, like a remote-control car) with **keyboard keys**.

The user should be able to set speeds using the GUI item(s) from Features 3 and/or 4. If the keys themselves also modify speeds, the current speed should be displayed.

7. The GUI allows the user to:

- **Display the current values of the reflectance sensors.**
- **Set thresholds** for one or more of the reflectance sensors, and have the **robot move forward or backward until a threshold is exceeded** (and then stop).

8. [Team-coded, with your instructor] **Long-running loops can be interrupted.**

IMPORTANT: All features should be designed to re-use existing code and to be re-usable by yet-to-be-implemented code, wherever practical. For example, Feature #9 should use aspects of Feature #7.

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 have the robot go straight per Feature 4, but with greater accuracy – more straight** (using **Bang-Bang control** as well as using **PID control**) and stopping **more closely to the desired distance**.
Likewise, 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 (again using encoders).
 Additionally, the code contains **functions that the entire team can use** whenever the robot needs to spin a specified number of degrees.
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 Manhattan movement (i.e., all movement is along the x-axis or y-axis).
13. The robot can **“talk” to another robot** using movements and/or 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. Make the low-level communication more efficient, or otherwise ***augment what the Arduino can do*** on behalf of the Python program that is controlling the Arduino.
30. ***Interact with a different kind of robot***, e.g. a quadcopter or BERO robot.
- 31.** Do something interesting... ***[You suggest what!]***