Pràctica 4: Control semafòric amb supervisor extern

Programació a Baix Nivell — iTIC

Sebastià Vila-Marta Paco del Águila

23 d'abril de 2020

Índex

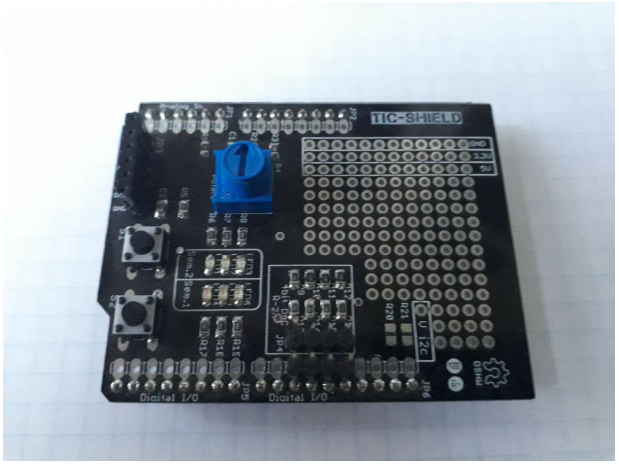

Figura 1: *Shield* d'Arduino per a la pràctica.

1 Introducció

Aquesta pràctica consisteix en un petit projecte l'objecte del qual és dissenyar i construir un semàfor de carretera controlat des d'un computador remot. La implementació de l'aplicació requerirà dissenyar i implementar diversos mòduls que caldrà implantar sobre el kit Arduino.

2 Organització

2.1 Material

Cada persona ha de disposar del següent material:

1. *Arduino ONE*. Cal disposar d'una placa Arduino per desenvolupar la resta de pràctiques de l'assignatura.

2. *Shield TIC* per Arduino. En aquest *shield* es disposa del material necessari per fer aquestes pràctiques, minimitzant la utilització de la placa de prototips que és més propensa a mals contactes. Cal adquirir-la (a preu de cost) al mestre de laboratori. La figura 1 mostra una imatge d'aquesta placa.

2.2 Lliurament

Lliureu aquesta pràctica seguint les mateixes pautes que s'han emprat a la pràctica 1.

2.3 Mètode de treball

Apliqueu les mateixes pautes que s'han emprat a la pràctica 1. De cara a repartir-se la feina de manera efectiva, cal 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 els dos mòduls que actuen de drivers: `gpio_device` i `serial_device`. Per poder provar el seu funcionament, caldrà construir sengles programes de prova que després descartarem. Cal continuar pel mòdul que farà de driver del semàfor `semaphore` i també fer el corresponent programa de prova.

Quan aquests mòduls es donin per vàlids, es dissenyarà el mòdul `controlsem` i, finalment, el mòdul `main`.

El darrer desenvolupament a fer és el del programa supervisor.

2.4 Control de versions

Apliqueu les mateixes pautes que s'han emprat a la pràctica 1.

2.5 Makefile

Cal incloure un `Makefile` per gestionar el projecte que també serà objecte d'avaluació. El `Makefile` s'ha d'incloure en el tarfile del lliurament.

2.6 Mètode de treball

Per fer aquesta pràctica és convenient, una vegada més,

3 Descripció del sistema

3.1 Arquitectura

En el giny que cal construir intervenen dos sistemes diferenciats:

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 el 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 2 mostra un esquema d'aquesta arquitectura.

El funcionament típic d'aquest sistema és el que es descriu en el diagrama de seqüència de la figura 3. En aquest diagrama intervenen tres entitats. El sistema supervisor, el sistema de control i el driver del semàfor. La comunicació entre el sistema de control i el driver del semàfor és redueix als següents missatges:

Init Inicialitza el semàfor. Cal enviar aquest missatge abans de començar a operar amb el semàfor.

setClear Posa el semàfor a verd i el prepara per anar canviant de color periòdicament. Aquest missatge pot enviar-se sigui quin sigui l'estat del semàfor.

doCycle Provoca un canvi cíclic de color en el semàfor. Per exemple, si estava en groc, passa a vermell.

