ROBOESL PROJECT

ROBOTICS-BASED LEARNING INTERVENTIONS FOR PREVENTING SCHOOL FAILURE AND EARLY SCHOOL LEAVING

Erasmus+ 2015-1-IT02-KA201-015141

Output 1: Curricula for 10 exemplary interdisciplinary robotics projects

Curriculum 10: Give precedence

Lead Partner: UNIPD (IT)

Authors: Michele Moro, Francesca Agatolio, Emanuele Menegatti (UNIPD)

Contributions

Dimitris Alimisis (EDUMOTIVA[1]) (Chapters 2 and 3)

Linda Daniela (University of Latvia) (Chapter 5)

Circulation: Public

Version: Final

Stage:

Date: March 31, 2017

[1] EDUMOTIVA stands for 'European Lab for Educational Technology

Declaration

This report has been prepared in the context of the ROBOESL project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Abstract

This document contains the description of the Curriculum n. 10, entitled ‘Give precedence’ which is part of the Intellectual Output 1 (Curricula for 10 exemplary interdisciplinary robotics projects) developed and tested in teacher training courses within the context of the ERASMUS+ ROBOESL project. Road education, precedence, traffic lights: stimulating subjects for this Curriculum which introduce the implementation of well known rules but also affords the delicate problem to synchronize two similar but independent robots.

Chapter 1: **Short description and scenario (O1.1)**

(source: maxcdn.icons8.com)

A couple of robots meet each other at a crossing point: let's impose some precedence rule.

1.1 The scenario

In this curriculum we define two slightly different scenarios: in the first we are going to use an ultrasonic sensor to take an appropriate decision in order to comply with the assigned precedence rule; in the second scenario the two robots synchronize through message exchange to take their own decision. Thus the second scenario introduces a new challenge which leads to several interesting observations. In order to simplify the layout, the crossing point is easily made up: two crossing stripes of black tape (useful to support line following) plus some short stripes of tape of different color that represent the beginning and the end of the intersection in both directions (fig. 1, red stripes). The rule to be established can be the usual in countries where you drive on the right, that is, when two cars reach an intersection almost at the same time, the one on the left gives way to the one on the right. Another precedence rule could be that the car which enters first the intersection gets precedence: this is the usual rule applied in a roundabout.

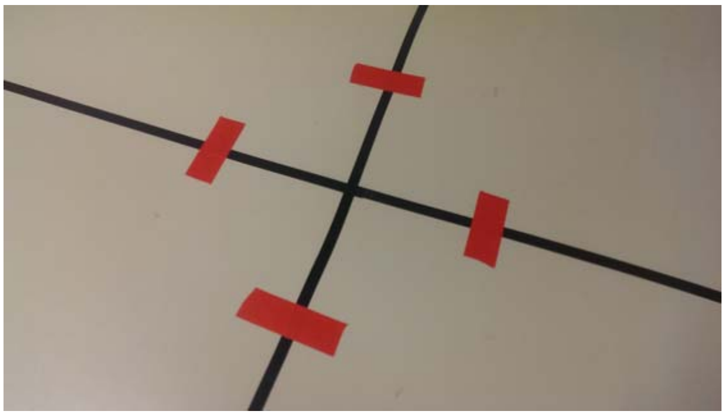
Figure 1. The intersection

1.2 Connections with subjects

1. Road education: how to comply with precedence rules for our own and others' safety.
2. Computer science: the mutual exclusion principle, synchronization, message exchange.
3. Physics: kinematics, motion profile.
4. Technology/Engineering: the feedback control principle.

Chapter 2: **Pedagogical objectives (O1.2)**

2.1 General objectives

- To show how important is to comply with driving rules programming the behavior of robots.
- To provide students with a stepwise approach for a step by step acquisition of technical skills in using robotic technologies (hardware and software) building on existing knowledge and skills.
- To offer the robotics benefits for all children, especially those at risk of school failure or early school leaving, putting robotics in real- life context.
- To engage students in STEM related subjects through interaction with the robotics technologies.
- To support self-directed action allowing learners to learn independently.
- To engage students in robotic constructions and problem solving through an interdisciplinary scenario that reflects aspects of real-life problems and situations.
- To align the robotics project to learners' needs and interests through tasks that derive from the initial activity but introduce new levels of complexity and difficulty.

2.2 Specific objectives

More specifically, upon successful implementation of the activities described in this curriculum students will achieve the following objectives:

- Build and program a robot that can detect another robot using the distance sensor
- Instruct robots to give precedence to the right
- Synchronize the robots' behavior through message exchange
- Simulate with their robots what happens in a real intersection where traffic lights impose a time-based precedence rule

Chapter 3: **Suggestions for learning methodologies (O1.3)**

This curriculum follows the methodology introduced in the previous curricula. However, it is tailored to the scenario introduced in the current curriculum. A special worksheet has been designed as a reference and supporting tool for the students. The students are encouraged to work in groups. The teacher acts as a scaffold and facilitator of the learning process. S/he provides feedback without revealing solutions and probing students through key questions to overcome emerging problems and difficulties.

The activity starts with the delivery of the scenario to the students.

The worksheet brings up the situation when vehicles often come into conflict with other vehicles because their intended courses of travel intersect, and thus interfere with each other's routes. The teacher asks questions for the students to discuss in their groups such as: who has the right to go first ? who has the right to use the conflicting part of the road and who has to wait until the other does so?

After a first discussion in groups, one student from each group presents their answers to the plenary. The teacher concludes this discussion summarizing the principles that establish the "right of way" or "priority".

Then students are asked to make a mock-up representing an intersection and two tribots representing the vehicles moving on this intersection.

Then students are challenged with the problems: How can you make your robots to give “precedence to the right”? what sensors and programming solutions are needed? Students work in groups designing and experimenting with different solutions. During students’ experimentations with this task, the teacher can advise the students, if needed, to mount a distance sensor to detect the other robot. In the end students’ groups present in the plenary of the class their solutions and reflect on them with critical mind. Feedback from peers and teacher is provided with a constructive spirit.

The next task is introduced with the question: how can the two vehicles communicate in order to make priority arrangements? The messaging block is introduced by the teacher. The students are invited to experiment with the messaging action between the two robots. After this familiarization task, students are challenged to program their robots to assign precedence alternatively to one of the two robots independently of their relative position and the time when they reach the beginning of the intersection. Students work as usual in groups experimenting with their own solutions. The teacher helps discretely when necessary introducing ideas and suggestions (for instance: make each robot to authorize the entrance of the other one just after having exited the intersection). A plenary presentation and discussion of students’ solutions follows. The next challenge for students is introduced with the question: can you give precedence to the robot which reaches first the intersection? Can you assure that while this robot is crossing the intersection, the entrance of the other robot is blocked until the first one clears the intersection? The students are allowed to try out their solutions and finally to present them in the class.

