Universitá degli Studi di Pisa, Scuola Superiore Sant'Anna

FlexBoardMeetsOBD

Technical Notes

Alessio Balsini, a.balsini@sssup.it

September 15, 2013

1 Repository Source Tree

All the project's source code and documentation related files can be found at *https://github.com/balsini/FlexBoardMeetsOBD*.
The repository has been structured in the following way:

- FlexBoardMeetsOBD: the root;
  - FLEX-OBD-LCD: Flex Demo Board code for the standalone mode (vehicle data is printed on LCD).
  - FLEX-OBD: Flex Light Board code to be used together with the PC GUI.
  - FLEX-SIMULATOR: PC program that simulates the FLEX-OBD, useful for the GUI debugging.
  - GUI: GUI to be used together with FLEX-OBD.
  - PC-OBD: PC program that retrieves vehicle data from Elm327 chip, used to better understand Elm327 and OBDII protocols.
  - doc: documentation folder.
    - developer: documentation useful for developers.
      - Doxygen: doxygen autogenerated documentation, with predefined configuration files.
      - uppaal: finite state machines that simulate systems behaviours.
    - user: user manual.

2 Code Documentation

Code documentation can be generated automatically by Doxygen, by launching:

```
doxygen doc/developer/Doxygen/DESIRED_CODE_FOLDER/Doxyfile
```

Actually on repository are available already the html and latex documentations.
These are contained in the following zip files:

```
doc/developer/Doxygen/DESIRED_CODE_FOLDER/html.zip
doc/developer/Doxygen/DESIRED_CODE_FOLDER/latex.zip
```

Inside the *html.zip* file, it is possible to open the *index.html* file with the preferred web browser.
Inside the *latex.zip* file, it is possible to open the *refman.pdf* file with the preferred pdf document reader.

3 State Machines

3.1 Common Module

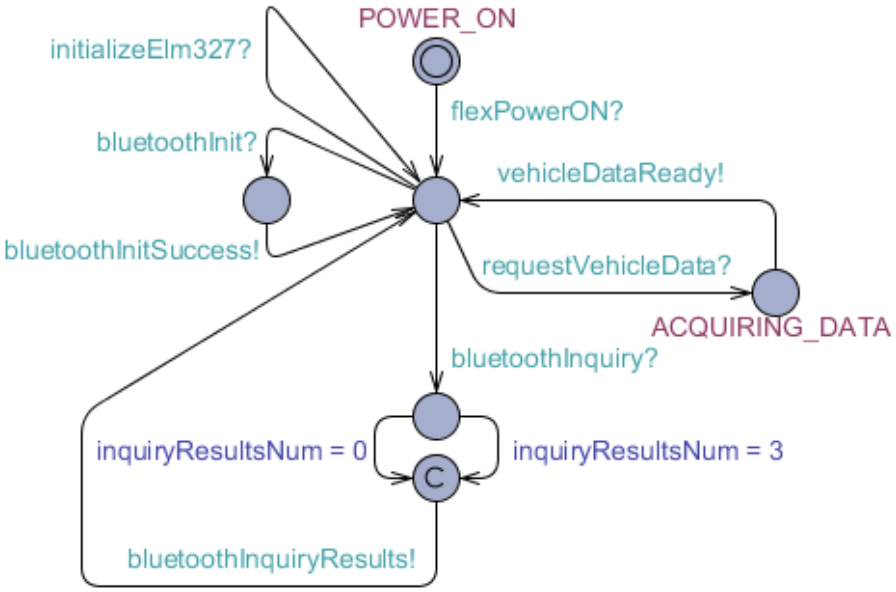

Figure 3.1: *Bluetooth*

As the picture shows, the Bluetooth module wakes up and simply replies to the requests. In case of inquiry request, nondeterministically returns 0 or 3, simulating the cases in which 0 or 3 devices have been discovered during the inquiry.

