CoSpace GRAND PRIX

U19

Presented to you by RoboCup Singapore

Email: CoSpace@robocupsingapore.org

RCAP RoboCup SINGAPORE CoSpaceRobot ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

Learning Point

1

Launch CoSpace Grand Prix

(Advanced)

Launch the CoSpace Grand Prix (Advanced)

U19

CoSpace Grand Prix Virtual Environment

U19

1. Click anywhere in the virtual environment

2. Press the Q, W, E, A, S, D keys

Up Zoom-in Down

Right Zoom-out Left

Turn 360°

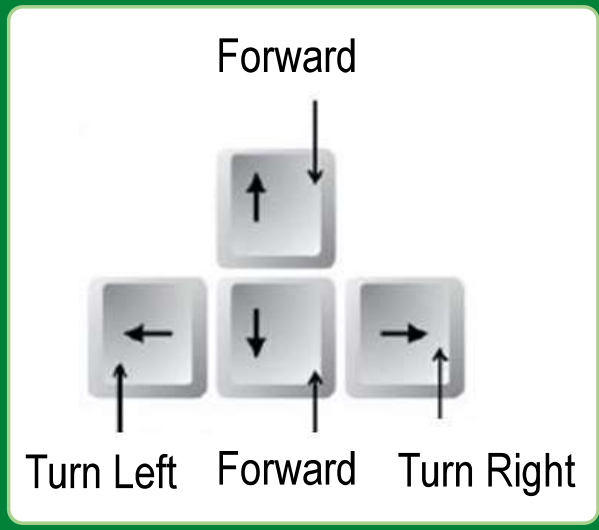
3. Press “Shift + the keys” to make the movement faster.

RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

Manual Control of Virtual Robot

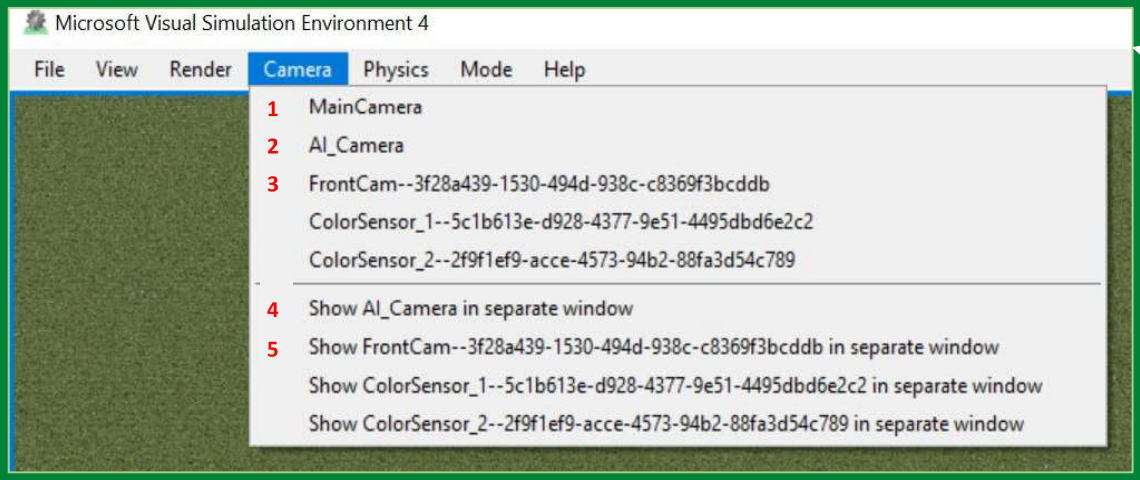
U19

2. Press the arrow keys below



Virtual Camera Viewpoint Settings

U19



1. Main Camera

2. AI Camera

3. Front Camera

4. Main Camera with front camera

5. Main Camera with AI camera

RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

2

My First CoSpace Grand Prix Program

My First Program

U19

Task:

To program our robot to follow the line and travel towards the finish line.

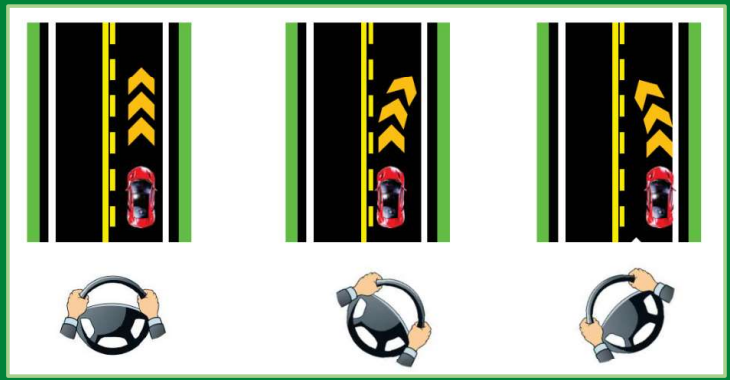
My First Program

Driverless Car U19

Real Situation

Real Robot

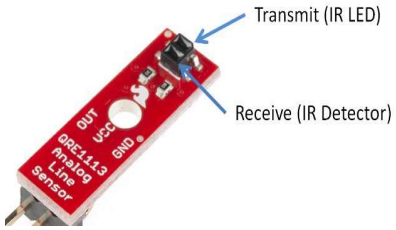
Virtual Robot



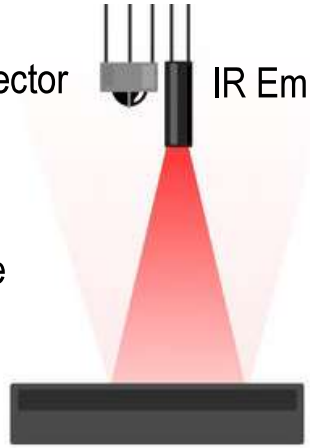
RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