Finally, students are challenged to simulate with their robots what happens in a real intersection where traffic lights impose a time-based precedence rule: robots cross the intersection only when light is green. The EV3 brick lights are suggested by the teacher as traffic lights. Students are advised to choose proper time interval between the three different traffic lights. Once again students design and experiment with their own solutions and finally present them in the class.

A. The role of the students

Students first discuss a scenario through a free dialogue in their group and after that they devise an action plan to realise it. They work in groups following their ideas and the discrete feedback they receive from the teacher. Students may extend their initial scenario devising further stories to play with. First, they find solutions making their own experimentations. Then they are supported to find additional solutions and realise further ideas. The final creations of the groups are presented in the class, are discussed and evaluated with students reflecting with critical mind on their work, expressing their views and recording their experiences in a diary or questionnaire.

B. The role of the teacher

The teacher in this constructivist learning framework acts as an organizer, coordinator and facilitator of learning for students. S/he organizes the learning environment, raises the task for making robotics theatre through a worksheet, introduces software tools when necessary for students' work, discreetly helps where and when necessary, encourages students to work with creativity, imagination and independence and finally organizes the presentation and evaluation of the activity in the plenary of the class.

Chapter 4: **Technical guidelines (O1.4)**

4.1 Building instructions

For this curriculum we implement a line following, object detection and message send-receive, therefore the usual *tribot* structure, equipped with an ultrasonic sensor in its front and a color sensor oriented towards the floor, is still suitable. But for this curriculum we need two similar robots with the same structure, and we assume they are named EV3A and EV3B.

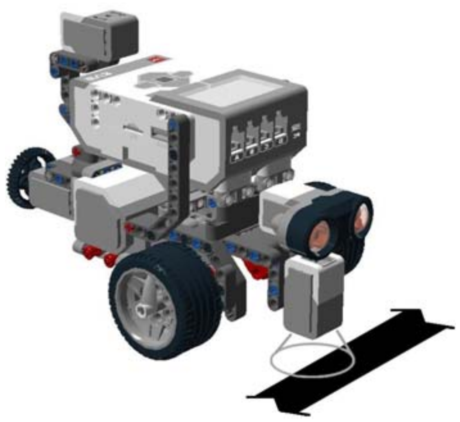

Figure 2. The tribot with US and color sensors

4.2 Illustrative solution

In the first scenario we concentrate our attention to impose the ‘precedence to right’ rule making each robot of the couple detect if the other one is on its right or not. So we imagine to put one robot over one of the intersecting black stripes, out of the intersection, and the other robot over the orthogonal stripe (fig. 3).

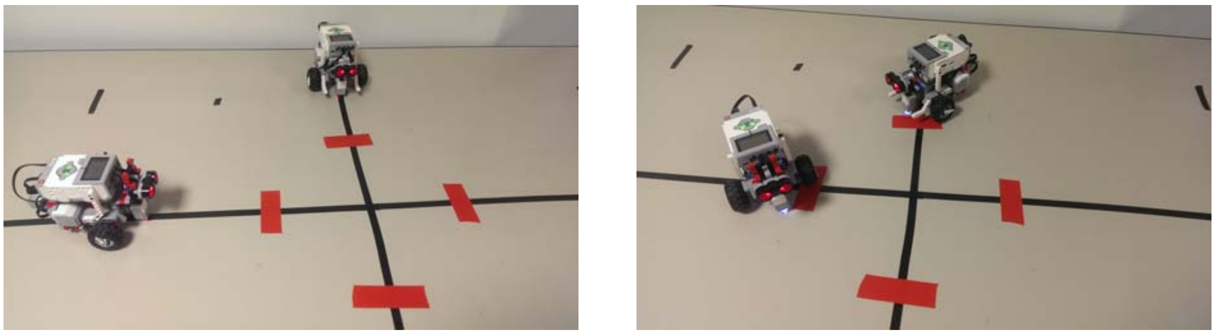

Figure 3. First scenario, right precedence

Taking care to put the two robots at about the same distance from the first red stripe crossing their line, and to start the program on the two robots at about the same instant, they also reach the intersection more or less at the same moment. At this point, each robot stops and turn a bit to the right. Tuning opportunely the angle of this rotation, one and only one of the robots detects the other one on its right and gives way to it. The solution is rather simple: the straight motion is stopped when the robot detects the red stripe, then it turns and waits until nothing is detected within a certain distance D (= 35 cm.). When this happens, the robot turns back, waits for 2 s and check again if there is anything within the D distance, that is occupying the intersection. When the way in front of it appears clear, it crosses the intersection; this is performed through the sequence of some commands: the enlightening of the brick yellow light (to show that the robots entered the intersection), an unlimited straight motion, a wait for the end of the first red stripe (that is simply waiting for a color change), a wait for the second red stripe signaling the end of the intersection, the switching off of the light, and a final straight motion for a while out of the intersection. It is worth to point out that the code is exactly the same for both the two robots (fig. 4).

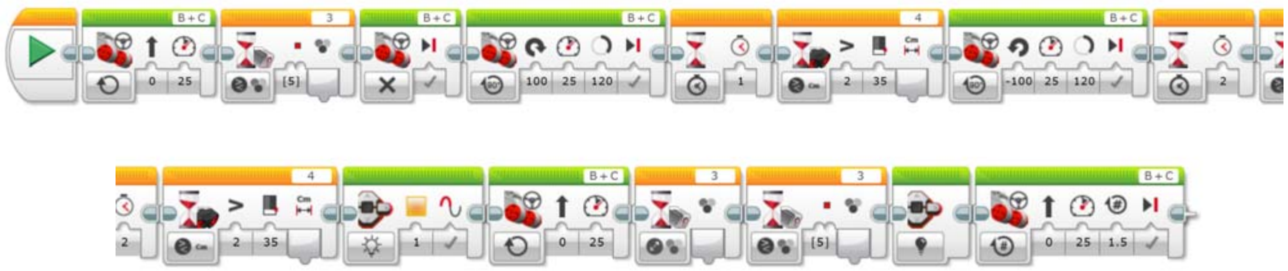

Figure 4. Simple right precedence

The first variant of the second scenario is again rather simple: we assume in this case that one robot gets always precedence to the other (fig. 5). Named EV3A this robot of higher precedence, for example an ambulance in emergency, apart from the enlightening of the brick light, it simply sends a message of title ‘rel’ (‘release’; notice that the content of the message is not significant) when it exits the intersection (second detection of a red stripe) (fig. 6). For this variant the code of the other robot (EV3B) is necessarily different: before entering the intersection, the robot must wait for the ‘rel’ message (see the `wait for the Message – Update – Text` command) (fig. 7).

