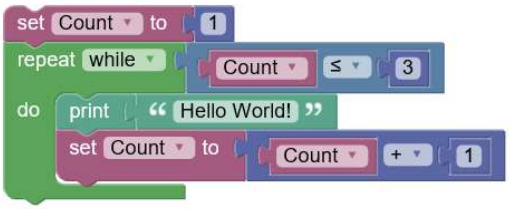
Adaptive Strategic Linked MOVA using Blockly

Craig Andrew Cameron
Principal Traffic Engineer
Siemens Traffic Solutions
Craig.Cameron@siemens.com
Tel: 07808824483

What is Blockly?

Blockly is library that adds a visual code editor to web and mobile apps. The Blockly editor typically runs in a web browser and uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops, and more. It allows users to apply programming principles without having to worry about syntax or the intimidation of a blinking cursor on the command line. Blockly uses visual blocks that link together to make writing code easier, and can generate code in JavaScript or Dart, Python, or PHP. It can also be customised to generate code in any textual programming language.



Both the Siemens Stratos Outstation (SOS) and the ST950 controller have a built in Blockly editor which allow users to write scripts; this allows conditioning to be written and run without the need to change the controller configuration. These scripts can be used to send and receive bits on the system IO which can manipulate other applications reading those interfaces (Controller applications, MOVA, UTMC etc.)s, in a similar way to controller special conditioning. The Blockly editor on the Siemens SOS and ST950 write the code generated to Javascript.

Initial Problem

The Sports Centre roundabout in Bracknell is controlled on two approaches by simple two stage streams on a single controller, and a further two unsignalized approaches. MOVA control had been implemented on both streams with simple linking programmed into the controllers special conditioning to allow each stream to be held off against the other while the approaches ran. This effectively holds the circulatory on the opposite controller while the main approach to the stream runs.

There was a delay between the approach on one controller finishing and the hold on the circulatory of the other stream from starting; this was fine when queues were present on the internal reservoirs but was seen to be incredibly inefficient when the internal circulatories were clear.

It was also noted that when one approach was extremely busy, but the opposing approach was not, while the greens on the busy stream were often curtailed under oversaturation condition in the MOVA kernel the opposing stream could continue optimising for vehicles after saturation flow had ended. This is a normal problem with MOVA linked sites.

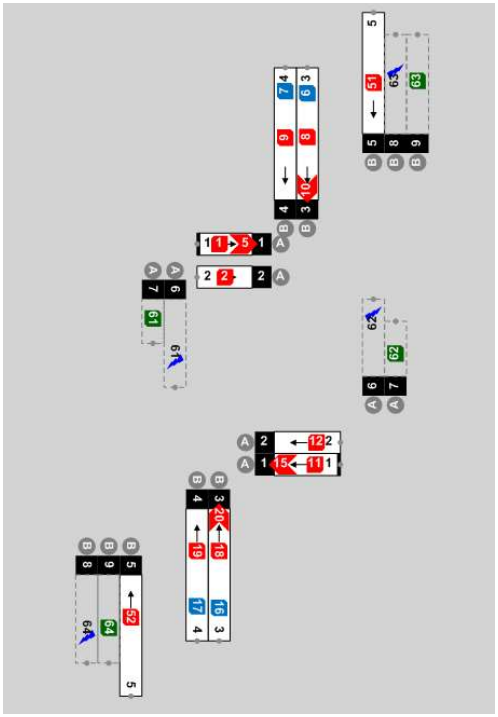
This effectively means that the oversaturated approach can often receive a shorter green than the undersaturated approach on the opposing MOVA stream.

Solution

To fix the problems identified above a new linking strategy has been written with the conditioning written on the Stratos Outstation in the blockly visual editor. This allowed us to keep the original linking on the controller if we wanted to switch back to it, and the controller configurations did not need to be changed. This also allows for changes to the linking code without loading new configurations onto the controller.

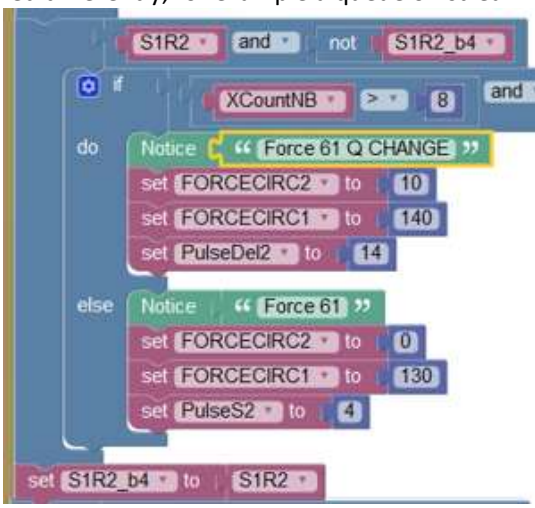
The strategy for linking is similar to before but with some big alterations to allow it to be more adaptive, the basic way in which this works is as follows:

- When stream 2 stage 2 (A322 NB) loses ROW the hold on stream 1 stage 1 (Circulatory) is removed to allow a move to stream 1 stage 2 (A322 SB); stream 1 is forced to move to stage 2.
    - Logged as “Force 62”
    - The Hold detector output is applied to stream 2 stage 1 to keep the stream on the circulatory.
    - The code waits until stream 1 stage 2 loses ROW.
- When stream 1 stage 2 (A322 SB) loses ROW the hold on stream 2 stage 1 (Circulatory) is removed to allow a move to stream 2 stage 2 (A322 NB); stream 2 is forced to move to stage 2.
    - Logged as “Force 61”
    - The Hold detector output is then applied to stream 1 stage 2 to keep the stream on the circulatory.
    - The code waits until stream 2 stage 1 loses ROW.