My First Program

IR Sensor U19

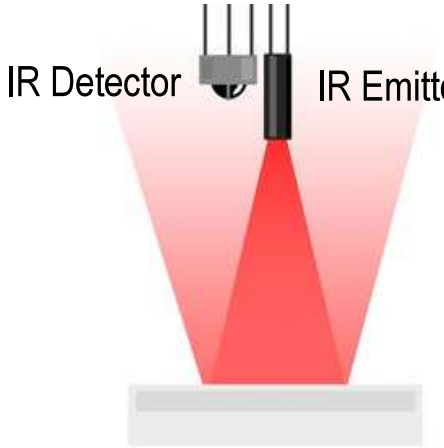


Less IR light will be reflected by darker coloured objects



Dark Surface

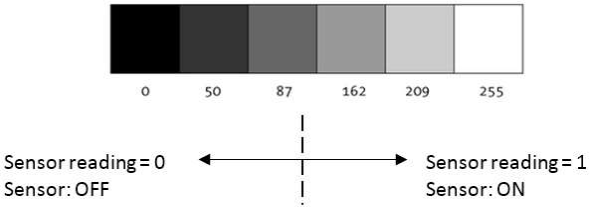
(Dark surface absorbs Infrared light)



Bright Surface

(Bright surface bounces Infrared light)

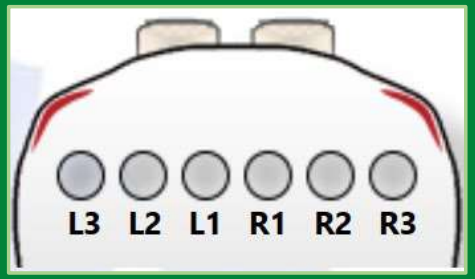
More IR light will be reflected by light coloured objects



My First Program

IR Sensor U19

A 6-IR sensor array is mounted at the bottom of the virtual robot to sense / detect black or white coloured surface.