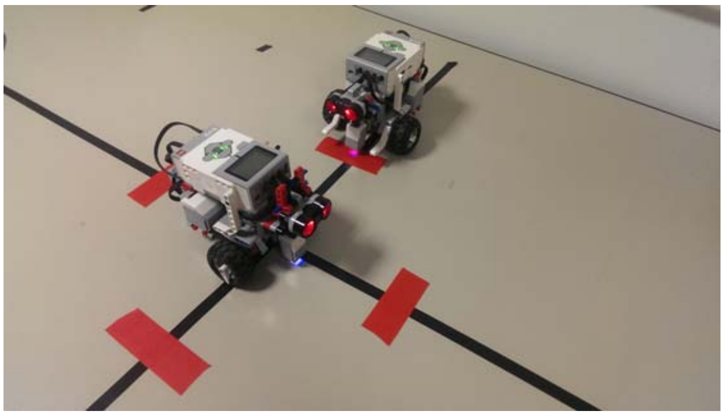

Figure 5. EV3A gets precedence on EV3B

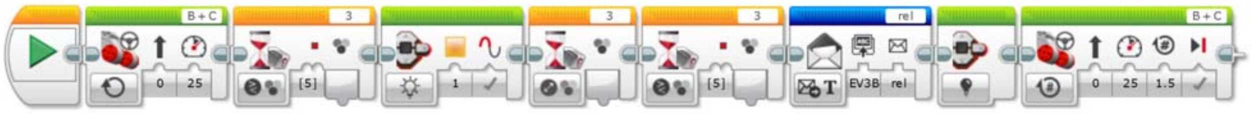

Figure 6. Fixed higher priority (robot EV3A)

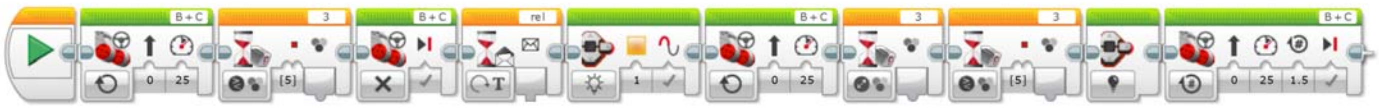

Figure 7. Fixed lower priority (robot EV3B)

4.3 Implementation suggestions

The only attention to put in these scenarios is to prepare the intersection with black and red tapes as explained above. For symmetry we suggest to put the couple of red stripes on one of the two black stripes at the relative distance more or less equal to the one between the other couple of red stripes, so that the intersection appears more or less as a square.

4.4 Extensions and variants

4.4.1 Variant a: First scenario with line following [easy/medium]

In this variant, we add a proportional feedback line following instead of a simple straight motion. This is why the scenario includes two types of stripes, one black for line following, which uses the reflected light option in the color sensor, and one red to delimit the intersection, detected using the color option of the sensor. The line following is performed by the robot when approaching the intersection and within it: these two phases are controlled by the two similar loops of the code shown in fig. 8. Two variables, `level` and `factor`, represent respectively the set point and the factor of the proportional feedback control as explained in curriculum 9. Notice that, as already mentioned, the color sensor is used in the loop alternatively to measure the reflected light and the color.

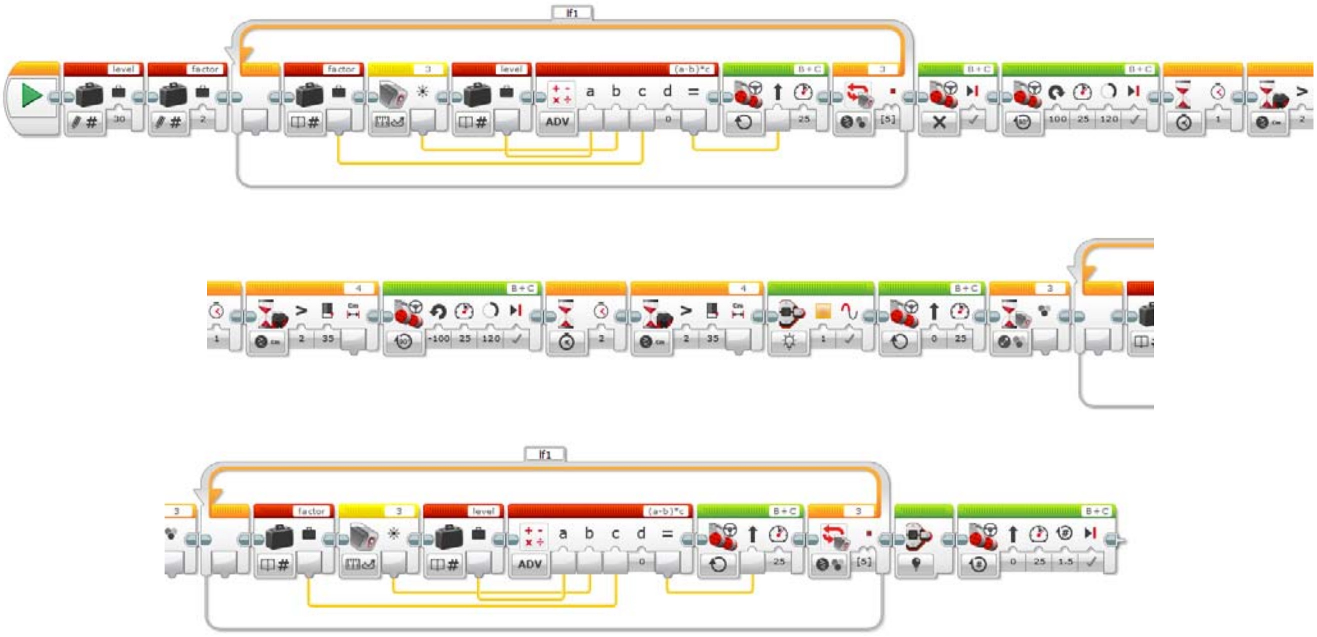

Figure 8. Right precedence with proportional line following

4.4.2 Variant b: Second scenario with line following as a customized block [easy/medium]

In this variant we add a line following in the second scenario, similarly as in the variant a) but here as a customized block. Therefore we first use the *My Block Builder* tool to define a new `plinef` block containing the loop we added in variant a) (fig. 9). Notice that this definition is strictly related to our scenarios because it is expected that the motion ends when the color sensor detects a red stripe. Once defined the block, we use it to simply add the line following function to the second scenario, that is to both the high and low priority codes (fig. 10 and fig. 11). To show better which one gets precedence, the intersection is repeatedly travelled by the two robots: so the code is inserted in a loop with a full inversion of the robot at the end of each cycle. The value of turning degrees is experimentally established to facilitate the realignment on the black stripe after the inversion. Remember to insert the initialization of the two variables `level` and `factor` at the beginning of the main program.