3.2 FLEX-OBD-LCD (STANDALONE)

In this case, the active parts are:

- user: human who pushes buttons and reads the LCD;
- bluetooth: Bluetooth module, which also abstracts the Elm327 and the vehicle;
- scheduler: tasks manager;
- taskMain: task which initializes all the environment and, in the end, activates the other tasks;
- taskReceive: task which retrieves vehicle data;
- taskUpdateLCD: task which updates the LCD with the data proveded by the taskReceive task.

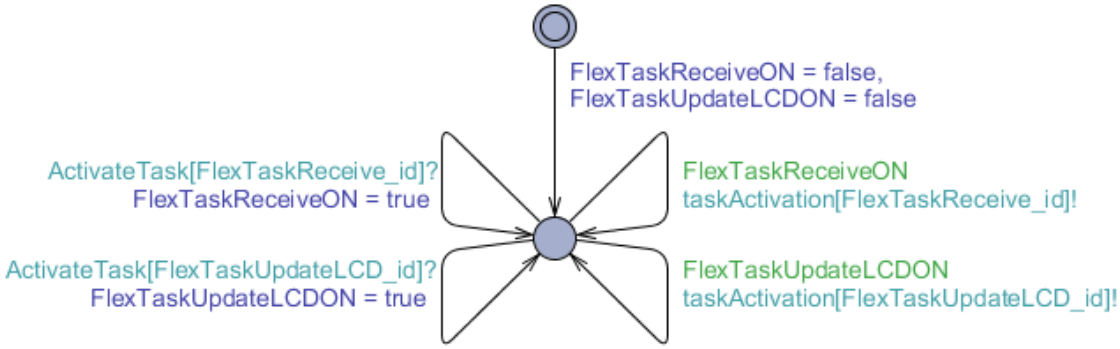

Figure 3.2: *Scheduler*

The scheduler is not the real scheduler. It simply provides a minimal concurrency management between taskUpdateLCD and taskReceive tasks after their activations.
These tasks are activated by the taskMain.