My First Program

Flowchart U19

Line-Following Logic

L1 senses the black line

Forward

R1 senses the black line

Forward

L2 senses the black line

Slight Left

R2 senses the black line

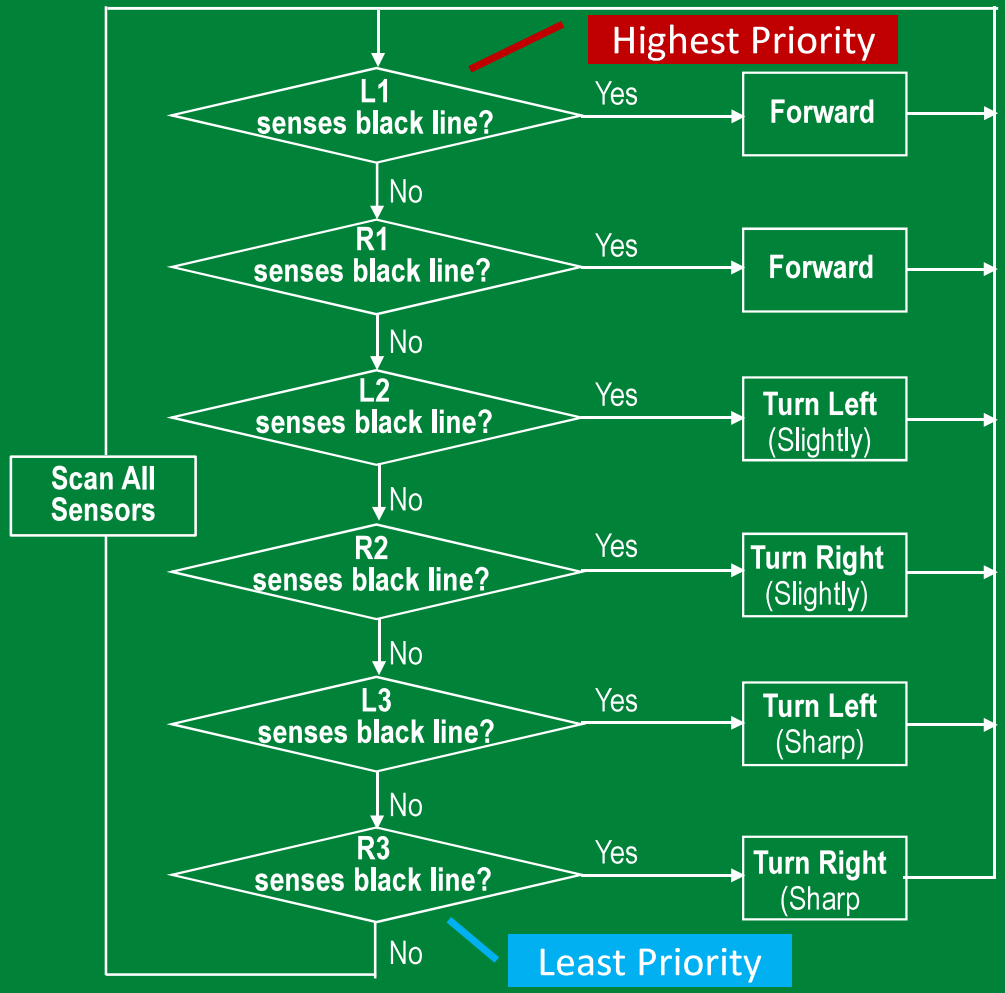
Slight Right

L3 senses the black line

Sharp Left

R3 senses the black line

Sharp Right

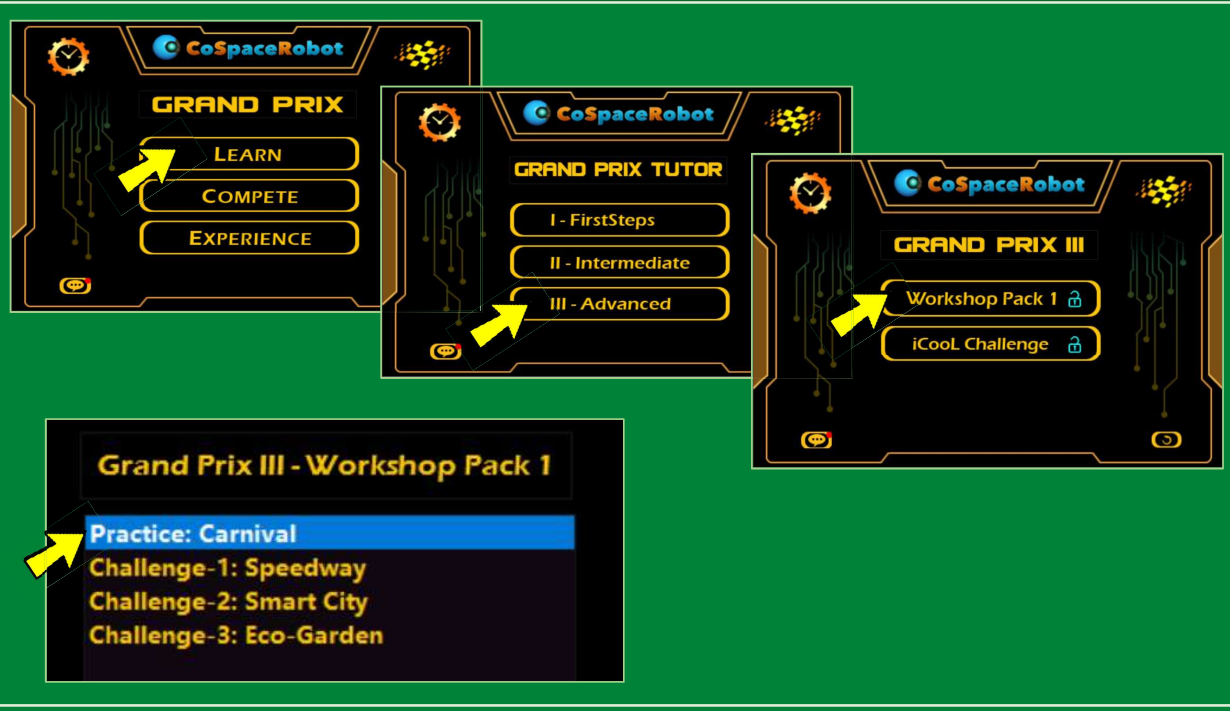


My First Program

Step-by-Step U19

Virtual Map - Practice: Carnival

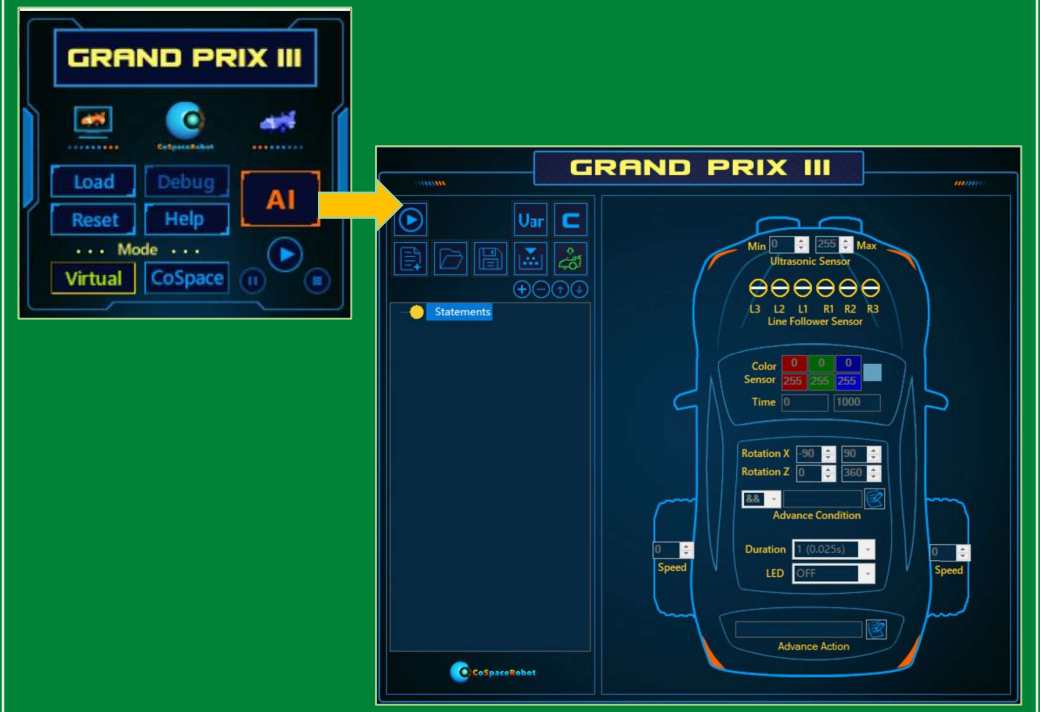
Step 1: Launch CoSpace Grand Prix Intermediate