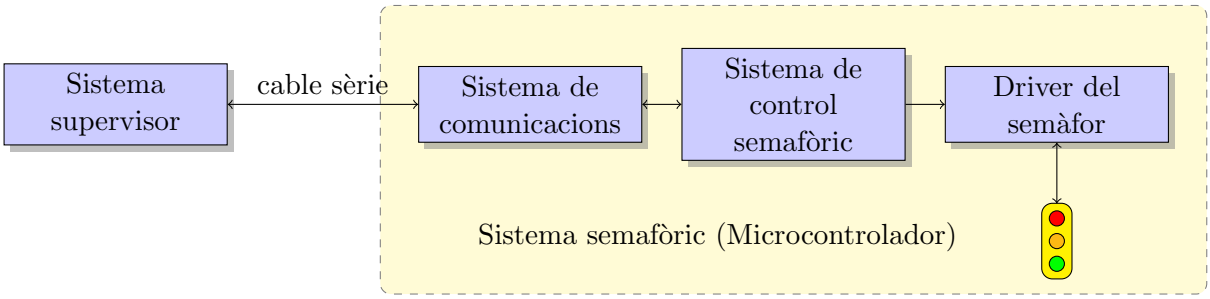

Figura 2: Arquitectura del sistema

setOff Apaga el semàfor completament.

D'altra banda, el sistema de supervisió es comunica amb el sistema semafòric a través del següent protocol:

E Emergència. Aquest missatge no té efecte si el semàfor està apagat. Sigui quin sigui l'estat del semàfor (verd, groc o vermell), aquest passa a verd. Una vegada modificat torna el missatge `EMERGENCY`.

S Shutdown. Aquest missatge no té efecte si el semàfor està apagat. Atura el semàfor. Una vegada aturat torna el missatge `SHUTDOWN`.

R Restart. Aquest missatge només té efecte si el semàfor està apagat. Engega de nou el semàfor en l'estat verd. Una vegada en marxa altre cop torna el missatge `RESTART`.

3.2 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 4.

Els mòduls tenen les següents responsabilitats:

gpio_device És el driver del port(s) d'entrada/sortida digital de l'AVR. La seva responsabilitat és aïllar la resta de l'aplicació de l'accés a les potes físiques.

semaphore És el driver del semàfor. La seva responsabilitat és aïllar la resta de l'aplicació del semàfor físic i la forma de comandar-lo a través dels ports del microcontrolador.

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 polling.

controlsem És el sistema central del control semafòric. Implementa la política de gestió del semàfor i de comunicacions amb el supervisor.

main És el mòdul principal. La seva responsabilitat principal és arrencar la resta de sistemes.

