

PROGRAMACIÓ DE SIMULACIONS I INSTRUMENTS DE MESURA

LABVIEW

Enfonsar la flota



Albert Ripoll i Josep Morancho

Gran de Física
Universitat de Barcelona

4 de desembre de 2019

ÍNDEX



Introducció.....	3
Estructura de l'informe.....	3
Descripció del joc.....	3
El circuit.....	4
Panell frontal del VI principal.....	5
Diagrama de blocs del VI principal.....	6
Estructura general.....	6
Fase prèvia.....	6
Calibració.....	7
Generació dels vaixells.....	9
Apuntar.....	9
Comprovació del tret.....	10
Resposta al tret.....	10
Tancament.....	11
SubVIs.....	11

Introducció

Estructura de l'informe

L'informe s'estructura de general a concret.

Primer, en l'apartat «Descripció del joc», es descriu el joc tal com un usuari que pretén jugar el percebrà.

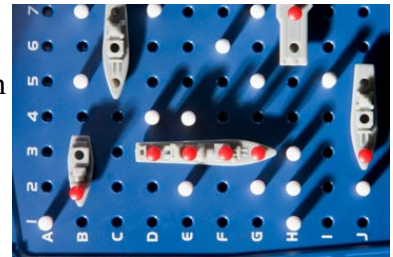
En segon lloc, en l'apartat «Circuit» es presenta el circuit muntat a l'Arduino.

En tercer lloc, en l'apartat «Panell Frontal del VI principal», s'expliquen les funcions al taulell de joc configurat al panell frontal del VI principal.

En quart lloc, en l'apartat «Diagrama de blocs del VI principal», s'exposa com s'estructuren les diferents parts que configuren el diagrama de blocs del VI principal i després s'expliquen com s'han aconseguit aquestes parts juntament amb l'ús de certs SubVIs.

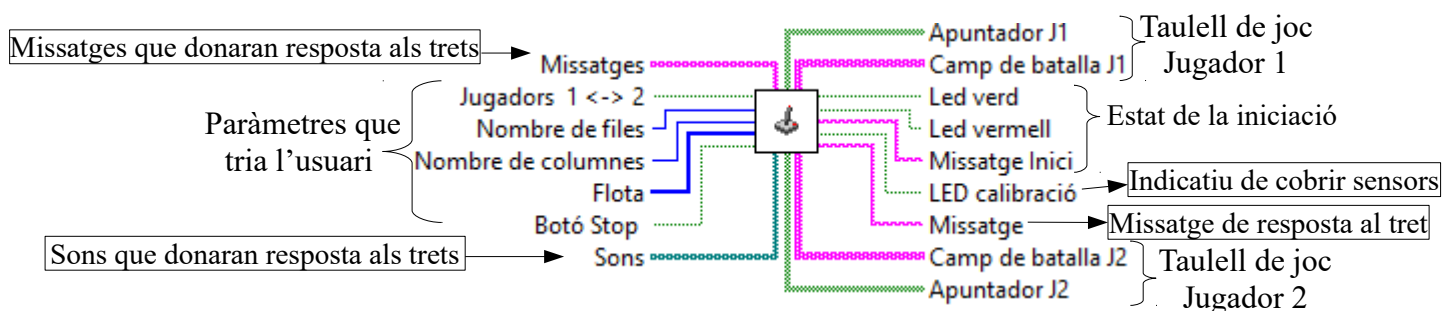
En últim lloc, en l'apartat dels SubVIs, s'explica la funcionalitat i programació que aquests inclouen.

Dins de cada apartat s'ha anat alternant una estructura ordenada cronològica amb la ordenació característica del treball de general a concret. En l'apartat dels SubVIs aquest fet es veurà molt clar ja que d'entrada s'explicaran les primeres solucions proposades seguit de les millores implementades.



Descripció del joc

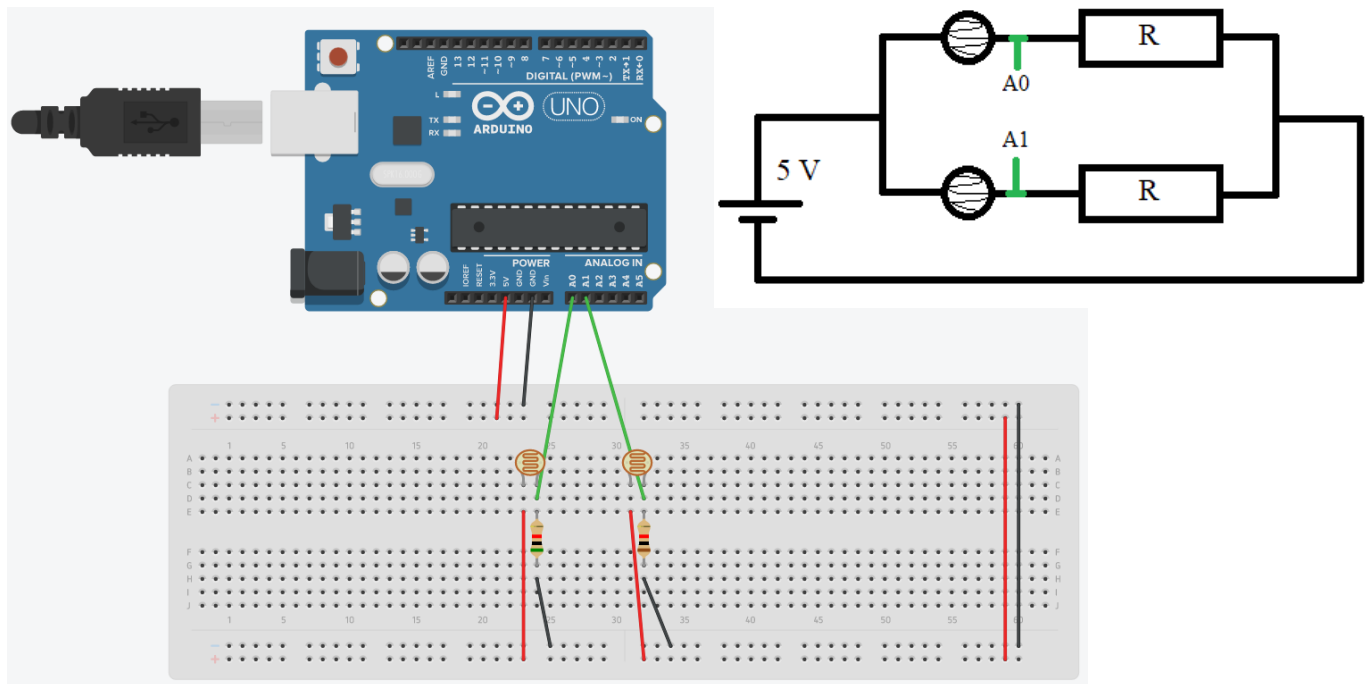
El joc d'enfonsar la flota consta d'un taulell de joc de dimensions files x columnes amb vaixells ocults en ell. L'usuari pot triar les mides del taulell de joc (entre 5x5 a 10x10), el nombre de vaixells (entre 1 i 5) i la seva mida (entre 0 i 5). L'usuari també pot triar si jugar sol o dos jugadors amb un taulell per jugador.



L'objectiu del joc és tocar i enfonsar tots els vaixells ocults (i fer-ho abans que l'altre jugador). Per apuntar, el jugador compta amb dos sensors de llum a la placa d'Arduino que prèviament haurà calibrat (seguint les instruccions de calibració). El sensor de l'esquerra controla el moviment horitzontal i el de la dreta controla el moviment vertical. El jugador deixa il·luminat amb més o menys mesura els sensors per apuntar a la posició que veurà al taulell de joc «Apuntador». Un cop passin 5 segons, s'efectuarà el tret. El resultat del tret s'indicarà de tres maneres:

- 1- Es representarà al «Camp de batalla» el resultat del tret O per aigua i X per tocat.
- 2- S'escoltarà un so d'error (quan ja havia disparat en aquella posició), un so d'aigua (ha fallat), un so de tocat (ha tocat un vaixell) o un so de victòria (quan hagi enfonsat tots els vaixells).
- 3- Sortirà per escrit a l'indicador «Missatge». *Jugador 1* o bé *Jugador 2* seguit de: *Ja hi has disparat, Aigua, Tocat o Victòria!*

Circuit



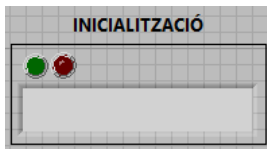
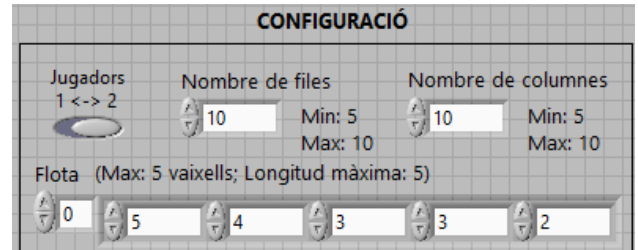
[↑] Fig. 1: Circuit esquemàtic i circuit muntat sobre la placa d'Arduino.

