**The algorithms that your robot uses: how they work (in reasonable detail) and why you chose them;**

From the ev3dev.ev3 library we imported everything, so that we could just call upon motors, sensors etc.

We have also imported time, as we will be using it to sleep parts of code so that actions can be performed without anything being overridden.

We initialised a button at the beginning so that we could call upon the button later as a way to stop the script, and this button was any button on the EV3 controller unit.

We initialised the ultrasonic sensor as “ussr” so that we could use it to detect the milk bottle, as opposed to letting the robot go in a straight line and hoping that it is on a path that will allow it to hit the milk bottle.

We had to initialise our motors, and attach them to ports, and then set their position and speeds.

We have a small set of motor commands to get the robot to move from the black arrow start tile to the first black tile on the black/white tile strip, to begin the next stage of the assessment.

We initialised our robots touch sensors, so that later on we can call on them to see if we have hit the milk bottle.

We initialised the colour sensor, with a count variable, and we had the colour mode set to COL-REFLECT as opposed to using COL-AMBIENT.

We have a lock on variable to see if we have locked on to the desired target, which is the milk bottle.

We have a variable “vodka\_bottle” that is assigned to 150, which is used as a base distance for the ultrasonic sensor. We are having the sensor search range be 1.5m, as it should be more accurate, and not be confused by anything that could be in its search radius.

We have set the mode to “US-DIST-CM”, which sets the measurements for the ultrasonic sensor to cm, which is good as we are using 150cm, or 1.5m, as our ultrasonic sensor search range.

We then have assigned black (blk) to a colour value so that we can use it for colour detection with the colour sensor, which will also be tied into an audible noise when a black square is under the colour sensor. This is the “colour calibration” method we have used, to get it to call out that it can see the black or white, and its colour value.

We have assigned white (wht) to a colour value as well, so that we can use it for colour detection with the colour sensor, like we did with the black.

We used time so that we could sleep our colour checker so that it didn’t get tripped up by the grout lines around the black and white squares. This time is proportional to speed, so if we were to speed up the motors, we would have to decrease the sleep timer so that it was still waking up at the appropriate time to check what colour was where.

We have two methods by the names of move\_one\_blk() and move\_one\_wht(), which

We have a for loop that takes a range of 2 to 16, the 2 is the minimum parameter because the first black square that we used to calibrate with is counted as 1, and 16 as our maximum parameter because we already have one square counted from the calibration square. This allows us to count 15 black squares effectively. This means that after the range is complete, the loop will be finished with and the next part of the code, and the assessment can happen. We also have our button exit command here, so that if any button is pressed the loop will break.

We have the robot now turn to the right, and continue on this trajectory for some time, at a rather high speed as it does not need to be very accurate due to the fact that it merely needs to cover a distance to get close to the bottle because the ultrasonic sensor will be taking over next, and the range is set to 1.5m.

Next is our loop that incorporates the touch sensors and the ultrasonic sensors. It is designed so that if it finds something, it can lock onto it and travel towards it. If it does not find something, it has the ability to scan in an arc, and will lock on to the first item that it finds in the scan, and travel towards it. The only way out of this loop is the touch sensors, which should get pressed in when the robot hits the bottle.

The robot is then being halted, it will play a confirmation note to say that it has hit the bottle, and will proceed to turn the wheels at a low speed as to get power down so that the bottle can be moved off of the black square.

**The problems that you overcame (or not) while coding and testing your robot.**

**Problems:**

Motor bias pulling to the right depending on robot?

With the straight line approach, the motor bias can not really be overcome. We have to hope we get a somewhat balanced robot (motor wise) and that we position the robot very square at the start, and that there is no harsh torqueing action from the motors stopping and starting suddenly that will skew the course of the robot.

Getting the robot to actually count the black squares correctly

We managed to get it through some refining of colour parameters, and getting our speed and sleeps correct for speech confirmation that a black was spotted and counted.

Getting the robot to turn right

To fix this all we had to do was gain an understanding of how the speed and position of the wheel worked, and from here it was not too hard to get it to turn right after some trial and error, based on our speed of rotation and a well-placed sleep to stop it from going off on an angle that would not allow the bottle to be found.