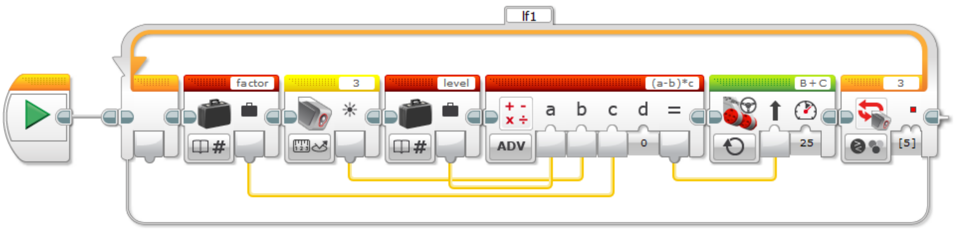

Figure 9. The plinef customized block

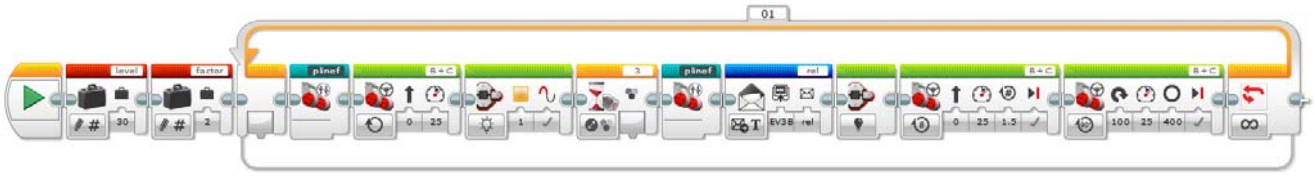

Figure 10. Fixed higher priority with proportional line following (robot EV3A)

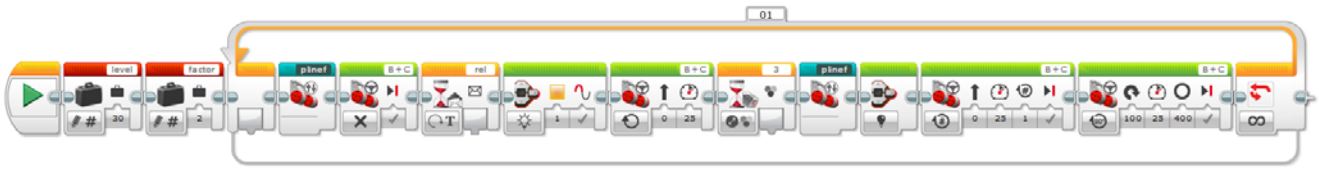

Figure 11. Fixed lower priority with proportional line following (robot EV3B)

4.4.3 Variant c: Alternative priority [easy/medium]

In the next variant we want to assign precedence alternatively to one of the two robots. This means that, starting from EV3A, and independently of their relative position and the time when they reach the beginning of the intersection, they can enter it alternatively. One simple method to force this behavior is to make each robot authorize the entrance of the other one just after having exited the intersection. This authorization is provided sending a message of type *go* to the other robot, so that each robot must wait for a message update of this type before entering the intersection. The first authorization is sent by EV3B to EV3A before executing the main loop to start the precedence alternation (fig. 12 and fig. 13).

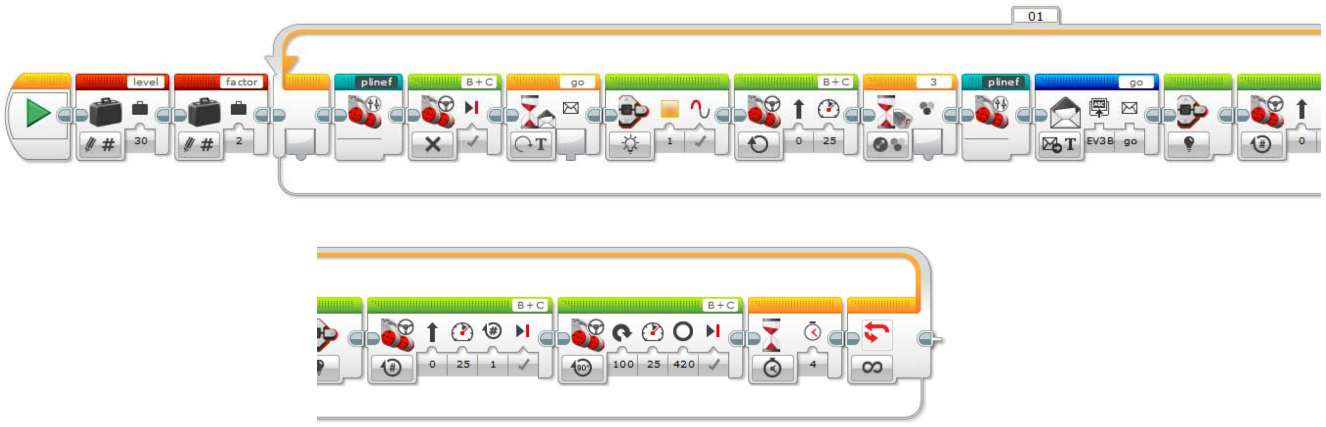

Figure 12. Alternative precedence (robot EV3A)