Panell frontal del VI principal

En el panell frontal hi figuren:

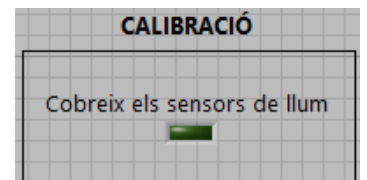
- Les instruccions del joc que indiquen els passos a seguir per començar el joc.
- Uns controls de configuració. Defineixen el tauler de joc (files i columnes), la quantitat i dimensió dels vaixells així com el mode de jugador: un jugador o dos jugadors.

Per assegurar el correcte funcionament del joc, nombre de files i de columnes s'ha fixat entre 5 i 10 amb la funció *Coerce* de la pestanya *Data Entry* dins de les propietats del control. De la mateixa manera, els cinc primers elements del control Flota estan acotats entre 0 i 5.



- Una finestra d'inicialització. Mostrarà la llum verda i un missatge de «Tot correcte» quan la inicialització de l'Arduino i la configuració escollida sigui correcte. En canvi, quan algun aspecte no sigui correcte s'engegarà el LED vermell i detallarà el que no permet iniciar el joc.

- Una finestra de calibració. S'engegarà un LED quan haguem de cobrir els sensors de llums per calibrar-lo en baixa il·luminació.



- Indicador del Joc. Un indicador de caràcters que anirà indicant les tirades dels jugadors: Ja disparat, aigua o tocat. També inclou un botó per tancar el joc abans de finalitzar-lo.

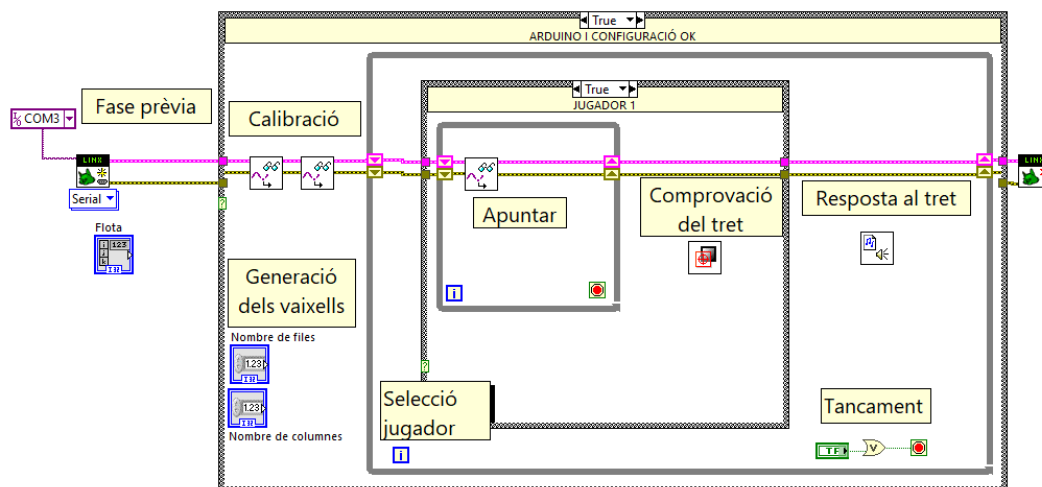
- Quadrícula de joc. Consisteix en una matriu de LEDS on es podrà veure la posició on s'apunta i un camp de batalla que mostrarà les tirades fetes anteriorment. Hi ha una quadrícula per jugador.



- Els controls dels missatges i sons de sortida. Seran els missatges i sons de resposta als trets. Al iniciar el programa per primera vegada a un ordinador nou, convé verificar que els *paths* dels arxius de sons estan correctament assignats.

Diagrama de blocs del VI principal

Estructura general



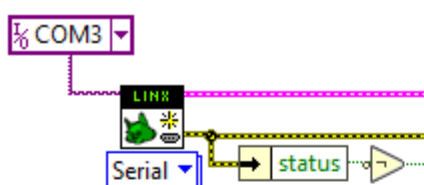
L'estructura general del programa és la següent:

- Fase prèvia on es comprova la obertura de l'Arduino i la configuració de la flota. Ens porta al Case True (tot correcte) o False (hi ha error i tanca). En el cas correcte se segueix:
- Fase de calibració en paral·lel amb la generació dels vaixells.
- Bucle while que dona pas al joc d'un dels jugadors (case jugador 1, case jugador 2) i dona una resposta al tret. En aquest bucle hi ha els *Shift Registers* que assegurin que tret rere tret no es perdi i estigui actualitzada la informació de: el nombre de «Tocats» que porta cada jugador, el camp de batalla i la informació de l'Arduino. Aquest bucle només s'acaba quan la resposta és que s'enfonsen tots els vaixells o amb el botó STOP.
- Case jugador té una fase repetitiva d'apuntar i una comprovació del tret.
 - Apuntar: és un bucle While que s'executa durant 5 segons realitzant mesura rere mesura permetent al jugador apuntar. En acabar els 5 segons es tanca el bucle i s'efectua el tret.
 - Comprovació del tret: per a cada jugador es comprova el resultat del tret i es passa el resultat a la fase de resposta del tret.

Fase prèvia

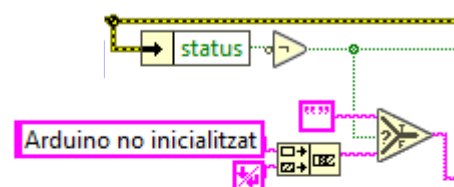
A la fase prèvia es comprovarà que obertura de l'Arduino hagi estat sense errors així com també es comprova la correcte configuració de la flota. Les respostes a aquestes comprovacions s'indiquen a la finestra d'inicialització del panell frontal.

Estat de l'Arduino



S'inicialitza l'Arduino i es comprova l'estat del seu error. **status** en principi retorna True si hi ha algun error i False en cas contrari. Es nega perquè la lògica sigui: està bé = True, està malament = False.

D'aquesta manera, quan la inicialització de l'Arduino està bé (True) s'agafa un caràcter buit i quan hi ha algun error s'agafa el missatge «Arduino no inicialitzat»

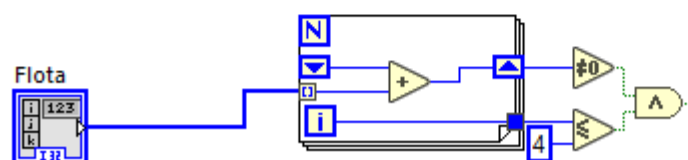


Estat de la configuració

Com s'ha explicat en els controls de configuració a l'apartat del panell frontal, les files i les columnes estan fixades entre 5 i 10 i els valors dels elements de Flota estan fixats entre 0 i 5. Això disminueix possibles errors en definir la configuració.

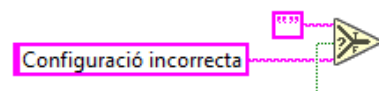
No obstant, per equivocació de l'usuari, es poden produir dues configuracions incorrectes: La introducció d'una Flota composta només d'elements nuls i la introducció d'una Flota amb més de 5 elements.

En aquest apartat es comprova si la suma dels elements de Flota és diferent a 0 i (AND) si la quantitat d'elements de Flota és menor o igual a 5 (índex d'iteració de 0 a 4).



Se segueix la lògica està bé = True, està malament = False.

El resultat lògic es porta a una selecció de missatge.