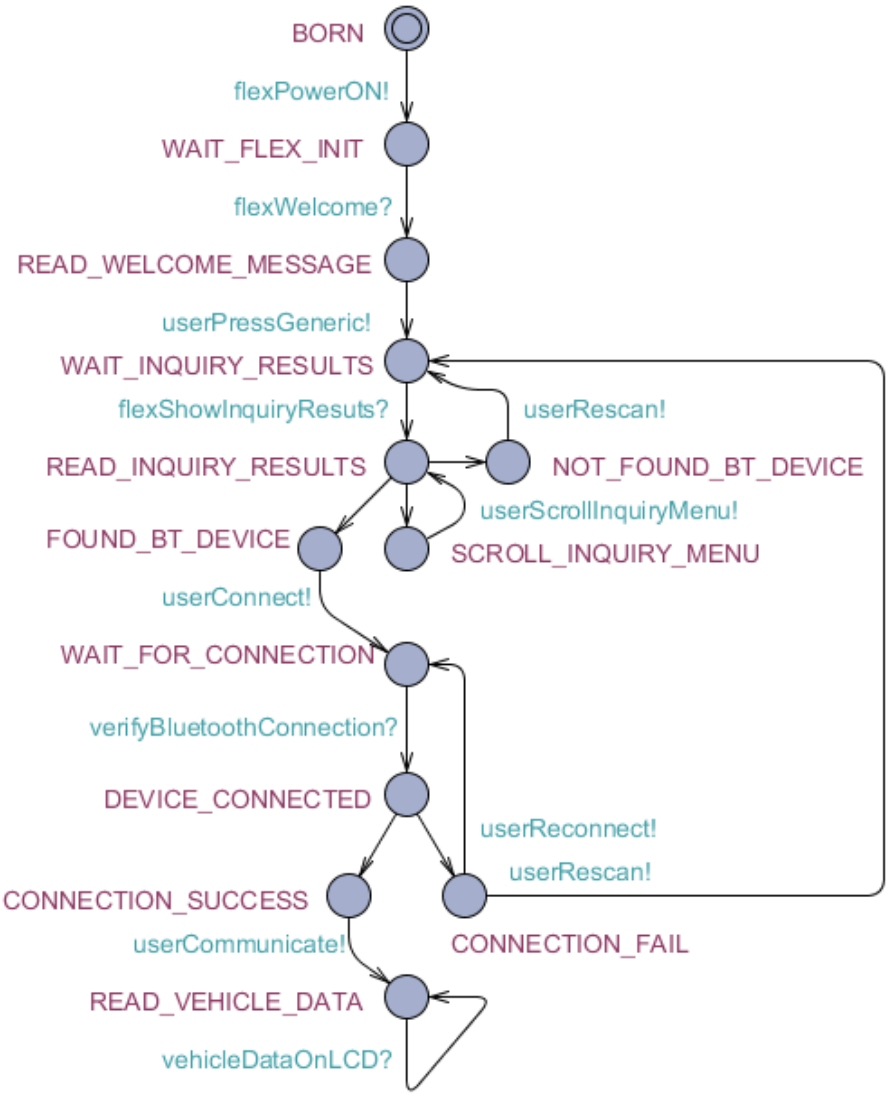

Figure 3.3: *User*

User's first action is to power-up the Flex.
After the Flex initialization, the user has to push a button.
After the Flex returned the inquiry results, the user can scroll the inquiry menu, request another inquiry scan or connect to a specific device.
If the user requested the connection, he must also check if the connection has been correctly established (on the Bluetooth module the red led should stop blinking and green led should be turned on).
If the connection failed, then the user can perform another inquiry or retry the connection.
If the connection succeded, then it is possible to start the vehicle communication and the user will be able to read the vehicle data on the LCD screen.