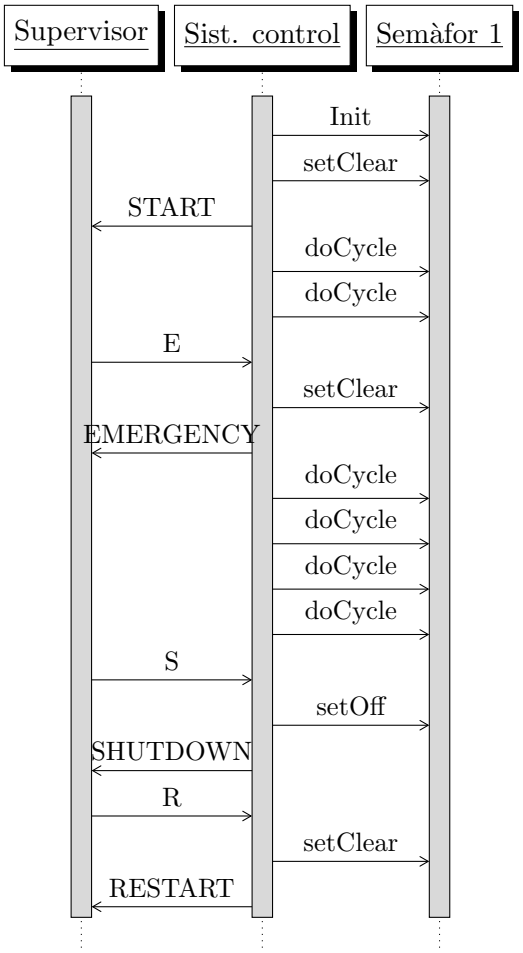

Figura 3: Exemple de comunicació entre supervisor i sistema semafòric.

4 El mòdul «gpio_device»

4.1 Implementació del mòdul

Aquest mòdul implementa una capa d'aïllament entre els ports digitals de l'AVR i l'aplicació. La seva principal funció és simplificar l'accés a aquests ports pagant un petit peatge en termes d'eficiència.

El mòdul es basa en el concepte de «pin», que és l'abstracció d'un pin físic d'un port. Cada pin té un sentit que indica si és d'entrada o sortida i està associat a un bit concret d'un port específic.

Un objecte de tipus pin_t representa aquesta abstracció i està dotat de les operacions que es consignen més endavant. Aquestes operacions permeten associar l'objecte a un pin físic i operar amb ell convenientment.

En certa forma el concepte de pin actua «d'intermediari» amb el pin físic, de ma mateixa manera que succeeix amb els `FILE *`, que actuen d'intermediaris amb els fitxewrs físics. Com en els cas dels fitxers, hi ha una operació específica per vincular el objecte pin al pin físic i una altra operació per desvincular-lo, són els operacions pin_bind() i pin_unbind() respectivament.

L'especificació del mòdul és la que segueix:

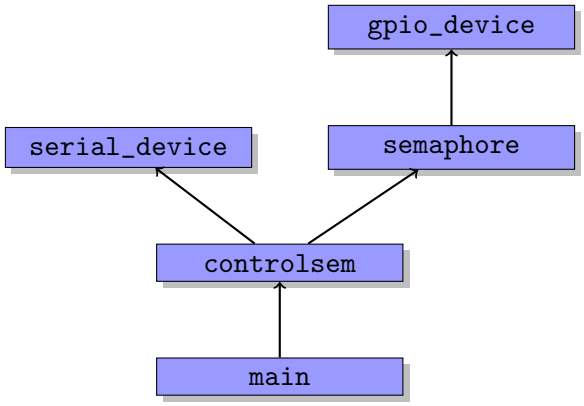

Figura 4: Diagrama de mòduls de la pràctica

typedef enum {Input, InputNP, Output} pin_direction_t;

Usat per especificar la direcció del pin. Quan la direcció és Input, s'activa sempre el *pull-up* del port, quan és InputNP no s'activa el *pull-up*.

```
typedef struct {
  volatile uint8_t *port;
  uint8_t pin_mask;
} pin_t;
```

Representació d'un pin. pin_mask és la màscara que correspon al pin en qüestió.

pin_t pin_bind(**volatile** uint8_t *port, uint8_t pin, pin_direction_t d);

Estableix el vincle entre el pin físic i el pin lògic. Retorna un objecte pin_t associat al pin número pin del port port i inicialitzat en mode d. El mode defineix també si s'activa o no el *pull-up* en cas de pins d'entrada.

void pin_w(pin_t p, bool v); — Escriu un valor en el pin p. p ha d'estar associat i en mode Output per a que la funció tingui efecte.

bool pin_r(pin_t p); — Llegeix un valor del pin p. p ha d'estar associat.

void pin_toggle(pin_t p); — Commuta el valor del pin p. p ha d'estar associat i en mode Output.

void pin_unbind(pin_t *const p); — Desassocia el pin p d'un pin físic.

Tasca 1 Dissenyeu el mòdul `gpio_device`. Per tal d'accedir físicament als pins del port corresponent des de `C`, consulteu [RW+02, Library manual, `<avr/sfr_defs.h>`]. Tingueu en compte que a `<avr/io.h>` teniu les definicions dels apuntadors que defineixen les adreces dels registres de propòsit específic de l'AVR i que aquesta capçalera també inclou `<avr/sfr_defs.h>`.

La funció d'unbind() pot semblar supèrflua però el seu ús permet fer implementacions «sofisticades» del mòdul. Per exemple, si tenim sospites que poden haver-hi conflictes en un programa per que diversos mòduls —sense intenció—, fan bind del mateix pin, podríem implementar el mòdul de forma que bind() prengués nota dels pins «ocupats» i unbind() els alliberés. D'aquesta forma es podríen detectar fàcilment colisions entre pins.

4.2 Prova del mòdul

TASCA PRÈVIA 2 Instal·leu els paquets necessaris per compilar de forma creuada cap a Arduino: `gcc-avr`, `avrdude`, `avr-libc` així com les seves dependències.

Aquest mòdul es pot provar fàcilment usant un LED i amb un petit programa que generi un patró d'encesa/apagada sobre aquest LED.

Com que aquest programa cal que s'executi sobre la plataforma Arduino, haureu de compilar-lo usant el compilador creuat d'AVR, convertir-lo a format `hex` i implantar-lo en l'Arduino usant `avrdude` tal i com fèieu en el cas dels programes assemblador de DISPRO. En aquest cas la seqüència seria la següent:

```
$ avr-gcc -std=c99 -Os -mmcu=atmega328p -fshort-enums \
          -DF_CPU=16000000UL gpio_device.c prova_led.c -o prova_led
$ avr-objcopy -Oihex prova_led prova_led.hex
$ sudo avrdude -c arduino -p atmega328p -P /dev/ttyACM0 -U prova_led.hex
```

Per tal de temporitzar la commutació del led podeu usar la funció `_delay_ms()` de la llibreria d'AVR, [RW+02, Library manual, `<util/delay.h>`].

TASCA 3 Implementeu un petit programa per provar el mòdul `gpio_device` a base de generar sobre un pin concret un patró de senyal que pugui ser observat amb un LED. Comproveu el seu funcionament.

Aquest mòdul fa més còmode l'accés a un pin però també més ineficient. Us proposem la següent tasca per tal d'apreciar la diferència.

TASCA 4 Implementeu dos programes que commutin un pin en mode sortida de l'AVR tant ràpidament com sigui possible i durant un temps indefinit. Per accedir al pin:

- En el primer programa accediu usant la forma tradicional d'accés als pins mitjançant màscares.
- En el segon programa accediu usant les operacions del mòdul `gpio_device`.

Un i altre programa generen un senyal quadrat a la sortida del pin la freqüència del qual depèn de la velocitat d'accés al pin del programa. Usant un oscil·loscopi determineu a quina freqüència arriba un i altre programa.

Sense alterar l'especificació del mòdul, com podríeu variar la implementació per tal que la seva eficiència augmentés?

5 Mòdul «semaphore»

5.1 Implementació del semàfor físic

Per tal de poder implementar el mòdul cal tenir la *shield TIC* o, en el seu defecte, el muntatge del semàfor físic de la pràctica anterior.

5.2 Implementació del mòdul

El mòdul `semaphore` té com a responsabilitat principal actuar de driver o capa d'aïllament del semàfor físic. L'API del mòdul, que cal codificar en el fitxer `semaphore.h` és la següent:

typedef enum { SemaphoreOff, SemaphoreClear, SemaphoreApproach, SemaphoreStop, } semaphore_state_t;	Defineix una enumeració que conté els estats possibles en que pot trobar-se un semàfor. SemaphoreOff indica tots els discs apagats. SemaphoreClear indica el disc verd encès i la resta apagats. SemaphoreApproach indica el disc groc encès i la resta apagats. SemaphoreStop indica el disc vermell encès i la resta apagats.
void semaphore_init(**void**);	Inicialitza el mòdul. Cal cridar-la obligatòriament abans d'usar cap altra funció del mòdul. Una vegada inicialitzat el mòdul el semàfor es troba en estat SemaphoreOff.
void semaphore_next(**void**);	Força el semàfor al següent estat. Aquesta funció només pot aplicar-se en cas que el semàfor estigui en estat SemaphoreClear, SemaphoreApproach o SemaphoreStop. La seva funció és desplaçar de forma circular l'estat al següent entenent que els estats estan en l'ordre expressat anteriorment.
void semaphore_set(semaphore_state_t s);	Força el semàfor a un estat concret determinat pel paràmetre s.

TASCA 5 Implementeu el mòdul `semaphore` d'acord amb l'API indicada. En la implementació vetlleu per organitzar el codi de forma que la part més física d'accés als ports quedi delegada al driver `gpio_device`.

5.3 Prova del mòdul

Per provar el mòdul cal construir un petit programa principal, que anomenarem `prova_sempahore.c`. Aquest programa ha d'inicialitzar el mòdul `semaphore` i, posteriorment, fer diverses maniobres sobre el semàfor per poder observar que funciona correctament. Per tal de temporitzar les maniobres podeu usar la funció _delay_ms() de la llibreria d'AVR, [RW+02, Library manual, `<util/delay.h>`].

TASCA 6 Implementeu el programa per provar el mòdul `semaphore`. Implanteu-lo en l'Arduino i depureu el mòdul fins a tenir-lo llest.

6 Mòdul «serial_device»

6.1 Implementació del mòdul

Aquest mòdul té com a principal funció aïllar la resta del programari de les especificitats del dispositiu de comunicacions sèrie. Recordeu que a 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_device` interactua amb la UART del microcontrolador amb polling i ofereix operacions per obrir la connexió, enviar caràcters i rebre caràcters (sense ocupar-se dels possibles errors de transmissió detectats) i tancar la connexió.

TASCA PRÈVIA 7 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.

Les operacions del mòdul són les següents:

void serial_open(**void**);	Inicialitza la connexió i ho prepara tot per enviar/rebre caracters. La transmissió es configura amb 8 bit a 9600 bit s^{-1}, amb 1 bit d'stop, sense paritat i en mode asíncron.
uint8_t serial_get(**void**);	Requereix la conexió oberta. Retorna un byte llegit 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);	Requereix la conexió oberta. Envia un byte pel port sèrie. En cas que estigui ocupat enviant una altra dada, es bloqueja fins que l'enviament en curs acaba.
bool serial_can_read(**void**);	Requereix la conexió oberta. 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.
void serial_close(**void**);	Tanca la connexió. Si escau, espera a que s'hagi enviat el darrer caràcter.

En el moment d'implementar el mòdul cal estudiar amb cura en el manual de l'AVR la inicialització del temporitzador que determina la velocitat de transmissió. A tal efecte cal recordar que la velocitat de rellotge de l'AVR és de 16 MHz. També és convenient donar un cop d'ull a funcions de la llibreria d'AVR com ara loop_until_bit_is_set(), [RW+02, Library manual, `<avr/sfr_defs.h>`], que faciliten la implementació de moltes de les funcions d'aquest mòdul.

TASCA 8 Dissenyeu el mòdul `serial_device`. Escriviu-lo en els fitxers corresponents.

6.2 Prova del mòdul

Per provar el mòdul cal dissenyar un petit programa que anomenarem `prova_serial.c`. Aquest programa inicialitzarà el mòdul serial i després cada cert temps, per exemple 2 s, anirà enviant una paraula seguida dels caràcters `'\r'` i `'\n'`. Usant una terminal com `picocom` o bé simplement l'ordre de la terminal

```
$ cat /dev/ttyACM0
```

comproveu que es rep correctament la seqüència de paraules en l'estació de treball.