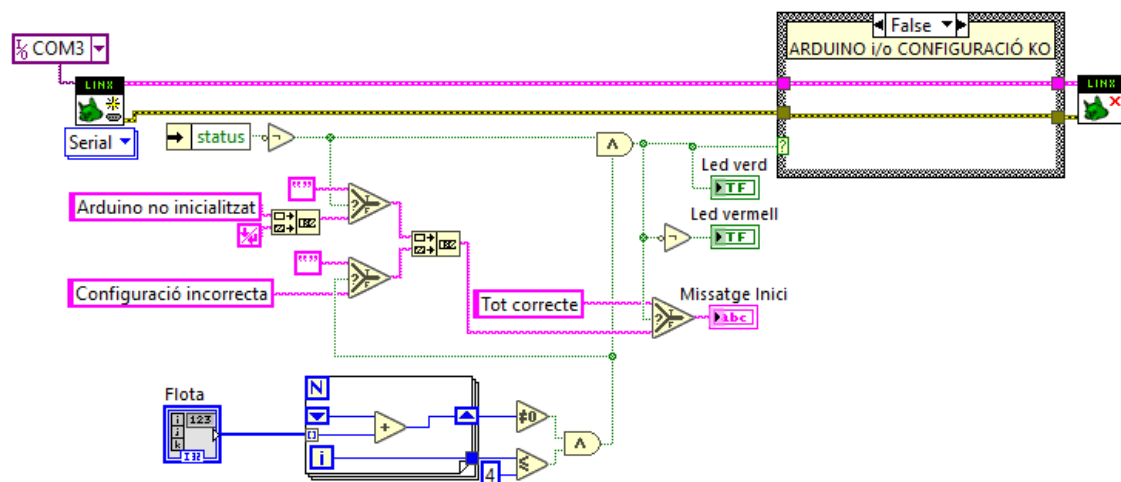


Resposta d'inicialització

La resposta de l'estat de l'Arduino i de l'estat de la configuració conflueixen en un AND.

Si els dos estats estan bé serà True, s'encendrà el LED verd, es seleccionarà el missatge «Tot correcte» i s'entrarà el programa en el *Case* «Arduino i configuració OK».

En el cas que algun dels estats anteriors no sigui correcte, s'encendrà el LED vermell, es seleccionarà el missatge concatenat dels errors i s'entrarà el programa en el *Case* «ARDUINO i/o CONFIGURACIÓ KO» que el que farà serà tancar l'Arduino.



Calibració

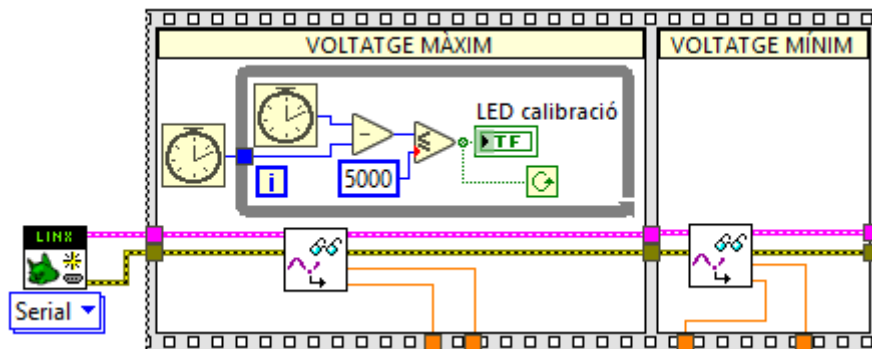
Un cop s'ha comprovat que l'Arduino s'ha inicialitzat sense errors i que la configuració és correcta, el primer que succeeix (en paral·lel a la generació dels vaixells) és la calibració.

La calibració en el nostre joc és necessària per a determinar el voltatge a màxima il·luminació (V_{max}) i el voltatge a mínima il·luminació (V_{min}) que serviran de referència per a associar voltatges entremitjos a un índex de fila o columna (veure SubVI MatchingIndex).

La calibració es realitza amb dues fases, primer es medeix el voltatge a màxima il·luminació i posteriorment el de mínima. Per a forçar aquesta seqüència, s'ha usat una estructura *Sequence*. Coherentment en el programa, els cables i voltatges referents al pin 1 (A0) han estat col·locats a la

part superior i els del pin 2 (A1) a la inferior.

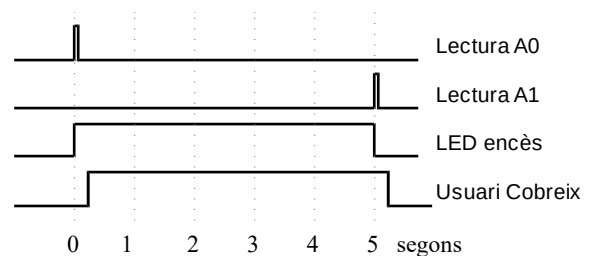
Seqüència de voltatge màxim i mínim



Amb el SubVI ReadVoltage es llegeix el voltatge al pin 1 i al pin 2. La lectura és immediata.

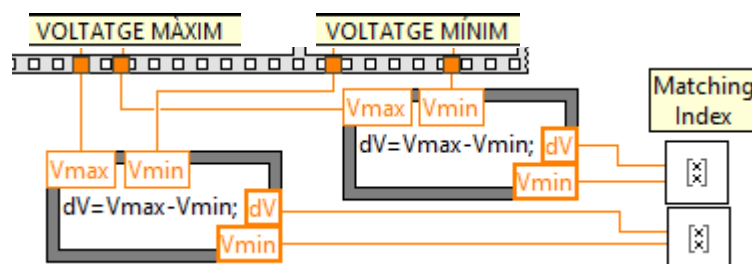
Per altra banda, s'encén el LED de calibració durant 5 segons. Passats 5 segons, s'apaga i es passa a la segona seqüència, que llegeix immediatament els pins A0 i A1.

És fonamental notar que la lectura és immediata mentre que el temps de l'usuari en reaccionar i cobrir els sensors a l'encesa de la llum i el retirar-se a l'apagar-se no és immediat. [→] Fig. 2: Representació dels moments de lectura, del LED encès i de quan l'usuari cobreix els sensors (s'ha usat el temps de reflex humà de 0,2 segons).



MatchingIndex

Les sortides de les lectures van a parar a un formula node que en calcula la diferència i retorna aquesta diferència juntament amb el voltatge mínim. Els valors s'entren al SubVI MatchingIndex.

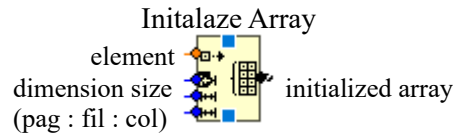
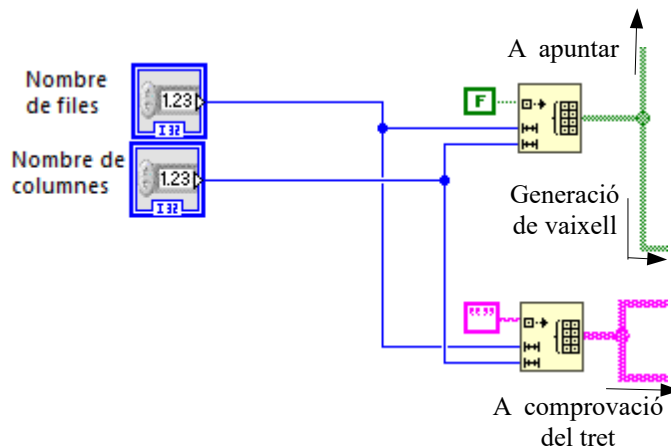


Generació dels vaixells

El nostre joc bàsicament treballarà a partir de 3 matrius:

- La matriu d'apuntar.
- La matriu Battlefield (on estaran anotades els tocats i aigües).
- La matriu oculta on estaran els vaixells col·locats.

Inicialització de matrius False i Battlefield

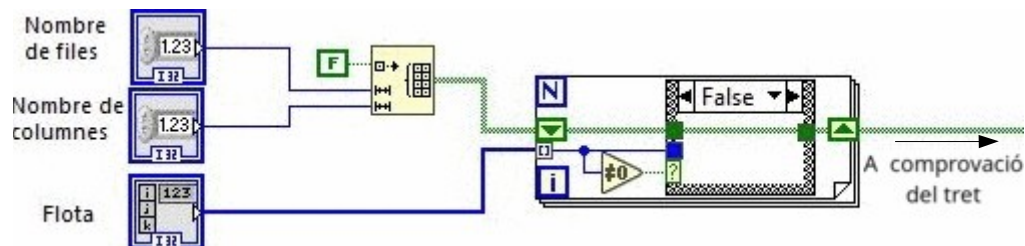


Com a matriu base del programa, es generarà una matriu 2D booleana plena de Falses de la mida definida en els controls «Nombre de files» i «Nombre de columnes». De la mateixa manera, es generarà una matriu 2D de caràcters buits.

La matriu False ens servirà de base per generar els vaixells i per a la fase d'apuntar.

La matriu de caràcters serà enviada a la fase de comprovació del tret.

Matriu amb la flota



Per generar la flota, s'iterarà sobre cada mida del vaixell de la llista del control «Flota». Quan aquest sigui 0 (case False) la matriu passarà al *Shift Register* directament i es mantindrà inalterada. Quan aquest sigui diferent de 1 (case True), s'executarà el SubVI placefleet que col·locarà un vaixell de la dimensió donada a un lloc de la matriu on no hi coincideixi amb cap vaixell prèviament col·locat. Quan es col·loquin tots els vaixells s'enviarà la matriu generada a l'apartat de comprovació del tret.