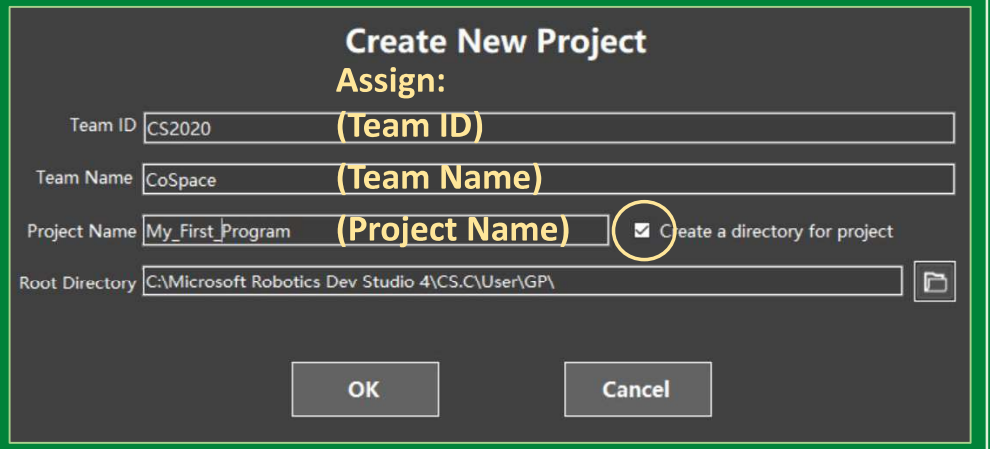
My First Program

Step-by-Step U19

Step 2: Launch AI Panel



Step 3: Create a New Project



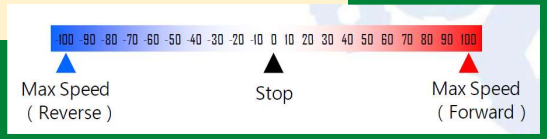
My First Program

Step-by-Step U19

Step 4: Add the 1st Statement

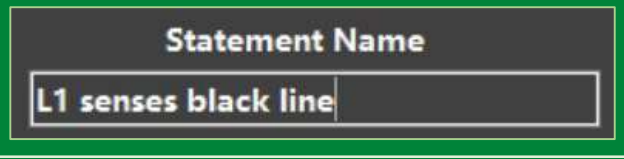
If L1 senses the black line, robot moves forward

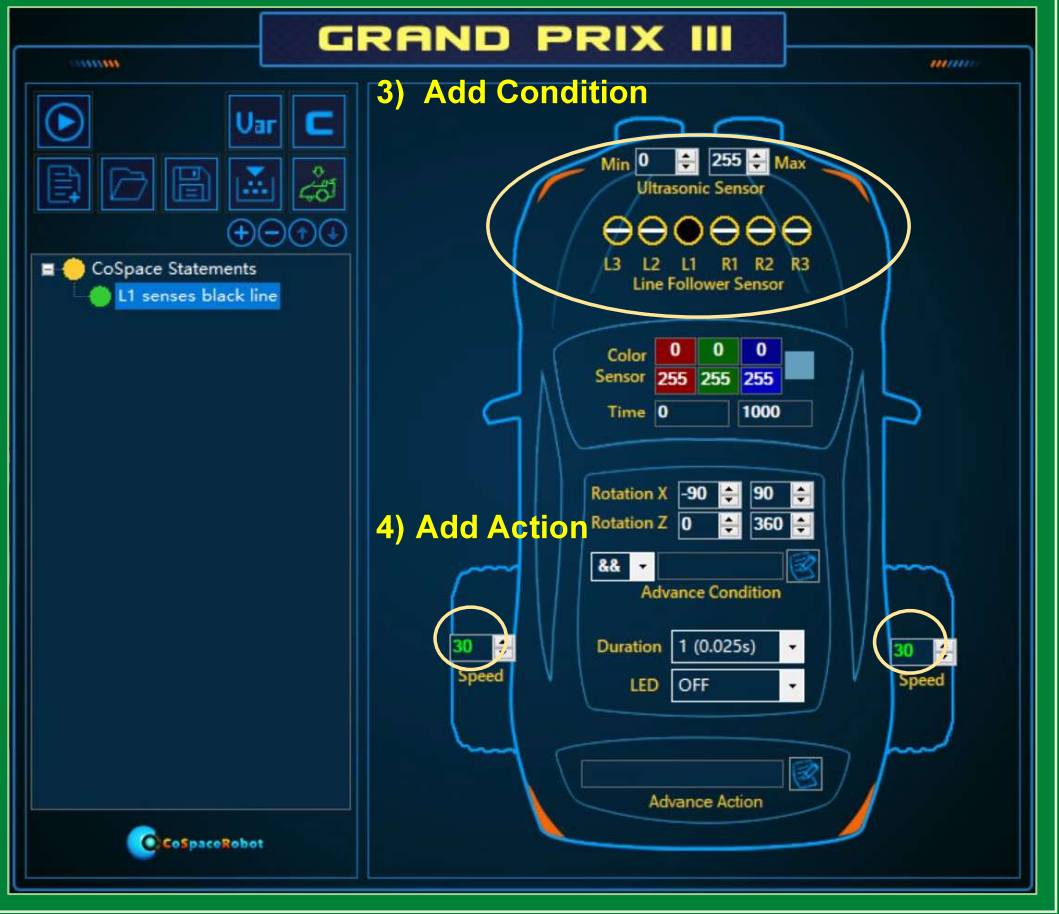
```
If  ( ⦸⦸●⦸⦸⦸ )
      {
          WheelLeft = +30;
          WheelRight = +30;
      }
```



1) Select “root” and click on “+”

2) Give a statement name:





My First Program

Step-by-Step U19

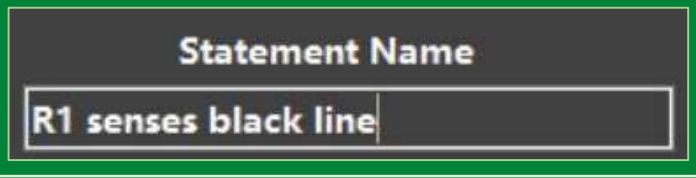
Step 5: Add the 2nd Statement

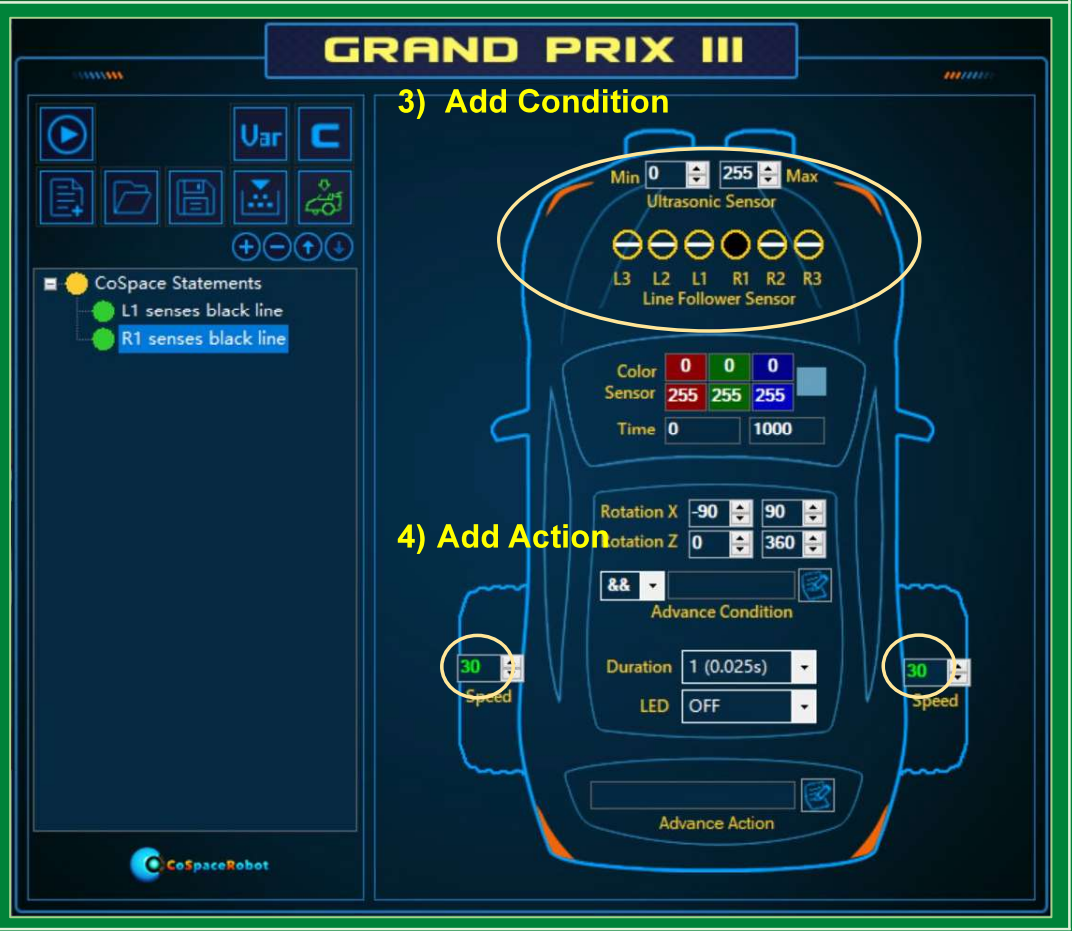
If R1 senses the black line, robot moves forward

```
If  ( ⦸⦸⦸●⦸⦸ )
      {
          WheelLeft = +30;
          WheelRight = +30;
      }
```