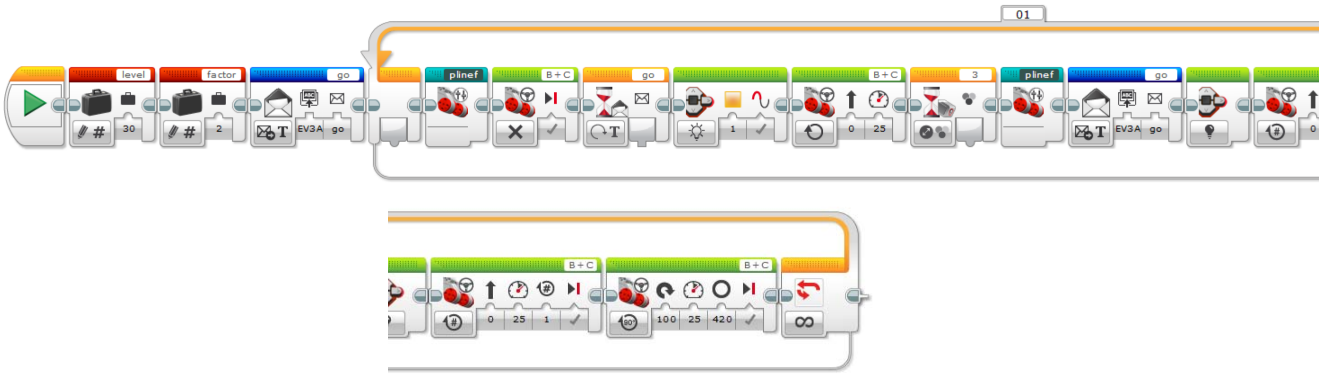

Figure 13. Alternative precedence (robot EV3B)

4.4.4 Variant d: Mutual exclusive intersection [medium]

Let's now consider to give precedence to the first of the two robots which reaches the intersection: we must assure that this robot is permitted to enter but this blocks the entrance of the other robot until the first one clears the intersection. This imposed situation, that is one resource can be used by only one actor at a time (for us the resource is the intersection), is usually called '**mutual exclusion**'. This means that only one actor can execute a particular section of its code but never concurrently with a corresponding section of another actor. For this reason these sections are called '**critical sections**'. In our case the critical section is, for each robot, the section of code executed when the robot is inside the intersection.

In order to force the behavior described above, the robot must wait for an authorization: this must be provided initially when the intersection has not yet crossed and every time one robot exits the intersection. In this variant we exploit again a message exchange to signal the other robot when it is permitted to cross. More in detail, each robot sends to the other a message of type *go* but it changes the content of this message, 'go' to authorize, 'stop' to remove the authorization. At the beginning the first message is sent by both robots. After waiting the authorizing message, a robot sends a 'stop' to the other and again a 'go' when it has crossed the intersection (fig. 14 and fig. 15). For adding a degree of variability, the final wait lasts a time randomly set to an amount of seconds with fraction. Unfortunately, due to not avoidable delays between successive transmissions, if the two robots reach the intersection almost at the same time, both read a 'go' message in their mailbox and both send concurrently a 'stop' message and enter the intersection: the mutual exclusion is not guaranteed in this case. This eventuality is rather improbable but clearly not impossible. The next variant provides a safer solution.

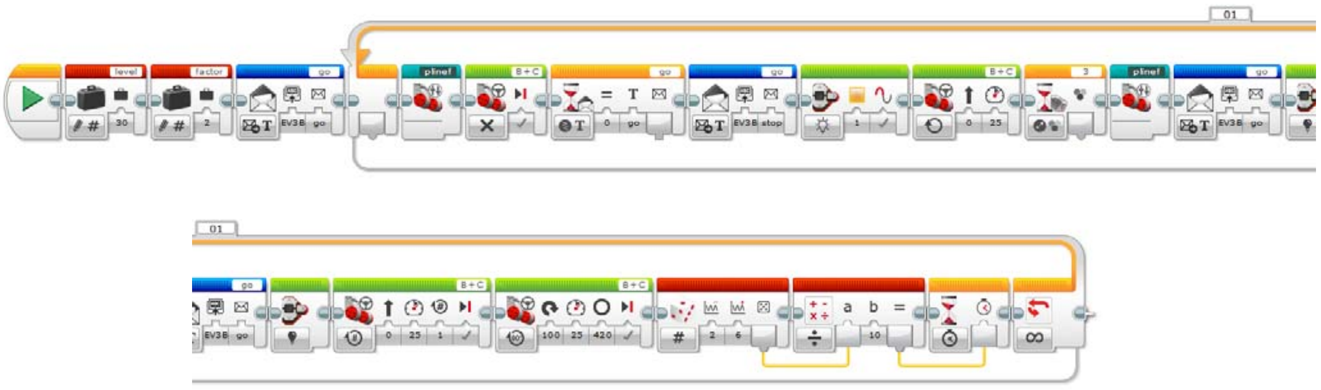

Figure 14. Precedence to the first, code for EV3A

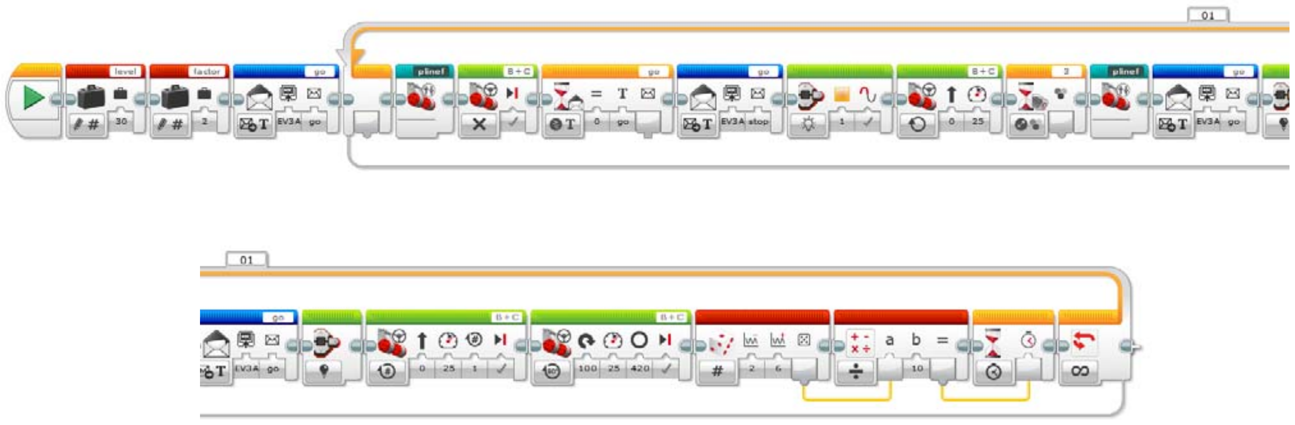

Figure 15. Precedence to the first, code for EV3B

4.4.5 Variant e: Mutual exclusive intersection with a semaphore [medium/difficult]

We recall that the mutual exclusion problem arises when independent actors try to use a common resource that cannot be used by them at the same time, in our case the intersection. Similar problems may stand also within a single robot when concurrent activities are programmed. Remember that EV3-G permits to insert in a program the so called ‘parallel sequences’, that is to open (apparently) independent flows of execution. For example in fig. 16, after a first motion, the second motion is executed concurrently with the production of a sound. Concurrency is also produced when you insert two full programs, each one starting with a `Start` command, like in fig. 17.

