Generally speaking, the task has been dividend into 2 stage with several smaller tasks:

* Stage1:

1. Start at a tile, towards the lane which is consisted of black and white tiles.
2. Walk in the lane.
3. Count encountered black tiles while making sounds to indicate that.
4. Stopped after encountered 15 black tiles and turn right

* Stage2:

1. The robot should move forward which the tower should be in front of it.
2. It should make contact the target
3. Push tower off the square it sitting on and make a sound to indicate it has finished.

The main associated problems need to be solved are:

1. The robot should distinguish the black and one tile. (stage1)
2. The robot should walk in the boundaries next to its both side. (stage1)
3. The robot should head towards to the tower which its position is not uncertain. (stage2)

When we try to implement the algorithm to finish the tasks. Our first intention is naturally try to solve them as human. We want to use the ROBOTC’s task ability to execute the different task concurrently. It is very nature to think that way. For example, for the task of walking ahead and counting the black tile it encountered. It is just “let it keep walking” and “count when it meets a black tile”. This executing multiple tasks functionality the ROBOTC gives us is very powerful. It works well at the beginning when we just want to finish two small tasks. However, as we develop more functions and add into the previous one, we found some annoying bugs which are very hard to debug. At last we cannot trace the logic in the code and everything seems in a mess.

So instead of executing multiple tasks at the same time, we decide to break the task into multiple same steps, and in each step we let it finish multiple function. For example, in stage1, there are 15 same steps, in each step, the robot should should finish those simple tasks:

1. It moves ahead.
2. Stops and makes a sound once it encountered a black tile.
3. It detects its positon relative in the black tile.
4. It corrects its errors according to its position.

Once it repeated this step for 15 times, it will finish the stage1. This approach makes our code become much simpler to debug (After all, this is the nature way to write programming). Similar for stage2, the robot needs to finish basically two tasks:

1. Moves forward and stop when it encountered a black tile.
2. Scan for a certain degree ahead of it which ended towards the direction of target.

Except for this main problem, other problem seems easier to solve. For, example, the movement of robot depends heavily on the moving and turning. But the command for controlling the robot’s movement at specific degree is error prone. So, for keeping consistency and reduce the error, we create some helper functions to wrap those commands. The helper functions can separate the functionality of movement away from the logic of our task. When we implement the turning and moving, we can just focus on manipulating the build-in functions instead of checking the code and the logic scenario back and forth.

We finished our code during the breaking week. Through this assignment, we really appreciate that it is **hard** to build some robot to finish even simple task. The environment is uncertain and making robot behave based on what it can sense is much more interesting than we expected.

The code is listed as below: