Pràctica 5: Control semafòric de cruïlla

Programació a Baix Nivell — iTIC

Sebastià Vila-Marta Paco del Águila

16 d'abril de 2015

Índex

1 Introducció

Aquesta pràctica és un *remake* de la pràctica anterior en que es varien diversos aspectes:

- El sistema controla els semàfors d'una cruïlla simple en comptes d'un sol carrer.
- El sistema té una estructura de mòduls més sofisticada que aïlla millor les funcionalitats i permet un reaprofitement més alt.
- El sistema funciona en base a interrupcions.
- El supervisor poc consultar i variar l'estat del sistema.

1.1 Organització

1. Cal fer la pràctica en equip de dues o tres persones.
2. Caldrà lliurar un petit informe del temps de dedicació.
3. Cal desenvolupar el programari i la documentació usant els serveis del sistema de gestió de versions de l'EPSEM.
4. La pràctica té una durada de tres sessions.

Pel que fa al material necessari, cada equip:

1. Cal que disposi de 6 LED (2 vermells, 2 grocs i 2 verds) per la realització d'aquesta pràctica i la propera.
2. Cal que disposi d'un Arduino i la corresponent placa de prototips amb els seus accessoris habituals.
3. Alternativament a aquests components i la placa de prototips, es pot fer servir el *shield* *iTIC* presentat a la pràctica anterior.

1.2 Lliuraments

Caldrà lliurar el resultat de la pràctica a través del sistema Atenea de la forma que ja s'anunciarà. Els lliuraments inclouen el següent:

1. Un tar file que conté els fonts (nets i sense objectes, còpies de seguretat, etc.) del projecte incloent el `Makefile` corresponent.
2. La taula de temps de dedicació.
3. El sistema muntat íntegrament i funcionant correctament.

1.3 Mètode de treball

Per fer aquesta pràctica és convenient, una vegada més, estudiar amb cura la forma de repartir-se la feina de manera que els membres de l'equip puguin treballar en paral·lel. Per aconseguir això cal estudiar prèviament el projecte i decidir quin pla de treball se seguirà per executar-lo.

Per simplificar el cicle de desenvolupament-prova, és convenient començar per implementar el mòdul `queue`. A continuació els dos mòduls que actuen de drivers: `lamp` i `serial`. Després el mòdul `semaph`, que abstreu el concepte de semàfor. Per poder provar el seu funcionament, caldrà construir sengles programes de prova que després descartarem.

Quan aquests mòduls es donin per vàlids, cal continuar pel mòdul `control` i, finalment, acabar en el mòdul `crossing`.

El darrer desenvolupament a fer és el del programa supervisor escrit en `Python`.

El desenvolupament cal fer-lo sobre el servei de control de versions de l'assignatura (`http://escriny3.epsem.upc.edu`). Per tant cal preparar prèviament la disposició de directoris necessària.

TASCA 1 Prepareu la disposició de directoris i fitxers que us convinguin per començar a desenvolupar el projecte sobre el sistema de control de versions.

2 Descripció del sistema

2.1 Arquitectura

En el giny que cal construir intervenen dos sistemes diferenciats que són anàlegs als de la pràctica anterior:

1. El sistema semafòric. S'encarrega d'operar el semàfor d'acord amb una rutina preestablerta i les indicacions del sistema de supervisió. El nucli del sistema semafòric és el control semafòric, que interacciona amb driver del semàfor i amb el sistema de comunicacions. El sistema semafòric s'implementarà usant el kit d'Arduino i un petit muntatge maquinari auxiliar.

2. El sistema de supervisió. S'encarrega de controlar el sistema semafòric. L'implementarem com un programa escrit en Python que s'executa en l'estació de treball. Es comunica amb el sistema semafòric a través del port sèrie usant un protocol textual molt senzill.

La figura 1 mostra un esquema d'aquesta arquitectura.