1) Select the 1st statement and click on “+”

2) Give a statement name:





My First Program

Step-by-Step U19

Step 6: Add The Rest of Statements

S/N	Statement Name	IR	Left wheel	Right wheel	LED	Action Time
1	L1 senses black line	⦸⦸●⦸⦸⦸	+30	+30	OFF	0.025 s
2	R1 senses black line	⦸⦸⦸●⦸⦸	+30	+30	OFF	0.025 s
3	L2 senses black line	⦸●⦸⦸⦸⦸	+10	+30	FLASH	0.025 s
4	R2 senses black line	⦸⦸⦸⦸●⦸	+30	+10	FLASH	0.025 s
5	L3 senses black line	●⦸⦸⦸⦸⦸	-10	+30	ON	0.025 s
6	R3 senses black line	⦸⦸⦸⦸⦸●	+30	-10	ON	0.025 s

Note: Wheel speed can be any value from -100 to 100.

Sample Code: U19_My_First_Program

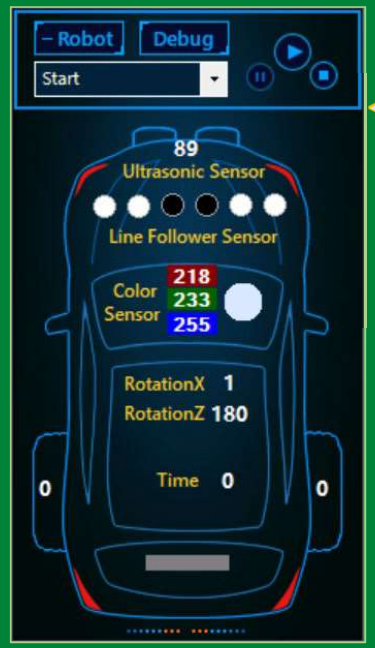
My First Program

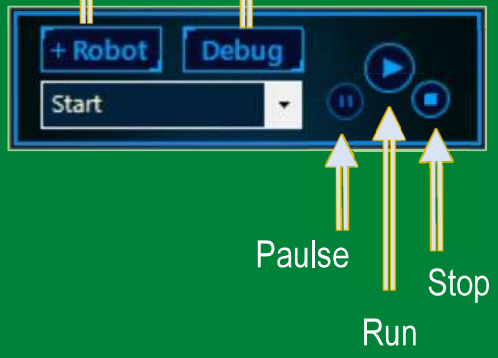
Step-by-Step U19

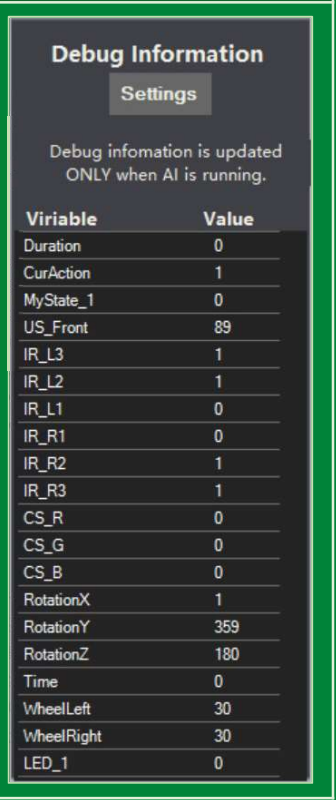
Step 7: Save Project

Step 8: Trial Run

Step 9: Monitor Robot Performance and View Debug Variables







Viriable	Value
Duration	0
CurAction	1
MyState_1	0
US_Front	89
IR_L3	1
IR_L2	1
IR_L1	0
IR_R1	0
IR_R2	1
IR_R3	1
CS_R	0
CS_G	0
CS_B	0
RotationX	1
RotationY	359
RotationZ	180
Time	0
WheelLeft	30
WheelRight	30
LED_1	0

My First Program

Step-by-Step U19

Step 10: Build Project

After building, there will be 3 files in the project folder:

1) My_First_Program.smp
2) My_First_Program.dll
3) My_First_Program.c

Path: C:\Microsoft Robotics Dev Studio 4\CS.C\User\GP\My_First_Program

Step 11: Load Project From Control Panel

Load "**My_First_Program.dll**" file

Step 12: Execute the Project

Set a virtual game

Pause / Run / Stop

You can select different camera viewpoints to monitor the robot's movement from different viewing angles.