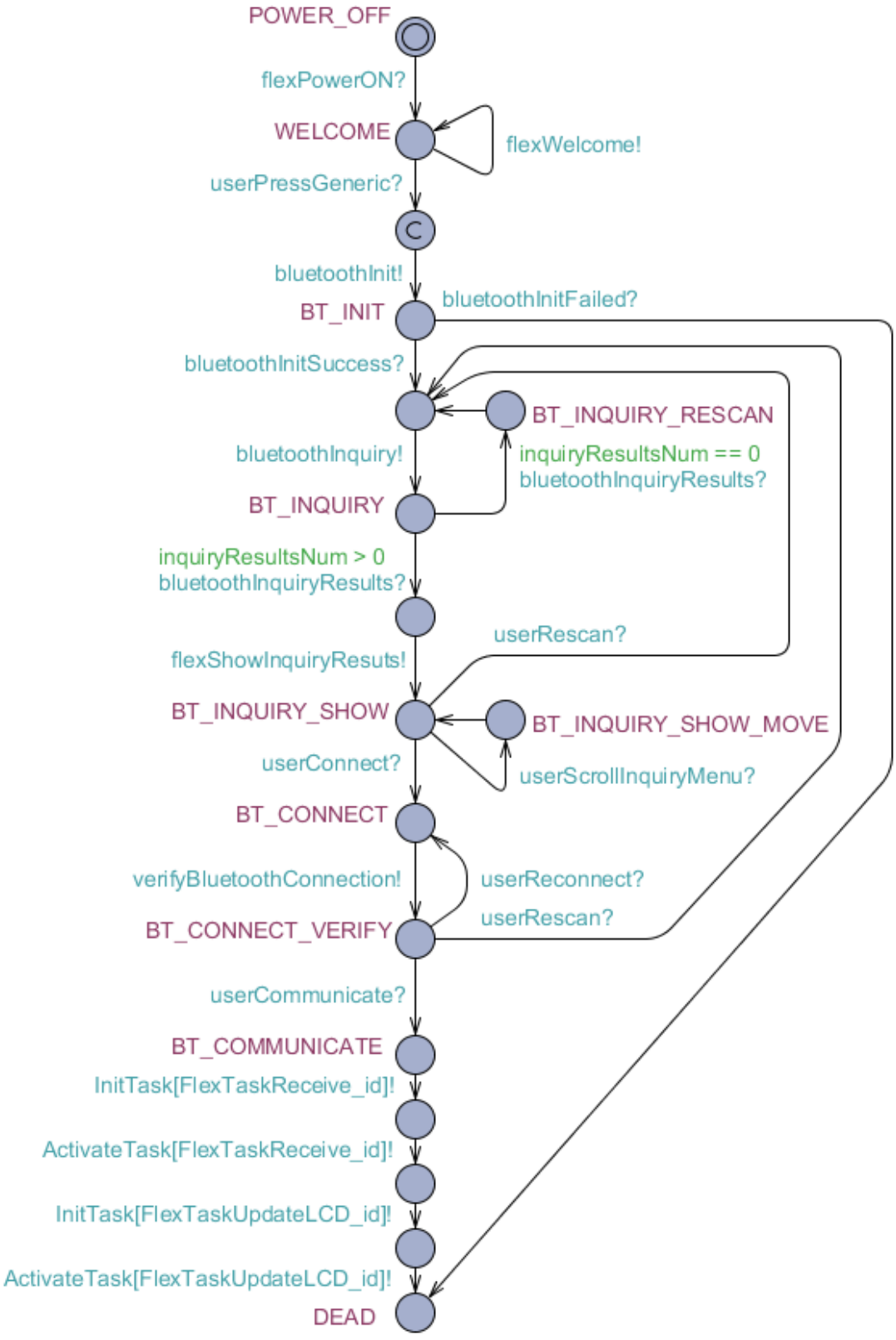

Figure 3.4: *taskMain*

The taskMain is the first task to be executed.
First of all, it shows a welcome message on LCD and waits an human response.
After the human response, the Bluetooth module is configured. If the configuration goes wrong, then the task dies and the whole system becomes idle. Otherwise, an inquiry scan is performed.
Inquiry goes on until at least one Bluetooth device is found, then the scan results are shown to the user through a dynamic menu.
User can navigate the menu and decide if perform another inquiry or connect to one of the devices.
After the connection attempt, is requested to the user what to do next, in case of connection success or failure: perform another inquiry, retry to connect, or start the vehicle data transfert.
If the data transfert began, then the other two tasks are initialized and activated.

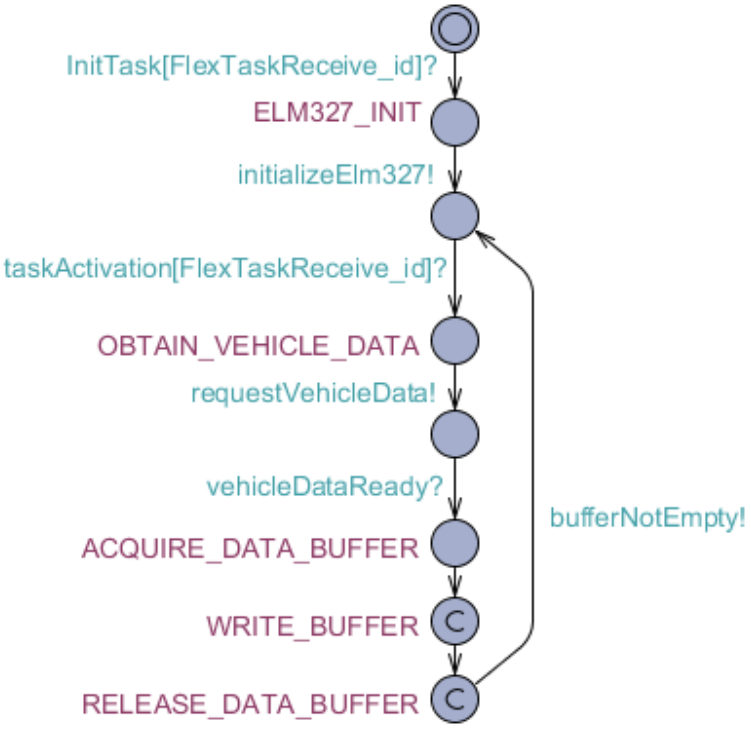

Figure 3.5: *taskReceive*

The taskReceive is waken up by the taskMain.
It's first operation is to initialize the Elm327, then, periodically, it performs the operations described next.
It request the vehicle new data.
It locks the mutex associated to the shared buffer which contains the vehicle data.
It fills the buffer with new data.

