Embedded Software and Systems 2020

SCT interface manual

1. Introduction

This document outlines every variable that can be found in the YAKINDU Statechart Tools interface for the Embedded Software and Systems course.

If you need new variables for the statechart, you can add a new interface. Within this interface you can then add the variables.

2. Annotations

These annotations determine the type of execution of the statechart. @Eventdriven means it only updates when an event happens, or an update is called from code. @ChildFirstExecution means that the children state will be executed before the parent state. There is no need to change these annotations.

```
@EventDriven
@ChildFirstExecution
```

3. BaseValues interface

The first pre-programmed interface to use is the baseValues interface. It contains values that remain static during the execution of the statechart.
The first set of variables is the maximum speed and maximum rotation that the TurtleBot3 can achieve. The second set of variables determine the number of degrees that is used to determine the minimum, maximum and mean around the 0° (Front), -90° (Right), 180° (Back) and 90° (Left) angles. This value should be kept at an even number. For example, if **degreesLeft** is set to 10, then the value **dLeftMean** from the *laserDistance* interface will be calculated by: **dLeftMean** = mean(distance at 85° till the distance at 95°)

```
// Static values
interface baseValues:
  var maxSpeed:real = 0.22
  var maxRotation:real = 2.84

  var degreesFront:integer = 10
  var degreesRight:integer = 10
  var degreesBack:integer = 10
  var degreesLeft:integer = 10
```

4. Output interface

The output interface contains variables that will either be outputted to the TurtleBot3, or used in the main.py file that runs the program.
The speed and rotation variables are outputted to the TurtleBot3 as velocity in the x direction (positive = forward) and rotation around the z-axis (positive = left turn).
The second set of variables are the count output variables, to be displayed upon completion of the program. These are not currently used in the assignment.
The last variable finish, is used to end the program. When this variable is set to anything other than 0 the program will stop.

```
////////////////////////////////////////////////
// These values are output.
// The code sends these either
// to the TurtleBot3, or
// uses it within the code
interface output:
  var speed:real
  var rotation:real

  var obstacles:integer = 0
  var gems:integer = 0

  var finish:integer = 0
```

5. Grid interface

The grid interface handles all interactions with the grid.
The **update** and **receive** variables, can be set to true. When **update** is set to true, information about the walls in the current grid is stored in the data structure. The data that is stored is as follows: **wallFront**, **wallRight**, **wallBack** & **wallLeft** will be stored in the datastructure at the current **row** and **column.** The **orientation** will be used to store all walls in the correct cardinal direction.
column and **row** are the current position of the robot on the grid, which need to be updated in the statechart. **orientation** is the cardinal direction of the front of the robot. Which is a value from 0 to 3, with 0 being north and 3 being west. This also needs to be updated in the statechart.
wallFront to **wallLeft** can contain a value from -1 to 1. -1 means that there is no information about the wall. 0 means there is no wall. 1 means there is a wall.
gridSize is the size in metres of a single cell of the grid. **maxCol** and **maxRow** determine the size of the grid, counting from (0,0). Thus in this example, there is a 4x4 maze with each cell being 0.51 cm wide and long. **maxCol** and **maxRow** also determine the size of the data structure in which the maze will be saved.

```
////////////////////////////////////////////////
// Keeps track of variables
// in the grid
interface grid:
  var update:boolean = false
  var receive:boolean = false

  var column:integer = 0
  var row:integer = 0
  var orientation:integer

  var visited:boolean

  var wallFront:integer
  var wallRight:integer
  var wallBack:integer
  var wallLeft:integer

  var gridSize:real = 0.51
  var maxCol:integer = 3
  var maxRow:integer = 3
```

6. StartPos interface

This interface is used to calibrate the robot. When the robot is put into the calibration position, **setZero** can be set to true to start calibration. This will set **zeroX** and **zeroY** to the current odometry values for x and y.
It will also set **zeroSouthDegree** to the current yaw of the robot. However, because of drift this value can be inaccurate over time.
Lastly, **laserDegOffset** is the offset of the lasers to what laser is orthogonal to the wall. This is because the 0° laser is not pointing directly forward, and the offset counteracts this.

```
////////////////////////////////////////////////
// Calibration variables
interface startPos:
  var setZero:boolean = false
  var zeroX:real
  var zeroY:real
  var zeroSouthDegree:real
  var laserDegOffset:integer
```

