For the lightSeeker behavior, we will first need to normalize the light sensor value to the motors value by defining a normalize() function. Then we plan to program the robot to use the three light sensors to determine its path. If there is more light on the left side, the robot will turn to the left. If there is more light on the right side the robot will turn to the right. If there is more light in the center, the robot will move forwards.

We ended up making a program that makes the robot seek out darkness. It still does the previous things listed, if it is darker on the right, the robot will turn right. If it is darker on the left, the robot will turn left. If it is darker on the center, the robot will move forward. What we did not include in our previous plan is a while loop or timing. This is important because while we are in the while loop for a certain amount of time it will continuously check the time and make adjustments to the movements. In order to fix our error, we realized that the lower values of light are the brighter spots. Previously, we programmed the robot to follow the highest light sensor values that are the highest(in other words, the darkest spots). Now we programmed it to follow the lowest values(the brightest spots).

———————

For the avoid(time) challenge we decided to make our robot stationary until something moved in front of it or seemed to be coming towards it. The getIR() or getObstacle() function spits out numbers ranging from 0 to 6500. 6500 meaning that something is in the way. For our avoid function we decided to pick a high value (>4500)and when the robot returns those numbers it will evade, or move backwards from the object.

We started by using the getObstacle() function and by using while and if statements able to make it so that the robot stayed stationary for the duration of “time”. But if something were to come into its path or move towards it, it would evade it. We also tried the getIR() function but found getObstacle() easier to use for our purposes.

————————————

For the security guard, we knew we needed to count in for the ambient light. When the light value passes the threshold, it will take a picture and beep.

We ended up using the threshold, so when the central light sensor passes a reading below 5500, then the program will activate. If it does not, we used a recursive function that would make the loop repeat itself until the light sensor does go below the value threshold. When the value of the central light sensor goes below 5500, and beeps for five seconds at a hertz of 600

——————————————
‘

For the digitalCamera() program, we decided to write it almost how we wrote the program that moves towards the light except we would change a few things. We plan on just changing the if statements so that when each sensor becomes completely dark (aka reaches 6300) to take a picture or grayscale picture depending on the sensor covered.

We used if statements to deligate what each sensor would do. One if statement made it so that when getLight() for the left sensor read over 6300 it would take a picture and vise versa on the

right it would take a grayscale picture. We also added an else statement that if no input was found it would continue to print a prompt until it received a finger on the sensor.