Secret Agent Man!

Capstone Python Project

Features

CSSE 120, Introduction to Software Development – Robotics

Spring term, 2013-2014

Features: (***SAM*** stands for ***Secret Agent Man – your robot***).

Per the song *Secret Agent Man*, as performed by Johnny Rivers at http://www.youtube.com/watch?v=6iaR3WO71j4, your robot is a Secret Agent Man. As such, he has to find various Bond Girls – Anya Amasova, Wai Lin, Honey Ryder and others. Sometimes he just finds her, sometimes he talks with her, whatever. This document lists the features by which he can do so. [Note: SAM and Bond Girls are just a "theme" for this project – implementing the features does not require a Secret Agent Man or Bond Girls, although you can incorporate them explicitly if you wish.]

All features must be implemented in a nice ***Graphical User Interface* (GUI)**. Each person must provide a GUI for their features.

- All input and output must be from your GUI (plus any optional external devices you might use, like Wiimotes). There must be ***no input or output from/to a Console window*** (except for debugging purposes).
- The more different kinds of GUI widgets, the better.
- The more your follow good GUI design principles (and you can explain how your GUI does so), the better.

There are ***Basic***, ***Advanced*** and ***Unmarked*** features. Your instructor will explain to you the role of each, along with the role of the color-coding, in your grading. But the spirit of the grading is:

- ***Green features must be implemented.***
- There are 3 pairs (Basic/Advanced) of Blues, and 3 pairs (Basic/Advanced) of Yellows. ***Each student*** implements a ***Basic Blue***, an ***Advanced Blue***, a ***Basic Yellow*** and an ***Advanced Yellow***. Restriction: **for any Basic that you implement, a teammate (not you) must implement its matching Advanced, using your code where practical.**
- ***The team must ALSO implement more*** to earn a high grade. There are LOTS of options for what to do!

Greens and blues are simpler robot ideas. Yellows are more sophisticated robot ideas (that are more challenging to implement).

Each feature has ***multiple sub-features***. Generally speaking:

- For ***Basic*** features: you must implement ***all the sub-features*** to achieve full credit for that feature.
- For advanced and unmarked features: There are have lots of options for which sub-features you can implement. The more, the better, but doing ALL the sub-features is (for most features) far beyond what is reasonable. Sometimes the sub-features have nothing to do with each other.

0. ***Each student does this feature:*** ***Your portion of the GUI indicates,*** for each Sprint, your name and ***the total hours that you worked on that Sprint***. This information must appear at the end of each Sprint! Additionally, display the total hours worked on the project. For full credit, read this information from a file (instead of putting the data in the code). Since each student does this feature, EACH of you makes a card for this.

0. ***There is a Connect and Disconnect button, plus a way to specify the port (possibly 'sim')***. Those buttons behave as their names suggest. There should be a SINGLE Connect button in the project.

1. ***Basic:*** SAM can be ***tele-operated*** (i.e., remote-controlled, like a remote-control car). The user can use ***buttons*** to move SAM forward and backward, spin her clockwise and counterclockwise. Additionally, the user can change her speed while she is moving/spinning.

2. ***Advanced:*** SAM can be ***tele-operated*** (i.e., remote-controlled, like a remote-control car). The user can make SAM move in curves (i.e., linear and angular motion at the same time). Uses easy-to-operate interfaces like keys (without interfering with other features!), gamepads, wiimotes, or other remote-control devices (perhaps wireless).

3. ***Basic:*** SAM can ***move autonomously***, by going a specified ***distance*** in a specified ***direction*** at a specified ***speed***. That is, the user can set the direction (forward, backward, spin left or spin right) and the distance (in some reasonable units). Then, the user can make the robot go (e.g. by pressing a *Go* button) and the robot should move the specified direction for the specified distance, with some reasonable accuracy.

4. ***Advanced:*** SAM can ***move autonomously***, by going a specified ***distance*** in a specified ***direction*** at a specified ***speed***. There are multiple implementations (any of which can be chosen by the user), with demonstrated understanding of when and why one is better/worse than another. For example, one implementation is the "time" approach and another is the "distance sensor" approach. There is high accuracy for the best implementations. Can move linearly and angularly at the same time, with some reasonable understanding of "distance" and "speed" in that case.

5. ***Basic:*** SAM can ***move autonomously***, by going ***until an event occurs***. The user can set the speed and which bumpers to use (both, just-left or just-right). Then, the user tells the robot to start, at which point the robot moves until the relevant bumper(s) are pressed. Likewise, the user can set the speed and a "darkness level." Then, the user tells the robot to start, at which point the robot moves until its front cliff sensors (one or both, your choice) see a black line of sufficient "darkness".