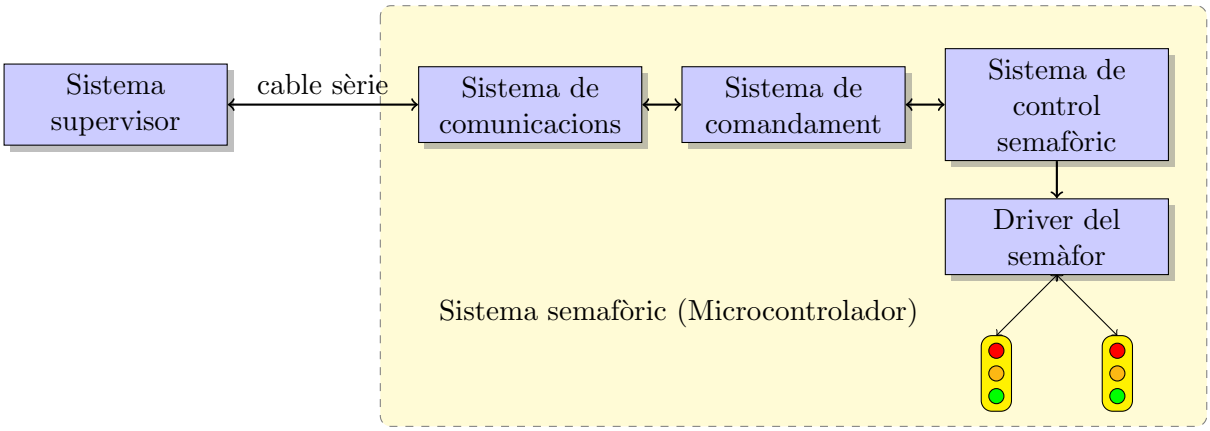

Figura 1: Arquitectura del sistema

El sistema te un funcionament similar al seu predecessor en la pràctica anterior. El sistema de control semafòric governa els semàfors de la cruïlla i els va fent canviar d'estat d'acord amb un cicle de treball preestablert. El sistema de comandament rep ordres del supervisor a través del sistema de comunicacions i les executa afectant el funcionament del control semafòric. El sistema de supervisió es comunica amb el sistema de comandament a través d'un petit protocol.

2.2 Protocol de supervisió

És el protocol que usen el sistema de supervisió i el control semafòric per a col·laborar. És un protocol en el que els missatges sempre són línies de text acabades en *carriage return* i *linefeed* (`"\r\n"`).

El sistema semafòric envia un primer missatge al sistema de supervisió quan ja s'ha inicialitzat i està a punt per funcionar. El supervisor no pot actuar fins que ha rebut aquest missatge. Aquest missatge és `START`. A partir d'aquest moment la iniciativa sempre la té el supervisor.

El sistema supervisor identifica els dos carrers que controla amb els noms `A` i `B`.

Els missatges que el supervisor pot enviar són els següents:

FA *Force A.* Força el semàfor del carrer `A` a verd. El pas a verd pot no ser immediat atès que sempre cal respectar el tems de carabassa corresponent. El sistema torna el missatge `OK` si tot va bé.

FB *Force B.* Idènticament a `FA` però pel carrer `B`.

?A *A state?.* Consulta quin és l'estat del semàfor del carrer `A`. El sistema torna els missatges `CLEAR`, `APPROACH` o `STOP` segons sigui el cas.

?B *B state?.* Idènticament que `?A` pel semàfor del carrer `B`.

R *Run.* Si el sistema està parat l'engega i aquest contesta amb el missatge `OK`. En cas contrari no fa res i contesta també amb `OK`.

H *Halt.* Atura els semàfors i els apaga fins a nova ordre. Si tot va bé el sistema contesta `OK`. Si el sistema està aturat els missatges `FA` i `FB` no tenen cap efecte.

Qualsevol altre missatge que envii el supervisor es considera un error i el sistema de semàfors respòn amb el missatge `COMERR` per indicar aquesta condició. Noteu que el protocol en cap cas fa echo del que s'escriu.

2.3 Estructura de mòduls

A banda del programa `Python` que actua de sistema supervisor, el programari que governa el microcontrolador s'escriurà usant `C` i s'organitzarà en diversos mòduls que responen al diagrama de mòduls de la figura 2.

Els mòduls tenen les següents responsabilitats:

gpio_device És el driver dels pins físics.

lamp És el driver d'un semàfor. Agrupa els LED que formen el semàfor i li permet maniobrar els LED que el formen de manera senzilla.

semaph És l'abstracció que representa un semàfor. Coneix els estats habituals d'un semàfor i permet maniobrar-lo convenientment.

control És el sistema de comandament dels semàfors d'una cruïlla. Actua sobre dos semàfors i coneix els cicles de treball que han de seguir per gestionar el flux de vehicles de la cruïlla. Actua conduit per les interrupcions del temporitzador.

queue És una cua de bytes de mida afitada. S'usa com a buffer en les comunicacions a través del port sèrie.