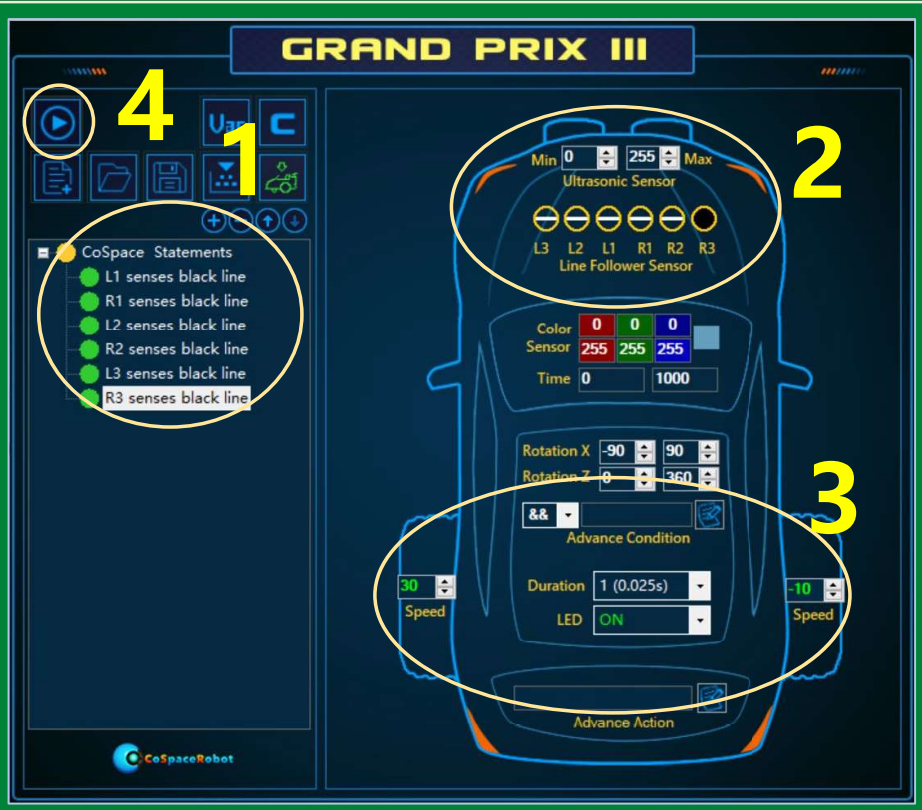
My First Program

Step-by-Step U19

Summary

Writing CoSpace AI, Steps:

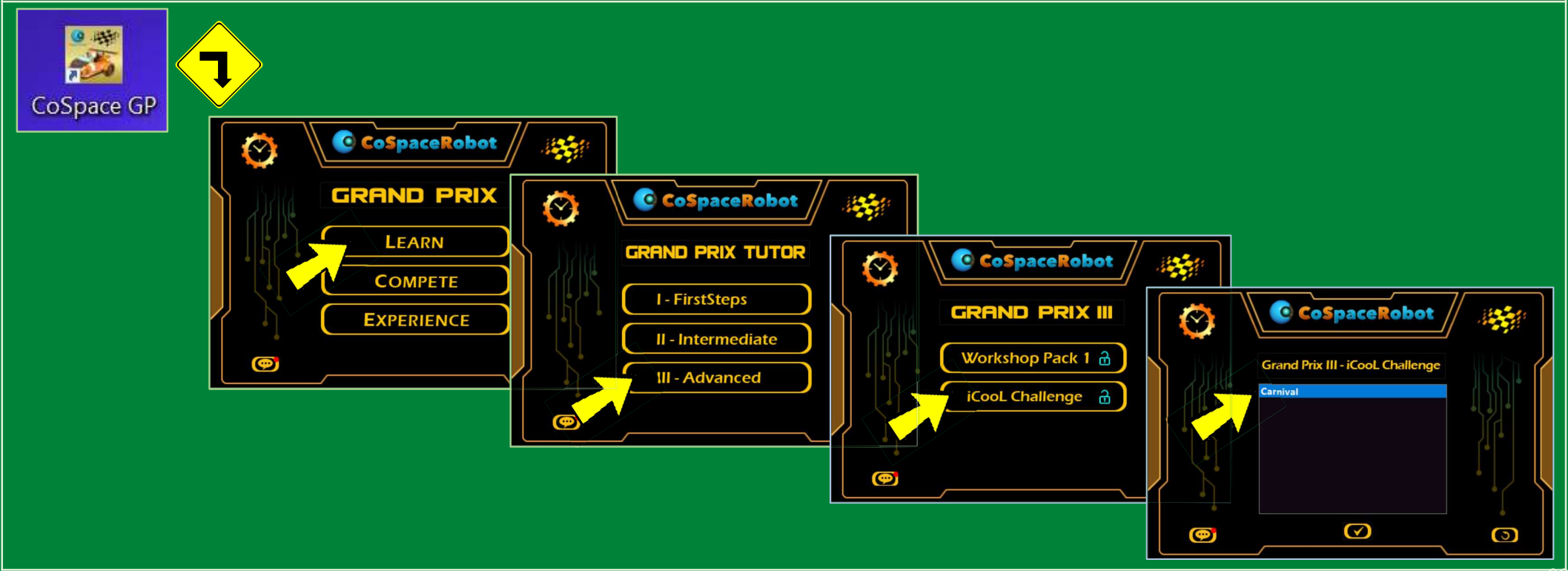
1. Define statement
2. Set conditions
3. Set actions
4. Trial Run