El codi anteriorment presentat es troba repetit dues vegades, per generar dues disposicions diferents de vaixells pels dos jugadors.

Selecció del jugador

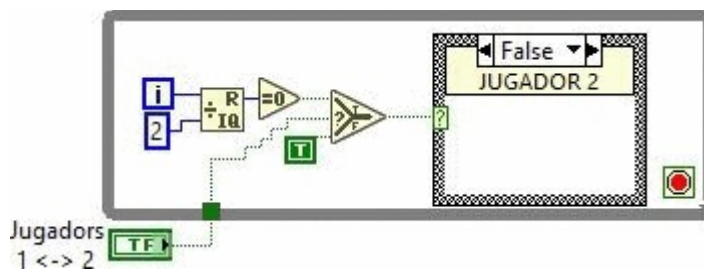
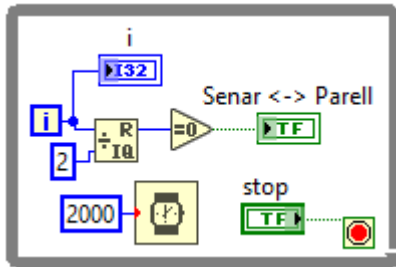
Un cop s'han generat els valors de calibració i els vaixells, la següent fase serà la de seleccionar el jugador que apuntarà i dispararà. Aquesta selecció es durà a terme en el bucle while que dona pas al joc d'un dels jugadors (3^r epígraf de l'apartat Estructura general).

El codi que permet seleccionar el torn dels jugadors consisteix en distingir números parells de números senars. Els números parells tenen la propietat de ser divisibles per 2 i donar com a residu enter zero. En canvi, els nombres senars sempre donen un residu al ser dividits per 2. És a dir,

$\forall \mathbb{Z}$ es compleix que:

$$1^{\text{r}} \text{ decimal} \left(\frac{\text{Número parell}}{2} \right) = 0 \quad ; \quad 1^{\text{r}} \text{ decimal} \left(\frac{\text{Número senar}}{2} \right) = 5 \neq 0$$

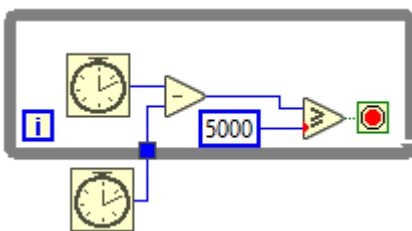
La funció Quotient & Remainder ens permet saber si hi ha residu i fer aquesta funcionalitat. Veiem-ho en el següent codi. Aquest ens genera, alternativament, True i False identificant si la iteració és parell o senar i, en conseqüència seleccionant alternativament el *Case* Jugador 1 o el 2.



Per implementar la opció de que l'usuari pugui triar si jugar sol o jugar 2 jugadors, s'incorpora un botó que selecciona la opció de triar alternar un jugador o altre o, en canvi, sempre seleccionar el mateix jugador.

Apuntar

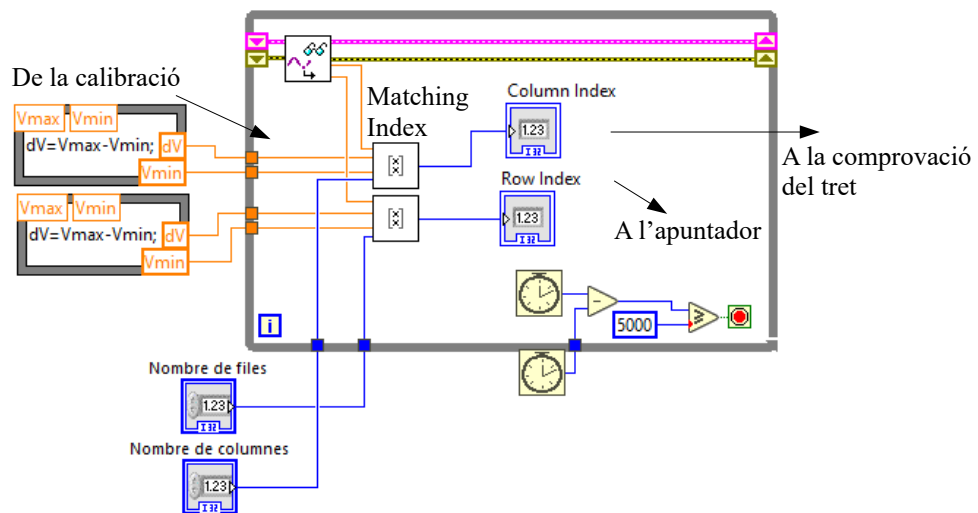
Com s'ha vist en l'apartat d'Estructura general (3^r epígraf), la fase d'apuntar és un bucle While que s'executa repetidament durant els 5 segons que dura aquesta fase. En aquests 5 segons es van realitzant mesures del voltatge dels pins i comparant-se amb les mesures del calibratge. Alhora es van mostrant, en cada moment, la posició on s'apunta.



L'execució repetida del bucle es controla comparant el temps del sistema abans d'entrar-hi i el temps del sistema dins del bucle. Quan la diferència entre aquests dos temps supera o iguala 5000 ms, fet que vol dir que el bucle porta 5 segons executant-se, el bucle s'atura (Stop if True).

Lectura

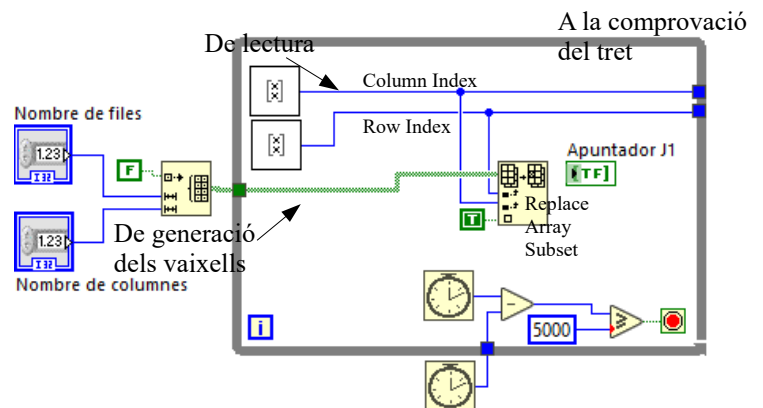
En cada iteració del bucle while, que hem de pensar que són molt ràpides, el programa crida el SubVI ReadVoltage per llegir el voltatge actual dels dos sensors. Recordem que el voltatge actual és proporcional a la il·luminació del sensor, és a dir, que la informació que s'extreu en definitiva és quant d'il·luminats estan els sensors.



El voltatge actual de cada sensor s'introdueix al paràmetre V del MatchingIndex. També se li introdueixen els paràmetres de calibració V_{min} i $V_{max} - V_{min}$ així com el valor d'una de les dimensions. El SubVI MatchingIndex ja s'encarrega, a partir d'aquests valors, d'associar l'índex de la fila o columna a la que s'està apuntant.

Apuntador

L'apuntador que es visualitza al Panell Frontal és una matriu de LEDs amb el LED al que s'apunta encès i la resta apagats. Es tracta de la matriu base de Falses creada a la fase de generació dels vaixells amb el valor corresponent a on s'està apuntant (és a dir al Row Index i Column Index) substituït per True amb el Replace Array Subset.



Comprovació del tret

La comprovació del tret es realitza dins de cada jugador. Perquè cada jugador té la seva matriu de batalla (Battlefield In i Out) i té el seu comptatge dels vaixells que ha tocat (Hits In i Out).

La fase de comprovació del tret consisteix en verificar si el Row Index i el Column Index provinent de la fase d'apuntar representen una posició on ja s'ha disparat, on no hi ha vaixell (aigua) o on sí hi ha un vaixell (tocat). Tota la comprovació es fa dins del SubVI CheckShot i només cal passar-li i extreure-li les variables correctament.

Les entrades d'aquesta fase són:

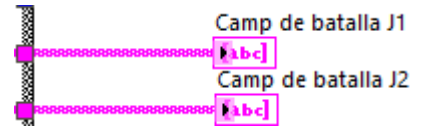
- Column Index** } Provenent de la
- Row Index** } fase d'apuntar
- Fleet Size** —————> Provenent de la fase prèvia (suma de dimensions de tots els vaixells)
- Hits In** (serà un comptatge diferent per cada jugador, començant per 0)
- Fleet Position** } Provenent de la fase de generació dels vaixells
- Battlefield In** } (serà la matriu de cada jugador, inicialment buida i actualitzada en aquesta fase)

Les sortides que dona aquesta fase són *Sound Index*, *Battlefield Out* i la indicació de quin jugador ha realitzat el moviment que aniran a la fase de resposta del tret. També es dona la sortida *Victory?* a la fase de resposta del tret i de tancament. Les sortides *Hits Out* surten de la comprovació del tret per ser tornades a agafar posteriorment amb un Shift Register.