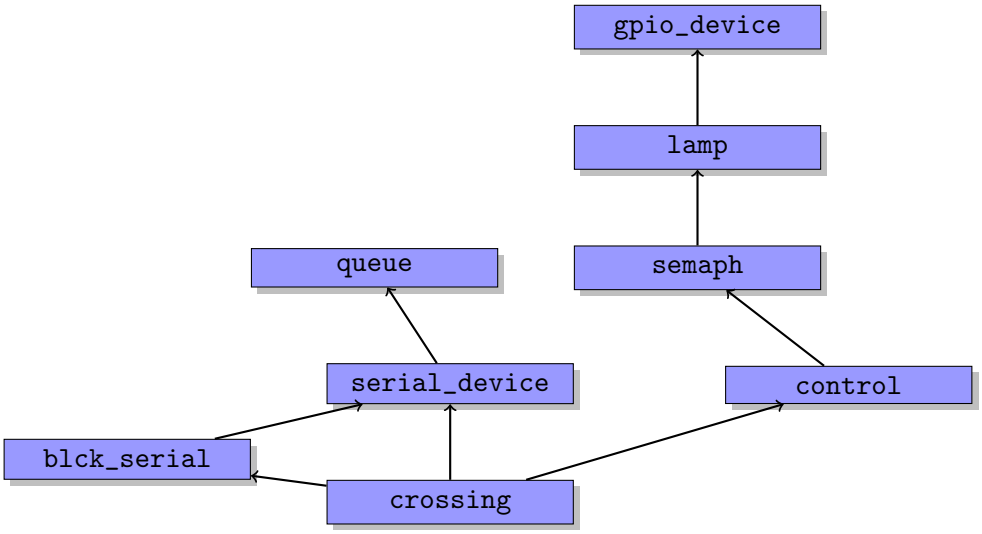

Figura 2: Diagrama de mòduls de la pràctica

serial_device És el driver del port de comunicacions sèrie. La seva responsabilitat és aïllar la resta de l'aplicació del detalls d'implementació de la UART. La seva implementació es basa en interrupcions. Escriu/llegeix exclusivament a través de cues.

blck_serial És un mòdul que permet escriure / llegir un bloc de caràcters en un dispositiu serie.

crossing És el mòdul principal. La seva responsabilitat és implementar el protocol de supervisió. Llegeix les ordres que venen del port sèrie, les decodifica i actua sobre el sistema de control de la cruïlla. En realitat és un intèrpret d'ordres.

3 Implementació del semàfor

Per tal de poder implementar el mòdul cal fer prèviament el muntatge del semàfor físic. Procediu com en la pràctica anterior però muntant ara dos semàfors en comptes d'un.

TASCA 2 Identifiqueu adequadament els pins dels leds que formen part del shield per definir els dos semàfors.

4 Mòdul «gpio_device»

Aquest mòdul és el mateix que s'ha fet servir a la pràctica anterior.

5 Mòdul «lamp»

El mòdul `lamp` abstreu un senyal de tres LED. A través seu se simplifica considerablement la maniobra d'una senyal.

EL mòdul ofereix un tipus de dades lamp_t que representa a les tres làmpades d'un semàfor i un conjunt d'operacions que permeten maniobrar aquestes làmpades.

typedef enum {Green, Yellow, Red} color_t	Aquest tipus s'usa per a identificar la làmpada contra la que interaccionen les operacions.

typedef struct {
  pin_t green;
  pin_t yellow;
  pin_t red;
} lamp_t;
	El tipus lamp_t representa una agrupació de leds amb els tres colors. Consta de 3 pin_t.

void lamp_init(lamp_t ∗**const** l,
  volatile uint8_t ∗prtg, uint8_t pg,
  volatile uint8_t ∗prty, uint8_t py,
  volatile uint8_t ∗prtr, uint8_t pr)
	Fa el bind entre un lamp_t i cadascun dels 3 pins físics. Inicialitza el lamp_t amb tots els LED apagats.

void lamp_on(lamp_t l, color_t c)	Encen el LED de color c de la làmpada l.
void lamp_off(lamp_t l, color_t c)	Apaga el LED de color c de la làmpada l.
void lamp_toggle(lamp_t l, color_t c)	Commuta l'estat del color c en la làmpada l.
bool lamp_is_on(lamp_t l, color_t c)	Retorna true sii en la làmpada l el color c està encès.