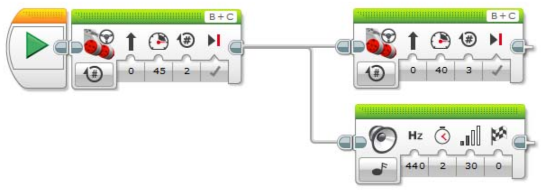

Figure 16. A parallel (sub)sequence

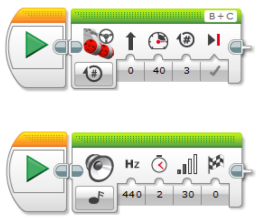

Figure 17. Two parallel sequences

As long as parallel actions act on independent actuators, like one motor and the sound generator, or different variable, the parallelism is not problematic, but when the parallel actions act on the same actuator(s), some form of interference may arise. Have a look at fig. 18. The first sequence makes the robot move around regularly, while the second, if executed alone, when you put your hand in front of the ultrasonic sensor, it makes a sort of ‘escape’ motion. But if you run both programs concurrently, the interference between the two actions is manifested clearly by the strange behavior of the robot when you put your hand in front of the sensor.

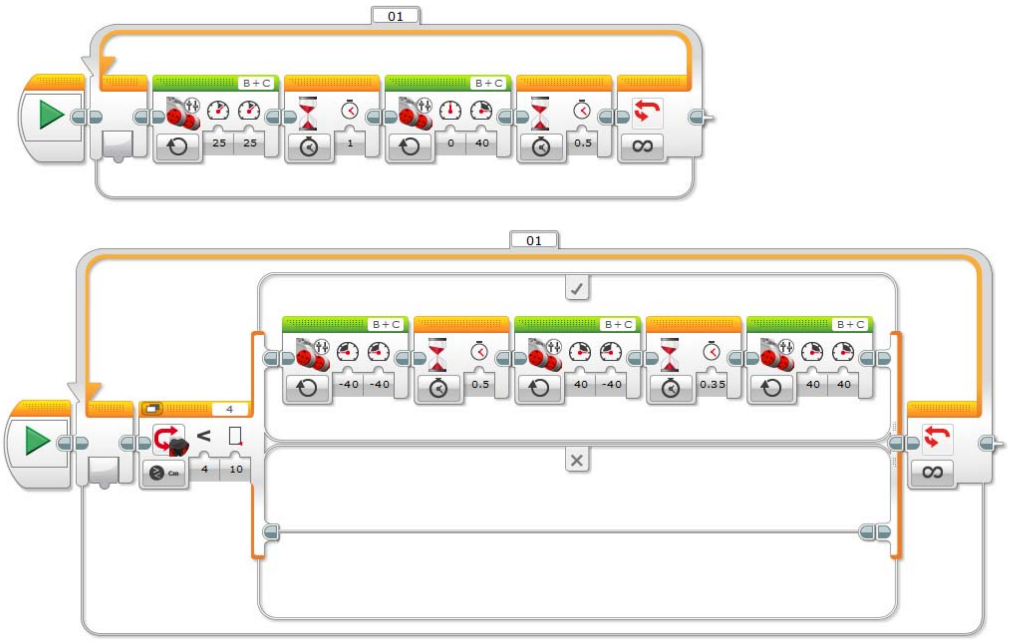

Figure 18. Two parallel sequences acting on the same motors

To avoid such an interference you should assure that the two subsequences of commands inserted in the two loops are executed in a mutual exclusive way. To force this discipline, a common solution is to used a synchronization tool which is appropriately called *semaphore*. It is a logical variable, initially true, on which we must define two main operations, traditionally called *wait* and *signal*. After having recognized the sequence of commands to be executed in an exclusive way, the discipline requires to call *wait* just before each sequence and *signal* after it.

We implement these two basic functions as the customized `mwait` and `msignal` blocks (fig. 19 and 20). The `mwait` block suspends the calling program within a loop until the logical `mutex` variable returns *true* and then it put the variable to *false*. So, if one of the two 'exclusive' sequences successfully completes the `mwait` blocks, it leaves the `mutex` variable to *false* and therefore this 'blocks' the other sequence to enter its critical section until the first one calls `msignal` to unblock the waiting program.

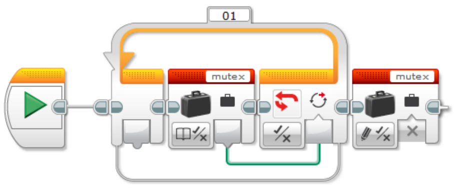

Figure 19. mwait *customized block, wait on a mutex semaphore*

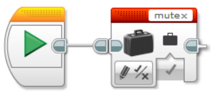

Figure 20. msignal *customized block, signal on a mutex semaphore*

Now, going back to the example of fig. 18, simply adding the two calls respectively at the beginning and at the end of each critical section, the abovementioned 'strange behavior' disappears (fig. 21) The first `msignal` call in the first sequence initializes the `mutex` variable to *true*.