Resposta al tret

Com dèiem a l'últim epígraf de l'apartat Estructura general, la resposta al tret consta de:

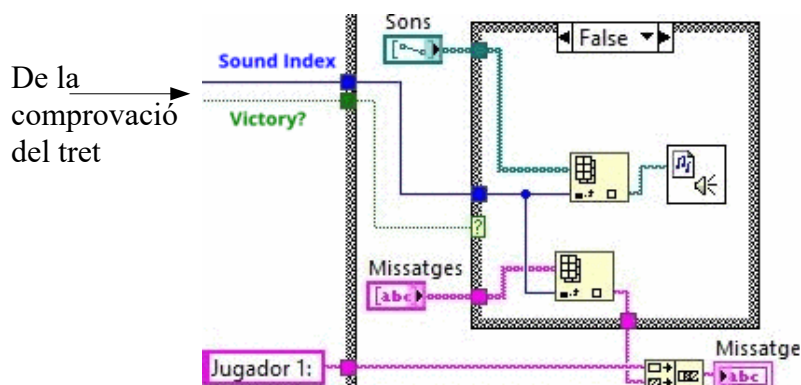
1- Representar la matriu *BattleScene* provinent del SubVI *CheckShot*



2- Presentar els missatges «Ja hi has disparat», «Aigua», «Tocat» o «Victòria» que es concatena amb el número del jugador.

3- Emetre el so d'error, d'aigua, de tocat o de victòria.

Els punts 2 i 3 es controlen a través de la variable *Sound Index*, que ve de la comprovació del tret. Aquest selecciona l'element de la llista de Sons i de la llista de Missatges. El cas de victòria es tracta per separat ja que és necessari que el programa esperi un temps raonable (2,5 segons) abans de tancar-se ja que sinó el so i missatge de victòria no s'acaba de sentir ni veure ja que el programa es tanca de seguida.



Tancament

El tancament de l'Arduino es pot produir per 3 motius:

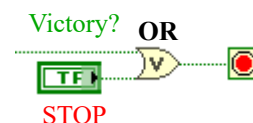


1) A la fase prèvia s'ha detectat que alguna cosa no estava bé. → L'estructura Case que s'executa tanca directament l'Arduino.

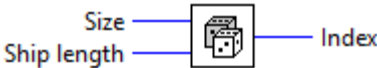







2) Un dels jugadors ha aconseguit realitzar tants tocats com la suma de dimensions dels vaixells (és a dir *Victory?* = True)

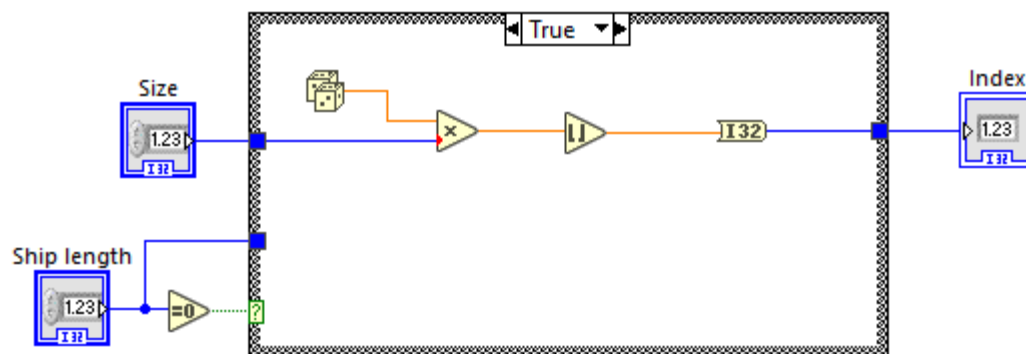
Es tanca el While gran amb un OR.

3) S'ha premut el botó STOP.

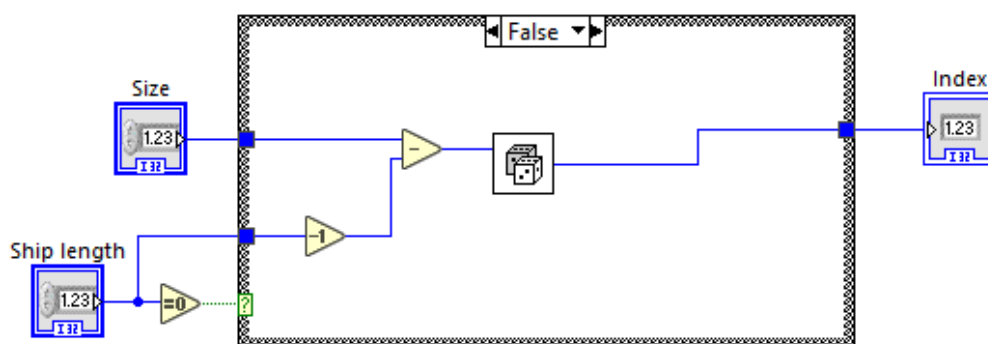



SubVIs

Nom	Icona	Entrades	Sortides
RandomIndex		<p>Size</p> <p>Ship length</p>	Index
Funció			
<p>Retornar un número enter aleatori a partir del qual es construirà el vaixell.</p> <p>Amb entrada de <i>Ship length</i> inexistent, aquest número podrà ser qualsevol índex dins de les dimensions. És a dir, serà entre 0 (la primera posició) i $Size - 1$ (la última posició).</p> <p>Amb entrada de <i>Ship length</i> existent, el número aleatori estarà restringit per la mida del vaixell. És a dir, serà entre 0 (primera posició) fins a $Size - Ship Length - 1$.</p>			
Explicació			
<p>La idea és construir el vaixell a partir d'un índex cap a índexs creixents i que no se'n surti de la mida de la matriu. Posem per exemple que volem construir un vaixell de 4 caselles en una línia de mida 7. Les possibilitats que tindrem per fer-ho seran 4:</p> <div style="display: flex; justify-content: space-between; align-items: flex-start;"> <div style="width: 45%;"> <p>0 1 2 3 4 5 6</p>  <p>0 1 2 3 4 5 6</p>  <p>0 1 2 3 4 5 6</p>  <p>0 1 2 3 4 5 6</p>  </div> <div style="width: 45%;"> <p>Començant per l'índex 0</p> <p>Començant per l'índex 1</p> <p>Començant per l'índex 2</p> <p>Començant per l'índex 3</p> </div> </div> <p>En aquest cas $Size = 7$ i $Ship length = 4$. La sortida ha de ser un índex aleatori entre 0 i 3. Per tant, les posicions seran entre 0 i $Size - Ship length - 1 = 7 - 3 - 1 = 3$.</p>			
Detall			
<p>Utilitzem la funció random  que genera un número aleatori en l'interval [0,1) i el multipliquem per <i>Size</i> perquè el número aleatori generat sigui entre [0, <i>Size</i>). Com que, el número és igual o major a 0 però menor que <i>Size</i>, l'arrodonim al enter més petit . Així, el número generat és un enter (encara que de type=float) totalment aleatori entre [0, <i>Size</i>-1]. Es converteix a enter .</p> <p>En resum, la funció genera un índex totalment aleatori de la mida <i>Size</i> entre 0 i <i>Size</i>-1, quan la funció es crida sense donar cap entrada a l'entrada <i>Ship Length</i> ja que per defecte agafa el valor 0.</p>			



Quan a la funció sí que se li entra un valor a *Ship Length*, llavors la condició que s'executarà serà, per força, la *False*. En aquest cas, les posicions aleatòries que es generaran seran fins l'índex $Size - Ship\ length - 1$. Per fer-ho es crida el propi VI però amb l'entrada *Ship length* sense indicar; és a dir 0. Cosa que fa més compacte el codi.



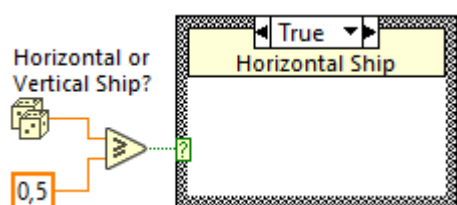
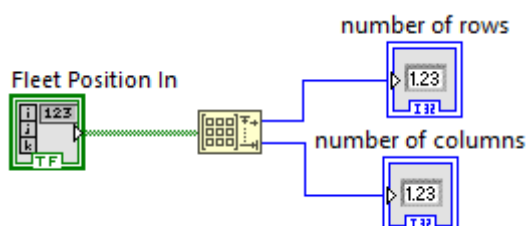
Nom	Icona	Entrades	Sortides
PlaceFleet		Fleet Position In Ship Length	Fleet Position Out

Funció

A partir d'una matriu booleana d'entrada i la mida d'un vaixell, retorna una matriu booleana que posiciona aleatòriament amb Trues la posició d'un vaixell de mida *Ship Length*. La posició del vaixell que s'afegeix mai se sobreposa a un vaixell ja existent.