Noteu que l'estat de cada LED no es conserva en cap variable sinó que, quan s'ha de menester, es consulta l'estat del port. De manera similar, per commutar l'estat del LED s'usa una funcionalitat directa del maquinari. Consulteu [Atm11, apartats 14.2.4 i 14.2.2].

Amb aquest mòdul implementat és molt senzill comandar una làmpada. Fixeu-vos en el següent exemple:

```
lamp_t l;

lamp_init(&l, &PORTD, 4, &PORTD, 5 &PORTD, 6 );
lamp_on(l,Green);
lamp_toggle(l,Green);
```

TASCA 3 Implementeu el mòdul `lamp` i també un programa de test `test_lamp` per provar el seu funcionament correcte.

6 El mòdul «semaph»

El mòdul `semaph` és una capa damunt del mòdul `lamp` que ofereix una visió més abstracta del semàfor. En particular, un objecte de tipus semaph_t pot estar només en certs estats coherents i té consciència de l'estat en que es troba. Així, per exemple, un objecte semaph_t mai pot tenir dos LED encesos.

L'API del mòdul és la següent:

typedef enum {
  SemOff,
  SemClear,
  SemApproach,
  SemStop,
} semaph_state_t;
	Defineix una enumeració que conté els estats possibles en que pot trobar-se un semàfor. SemOff indica totes els discs apagats. SemClear indica el disc verd encès i la resta apagats. SemApproach indica el disc groc encès i la resta apagats. SemStop indica el disc vermell encès i la resta apagats.

typedef struct {
  lamp_t l;
  semaph_state_t s;
} semaph_t;
	Un objecte de tipus semaph_t representa un semàfor implementat per una làmpada.

void semaph_init(semaph_t ∗**const** s, **volatile** uint8_t ∗port_green, uint8_t pin_green, **volatile** uint8_t ∗port_yell, uint8_t pin_yell, **volatile** uint8_t ∗port_red, uint8_t pin_red)	Inicialitza el mòdul. Cal cridar-la obligatòriament abans d'usar cap altra funció del mòdul. Els paràmetres port_X i pin_X indiquen on es troba connectada la làmpada. Una vegada inicialitzat el mòdul el semàfor es troba en estat SemOff.
void semaphore_set(semaph_t ∗**const** sem, semaph_state_t s)	Força el semàfor sem a un estat concret determinat pel paràmetre s.
semaph_state_t semaph_get(**const** semaph_t sem)	Retorna l'estat en que es troba el semàfor sem.

Tasca 4 Implementeu el mòdul `semaph` d'acord amb l'API indicada. Implementeu també un mòdul `test_semaph` que us permeti comprovar-ne el correcte funcionament.

7 El mòdul «queue»

Aquest mòdul facilita el treball amb cues de mida afitada i fixa representada pel tipus queue_t. L'API d'aquest mòdul és la que segueix:

queue_t	Els objectes d'aquest tipus són cues de uint8_t. El nombre d'elements màxim de la cua és fix i afitat.
void queue_empty(queue_t *const q)	Inicialitza la cua q a l'estat buit.
bool queue_is_empty(**const** queue_t *const q)	Retorna true ssi q és buida.
bool queue_is_full(**const** queue_t *const q)	Retorna true ssi q és plena.
void queue_enqueue(queue_t *const q, uint8_t v)	Encua l'element v a la cua q. Si la cua és plena no encua res.
void queue_dequeue(queue_t *const q)	Desencua l'element del davant de la cua q. Si la cua és buida no fa res.
uint8_t queue_front(**const** queue_t *const q)	Retorna l'element del davant de la cua q. Si la cua és buida retorna un valor arbitrari.

La implementació de les cues segueix una estratègia circular. Podeu buscar en que consisteix aquesta tècnica per implementar cues en qualsevol llibre d'estructures de dades com, per exemple, [AHU83].

> TASCA PRÈVIA 5 Estudieu com s'implementa una cua circular. Poseu una atenció especial en la detecció de les condicions de cua buida i cua plena. Dissenyeu quina hauria de ser la representació del tipus queue_t.

TASCA 6 Implementeu el mòdul `queue`. Per provar el funcionament d'aquest mòdul podeu construïr un programa de test i provar-ho directament en l'estació de treball en comptes de fer-ho sobre l'arduino, on les coses són més sofisticades.