6. ***Advanced:*** SAM can ***move autonomously***, by going ***until an event occurs***. This is like the basic, but using different, multiple and more sophisticated sensors – the infrared knob on the top of the robot and/or the camera, for example. In each case, the user sets some values, then makes the robot start moving, then the robot continues until the sensors give values in the ranges that the user has set. The best implementations will require multiple sensors mixed in interesting ways, e.g. the infrared hears 100 followed a second later by 200.

7. ***Basic:*** SAM can ***follow a black line.*** ***Uses P (proportional) control***. Can follow a curvy black line about 2 inches wide, with reasonably gentle curves, using the left front signal (for the left wheel speed) and the right front signal (for the right wheel speed). (You can also use other sensors if you wish.) The P constants are tuned reasonably. Auto-calibrates the darkness of the lines under current lighting conditions by the human placing the robot in positions as desired (with no changes to the program needed for this process). Must work under various lighting conditions and with various robots. (Their IR light sensors vary wildly from robot to robot, and even within a robot!)

8. ***Advanced:*** SAM can ***follow a black line using PID and other control, and possibly can follow a wall too.*** The I and D constants are implemented (but possibly not tuned perfectly, as that may be hard). The user can set all the parameters at run-time, ideally even while the robot is doing line-following. Uses additional sensors. Can follow a wall, using "bump and bounce" and perhaps also PID. Performs wall-following as well or better than the demo.

9. ***Basic:*** SAM can move to ***user-specified waypoints***. That is, the user can enter a sequence of (x, y) coordinates and tells the robot to go. Then, the robot moves to each, one after the other. (The origin of the coordinate system is where the robot began the sequence of moves.)

10. ***Advanced:*** SAM can move to ***user-specified waypoints***. As per the Basic version of this feature, but additionally some or all of the following: The robot can move around obstacles as it moves from waypoint to waypoint. There is a nice way to enter coordinates (e.g. by clicking on a map). Coordinates come from a file. User can control speeds as well (perhaps via pre-specification, perhaps via teleoperation, perhaps both). The robot remembers paths that it is teleoperated and then can reproduce the paths autonomously. The robot keeps track of its position through ALL its movements (even those produced by teammate's code) and the way-point movement is relative to the position at which the robot began its operation, not the position at which it began the sequence of way-point moves.

11. ***Basic:*** SAM can ***chat with another robot via IR or another appropriate sensor***. [This description uses IR, but other sensors might be able to be used in a similar way.] User can make SAM start/stop emitting a user-specified IR signal. SAM displays whatever IR signal it is currently hearing. SAM can "chat" via user-specified IR numbers sent synchronously: SAM starts sending, then listens until it hears something from the other robot, then starts sending something different, then listens until it hears something from the other robot, etc. You can assume that the other robot never sends back immediately the same number SAM just sent, that no robot sends the same IR signal twice in a row, and any other simplifying assumptions that are required (ask your instructor about any such assumptions as needed).

12. ***Advanced:*** SAM can ***chat with another robot via IR or another appropriate sensor.*** As per the Basic version of this feature, but additionally some or all of the following: Uses codes to send letters, words and entire phrases. Encrypts and decrypts (perhaps as simple as Caesar's cipher, or as complicated as a public key encryption system). Can use a file-specified encoding system. Communicates asynchronously, or uses more advanced protocols than the basic (either standard ones sort of like TCPIP

or ones that you develop yourself). Does handshaking to identify itself.

13. SAM ***follows another robot***. Uses the camera, or uses the "caps" for directionality with the other robot emitting IR, or the other robot sends codes to indicate directionality, or …

14. SAM ***sings and dances*** with a light show. Songs of more than 16 notes. Plays MIDI from a file. Composes songs – randomly, or with principles from music theory. Likewise for dances and/or light shows. Does the light show while dancing and singing, perhaps choreographed.

15. SAM does interesting things with ***external motors, servos and/or sensors***: moving something, shooting something, sensing something, or …

16. SAM offers ***Rogerian psychotherapy***, ala Eliza (http://en.wikipedia.org/wiki/ELIZA).

17. SAM uses ***swarm techniques*** and/or distributed algorithms to accomplish interesting things.

18. SAM uses ***parallel algorithms*** (in processes and/or threads, in a single processor or across cores) to accomplish interesting things.

19. SAM can go until it is "stuck" (still trying to move), no matter what the direction (not just forward). (Hint: this is easy if you figure out the surprising sensor to use.)

20. SAM does interesting things with ***computer vision***: e.g. finding objects, using semaphores to communicate, or … [Note: this item requires figuring out how to connect the camera to the robot.]

21. SAM uses ***files or internet communication*** to do interesting things beyond that described above.

22. SAM displays a short, fictitious bio (for him/her). Possibly with fictitious bios for each of you (students) as well.

23. SAM uses a Leap Motion device (and accompanying Python software) to control the robot with hand movements.

24. SAM … ***[You suggest something interesting!]***