Project

Real-Time Computer Systems [ELEC-H-410]

Design a real-time distributed alarm

Corrigé

2013–2014

1 Purpose

On the basis of the previous labs you will now create a distributed alarm system working on the CAN network.

2 Useful documents are stored on the network share:

`labo\ELEC-H-410\Useful Documents\`

- `dsPIC30F-33F Programmer's Reference.pdf`
- `dsPIC33 Data Sheet.pdf section 21`
- `Introduction to MPLAB.pdf`
- `Explorer 16 User Guide 51589a.pdf`
- `MPLAB C30 C Compiler User's guide.pdf`
- `uCOSII_RefMan.pdf`
- `Enhanced Controller Area Network.pdf`
- `Introduction to language C for microcontrollers.pdf`
- `Table of ASCII codes.`

3 Specifications

In this distributed alarm system, all nodes are equal (i.e. there is not any master node), can work autonomously but also exchange information through CAN. The application of the node should run under µC/OS-II.

Each node of the alarm system will have to carry the following functionalities out:

- Monitor the presence of intruders. The sensor will be a push button connected to a digital input of the microcontroller.
- Arm/disarm the system thanks to a secret code typed on the keyboard. For the moment, you can use the value B169. Functions to change this code are optional.
- If an intruder is detected while the system is armed, leave the possibility to disarm the system using the same code during the first 30s.
- If the system has not been disarmed within 30s, the alarm should start. You will get a buzzer or a flashing light that you can connect to an output port.

In a distributed system, each node must cooperate with the other members of the network:

- A node signals its presence when it appears on the network and repeat this message periodically at least every 5 seconds. This "heartbeat" enables to detect the disappearance of the node (failure or sabotage).
- If a node detects an intrusion, it must signal it so that all other nodes pass in the "alert" state, even if they haven't detected anybody.
- (Dis)arming a node should be propagated to all other nodes; there is no obligation to disarm at the node which has detected the intrusion.
- Think about any way to attack your alarm and try to reduce possibilities succeed in doing that.

To facilitate the management of the communication, we propose that you use the following protocol:

- `Message_ID` will be fixed during the first brainstorm and is linked to the priority of the messages on the network. Nodes must specify their `Node_ID` in the data.
- The alarm system is limited to 10 nodes whose `Node_ID` is unknown a priori.
- Then a node boots, it has to find a free identifier, then signal its presence by broadcasting it regularly.

Message type	**Message_ID**	**Data**
Heartbeat	0	node ID
Intrusion detected	1	node ID
Node has been armed	5	node ID
Node has been disarmed	4	node ID
Alarm started	6	node ID

Table 1: Table of messages

4 Operating mode

4.1 Timetable

Five sessions are dedicated to the project:

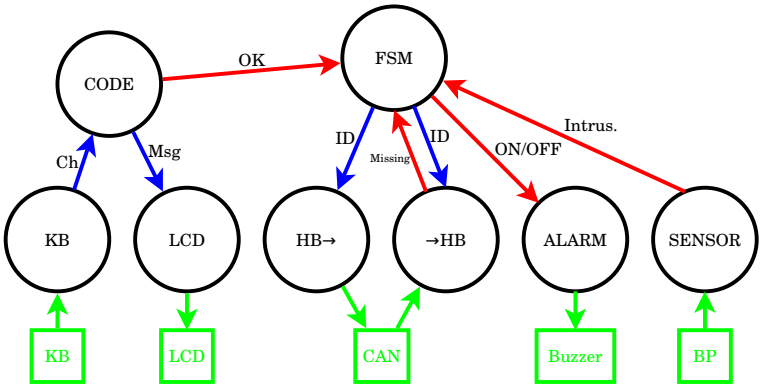
- During the *first project session*, you have to brainstorm about the specifications of a distributed alarm. Don't hesitate to use timing diagrams, block diagrams, state machines or other operating/functional schemes to share your ideas. You must **NOT** code anything during this session.
- During the *next four sessions*, you will use your knowledge and the result of your brainstorming to implement your node of the distributed alarm system.
- The *last session* will be split in two times two hours. The first two hours will allow your group to check the effective functioning of your alarm. Your project will be evaluated during the last two hours.