Les operacions de la cua en aquesta pràctica poden ser cridades des de rutines d'interrupció. Això obre la possibilitat a conflictes d'accés. Imagineu, per exemple, que mentre s'està executant una operació de desencua una rutina d'interrupció afegeix un element a la cua. Aquest tipus de conflictes que es deriven de l'accés concurrent són molt importants i origen d'errors molt difícils de depurar. Per evitar-los en aquest cas determinarem quines zones del codi cal executar de forma *atòmica*, és a dir, que han d'executar-se com si es tractés d'una sola instrucció, d'una tacada. Per aconseguir aquesta atomicitat cal inhibir i restaurar les interrupcions en la zona.

La llibreria `avr-libc` ofereix una eina específica per a delimitar aquestes zones, que sovint s'anomenen d'*exclusió mútua*. Es tracta de la construcció `ATOMIC_BLOCK` que trobareu documentada a [RW+02, capítol `atomic.h`].

TASCA 7 Afegiu a les operacions del mòdul `queue` que ho necessitin les zones d'exclusió mútua que siguin necessàries. Recordeu que cal filar prim i no posar-ne ni més ni menys del compte.

8 Mòdul «serial_device»

Aquest mòdul té com a principal funció aïllar la resta del programari de les especificitats del dispositiu de comunicacions sèrie. Recordeu en l'Arduino el dispositiu sèrie "viatja" a través de la connexió USB i és compartit amb el sistema d'implantar codi usat per `avrdude`. El mòdul `serial` interactua amb la UART del microcontrolador via interrupcions.

El mòdul treballa usant dues cues fonamentals la de recepció i la de transmissió. La recepció de dades funciona de la següent forma: cada vegada que es rep una interrupció indicant que hi ha dades a llegir, la rutina d'interrupció corresponent llegeix el byte rebut i l'encua en la cua de recepció. De forma similar, quan es produeix una interrupció indicant que la UART pot transmetre. Com a resposta a aquesta interrupció s'obté un byte de la cua de transmissió i s'envia. Si la cua de transmissió és buida, es desactiva la interrupció fins que torni a haver-hi bytes a enviar.

> TASCA PRÈVIA 8 Recupereu la informació sobre la UART de l'AVR, la seva arquitectura i mode de configuració i operació del manual tècnic de l'AVR, [Atm11], i de les pràctiques de l'assignatura DISPRO. Estudieu amb cura com funciona el mecanisme d'interrupcions i com es configura.

Les operacions del mòdul són les següents:

void serial_open(**void**)	Inicialitza el modul i deixa la UART a punt per enviar/rebre caracters de 8 bit a 9600 bit s^{-1}, amb 1 bit d'stop, sense paritat i en mode asíncron.
void serial_close(**void**)	Deshabilita la UART per no rebre ni transmetre res. En cas que quedin caràcters al buffer de sortida, s'espera fins que acabin de sortir abans de deshabilitar la UART.
uint8_t serial_get(**void**)	Retorna un byte llegit de la cua de receptció del port sèrie. Es bloqueja indefinidament fins que hi ha un caràcter disponible per a ser llegit. En cas que no es llegeixi prou sovint es poden perdre caràcters.
void serial_put(uint8_t c)	Envia un byte pel port sèrie. En cas que la cua de transmissió estigui plena, es bloqueja fins que el byte es pugui afegir a la cua.
bool serial_can_read(**void**)	Retorna true sii hi ha un caràcter disponible per a ser llegit. Si aquesta funció retorna true es garanteix que una posterior crida a serial_get() no es bloquejarà.

Noteu que les operacions del mòdul realment escriuen i llegeixen de les corresponents cues de recepció i transmissió. Simultàniament les rutines d'interrupció privades que el mòdul conté van encuant i desencuant bytes a les cues de forma autònoma.

Per implementar les rutines d'interrupció usant `C`, empreu les eines que s'expliquen a [RW+02, capítol `interrupt.h`]. Les interrupcions que cal usar, tal i com indica el manual tècnic de l'AVR, són `USART_RX_vect` i `USART_UDRE_vect`.

TASCA 9 Dissenyeu el mòdul `serial` i també el programa de prova corresponent `test_serial`.

9 El mòdul «blck_serial»

Aquest mòdul ofereix les funcionalitats d'escriure i llegir blocs de caràcters a la UART. Per aquest motiu, aquest mòdul depèn del mòdul serial.

Les funcions que ofereix són aquestes:

void print(**char** s[])	Envia per la UART el bloc de caràcters que se li passa com a paràmetre. Quan troba el caràcter `'\0'` acaba.
int readline(**char** s[], uint_8t m)	Va llegint caràcters i els va guardant a la taula s fins que troba un caràcter no gràfic o bé el nombre de caràcters iguala a m. El valor retornat és el nombre de caràcters que ha llegit.

TASCA 10 Dissenyeu el mòdul `blck_serial` i també el programa de prova corresponent `test_blck_serial`.

10 El mòdul «control»

10.1 L'API del mòdul

Aquest mòdul és l'encarregat d'implementar la política de treball dels semàfors de la cruïlla: fa rotar els colors dels semàfors d'acord amb els temps establerts, permet forçar canvis de flux i aturar o engegar els semàfors de la cruïlla. Per distingir cada semàfor etiquetem els carrers amb els noms `A` i `B`.

El control dels semàfors de la cruïlla com a tal es considera un sistema autònom que pot estar en divesos estats cadascun dels quals explica com es troben els semàfors. Cada estat s'allarga durant un cert temps que ve donat per la temporització preestablerta. El pas del temps el marca una interrupció del rellotge.

Les funcions públiques del mòdul són les següents:

typedef enum {StreetA, StreetB} street_t	Declara els noms dels carrers de la cruïlla.
void control_init(**void**)	Inicialitza el mòdul de control. És imprescindible cridar la funció abans d'usar el mòdul. Una vegada inicialitzat deixa els semàfors apagats i cal engegar-los explícitament.
void control_force(street_t t)	Força el carrer `t` a l'estat Clear. Respecta els temps de seguretat establerts (vegeu descripció posterior).
void control_off(**void**)	Apaga els semàfors de la cruïlla.
void control_on(**void**)	Engega els semàfors de la cruïlla.
semaph_state_t control_get_state(street_t s)	Retorna l'estat del semàfor del carrer `s`.

10.2 Temporització i estats del sistema

El mòdul `control` en règim permanent funciona en base a una seqüència d'estats. Cada estat determina com s'il·luminen els semàfors de la cruïlla. Els estats són: `Aclear`, `AtoB`, `Bclear` i `BtoA` a més de l'estat que indica que els semàfors estan apagats i que anomenem `ABoff`. El cronograma de la figura 3 mostra com se succeeixen els estats en règim permanent de treball.

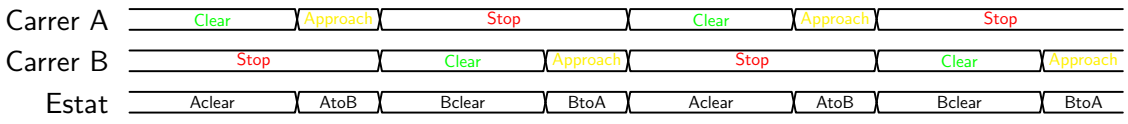

Figura 3: Cronograma del cicle ordinari de funcionament dels semàfors d'una cruïlla.

Els temps bàsics amb que definirem la temporització estaran basats en un temps de tick de 100 ms i seràn: 80 ticks per a l'estat `Aclear`, 100 per a l'estat `Bclear` i 20 ticks per a la resta.

Les operacions control_off(), control_on() i control_force() trencan el cicle de funcionament ordinari dels semàfors i forcen nous estats. En el cas de control_off() i control_on() apaguen i engeguen

els semàfors de la cruïlla. Quan estan apagats la única operació que té efecte és control_on() i quan s'engeguen ho fan en estat `Aclear`. Els cronogrames de la figura 4 mostren com es comporten aquestes operacions.

Figura 4: Efecte on/off sobre el control dels semàfors d'una cruïlla.

L'operació de control_force() força un carrer, per exemple `A`, a l'estat `Clear` (o de manera simètrica el carrer `B` a estat `Stop`). Aquest canvi està subjecte a les següents consideracions de seguretat:

- Si el carrer `B` es trobava en estat `Clear`, abans de commutar a `Stop` cal que passi per l'estat `Approach` obligatòriament.
- Si el carrer `B` es trobava en estat `Approach`, no cal intervenir atès que està a punt de canviar a `Stop`.
- Si el carrer `A` ja es trobava en estat `Clear`, la crida a control_force() té l'efecte de tornar a començar el cicle talment com si acabés d'entrar en estat `Clear`.
- Si el carrer `A` ja es trobava en estat `Approach`, la crida a control_force() té l'efecte de tornar a l'estat `Clear` i començar el cicle de nou.