The above linking works when the junction has no queues on the internal reservoirs, with no delay between a controlling stage ending ROW and the new controlling stage (on the opposite stream) gaining ROW. This reduces the cycle time when the queueing on the circulatory is not causing an issue.

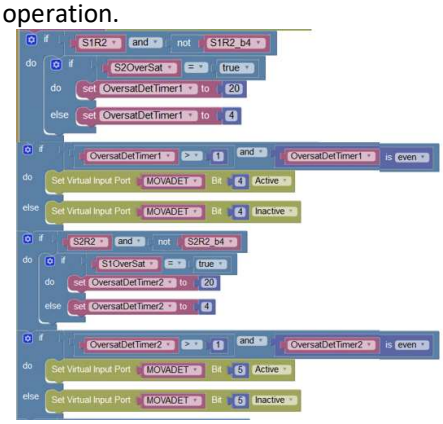
When queues are detected on the X detectors for the gyratory on one stream, the opposite stream is controlled differently, for example a queue on stream 1 circulatory:

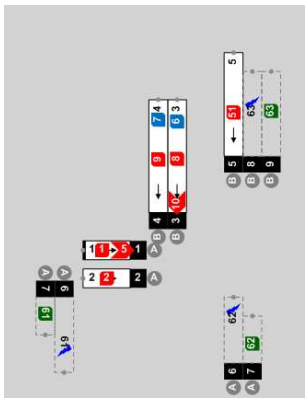


- X detector on stream 1 lane 2 covered for more than 4 seconds when stage 1 loses ROW
    - Logged as “Force 61 Q Change”
    - The hold is applied to circulatory at stream 1
    - The hold on the circulatory for stream 2 is kept on for a further 5 seconds to allow the queue to dissipate
    - The force to move to stream 2 stage 2 is also delayed

This method then controls the offset between the streams, preventing the platoon from one stream meeting a queue on the next. This control is independent to each direction, meaning that a queue on one circulatory does not trigger the delay action on the other, keeping the cycle time down as much as we can under these conditions; this check is done every cycle to allow as much adaption as possible.
If one stream is oversaturated while its opposite isn’t, the opposite stream will allow optimisation while the oversaturated stream does not; this causes the busier stream to have less effective green time than the other, a problem often experienced with linked MOVA sites. To prevent this the oversaturation flags on the approach lanes on both streams are read by the code:

When one streams approach lanes (lanes 3 and 4) are oversaturated pulses are sent to the opposite stream to a dummy lane (5) which is pulsed at the end of green forcing it into oversaturation. This passes the oversaturation to the opposing stream ensuring they are both in the same mode of operation.





	Lane	Auto Set Critical Vehicle Count	Detector For Oversaturation Checking [XOSAT]	Critical Vehicle Count [OSATCC] (veh)	Critical Vehicle Count Compact [OSATCX] (veh)	Critical Vehicle Time [OSATTM]
▶	1	☐	X detector	20	20	1.0
	2	☐	X detector	20	20	1.0
	3	☑	IN detector	9	5	
	4	☑	IN detector	9	5	
	5	☐	X detector	4	4	10.0

When there are no queues on the circulatories and there is no oversaturation on the approaches we can be less stringent with the linking allowing even lower cycle times and some cross over between the streams. To achieve this the code allows a stream to move to stage 2 (by removing the hold on stage 1) when its opposite stream is likely to end ROW on stage 2 soon, while still allowing optimisation on the circulatory. The code works by reading the end of saturation flags on the links:

The "endForc" timers are then set to give a window time for the MOVA stream to make its own decision to change, at this point both of the main approaches to the roundabout can be at green allowing free flowing traffic to be optimised by the two separate MOVA streams.

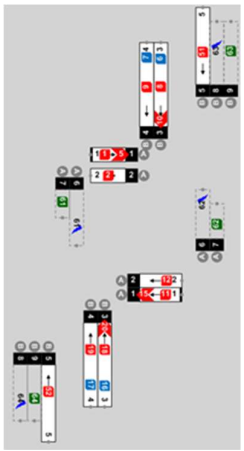
During the night the same linking is applied but its effect on MOVA is lessened with the SD codes for the priority links dropped to only allow holds instead of immediate moves. The maximum hold timers are also dropped to much lower values to allow MOVA more flexibility to make decisions, while still allowing platoons to get through the circulatories when they form.

	Stage	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7	Link 8	Link 9
▶	1	1	1				1	1		
	2			1	1	0			1	1

Linking – Day Time

	Stage	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6	Link 7	Link 8	Link 9
▶	1	1	1				0	1		
	2			1	1	0			0	0

Linking – Night Time



Conclusion

The use of Blockly as a conditioning language has several advantages over the controllers own special conditioning such as the ease in which the code can be changed and the amount of information available to the code without having to pass this information with other coding. It's also possible to write to the system log of the platform allowing easy testing in real time as well as allowing for fault finding and validation.

The solution provided above certainly showed how quickly and easily code could be written and adapted to provide adaptive linking to junctions; certainly the ease in which the code could be changed allowed for the end solution to be much more mature than using traditional controller conditioning (with the reloading of configurations required).

However, the Blockly editor provided on the Stratos Outstation and ST950 controller are currently undocumented and unsupported, and as such are not for general use. If this is a feature that you feel could be useful then please come and speak to us on our stand.