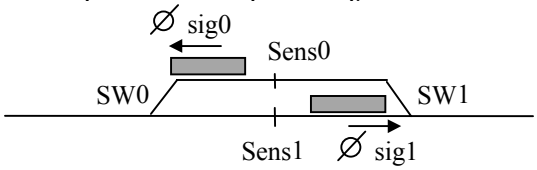
Μεταπτυχιακό Μάθημα ΟΣΥΛ
"Λειτουργικά Συστήματα Πραγματικού Χρόνου"
Μ.Στεφανιδάκης

Εργαστηριακή Ασκηση #1

Σκοπός της παρούσας άσκησης είναι η εξοικείωση με διάφορες τεχνικές συγχρονισμού διεργασιών.

Εισαγωγή: Η εφαρμογή σας θα χρησιμοποιήσει τη βιβλιοθήκη LXRT του interface πραγματικού χρόνου RTAI. Η βιβλιοθήκη αυτή επιτρέπει την ανάπτυξη εφαρμογών πραγματικού χρόνου σε user space. Για την δημιουργία της εφαρμογής σας θα συνδυάσετε τα εξής αρχεία:
α) βιβλιοθήκες RTAI: **rtai_hal rtai_lxrt rtai_sem rtai_msg rtai_mbx** (τα modules αυτά θα πρέπει να υπάρχουν στη μνήμη του συστήματος).
β) εφαρμογή **mainsim.c emu.c sim.h param.h task.c-edu1 task.c-edu3 task.c-edu4 doit** (αντιγράψτε τα αρχεία αυτά από το directory sim)
Ο κώδικας που θα τροποποιήσετε βρίσκεται στα αρχεία **task.c-edu?** όπου θα βρείτε λεπτομερείς οδηγίες για τα ερωτήματα που ακολουθούν.

Μέρος Α: (emulation setup 1)
Αντιγράψτε το αρχείο **task.c-edu1** στο **task.c**. Το 1^{ο} περιβάλλον εξομοίωσης αναπαριστά μία σιδηροδρομική γραμμή με διακλάδωση (σχ.1), όπου κινούνται δύο συρμοί. Τα άκρα της γραμμής ενώνονται μεταξύ τους. Οι συρμοί κινούνται με σταθερή ταχύτητα σε αντίθετες κατευθύνσεις. Σε κάθε ένα από τα δύο παράλληλα τμήματα υπάρχουν αισθητήρες και σήματα τα οποία ρυθμίζουν την κυκλοφορία των συρμών. (->) γραμμή ελεύθερη, (!>) γραμμή κλειστή. Η κίνηση των συρμών καθορίζεται αυτόματα από την κατάσταση των σημάτων.



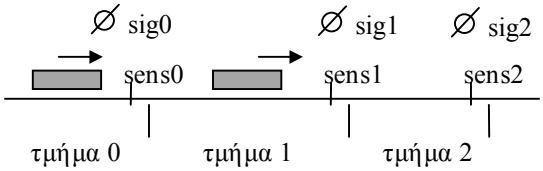
Σχ.1

Τα σήματα ελέγχονται με την κλήση της συνάρτησης turnsignal(id, STOP/PASS) και τα κλειδιά με τη συνάρτηση turnswitch(id, STRAIGHT/CURVE). Το ζητούμενο είναι η συγγραφή προγράμματος ελέγχου για την εκ περιτροπής χρήση της γραμμής, δηλαδή μόλις ένας συρμός φτάνει στον κόμβο, να ξεκινά ο άλλος.

Διαδικασία:
Στο αρχείο task.c περιλαμβάνεται μία περιοδική διεργασία, η οποία ελέγχει τους αισθητήρες sens1 και sens2. Το σήμα των αισθητήρων παρουσιάζεται στα bits 0,1 της σφαιρικής μεταβλητής **iport**. Το αρχείο περιλαμβάνει επίσης το πλαίσιο μίας διεργασίας ελέγχου, όπου θα τοποθετήσετε τον κώδικά σας. Μελετήστε τη δομή του προγράμματος. Εκτελέστε δοκιμαστικά το πρόγραμμα (**./sim**).
Υπόδειξη: Όταν ενεργοποιείται ένας αισθητήρας, ο συρμός που τον ενεργοποίησε πρέπει να σταματήσει και να επιτραπεί η κίνηση του άλλου που περιμένει. Αυτό συμβαίνει εναλλάξ για τους δύο συρμούς. Οδηγήστε κατάλληλα τα ζεύγη κλειδιών/σημάτων. Δεν απαιτείται η χρήση πρόσθετων σημαφόρων ή mailboxes.

Αρχικά τα δύο σήματα είναι κλειστά (STOP), το κλειδί 0 οδηγεί ευθεία (STRAIGHT), ενώ το κλειδί 1 προς τα πάνω (CURVE). Ο συρμός που κινείται προς τα αριστερά είναι ήδη μέσα στη διακλάδωση και έχει σταματήσει. Με SPACE το πρόγραμμα τερματίζεται.

