UPPSALA
UNIVERSITET

Real Time System Course, 1DT004

Lab 2: Programming in LEGO Mindstorms NXT

Report

Group Number: 2

Arijeet Laga Sagar Shubham Kedar Joshi
Aakash Goel

October 5, 2018

Contents

1 WARM-UP

1.1 Question 1

Do you see any variation in light sensor values depending on the color of the surface? Give an idea of how to distinguish between two different surfaces (light and dark) using this light sensor.

Yes, there is a variation in light sensor values depending on the color of the surface. As we know that light surface reflects more light as compared to the dark surface.There is variation of light sensor values while reading light and dark color. Surfaces of dark shades, such as black and brown, give a lower output value on the sensor, as they absorb most of the light incident on them. On the other hand, lighter shades, such as white and yellow give a higher value as they reflect most of the light incident on them. The maximum difference, was thus observed between surfaces of white and black shade, with reading on white surfaces between $60 - 70$, and that on black surfaces between $30 - 40$.

To differentiate between two different surfaces, we first calibrate the sensor, by putting it in close proximity of a complete black surface, and then on a complete white surface. We note down the lowest(on black surface) and highest(on white surface) values as output by the sensor. Now we can place the sensor on any other surface of different shade, and use the sensor output to differentiate it from other surfaces, based on the output's difference from complete black and complete white surface's readings.

2 EVENT-DRIVEN SCHEDULING

2.1 Question 1

Is it possible to implement part 2 using only one task? Explain advantages or disadvantages of using one task for this assignment according to your understanding.

Yes, it is possible to implement using only one task. The advantage of using only one task is that the total size of the program will be small, but the disadvantage is that we won't have any control over the occurrence of the events. For instance, we won't be able to decide when to read the light sensor data and when to operate the motors. Thus a single task model is akin to polling the touch sensor, which has lower space overhead, but might result in missing the actual event. While the multi-task model is similar to an interrupt based model, where the touch sensor activity generates an interrupt which can be used to start and stop the motors quickly,

3 PERIODIC SCHEDULING

3.1 Question 1

How did you implement the driving command and its execution?

We used the `NXT.Motor_Controls` package to drive the motors using the `Control_Motor` API. Initially, we had a seperate task `DriveMotorsTask` which had fourth priority(after main task, reading light sensor, distance sensor and PID calculations) but due to space constraints, we had to let go of that task, and incorporate its actions into the PID values (`calculatePIDValuesTask`) calculation task itself.

In first case, we provided a seperate protected object as `MotorObject`, which had its private members as a record, which in turn held a `Motor_ID`, and present speed and direction of the left and right motors. to which we set the current the speed and direction of left and right motors on the robot. These settings were done by the `calculatePIDValuesTask` Task, while the actual calling of `Control_Motor` API was done by `DriveMotorsTask`, every 50 ms.

In the actual final robot, we used `Motor_ID` directly as an input to calling the `Control_Motor` API from the `calculatePIDValuesTask` Task, as well as providing speed and direcional values according to the calculations from the PID control loop.

4 LINE TRACKER

4.1 Question 1

Explain the way (like algorithm, PID, etc) you have implemented the line tracking functionality in this exercise.

For this part, we had two PID algorthims running in a single task of `calculatePIDValuesTask`. One was used to set line following correction, using $K_{p,line}$, $K_{d,line}$, $K_{i,line}$ values, and the other one was used as a simple $K_{p,distance}$ for following the other robot on the track.

We calibrated our sensor to note down the lowest and highest values possible for the light sensor, by placing it alternatively on a dark and light surface. A set-point was then set as average of these two values. Based on the deviation from the set-point of the values actually received during the run on the track, appropritate compensation was provided to the base speed of the motors to keep the robot on track. We also attempted to remove the integral wind-up, as well as attempted to enforce a maximum motor speed to prevent runaway situations.

The second PID control loop locked on to a object if it was closer than a particular distance to the robot based on the distance sensor readings, and attempted to maintain a constant distance after that, as long as the object locked onto was still within the particular distance mentioned above. This was done by changing the base speed of the robot, using a single P controller. The compensated base speed of the motors was fed into the PID loop for correcting robot's course on the line mentioned earlier. We made sure than if the distance goes too small, the robot should stop and wait till the bot in front starts moving.