It unlocks the mutex.
It notifies the taskUpdateLCD that new data is ready.
It dies, waiting for next activation to be performed by the scheduler.

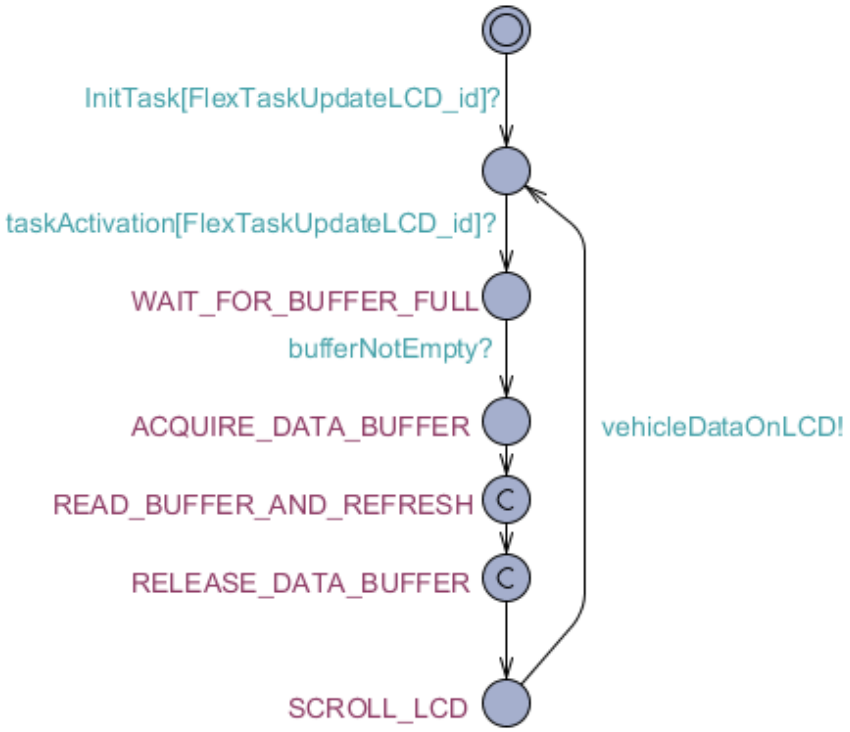

Figure 3.6: *taskUpdateLCD*

The taskUpdateLCD periodically performs the operations described next.
It waits until the vehicle data buffer contains new data.
It locks the mutex associated with the shared vehicle data buffer.
It locally copies data from the shared buffer.
It releases the mutex.
It checks if the user requested a scroll and shows the data on LCD.
It dies, waiting for next activation to be performed by the scheduler.

3.3 FLEX-OBD AND PC GUI

In this other case, the active parts are:

- PC: system that comprehends the human and the PC program which interacts with the Flex Light;
- bluetooth: Bluetooth module, which also abstracts the Elm327 and the vehicle;

- scheduler: tasks manager;
- taskMain: task which initializes all the environment and, in the end, activates the other tasks;
- taskGetData: task which retrieves vehicle data;
- taskSendData: task which forwards to the PC the data provided by the taskGetData task.