4.2 Evaluation

The project will we evaluated on the basis of a short presentation of your work and your answers to questions.
P. Mathys, Y. Allard and R. Sommeillier will spend several minutes in each group during the last two hours of the last session to check the effective functioning of your alarm, to assess the quality of your code and to ask you a few questions.

Good luck!

5 Task splitting



5.1 FSM task: Finite State Machine detail

5.2 HB→

Heartbeat generation

Input: mailbox from FSM with ID

Output: Heartbeat message on the CAN

Behaviour: Periodic message with ID=0, each 5s (max)

5.3 →HB

Heartbeat processing

Input: HB messages from the CAN

Input: Mailbox from FSM to add ID in the list

Output: Semaphore to FSM set when ID goes missing on the CAN

Behaviour: Maintains a list with Node ID seen on the CAN. Each message reset a counter for the corresponding NodeID. Periodically, the counter in decremented and when it reaches 0, the semaphore to FSM is set.

All messages can be processed as HB as the Node ID is the first data byte.

5.4 CODE

Code input and verification

Input: mailbox from KB

Output: Semaphore to FSM set when valid code

Output: Mailbox: message to display on the LCD, "*" when char typed, "Code OK", "Code wrong"...

Behaviour: Based on the char received from KB, send strings to print (stars and/or messages) to the LCD task and set a semaphore to FSM is the code is correct. Reset any partial input after some time.

5.5 ALARM

Ring the Alarm

Input: semaphore from FSM

Output: sound on the DAC

Behaviour: When the semaphore is set, play a sound, could use a global variable instead.

5.6 LCD

Displays string received from CODE task using a mailbox.

Input: mailbox from CODE

Output: message on the LCD

Behaviour: Reads the content of the mailbox and displays it on the LCD screen.

5.7 SENSOR

Detect an intrusion

Input: a push button on the explorer 16 extension board will be used.

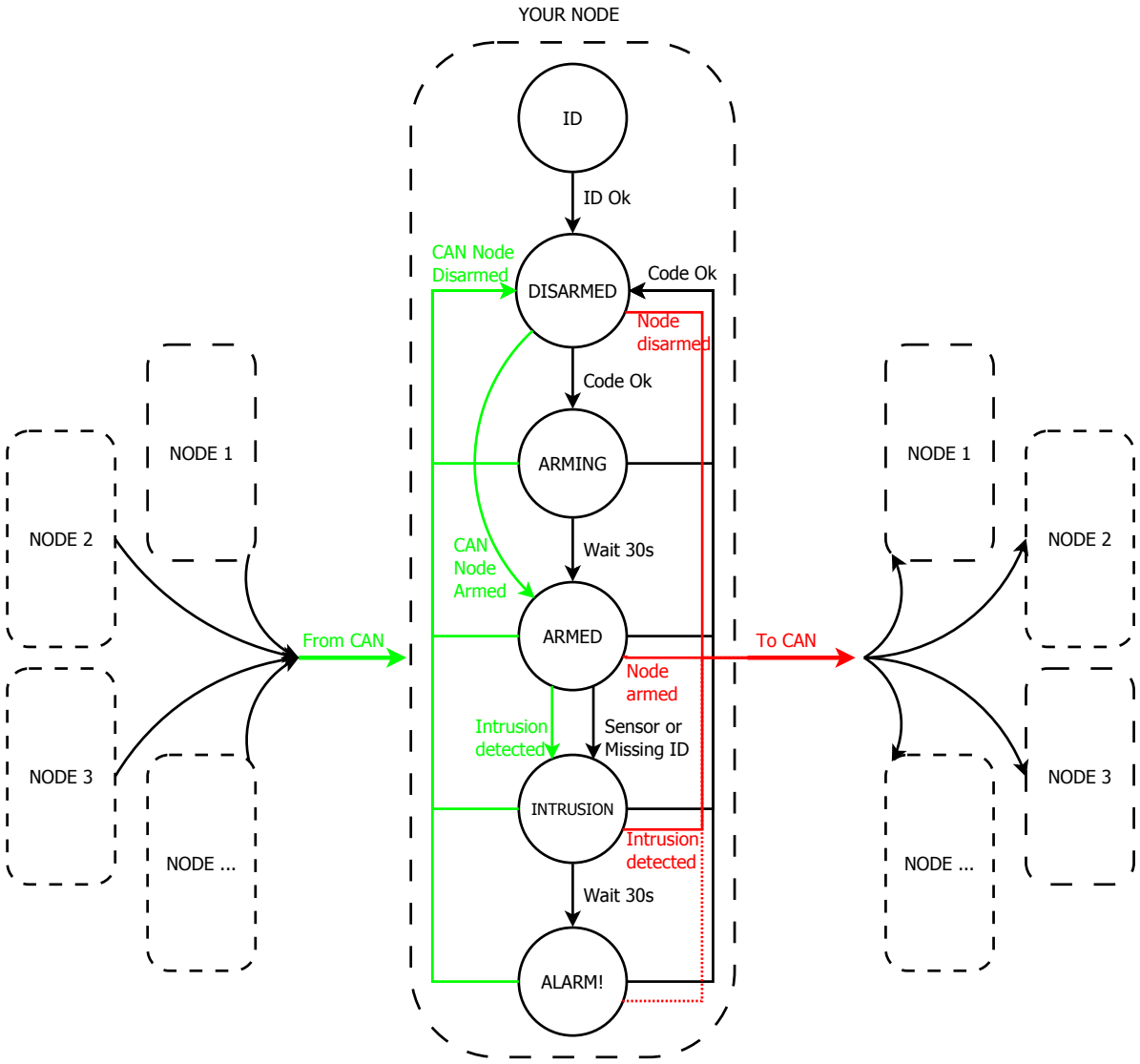
Output: set a semaphore to FSM when an intruder is detected