7. Computer interface

This interface contains keyboard input. These events are triggered by pressing the associated button + enter.

```
////////////////////////////////////////////////
// Keyboard input in the
// form of events
interface computer:
  in event m_press
  in event w_press
  in event a_press
  in event s_press
  in event d_press
  in event x_press
```

8. Imu interface

This interface contains the variables for the pitch roll and yaw of the TurtleBot3. The variables are on the range [-180:180] degrees.

```
////////////////////////////////////////////////
// The data from the IMU in
// Euler angles.
interface imu:
  var pitch:real
  var roll:real
  var yaw:real
```

9. Odom interface

The odom interface determines the location of the robot. Odometry tracks the location of the robot using the rotation of the wheels and the imu data. x and y determine the location on the 2D ground plane in metres. z will always remain on 0 as the wheels can only track 2D movement.

```
////////////////////////////////////////////////
// The data from the odometry in
// the x, y and z directions.
interface odom:
  var x:real
  var y:real
  var z:real
```

10. LaserDistance interface

This interface contains several abstractions of the laser data. The original laser data contains a list with 360 values indicating distance in metres in all directions. This is simplified to 3 different types of abstractions.

The first set of variables contains the values for 0° (front), 90° (left), 180° (back) and -90° (right) degree angles (**d0, d90, d180, dm90**).

Additionally, there are the minimum and maximum distance of all angles (**dMin**, **dMax**). Next to the value there is also the direction of the laser that measured the minimum and maximum value (**mindDeg**, **maxDeg**). This is on the range [-180:180]. The **dMean** variable shows the mean of all 360°.

The next 4 sets of variables are the minimum, maximum and mean distance in a range around the 0 (**dFront**), 90 (**dLeft**), 180 (**dBack**) and -90 (**dRight**) angles. This range is determined by the **degreesFront**, **degreesLeft**, **degreesBack** and **degreesRight** of the *baseValues* interface. Addittionally it also contains the degree of the minimum and maximum within the range.

The following preprocessing is done before these values are calculated: If no object is within range for a laser, this will give an incorrect value. Thus all these values are filtered out. If that means that a minimum, maximum and mean cannot be calculated, a distance of 4m will be set and the **minDeg** and **maxDeg** will be the centre of the range.

Additionally, all degrees are automatically calibrated to fit with the offset.

```
////////////////////////////////////////////
// Distance data from the
// LDS
interface laserDistance:
  var d0:real
  var d90:real
  var d180:real
  var dm90:real

  var dMin:real
  var minDeg:integer

  var dMax:real
  var maxDeg:integer

  var dMean:real

  var dFrontMin:real
  var minDegF:integer
  var dFrontMax:real
  var maxDegF:integer
  var dFrontMean:real

  var dRightMin:real
  var minDegR:integer
  var dRightMax:real
  var maxDegR:integer
  var dRightMean:real

  var dBackMin:real
  var minDegB:integer
  var dBackMax:real
  var maxDegB:integer
  var dBackMean:real

  var dLeftMin:real
  var minDegL:integer
  var dLeftMax:real
  var maxDegL:integer
  var dLeftMean:real
```

11. LaserIntensity interface

This interface contains two abstractions of the laser data. The laser data also contains a list with 360 values containing intensity in all directions

The first set of variables contains the values for 0° (front), 90° (left), 180° (back) and -90° (right) degree angles (**i0, i90, i180, im90**).

The next 4 sets of variables are the minimum, maximum and mean intensity in a range around the 0 (**iFront**), 90 (**iLeft**), 180 (**iBack**) and -90 (**iRight**) angles. This range is determined by the **degreesFront**, **degreesLeft**, **degreesBack** and **degreesRight** of the **baseValues** interface.

All degrees are automatically calibrated to fit with the offset.

```
////////////////////////////////////////////
// Intensity data from the
// LDS
interface laserIntensity:
  var i0:real
  var i90:real
  var i180:real
  var im90:real

  var iFrontMin:real
  var iFrontMax:real
  var iFrontMean:real

  var iRightMin:real
  var iRightMax:real
  var iRightMean:real

  var iBackMin:real
  var iBackMax:real
  var iBackMean:real

  var iLeftMin:real
  var iLeftMax:real
  var iLeftMean:real
```