

P2 Code Review

Introduction:

The code we are reviewing was written for our main robot: BigBot. Bigbot will start off by grabbing a gas link from the shared area and connecting it to the gas pipe. After this, he will turn around and grab the second link (brought by SmallBot). When done with the pipe connection, BigBot will move on to the electricity connections. The code was written by Sean McManus and Jovan Karlius. The review was done by Shaurya Singh and was conducted on 17/1/2019.

Best Practices Checklist:

1. Code uses functions for organisation. |DONE|
2. Code contains comments documenting each functions purpose. |NOT DONE|
3. Code contains comments documenting each function's arguments |NOT DONE|
4. Code contains comments documenting each function's return values |NOT DONE|
5. All variable names are descriptive and convey use in code |DONE|
6. No unnamed constants other than 0,1, or 2 |DONE|
7. Code is formatted to show flow control |DONE|
8. Removed commented out code that is no longer in use |NOT DONE|

Instead of commenting out redundant code, the code was completely removed from the syntax. Also, all team members understood the code so there was no need to comment it. This was an error on our part, and everyone involved has been notified so this does not occur in the future. Functions are used for organisation and all variables have appropriate names, with no unnamed constants. The program also shows the steps the robot will take when operating its claw in sequence.

General Code Analysis:

Reliability –

Although we have hard coded the route BigBot will take, we still wanted to make sure that we had a contingency measure in case things went wrong. This is where the black/white colour sensor comes in. We attached the sensor to the front of the robot, with Lego. With most of the board being white and the paths having black tape on them, we programmed the sensor so that whenever the robot went at an angle, the sensor would detect a colour change from black to white and then turn the robot back into position (detecting another colour change from white to black). This would ensure the robot followed the right path and got to its locations.

So far, we have are quite comfortable with the code and how it works. However, the robot does tend to turn increasingly frequently after encountering a colour change due to the angle at which it rotates not allowing it to continue a straight path. To make this less likely to happen, we can increase the speed of the robot and bring the sensor closer to the ground. This way, the sensor can detect a colour change faster and the speed allows the robot to arrive at its location before the minor correction turns begin to build up to a significant amount. We are currently undecided as to whether we should implement this piece of code to make it more accurate as the current code is already quite reliable.

Maintainability –

Before we began to code, our team gathered around and made a pseudocode version of everything BigBot would do. As such, all members agreed on the code and thus everyone has a rough idea of how the program works, making it easy to understand and modify. To make sure we don't lose any code, after every session,

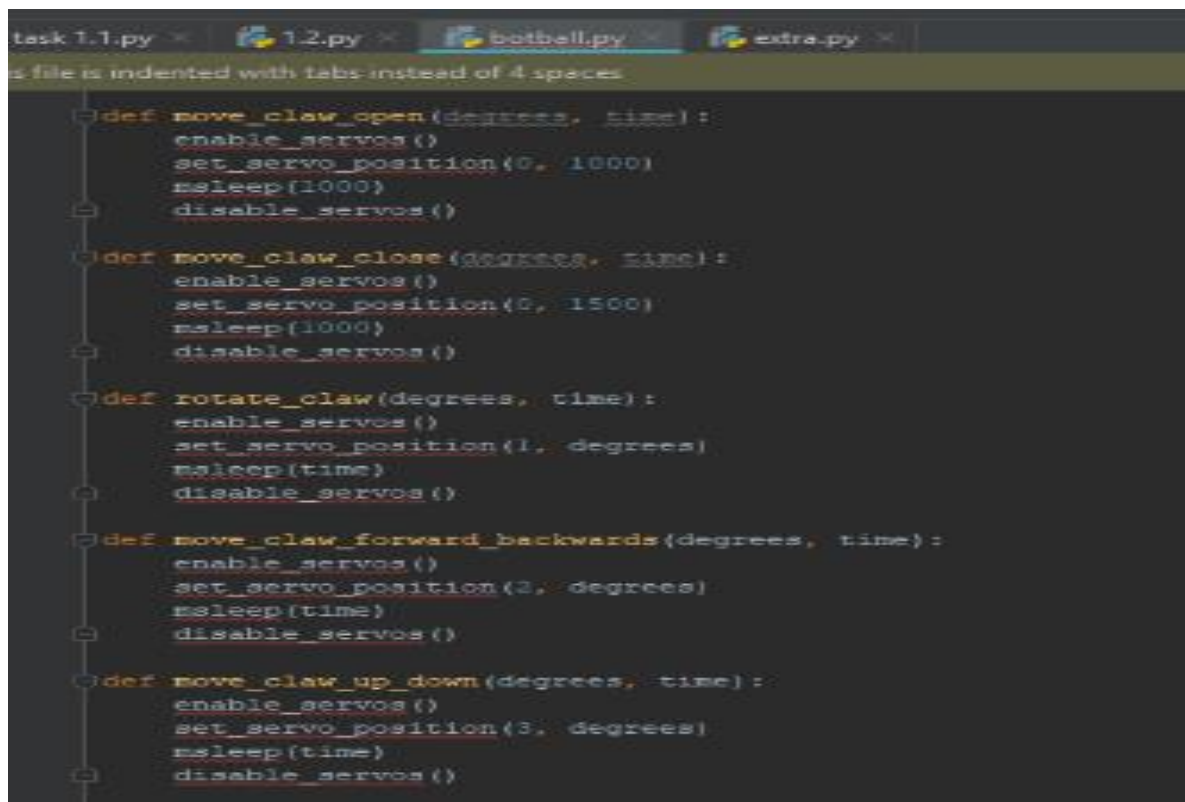
Sean sends the file to Jovan and Ahmed, so they have a copy in case Sean happens to **ACCIDENTALLY DELETE HIS!** Because BigBot and SmallBot have similar designs (ex: both have a effector and claw) we found that parts of BigBots code can be reused for SmallBot, with a few tweaks of course.

One way to improve the maintainability of the program is to set up a google drive folder (or something similar) and store the program files in it whilst sharing the folder with all team members. But that seems a bit extreme as we already have a reasonably good maintenance and security system.

Effectiveness –

Our program is not a 100% effective. Does it get all the tasks done? Yes. Is there a better way to code the tasks? Absolutely. Right now, Sean and Ahmed are focused on getting all the tasks done reliably (without bugs) while Jovan tries to improve on already existing code. So far, the program is completing all tasks (making it for all intents and purposes, effective).

However, to improve the code, we need to follow top down design, and separate monoliths of code into reusable functions. Jovan is tidying up all of Sean and Ahmed's raw code into simpler functions so that the rest of the team can understand the program and some parts of the code can be reused. An example of some of Jovan's improvements can be seen below



The image shows a screenshot of a code editor with four tabs: 'task 1.1.py', '1.2.py', 'botbell.py', and 'extra.py'. A message at the top states 'this file is indented with tabs instead of 4 spaces'. The code defines five functions for controlling a robot's claw:

```
def move_claw_open(degrees, time):
    enable_servos()
    set_servo_position(0, 1000)
    msleep(1000)
    disable_servos()

def move_claw_close(degrees, time):
    enable_servos()
    set_servo_position(0, 1500)
    msleep(1000)
    disable_servos()

def rotate_claw(degrees, time):
    enable_servos()
    set_servo_position(1, degrees)
    msleep(time)
    disable_servos()

def move_claw_forward_backwards(degrees, time):
    enable_servos()
    set_servo_position(2, degrees)
    msleep(time)
    disable_servos()

def move_claw_up_down(degrees, time):
    enable_servos()
    set_servo_position(3, degrees)
    msleep(time)
    disable_servos()
```