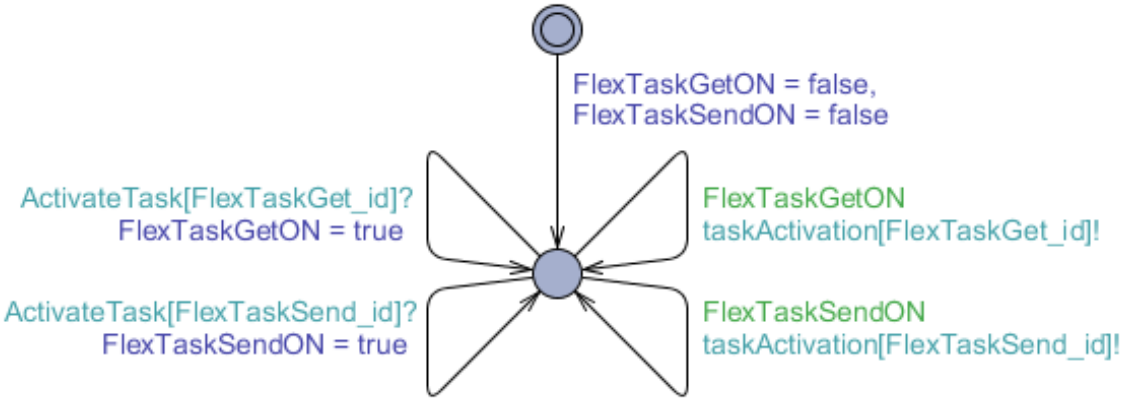

Figure 3.7: *Scheduler*

The scheduler is not the real scheduler. It simply provides a minimal concurrency management between taskGetData and taskSendData tasks after their activations.
These tasks are activated by the taskMain.

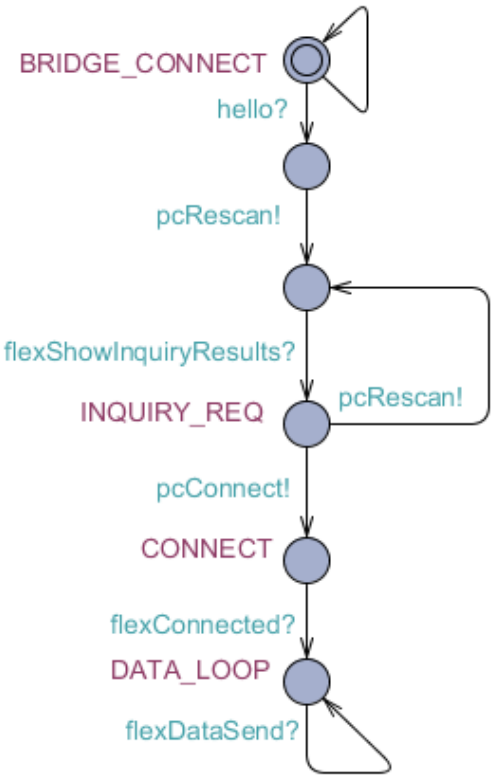

Figure 3.8: *PC*

The PC loops until the user requests to ping the Flex.
After the ping, the PC has to request an inquiry scan to the Flex.
Once the inquiry results have been received, the PC can reject them, requesting another inquiry, or accept one of them, asking the Flex to establish the connection.
After the Flex confirms that the connection has been succesfully performed, then the PC enters in a loop, receiving and showing vehicle data provided by the Flex.