Modifiqueu ara el programa de prova per tal que no comenci a enviar paraules fins que llegeixi un caràcter provinent de l'estació de treball. Per últim, feu que des de l'estació de treball es pugui controlar el flux de paraules que envia l'Arduino: enviant un caràcter comencem a rebre paraules i enviant-ne un altre s'atura l'enviament de paraules.

Per enviar caràcters podeu usar `picocom` () o bé, en una terminal diferent, executar la comanda (useu Ctrl+D enlloc de Return):

```
$ cat > /dev/ttyACM0
```

TASCA 9 Implementeu el programa per provar el mòdul `serial_device`. Implanteu-lo en l'Arduino i depureu el mòdul fins a tenir-lo llest. Si dubteu sobre l'ús de la comanda cat utilitzeu picocom tant en recepció com en transmissió.

7 El mòdul «controlsem»

7.1 Implementació del mòdul

Aquest mòdul és el centre del sistema semafòric. La seva funció principal és mantenir la rotació dels colors del semàfor en els temps adequats i, al mateix temps, "escoltar" les ordres que provenen del supervisor a través del port sèrie.

Els temps de rotació dels color del semàfor en el seu funcionament ordinari s'especifiquen en funció d'un temps base τ i són de 40τ per l'estat `Clear`, 10τ per l'estat `Approach` i 60τ per l'estat `Stop`. Al temps base τ l'anomenarem "tick".

La implementació del mòdul es basa en la següent estratègia. Les seves operacions públiques principals són dues: tick_monitor i tick_semaphore. Aquestes funcions són cridades pel programa principal a intervals regulars de $\tau = 100\,\text{ms}$ i s'encarreguen de:

1. Sondejar si hi ha cap ordre provinent del supervisor a través del port sèrie que cal llegir i executar. Aquest és el propòsit de tick_monitor.
2. Tenint en compte que ha passat un tick des de la darrera crida, determinar si ha arribat el moment en què cal rotar el color del semàfor i fer-ho si és el cas. Aquest és el propòsit de tick_semaphore.

Per implementar la funció tick_semaphore cal conservar en una variable estàtica local del mòdul l'estat del controlador. L'estat el formen dues dades:

1. L'estat del semàfor (Off, Clear, etc.).
2. El nombre de ticks que queden per a canviar de color el semàfor.

Cada crida a tick_semaphore ha d'actualitzar convenientment aquest estat i actuar en conseqüència.

void controlsem_init();	Inicialitza el mòdul i el deixa a punt per a ésser utilitzat. Això inclou la inicialització del port sèrie i del semàfor. Un cop fet això arrenca el semàfor posant-lo en `Clear` i envia el missatge `START`.
void tick_monitor(**void**);	Determina si hi ha alguna ordre del supervisor per llegir. Les ordres segueixen el protocol que s'ha explicat a l'apartat 3.1. Si és el cas la llegeix i executa l'acció associada a l'ordre.
void tick_semaphore(**void**);	Decrementa el nombre de ticks que falten per canviar l'estat del semàfor i, si escau, canvia l'estat del semàfor. Cal que tingui en compte que el semàfor pot estar apagat i, en aquest cas, no es produeixen rotacions ni es comptabilitzen ticks.

TASCA 10 Dissenyeu el mòdul `controlsem` i escriviu els corresponents fitxers.