Explicació

L'entrada *Fleet Position In* serà una matriu booleana de dimensions files-columnes. Per aquest apartat ens la podem imaginar plena de Falses. Amb la funció matrix size traurem el nombre de files i de columnes d'aquesta matriu.

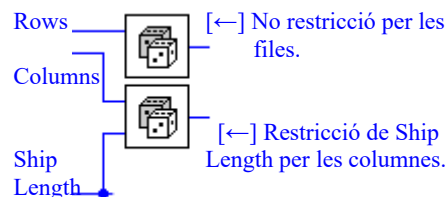


Per decidir si el vaixell que es construirà serà horitzontal o vertical, es genera un número aleatori entre $[0,1)$. Quan aquest número sigui 0,5 o major es farà un vaixell horitzontal. En cas contrari, el vaixell serà vertical. Això implica en 50% de probabilitats ja que a l'interval $[0, 0.5)$ hi ha els mateixos nombres que a $[0.5, 1)$.

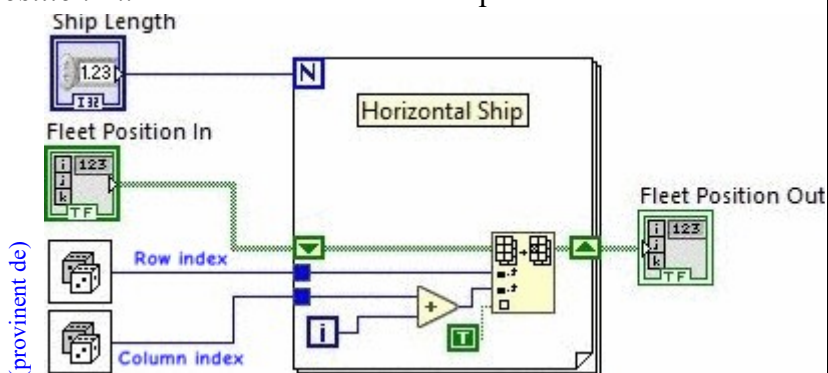
Per escollir la posició a partir de la qual ha de crear-se el vaixell, generarem un número aleatori amb la funció *RandomIndex*. En el cas que vulguem construir un vaixell horitzontal, la columna des d'on crear el vaixell no podrà ser qualsevol (com s'ha justificat a la funció *RandomIndex*). Però en canvi, la fila on es forma aquest vaixell pot ser qualsevol fila de la mida triada. I viceversa, al voler crear un vaixell vertical, podem escollir qualsevol columna però només unes files concretes segons la *Ship Length*.

Quan ha de fer el vaixell horitzontal, les files criden a la funció *RandomIndex* sense restricció de *Ship Length*. Però les columnes restringeixen els nombres aleatoris segons la mida del vaixell.

Quan el vaixell ha de ser vertical és al revés; les files estan restringides i les columnes no ho estan.



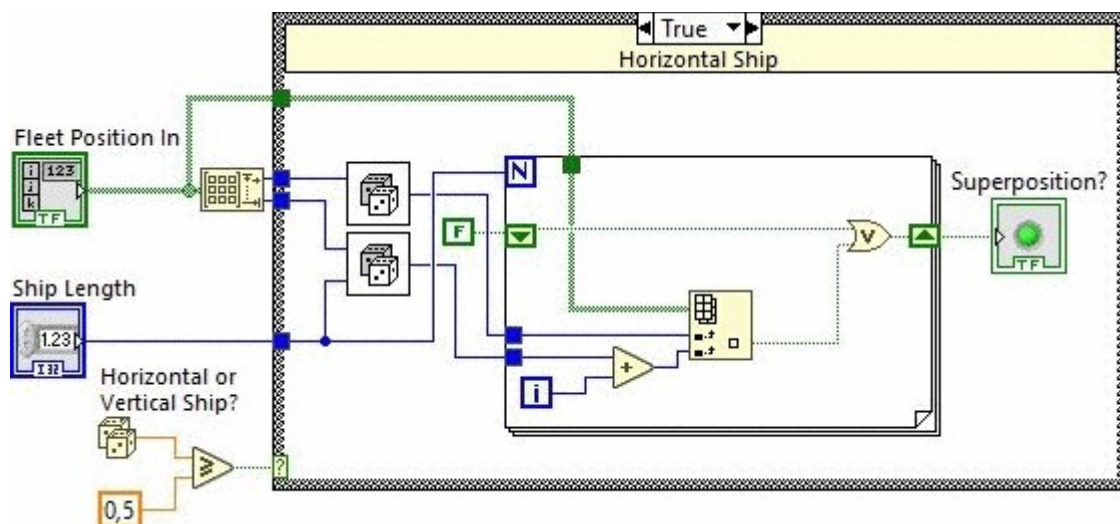
Les sortides *Index* del subVI *RandomIndex*, ens indicaran, doncs, la posició a partir de la qual ha de crear-se el vaixell. Ara, usarem la funció *Replace Array Subset* per situar Trues a la posició on ha d'anar el vaixell en la matriu *Fleet Position In*. Per fer-ho farem un bucle per cada dimensió del vaixell i situarem True en la posició indicades per les sortides *Index*; l'índex de la fila i l'índex de la columna. En la següent iteració, recuperarem la matriu amb la substitució feta anteriorment (amb un *Shift Register*) i li afegirem el True a la columna següent en el cas de vaixell horitzontal o fila següent en el cas del vertical.



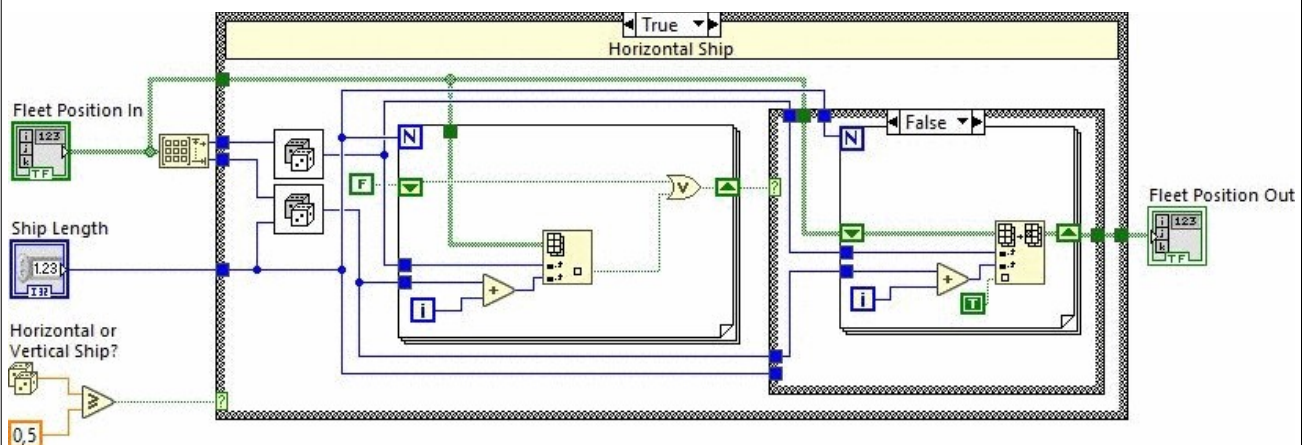
Detall

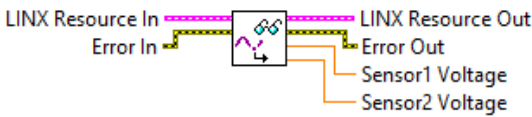
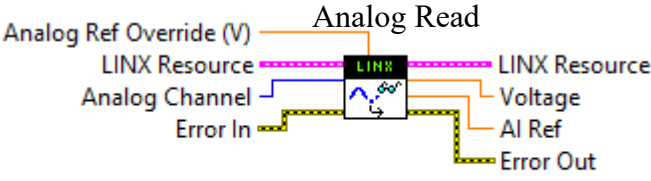
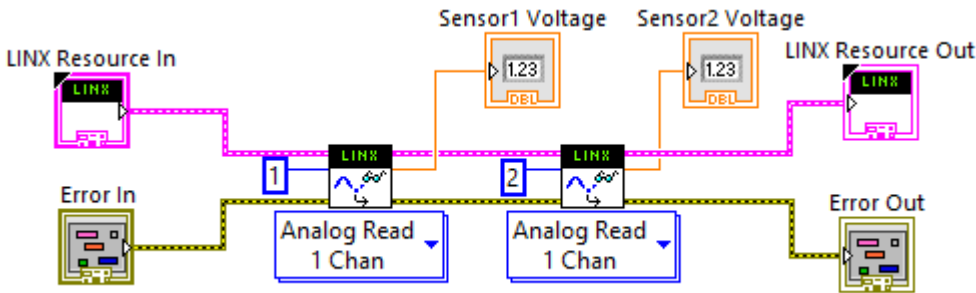
Fins aquí el nostre programa posiciona aleatòriament a la matriu *Fleet Position In* un vaixell de longitud *Ship Length* i retorna la matriu *Fleet Position Out* amb la posició d'aquest vaixell. D'inici la nostra matriu *Fleet Position In* serà una matriu plena de Falses (s'establirà des de fora del subVI), però un cop li haguem col·locat un vaixell ja no ho serà. Llavors, s'haurà de controlar si en alguna de les posicions on anirem a posicionar un segon vaixell coincideix en les posicions on ja hi ha un vaixell. Aquí es controlarà aquest problema. Es farà un cop creats els índexs aleatoris i abans d'accedir a la funció *Replace Array Subset*.