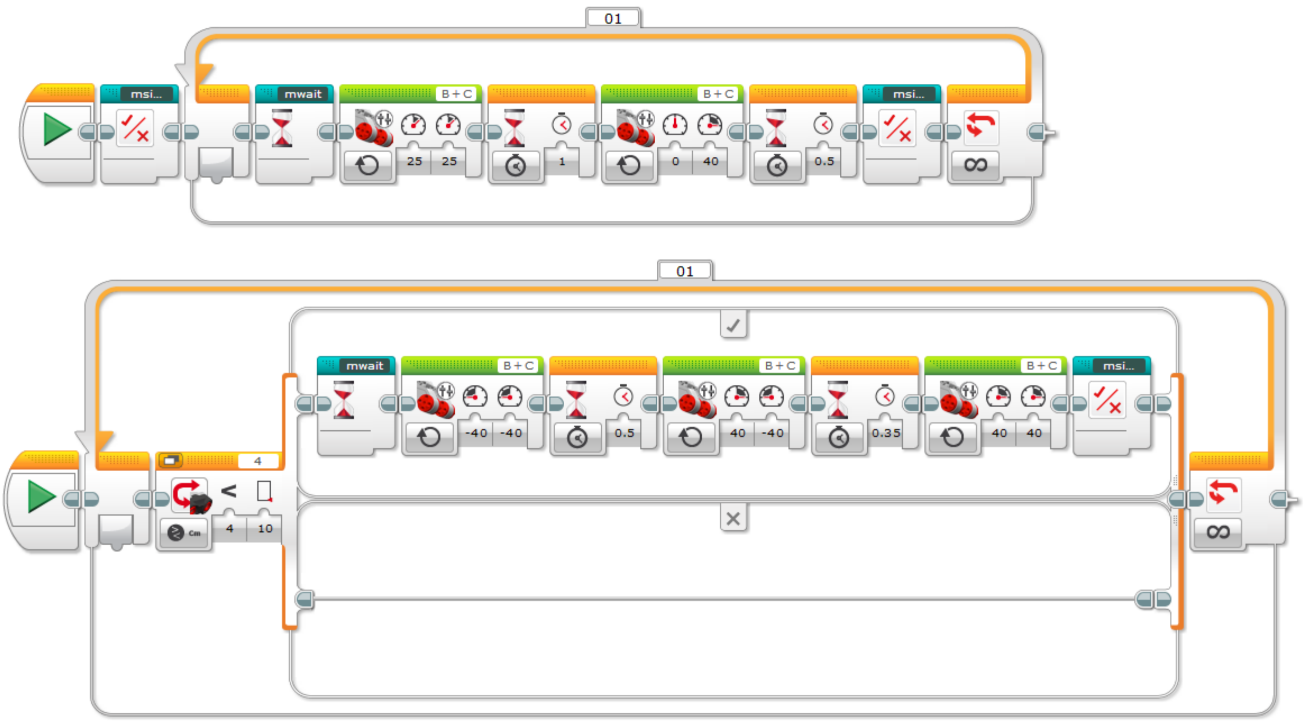

Figure 21. Two parallel sequences with mutual exclusion

4.4.6 Variant f: Second scenario with fixed distance [difficult]

Following the approach just presented, it is now possible to provide a solution to the intersection problem safer than the one presented in the d) section. Considering that, to synchronize in mutual exclusion two or more actors, only one semaphore is necessary, the `mutex` variable is maintained in one of the two robots, for example EV3A (fig. 22). The code in this robot that contains the motion commands can be protected directly inserting the two semaphore calls around its critical section. The other robot EV3B (fig. 23) must ask EV3A for the permission to enter the intersection using a message exchange: so we must add in EV3A a parallel sequence which is always waiting for a request of permission from the other robot and giving. This sequence gives such permission only if the `mutex` variable shows that the intersection is clear. Therefore EV3A acts as a sort of 'master' for providing permission to the 'slave' EV3B. This second parallel sequence, shown in fig. 22, is simply a loop with five commands: a wait for a request of permission, a `mwait` call which can delay the permission if the intersection is already occupied, a send of permission, a wait for the message signaling that EV3B has cleared the intersection, and a final `msignal` call to return to the initial state. We say that this solution is safer because, implementing test and set *false* of the `mutex` variable within the `mwait` block, guarantees a substantial 'atomicity' of the *wait* operation on the semaphore and therefore the problematic sequence of events described in the previous section are here almost impossible.