7.2 Prova del mòdul

Aquest és un mòdul complicat de provar. L'estratègia més escaient per provar-lo consisteix en provar-lo en l'estació de treball i no en la plataforma Arduino. Per fer això cal implementar uns "falsos" mòduls `semaphore` i `serial_device` que, en comptes d'interactuar amb el maquinari de l'Arduino el que fan és escriure/llegir missatges pels canals estàndard d'entrada/sortida. Aquests tipus de "falsos" mòduls s'anomenen *stub*. Aleshores, amb un unit test podem comprovar que el funcionament del mòdul `controlsem` és el que s'escau. Atès lo laboriós del procediment, considereu aquesta feina opcional.

8 Mòdul «main»

El mòdul `main` en aquesta pràctica és trivial. Simplement s'encarrega d'inicialitzar el `controlsem` i posteriorment invocar indefinidament les funcions `tick_` en intervals de temps τ.

Per provar el sistema complet, arranqueu el sistema i connecteu-vos amb `picocom`. Treballant directament amb el protocol especificat per al supervisor hauríeu de poder comandar el sistema correctament.

TASCA 11 Dissenyeu el mòdul `main`. Compileu i implanteu el sistema semafòric complet en la plataforma Arduino. Usant `picocom` comproveu el seu funcionament i depureu el que sigui necessari.

9 El supervisor

El supervisor és un programa que s'executa en la vostra workstation i és la interfície a través de la qual un usuari comanda el sistema semafòric. Es tracta d'un programa escrit en `Python` que llegeix les ordres que dóna l'usuari i les comunica al sistema semafòric a través del port sèrie usant el protocol definit anteriorment.

Per implementar aquest programa recorrereu a la classe `Interpret` que vàreu dissenyar en una de les pràctiques de TECPRO. Es tracta de dissenyar un programa que instanciï un `Interpret` i el configuri adequadament per a respondre a les següents ordres de l'usuari:

start Engega el cicle ordinari del semàfor.

stop Atura el cicle del semàfor.

emergency Força el semàfor a recomençar el cicle.

Cada vegada que l'usuari escrigui una d'aquestes ordres, el programa haurà d'enviar pel port sèrie el missatge escaient i esperar la resposta adequada. Si la resposta no arriba o bé no és la que toca, el programa haurà de mostrar un missatge d'error.

Per tal de comunicar-se a través del port sèrie useu el mòdul de `Python` anomenat `pySerial`, [Lie10]. Aquest mòdul us facilitarà la lectura/escriptura a través dels dispositius sèrie del vostre computador.

TASCA PRÈVIA 12 Repasseu la pràctica sobre el receptari de TECPRO i recupereu la classe `Interpret`.

TASCA 13 Usant la classe `Interpret` que vàreu dissenyar en una pràctica de TECPRO i el mòdul `pySerial` dissenyeu un programa `Python` que permeti a un usuari comandar el sistema semafòric des de l'estació de treball tal i com s'ha explicat prèviament.

TASCA 14 Responeu a les següents preguntes:

1. Quan el supervisor envia un senyal d'emergència, què determina el temps màxim que pot trigar el semàfor a canviar?

2. Quin és el valor del temps màxim en el cas anterior?

3. El temps que el LED groc, per exemple, està encès durant el cicle normal del semàfor és sempre exactament el mateix? En cas que no ho fos, què podria provocar canvis en aquest temps?

4. Durant el cicle normal del semàfor, podríem provocar un endarreriment en el canvi d'estat d'alguna forma? Per què?

Referències

[Atm11] Atmel. *ATmega48A/PA/88A/PA/168A/PA/328/P Complete datasheet*. Ang. Vers. Rev. 8271D–AVR–05/11. 2011. URL: `http://www.atmel.com/Images/doc8161.pdf` (cons. 20-02-2017).

[Lie10] Chris Liechti. *pySerial's Documentation*. Ang. Vers. 2.6. 2010. URL: `http://pyserial.sourceforge.net` (cons. 21-03-2012).

[RW+02] Theodore A. Roth, Jörg Wunsch et al. *AVR Libc Home Page*. Ang. 2002. URL: `http://www.nongnu.org/avr-libc` (cons. 20-02-2017).