Real Time Systems
Uppsala University – Autumn 2015
Report for Lab 2 by Team 4

Martin Kjellin Pei-Chun Chen Jai-Ying Lin

September 28, 2015

Part 2: Event-Driven Scheduling

In this part of the lab, we were asked to write a program using event-driven scheduling. The program should make a Lego car drive forward as long as a touch sensor is pressed and a light sensor indicates that there is a table underneath the car.

We have defined four different events. TouchOnEvent and TouchOffEvent indicate that the touch sensor is pressed or released, respectively. OnTableEvent and OffTableEvent indicate that the reading of the light sensor goes below or above 700, respectively. This threshold was experimentally determined to be a good indicator of the light sensor being above the table or not. The light sensor is mounted in the front of the car, and the car will therefore stop before going over the edge of the table.

According to the instructions, our program includes two tasks. The EventdispatcherTask generates the appropriate events when the touch sensor is pressed or released or the light sensor reading goes below or above the threshold. The MotorcontrolTask repeatedly first waits for a TouchOnEvent and an OnTableEvent before starting the motors and then waits for a TouchOffEvent or an OffTableEvent before stopping the motors. Unfortunately, a bug that we haven't been able to find sometimes makes the program hang.

The code can be found in the files `eventdriven_new.c` and `eventdriven_new.oil`.

Part 3: Periodic Scheduling

Next, we were asked to write using periodic scheduling. The program should make a Lego car keep a constant distance to some object in front of it. Also, users should be able to make the car to move backwards a little by pressing the touch sensor.

As indicated in the instructions, the program includes four tasks that are released periodically: MotorcontrolTask (released every 50 ms) sets the

speed of the motors, ButtonpressTask (released every 10 ms) makes the car go backwards if the touch sensor is pressed, DisplayTask (released every 100 ms) displays information about the current driving commands and the current distance to another object, and DistanceTask (released every 100 ms) tries to keep the car at a constant distance of 20 cm to that object.

The code can be found in the files `periodic.c` and `periodic.oil`.

Part 4: Lego Car Race

Finally, we have written a program that makes a Lego car follow a line on the floor while keeping a constant distance of around 20 centimeters to another car driving in front of it (if there is such a car).

Our idea is to make the car follow the left edge of the line on the floor by turning left if it the light sensor is over the line and turning right if the light sensor is outside the line.

The code can be found in the files `race.c` and `race.oil`.