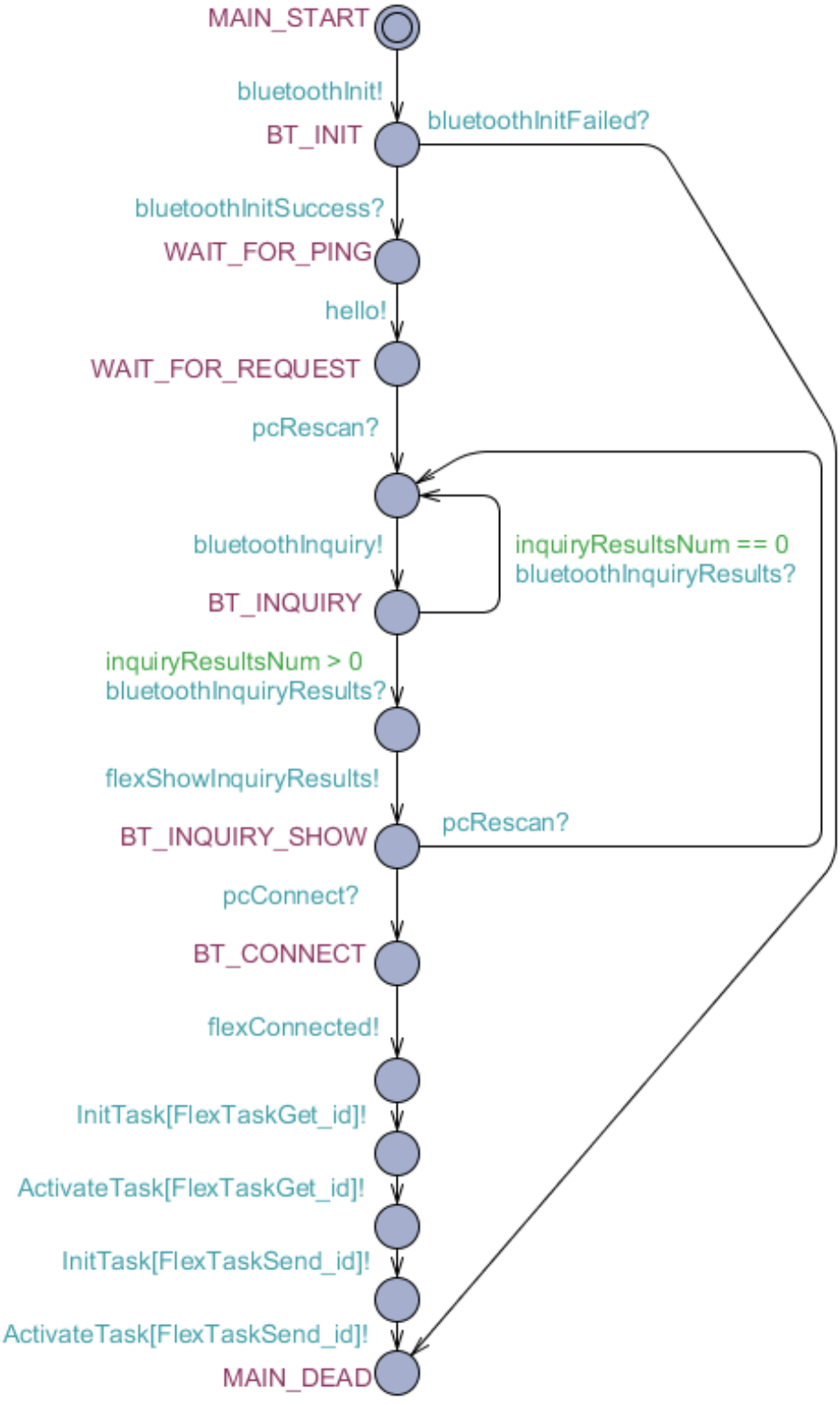

Figure 3.9: *taskMain*

This task automatically performs the following operations.
It initializes Bluetooth module. If initializzation fails, then this task dies, causing the whole system to become idle. Otherwise, it can wait for the PC ping.
After the PC ping, it waits for the PC to request an inquiry, so it cyclically performs inquiry scans until at least one Bluetooth device is found.
The inquiry results are sent to the PC, which can request another inquiry or can ask to connect to one of the devices discovered by the last scan.
After the connection, this task initializes and activates the taskGetData and taskSendData tasks and dies.