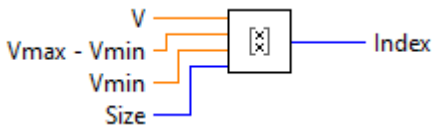
Amb *Index Array* es comprovarà si en alguna de posicions on el programa pretén crear un vaixell hi ha *True*. Es fa un bucle per tota la mida del vaixell i s'extreuen els valors de la matriu *Fleet Position In* corresponents als índex més les files o columnes conseqüents on es vol crear el vaixell en el cas vertical i horitzontal respectivament. Mentre aquesta sortida sigui *False*, la sortida de l'OR també serà *False*. Si en algun moment la sortida del *Index Array* és *True*, llavors l'OR serà *True* i a partir de llavors aquesta sortida sempre serà *True* ja que el *Shift Register* passarà *True* a l'entrada de l'OR. Per tant, *Superposition?* Serà *True* si el vaixell que es pretén construir coincideix amb un anterior vaixell. I *False* si no hi coincideix.



Finalment, es construirà una estructura *Case* controlada per aquesta sortida booleana que acabem de crear. En el cas *False*, es podrà crear sense problemes el vaixell utilitzant la funció *Replace Array Subset* explicada a l'apartat «Explicació». En el cas *True*, es tornarà a cridar el subVI perquè generi nous índexs ja que els que ha triat provoquen que el vaixell es construeixi sobre un altre. En definitiva el codi queda estructurat així:



Nom	Icona	Entrades	Sortides
ReadVoltage		LINX Resource In Error In	LINX Resource Out Error Out Sensor1 Voltage Sensor2 Voltage
Funció			
Llegir la placa Arduino UNO i retornar els valors dels voltatges mesurats en els pins A1 i A2.			
Explicació			
<p>Per a la lectura dels canals fem servir la funció Analog Read que el que fa és llegir els pins catalogats a la placa com «ANALOG IN». El pin que llegeix és aquell que indiquem amb un número a l'entrada <i>Analog Channel</i> i el voltatge que mesura és el que surt per <i>Voltage</i>.</p> <p>Read the value of the specified analog input channel(s).</p> 			
<p>Es col·loquen dos sensors. Un pel pin 1 i un pel pin 2 (Analog Channel=1 i = 2). La sortida del voltatge es col·loca al dos indicadors. En tot moment es passa la informació del LINX i l'Error.</p> 			

Nom	Icona	Entrades	Sortides
Matching_index		<p>V</p> <p>$V_{max} - V_{min}$</p> <p>V_{min}</p> <p>$Size$</p>	$Index$

Funció

Associar un índex al voltatge rebut dels sensors lumínics en el rang de $Size$.

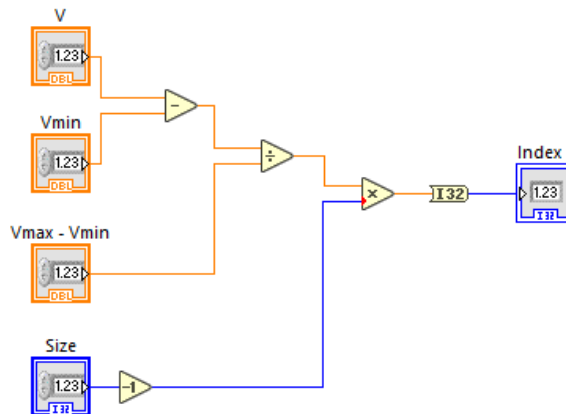
Explicació

Durant la fase de calibració es mesura el voltatge dels sensors lumínics en màxima il·luminació (V_{max}) i en mínima (V_{min}). Després, durant el joc, el jugador fa variar la il·luminació en els sensors de tal manera que aquests estan a voltatge V .

La idea d'aquest subVI és retornar l'índex al que correspon una certa il·luminació V dels sensors. Per fer-ho es parteix de la idea de que quan la il·luminació sigui mínima $V = V_{min}$, l'índex sigui el més baix. I quan sigui màxima $V = V_{max}$, l'índex sigui el més gran. La relació d'aquest tipus s'aconsegueix amb la següent expressió: $\frac{V - V_{min}}{V_{max} - V_{min}}$ de tal manera que és 1 quan $V = V_{max}$ i és 0 quan $V = V_{min}$. Per escalar aquest fet des de 0 fins a la mida d'aquell eix, ho multiplicarem pel valor de l'índex més gran, és a dir, $Size - 1$. Per tant, l'expressió serà $\left(\frac{V - V_{min}}{V_{max} - V_{min}} \right) \cdot (Size - 1)$

Això en programació en LabVIEW és:

Fins aquí podríem pensar que ja ho tindríem tot. Que aquesta relació que ens transforma perfectament voltatges V en índexs. Però hi ha un detall que podria causar un error en aquesta transformació. Mirem-ho tot seguit.



Detall

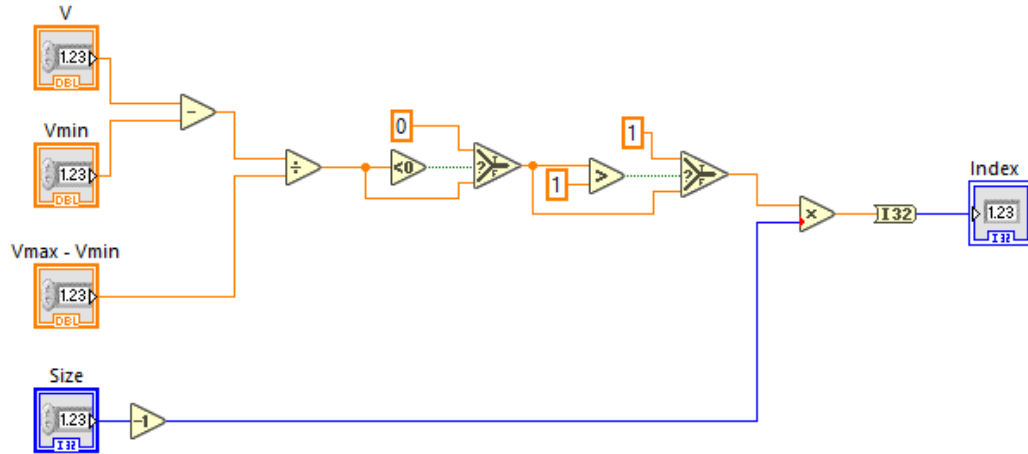
En l'expressió i programació anterior, què passaria si, pel que fos, el voltatge durant el joc V fos menor al voltatge mínim calibrat V_{min} ? Això implicaria un índex negatiu. I si fos major al voltatge màxim calibrat V_{max} ? Implicaria un índex major a la mida de la matriu. Aquest fet és una cosa que el nostre programa ha de controlar per evitar errors.


Per fer-ho, usem la eina *Select*. Aquesta eina té dues entrades numèriques, una booleana i una sortida numèrica. El seu funcionament és el següent: Si la variable booleana és True, la variable numèrica de sortida és la de dalt. Si en canvi és False, la variable numèrica de sortida és la de baix.



Ho construïm de la següent manera i amb aquest sentit: Si el valor de la relació $\frac{V - V_{min}}{V_{max} - V_{min}}$ és

menor que zero, vol dir que la il·luminació és menor a la mínima calibrada, que es buscava l'índex mínim (el 0). En cas que no sigui menor a zero, seguim. Si el valor de la relació $\frac{V - V_{min}}{V_{max} - V_{min}}$ és major a 1, vol dir que la il·luminació és major a la màxima calibrada, que es buscava assolir l'índex més gran i s'ha sobrepassat. Li associem 1 en aquest cas. Sinó compleix cap de les dues condicions vol dir que la dada no sobrepassa els límits [0,1] i passa sense problemes a escalar-se amb la mida.