Behaviour: this task sets the semaphore when the button is pushed.

6 Hardware constraints

The CAN has several users and so must be protected.

7 Finite State Machine



The following diagram represents the finite state machine of your node and its communication with the network.

7.1 States of a node

7.1.1 ID

Each node of the alarm system must have a CAN_ID. But this identifier is a priori unknown. Thus, when a node boots, it has firstly to find a free identifier and then signals its presence by broadcasting it regularly (heartbeat message).

This state "<ID"> is so the state in which is a node just after booting, i.e. the state of researching a free identifier.
Once this ID is found, the node move from this first state to the following one (DISARMED).

7.1.2 DISARMED

This state is the one in which the node must be logically as often as possible. The tasks that must be runned into the DISARMED state are to listen to all messages from the CAN and to signal its

presence and its state to the other nodes.

If the correct arming code is typed on the keyboard on this node, it enters in the ARMING state. And if the node receives via the CAN a message stating ”A node has been armed”, as all the other nodes, its enter in the ARMED state.

When the correct disarming code is type on the keyboard or when the node receives via the CAN, as all the other nodes, a message ”A node has been disarmed”, the node enters in DISARMED state; whatever was the previous state.

7.1.3 ARMING

This state is conceptually a delay of 30s. In practice, it allows the owner of the node (i.e. the house for example) to leave without being detected by the sensor.
As previously explained, as for the three next states, if the correct disarming code is pressed on the keyboard or if a disarmed CAN message is received, the node return to the DISARMED state.
This state is the only one clearly independant from the network. As a consequence, it is unuseful to communicate the transition towards this state of your node to the other nodes of the network.

7.1.4 ARMED

After waiting 30s, the node enters in the ARMED state and signals this transition to the other nodes.
From the ARMED state, the node can pass in INTRUSION state if:

- The sensor of the node detects someone. In this case, the node must transmit a ”detection message” to the CAN;
- If an CAN_ID from the other nodes that was viewed regularly before via heartbeats messages seems missing (no news during 15 seconds);
- If the node received a ”detection message” from the CAN. (???????????????? réfléchir là-dessus)

7.1.5 INTRUSION

This state is conceptually a delay of 30s. In practice, it allows the owner of the node (i.e. the house for example) to enter, access to the keyboard and type the correct disarming code without triggering the alarm.

7.1.6 ALARM!

When entering in this state

- receives informations from the other nodes connected to the network. These received messages should be:
  - heartbeat
- transmits informations to the other nodes connected to the network