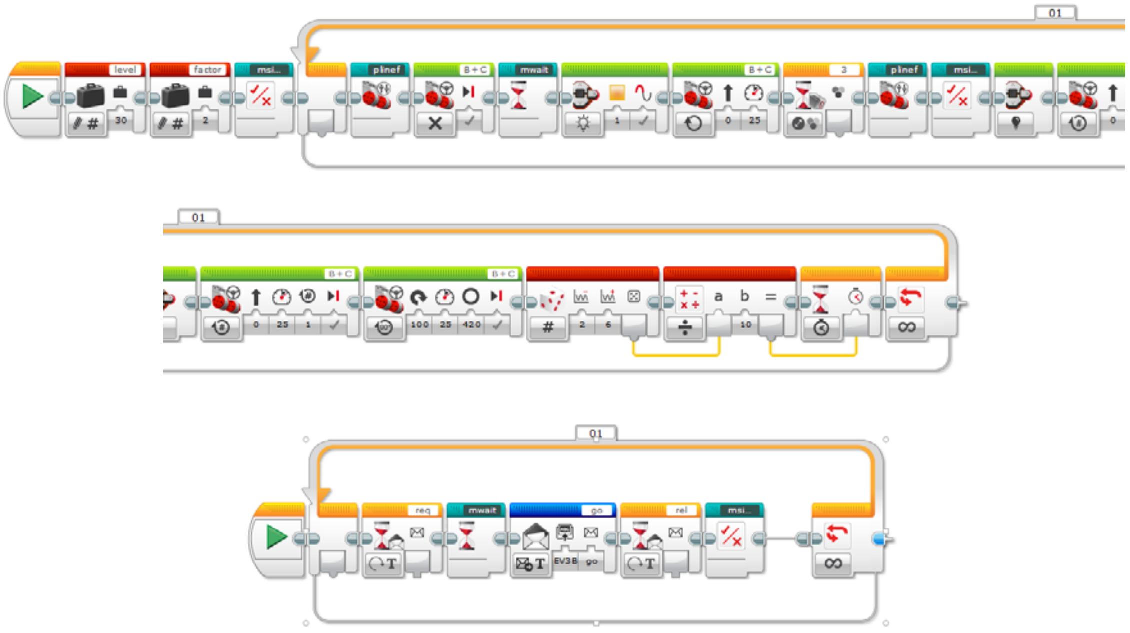

Figure 22. EV3A the Master, the two parallel sequences

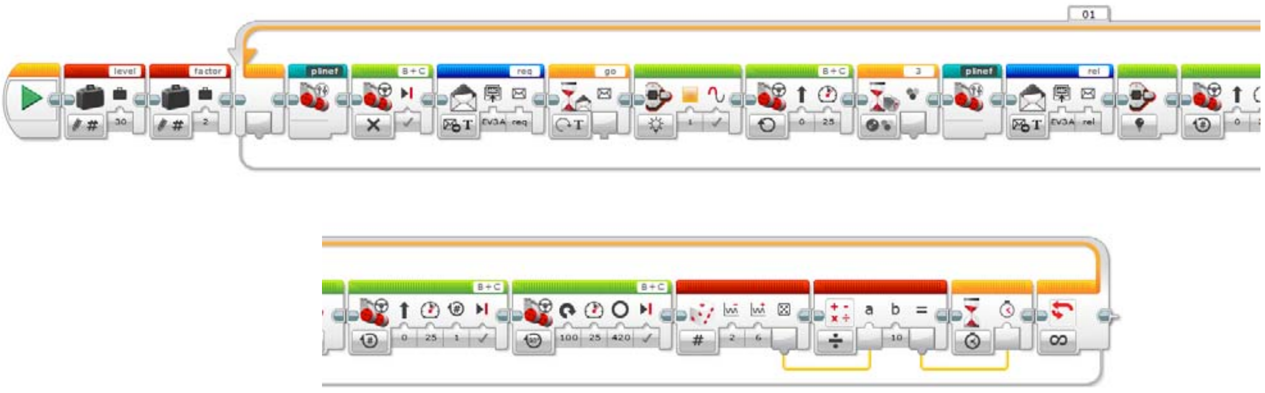

Figure 23. EV3B the Slave

4.4.7 Variant g: Mutual exclusive intersection with a traffic light [medium/difficult]

In a real intersection, often a traffic light imposes a time-based precedence rule: we would like to simulate a similar situation. We want to represent the traffic light in each one of the two direction with the brick lights which will be enlighten with one of the three colors, green, yellow and red in sequence. The timing will be controlled only by one robot (EV3A as master), while the other (EV3B) will receive a synchronization message anytime a change in the traffic light must be done. In particular such timing must be chosen so that the yellow interval is long enough (in the example, 4 s) so that the robot can entirely cross the intersection after entering it just before the transition green-yellow, assuring that the intersection is clear when the other robot ‘sees’ its traffic light becoming green. Both robots have to maintain a `tlight` numerical variable representing the state of their associated traffic light, 2=green, 1=yellow, 0=red. The entering of the intersection is conditioned by the value 2=green of the associated traffic light. For the rest the code of the two robots is similar to the one presented in the previous section.

Fig. 24 shows the two parallel sequences for EV3A (master), that is the sequence with motion commands, which checks the state of the traffic light variable and waits in a loop until it is green, and the code which controls the sequence of states of this variable, with 8 s of separation between green and yellow, and 4 s between yellow and red. This second code includes the sending to EV3B of messages of type *sync* when a change in the state of the traffic light variable of the other robot is requested. Fig. 25 shows only the code to control the traffic light variable of the EV3B robot (slave) because the other parallel sequence is equal to the one in EV3A.

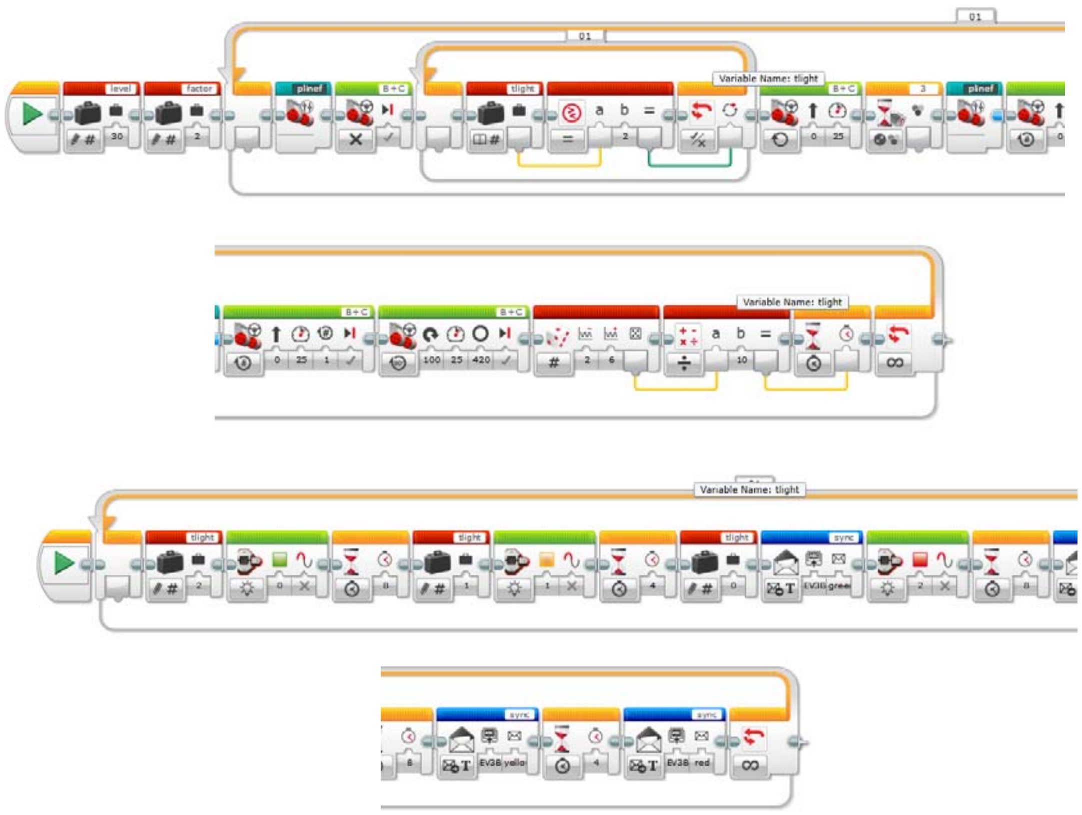

Figure 24. The two parallel seq. of EV3A the Master with traffic light

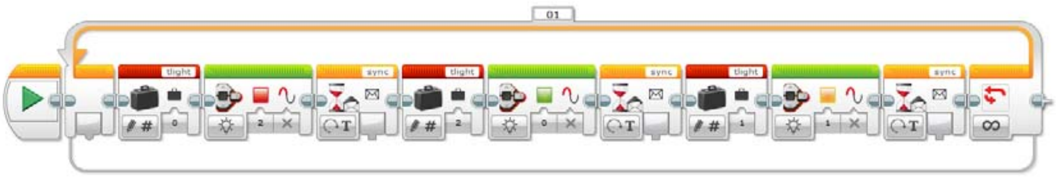

Figure 25. The second parallel seq. of EV3B the Slave with traffic light

Chapter 5: **Evaluation tools (O1.5)**

Use the rubric below to evaluate your students' achievement in each specific objective of this curriculum.

Name of student (or group of students): ………………………………………………………

upon completion of the activities described in this curriculum students achieved the following objectives	**Evaluation score** **0 = not attempted** **1 = attempted without success** **2 = partial success** **3 = completed with teacher's help** **4 = completed without teacher's help**
Mounted and used correctly the ultrasonic sensor	
Mounted and used correctly the color sensor	
Created a mock up with the intersection within which the robots would operate based on the scenario	
Instructed the robot to smoothly follow the line by using the P feedback control principle	
Sensed the presence of the other robot using the ultrasonic sensor	
Instructed the robots to give precedence to right	
Instructed both robots to correctly apply a precedence to always the same robot	
Instructed both robots to correctly apply an alternative precedence	
Instructed both robots to correctly apply the control of a simulated traffic light	