Nom	Icona	Entrades	Sortides
CheckShot	 <p>Column Index Row Index Fleet Size Hits In Fleet position Battlefield In</p> <p>Sound Index Victory? Hits Out Battlefield Out</p>	<p>Column Index Row Index Fleet Size Hits In Fleet position Battlefield In</p>	<p>Sound Index Victory? Hits Out Battlefield Out</p>

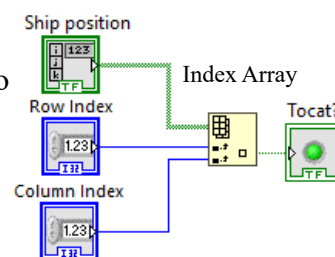
Funció

La seva funció principal és comprovar si al lloc on s'ha disparat és tocat, aigua o ja disparat. Però també fa altres funcions:

- Comprovar si s'ha enfonsat el vaixell. Retorna True o False a la sortida *Victory?*.
- Retornar el *Sound Index* segons si ja s'ha disparat (0), si és aigua (1) o si és tocat (2).
- Retornar la matriu de batalla de sortida indicant aigua (O) o tocat (X).

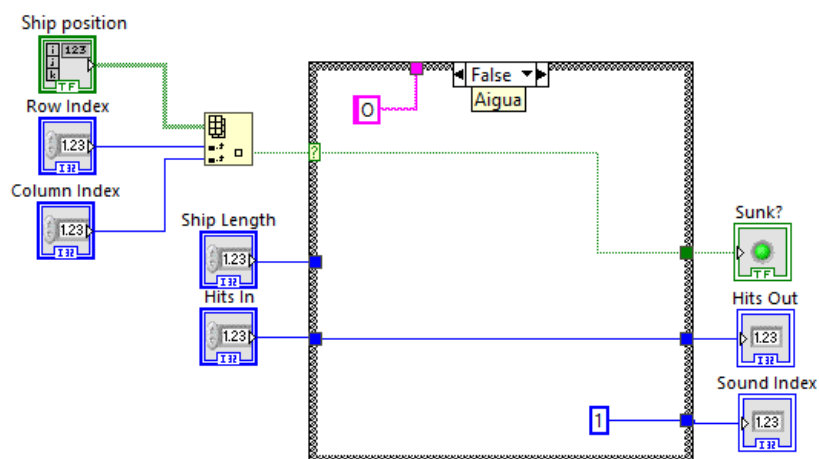
Explicació

La idea és comprovar si la matriu *Fleet position* té o no alguna posició pertanyent al vaixell en la posició *Row Index*, *Column Index*. Per fer-ho s'extreu de la matriu *Fleet position* el valor d'aquella posició amb el *Index Array*. Quan sigui True voldrà dir que hi ha vaixell, serà tocat. Quan sigui False voldrà dir que no hi haurà vaixell, serà aigua.

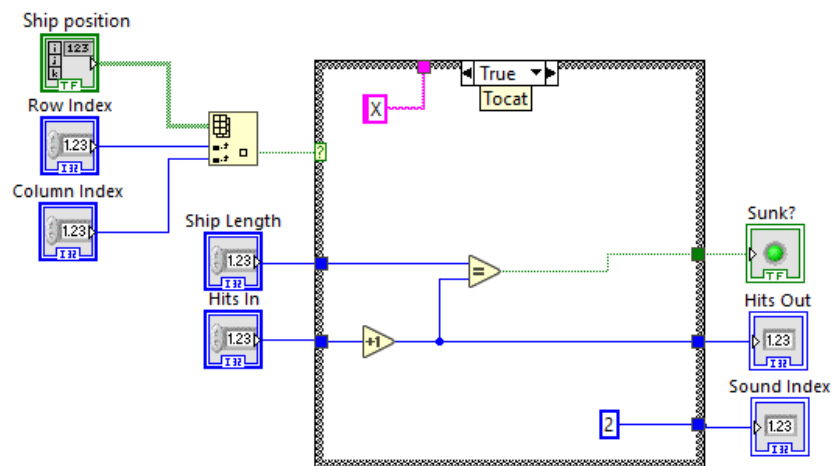


Aquí veiem que distingirem els dos casos amb l'estructura *Case*. El cas tocat i el cas aigua.

- En el cas aigua, no haurem tocat el vaixell. Per això Hits In = Hits Out. També serà impossible que en un tret d'aigua enfonsem el vaixell. Per tant podem connectar el False a *Victory?* Finalment retornem «1» a *Sound Index* i «O» a la *Battlefield* (tot seguit veurem com).



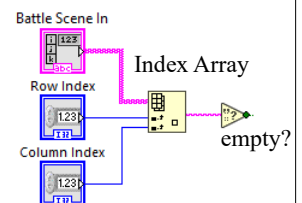
- En el cas tocat, sumarem 1 al nombre de disparats tocats que portàvem. Llavors comprovarem si el nombre de tocats que portem és igual a la mida del vaixell. Mentre no ho sigui, el vaixell encara no estarà enfonsat *Victory?*=False. Però quan el nombre de tocats sigui igual a la mida del vaixell, el vaixell estarà enfonsat *Victory?*=True. Finalment retornem «2» a *Sound Index* i «X» a la *Battlefield* (tot seguit veiem com).



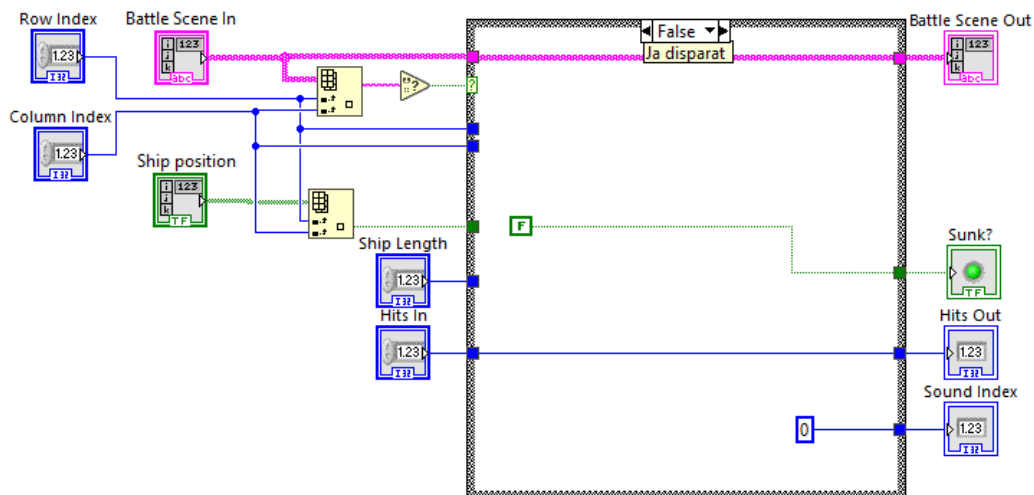
Fins aquí ja tenim la base feta que ens comprova si és Tocat o si és Aigua, compta el nombre de tocats que portem i detecta si el vaixell s'enfonsa. Tot i així, ens queda considerar una opció. La opció de que en el lloc assenyalat ja hi hem disparat anteriorment.

Detall

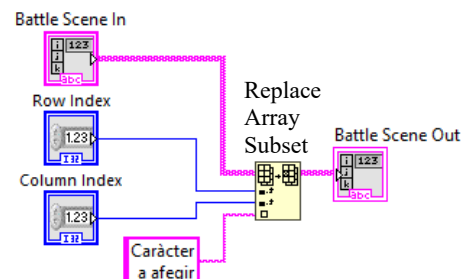
Per fer-ho entrarem una matriu de caràcters (*Battlefield In*) que en primera instància estarà tota buida. Comprovarem si aquesta entrada té el caràcter situat a la fila i columna que apuntem (*Row Index* i *Column Index*) buit.



- En cas negatiu, voldrà dir que ja hi hem disparat i per tant no hem de sumar cap *Hit* ni haurem enfonsat el vaixell (*Hits In* = *Hits Out*, *Victory?* = *False*). També retornarem la matriu de batalla sense cap canvi i indicarem el soroll d'error «0».



- En cas afirmatiu, es realitzarà la verificació de si és Aigua o Tocat com hem programat anteriorment i s'afegirà a la *Battlefield* el caràcter provinent d'aquesta comprovació. El caràcter s'afegeix amb la funció *Replace Array Subset*.



En definitiva queda:

