# EMBSYS 110 Assignment 5

Traffic Light Remake

## 1 Goal

In classes students learned about how a system can be partitioned into components, and how different components communicate with each other via well-defined event interfaces. In this exercise students put what they learned into practice by hooking up multiple components to form a functional system.

## 2 Readings

1. Finite State Machines for Integration and Control in ALICE. Giacinto De Cataldo, etc. Proceedings of ICALEPCS07, Knoxville, Tennessee, USA. 2007. (https://accelconf.web.cern.ch/accelconf/ica07/PAPERS/RPPB21.PDF)

   This paper illustrates the use of a hierarchy of finite state machines in ALICE (A Large Ion Collider Experiment) on the Large Hadron Collider (LHC) in CERN. This is one of the many examples of the application of formal state machines in critical systems. Note that in this system each state machine itself is not hierarchical.

2. Use of statecharts in the modelling of dynamic behaviour in the ATLAS DAQ prototype-1. P. Croll', P.-Y. Duval', R. Jones3, S. K010s4, R.F. Sari' and S. Wheeler. IEEE Transactions on Nuclear Science, Vol. 45, No. 4, August 1998.

   (https://www.researchgate.net/publication/3136235_Use_of_statecharts_in_the_modelling_of_dynamic_behaviour_in_the_ATLAS_DAQ_prototype-1)

   This earlier paper documents the evaluation of employing statecharts and CHSM tools in ALICE on LHC. Searching for CHSM on Github still yields active projects.

## 3 Setup

1. *Carefully* plug in the following extension modules in the exact order onto your Nucleo board:

   (a) X-NUCLEO-IKS01A1 (IMU sensor).

   (b) X-NUCLEO-IDW04A1 (WiFi).

   (c) LCD module.

   Please ensure you align the pins correctly and do not bend any pins.

In this assignment we are only using the LCD module. Plugging in other modules now avoids the need to unplug the LCD module later and gives us more space to press the USER (blue) button.

2. Download the compressed project file (platform-stm32f401-nucleo_assignment5.tgz) from the Assignment 4 course site.

   Place the downloaded tgz file in your VM under **~/Projects/stm32**.

3. Move your existing project folder to a backup folder, e.g.

   ```
   mv platform-stm32f401-nucleo platform-stm32f401-nucleo.bak
   ```

   Note – You may want to pick another backup folder name if the one shown above already exists.

4. Decompress the tgz file with:

   ```
   tar xvzf platform-stm32f401-nucleo_assignment5.tgz
   ```

   The project folder will be expanded to **~/Projects/stm32/platform-stm32f401-nucleo**.

5. Launch Eclipse. Hit F5 to refresh the project content.

   Or you can right-click on the project in Project Explorer and then click "Refresh".

6. Clean and rebuild the project. Download it to the board and make sure it runs.

   In minicom, you should see log statements printed out by the USER_BTN component when you press the USER button. They look like these:

   ```
   38759 USER_BTN(23): Inactive TRIGGER from UNDEF(0) seq=30
   38759 USER_BTN(23): Inactive EXIT
   38760 USER_BTN(23): Active ENTRY
   38760 USER_BTN(23): PulseWait ENTRY
   ```

   The LCD shows a white background which is normal.

7. If you have not installed UMLet, you will need to do so for this assignment. Please refer to the Setup notes in the previous assignment.

# 4 Tasks

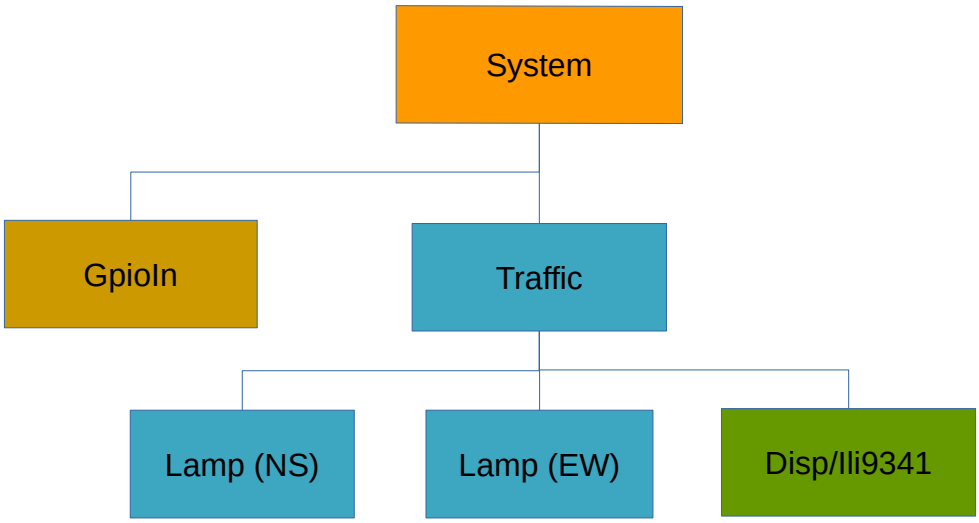
Our demo system involves the following components:

- System (SYSTEM) - System manager coordinating other components.
- GpioIn (USER_BTN) - GPIO input connected to the USER button.
- Traffic (TRAFFIC) - Traffic light controller with two Lamp (LAMP_NS and LAMP_EW)

orthogonal regions.

- Disp/Ili9431 (ILI9341) - Display controller.

The following diagram shows the control hierarchy of these components.




Tasks for this assignment are:

1. In **src/System/System.cpp** *Started* state, handle GPIO_IN_PULSE_IND and GPIO_IN_HOLD_IND by sending the TRAFFIC_CAR_NS_REQ and TRAFFIC_CAR_EW_REQ respectively.

   This simulates a car arriving along the NS direction with a short press (<1s) on the USER button. It simulates a car arriving along the EW direction with a hold (>1s) on the USER button. If you keep holding the button, it simulates cars keep arriving along the EW direction.

   In this example, the System acts as a coordinator between the GpioIn component and the Traffic component

   Hint: You can send an event like the following example:

   ```
   Evt *evt = new TrafficCarEWReq(TRAFFIC, GET_HSMN());
   Fw::Post(evt);
   ```

2. In **src/Traffic/Lamp/Lamp.cpp**, call the Draw() member function at appropriate places to draw traffic lights on the display module. See the following function for details of the implementation:

   ```
   void Lamp::Draw(Hsmn hsmn, bool redOn, bool yellowOn, bool greenOn)
   ```

3. Test your code by pressing and holding the USER button. Ensure the traffic lights on the LCD displays show the expected patterns. Use log messages on minicom to debug and verify your application.

   Note – You can control log output with the "**log on/off**" command. You can check component states and component numbers (HSMN) with the "**state**" and "**hsm**" command.

# 5 Submission

The due date is 5/13 Monday 11:59pm. Please submit:

1. A zip file containing the source code in **src/System and src/Traffic.**
2. One photo showing the green light turned on along the NS direction.
3. One photo showing the green light turned on along the EW direction.

Upload to the usual Canvas assignment upload location.