La figura 5 mostra en un cronograma l'efecte de l'operació control_force() sobre el carrer `A` en diverses circumstàncies.

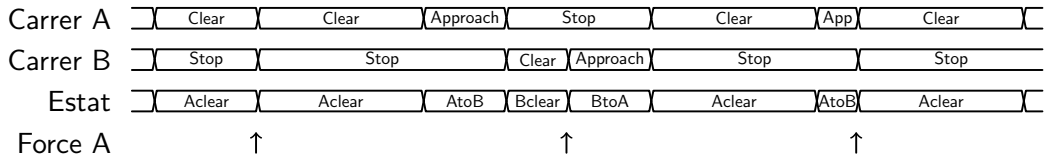

Figura 5: Efecte del force sobre el control dels semàfors d'una cruïlla.

10.3 Implementació del mòdul

La implementació del mòdul es basa en l'ús de les interrupcions generades pel comptador/temporitzador. Durant la inicialització del mòdul cal programar el comptador de 16 bit de manera que provoqui una interrupció cada 100 ms. El comptador s'usa en mode CTC. Això implica que cal determinar el *prescaler* a usar i també el llindar de comptatge a usar per que la interrupció esdevingui quan toqui.

> TASCA PRÈVIA 11 Estudieu amb cura el funcionament dels comptadors/temporitzadors de l'AVR. El manual tècnic del microcontrolador, [Atm11, cap. 16], és la font d'informació més recomanable. Feu-vos al càrrec de l'arquitecura de temporitzador, dels diferents modes de funcionament, de l'ús del prescaler, de com aturar i engegar-lo, etc.

La rutina d'interrupció corresponent al temporitzador entrarà cada 100 ms i s'encarregarà de actualitzar l'estat del sistema de semàfors de la cruïlla.

Noteu els següents detalls:

- Fixeu-vos que en aturar els semàfors cal també deixar de provocar interrupcions ja que aquestes no tindrien cap efecte. El temporitzador ha de tornar a interrompre quan el semàfor s'engega de nou.
- També cal tenir present que quan el semàfor engega, cal inicialitzar el temporitzador de manera que la primera interrupció sigui exactament al cap de 100 ms.

Tasca 12 Implementeu el mòdul `control`. Implementeu també un programa per a comprovar el funcionament correcte de `control`. Anomeneu-lo `test_control`.

11 El mòdul «crossing»

Aquest mòdul conté el programa principal i és l'encarregat d'interpretar el protocol de supervisió. A banda d'inicialitzar els mòduls necessaris i arrencar el sistema, la seva tasca principal és llegir missatges del canal sèrie usant el mòdul `serie`, determinar si són correctes i en el seu cas executar les corresponents ordres interactuant amb el mòdul `control`.

Tasca 13 Implementeu el mòdul `crossing` i proveu-lo usant qualsevol emulador de terminal sèrie que tingueu a mà.

12 El supervisor

Usant la mateixa estratègia que vàreu usar en la pràctica anterior, en aquesta pràctica el supervisor serà un programa `Python` que parlarà el protocol i facilitarà a l'usuari un interpret de comandes per controlar el sistema de semàfors. També, com a la pràctica anterior, cal usar la classe Interpret implementada a les pràctiques de TECPRO.

Tasca 14 Dissenyeu un joc d'ordres escaient per a un usuari i implementeu el programa supervisor del sistema de control.

Referències

[AHU83] Alfred V. Aho, J.E. Hopcroft i Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1983.

[Atm11] Atmel. *ATmega48A/PA/88A/PA/168A/PA/328/P Complete datasheet*. Anglès. Versió Rev. 8271D–AVR–05/11. 2011. URL: `http://www.atmel.com/Images/doc8271.pdf` (consultat 7 de jun. de 2012).

[RW+02] Theodore A. Roth, Jörg Wunsch et al. *AVR Libc Home Page*. Anglès. 2002. URL: `http://www.nongnu.org/avr-libc` (consultat 19 de mar. de 2012).