Μέρος Β: (emulation setup 3)
Το περιβάλλον εξομοίωσης αναπαριστά μία σιδηροδρομική γραμμή (σχ.2), όπου κινούνται δύο συρμοί. Τα άκρα της γραμμής ενώνονται μεταξύ τους. Οι συρμοί κινείται συνεχώς με διαφορετική ταχύτητα, προς την ίδια κατεύθυνση. Η γραμμή έχει χωριστεί σε 3 τμήματα, και 3 σήματα με τους αντίστοιχους αισθητήρες χρησιμοποιούνται για τον έλεγχο της κίνησης των συρμών.



Σχ.2

Σε κάθε τμήμα επιτρέπεται να υπάρχει μόνο ένας συρμός. Το ζητούμενο είναι το αντίστοιχο πρόγραμμα ελέγχου.

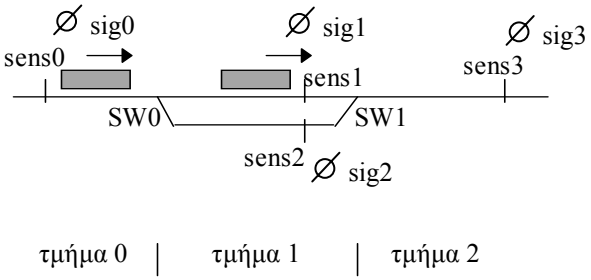
Διαδικασία:
Αφήστε τους συρμούς να κινηθούν χωρίς έλεγχο. Τι παρατηρείτε;
Αναθέστε 3 σημαφόρους σε κάθε τμήμα γραμμής. Χρησιμοποιήστε τη διαδικασία ανίχνευσης αισθητήρων της προηγούμενης άσκησης. Οι αισθητήρες βρίσκονται προς την έξοδο κάθε τμήματος.
Για κάθε ενεργοποίηση αισθητήρα i ξεκινήστε την εξής διαδικασία (**blocker()**):
α) κλείστε το σήμα προς το επόμενο τμήμα
β) Προσπαθήστε να πάρετε τη σημαφόρο του επόμενου τμήματος (i+1, μεταβλητή **nid**)
γ) Οταν το καταφέρετε, ανοίξτε το σήμα προς το επόμενο τμήμα, ελευθερώνοντας ταυτόχρονα τη σημαφόρο του τμήματος i (μεταβλητή **id**).
Προσοχή! Επειδή οι συρμοί βρίσκονται ήδη στα τμήματα 0 και 1, αρχικοποιήστε κατάλληλα τις σημαφόρους.

Σημαφόροι στο Linux-RTAI.
1.Δηλώστε ένα handle σημαφόρου:
SEM *semptr;
2. Δημιουργήστε τη σημαφόρο:
sem = rt_sem_init(id,value); όπου αν value=1, η σημαφόρος είναι ελεύθερη, ενώ αν value=0, η σημαφόρος είναι κατειλημμένη.
3. Ζητήστε τη σημαφόρο: **rt_sem_wait(semptr);**
4. Ελευθερώστε τη σημαφόρο: **rt_sem_signal(semptr);**
5. Στο τέλος διαγράψτε τη σημαφόρο: **rt_sem_delete(semptr);**

Επίσης: όταν τερματίζεται το πρόγραμμα θα πρέπει να φροντίσετε να τερματίσουν και όσες διεργασίες περιμένουν σε σημαφόρους κλπ. Στην scansensors(), στο σημείο εξόδου, απελευθερώστε όλες τις σημαφόρους που χρησιμοποιείτε.

Μέρος Γ: (emulation setup 4)

Στο προηγούμενο σχήμα έχει προστεθεί και μία παράπλευρη γραμμή (σχ.3), η οποία επιτρέπει την προσπέραση του αργού συρμού από τον ταχύτερο. Ζητείται το πρόγραμμα ελέγχου της κίνησης των συρμών.



Σχ.3

Διαδικασία:

Χρησιμοποιείστε την ανίχνευση των αισθητήρων, όπως προηγουμένως. Τα δύο ακραία τμήματα της γραμμής ελέγχονται από 2 σημαφόρους. Για το μεσαίο διπλό τμήμα χρησιμοποιήστε μια ουρά μηνυμάτων (mailbox). Η ουρά αυτή έχει δύο θέσεις και περιέχει μηνύματα με την τιμή 0 και 1. Όταν ένας συρμός θέλει να προσπελάσει το μεσαίο τμήμα, διαβάζει από την ουρά μηνυμάτων έναν αριθμό γραμμής που θα χρησιμοποιήσει. Μόλις ελευθερώσει το τμήμα, επιστρέφει τον αριθμό στην ουρά.

Mailboxes στο Linux-RTAI
1.Δηλώστε ένα mailbox handle
MBX *mbxptr;
3. Δημιουργήστε το mailbox:
mbxptr = rt_mbx_init(id,MBX_SIZE);
4. Διαβάστε από το mailbox:
rt_mbx_receive(mbxptr,&msg,MSG_SIZE));
5. Στείλτε μήνυμα στο mailbox:
rt_mbx_send(mbxptr,&msg,MSG_SIZE);
6. Διαγράψτε το mailbox:
rt_mbx_delete(mbxptr);

Τα υπόλοιπα τμήματα γραμμής ελέγχονται με σημαφόρους όπως στην προηγούμενη φάση. Θυμηθείτε να αρχικοποιήσετε κατάλληλα τις σημαφόρους. Επίσης τι πρέπει να περιέχει αρχικά η ουρά μηνυμάτων;