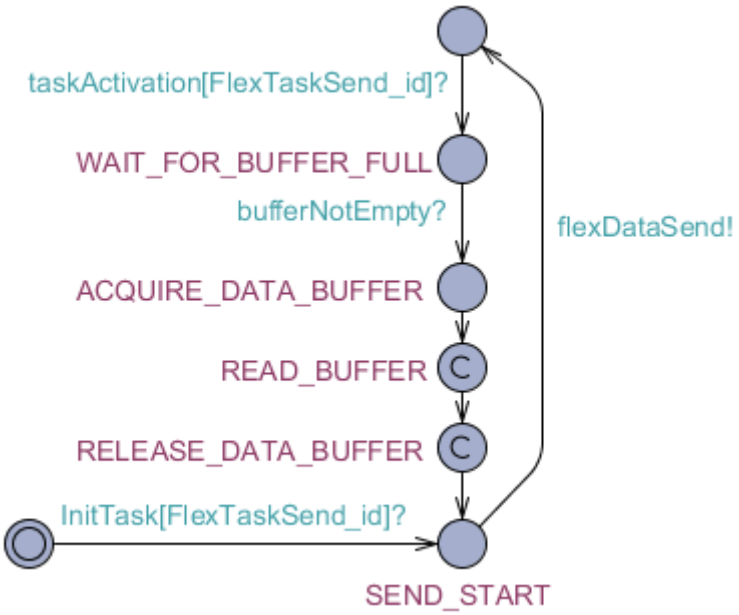

Figure 3.10: *taskSendData*

This task periodically performs the operations described next.
It waits for new vehicle data.
It acquires the mutex associated to the shared vehicle data buffer.
It puts the vehicle data buffer contents to local variables.
It releases the mutex.
It sends the new vehicle data to the PC.
It dies, waiting for next activation.

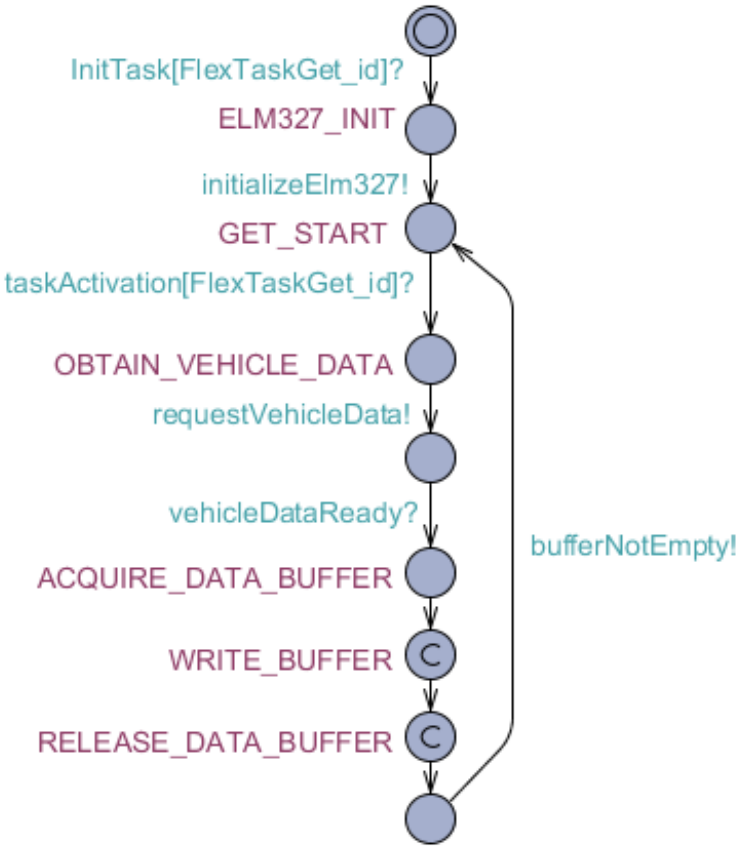

Figure 3.11: *taskGetData*

First of all this task initializes the Elm327.
After that, periodically performs the operations described next.
It queries the vehicle to receive new data and locally stores this data.
It locks the mutex associated to the shared vehicle data buffer.
It fills the shared vehicle data buffer with the new data.
It releases the mutex.
It notifies the taskReceive that new data is available and dies, waiting for next activation.