My First iCooLChallenge Program

Step-by-Step U19

Step 1: Launch iCooLChallenge Virtual Field

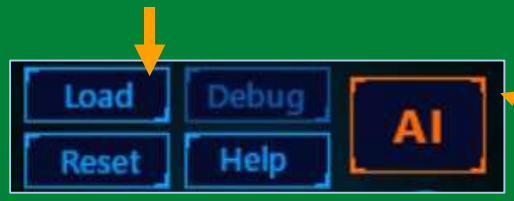


My First iCooLChallenge Program

Step-by-Step U19

Step 2: Load Program and Run

1. Load “My_First_Program.dll”



2. Set a virtual game

Need to change

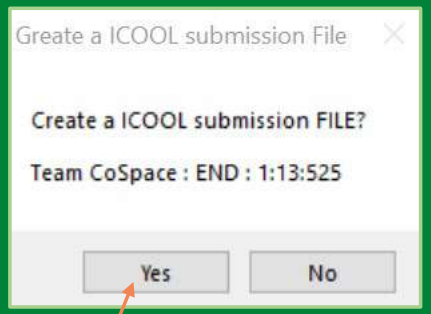
3. Pause / Run / Stop

My First iCooLChallenge Program

Step-by-Step U19

Step 3: Generate ".ics" file

When the robot reaches the finish line, the following dialogue box will pop out:



Click "Yes" to save "My_First_Program.ics" file.

During the iCooLChallenge, you need to submit the ".ics" file to the iCooLChallenge server. The ".ics" file will include the ".dll" file, a screen capture of the score bar and the robot's trajectory during the iCooLChallenge.

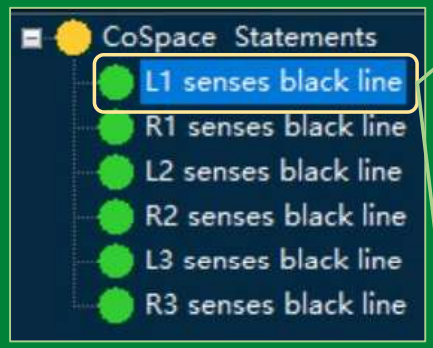
3

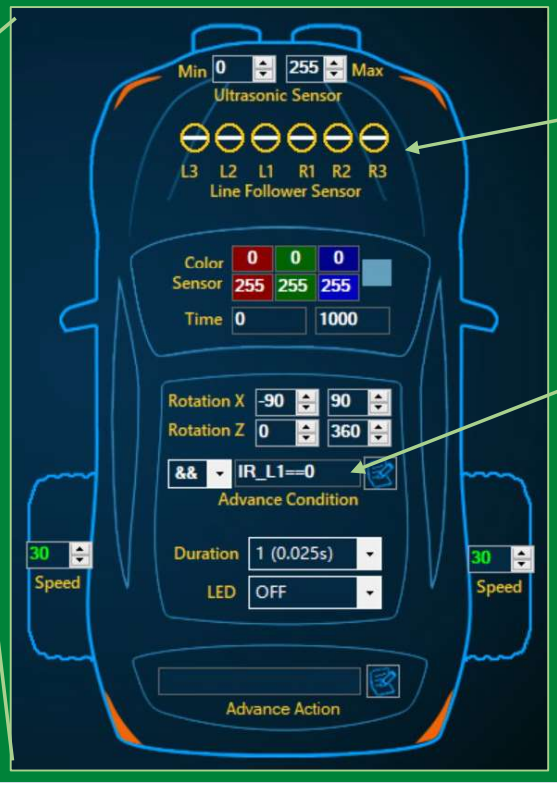
Practice & Challenges

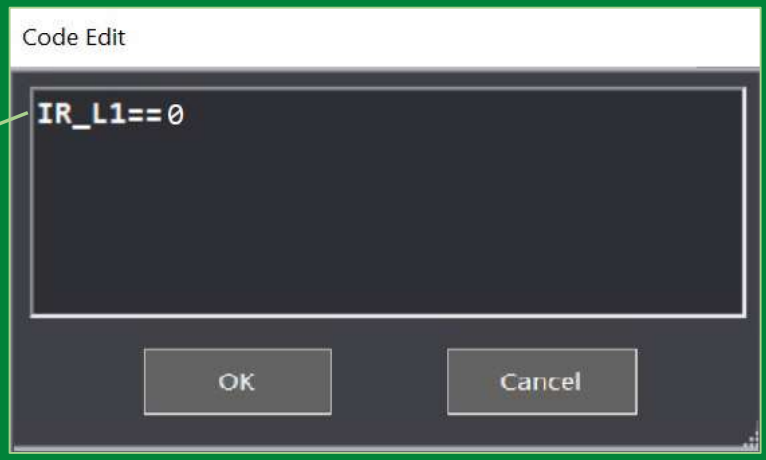
Using Advanced Condition

U19

1) Open “My_First_Program.smp.smp” in AI
2) Select the 1st statement and change the original statement as follows:
4) Complete the whole program.
5) Save the new program as “ **Using_Advanced_Condition** ”.
6) Build and run the program.
7) Monitor the robot’s performance.







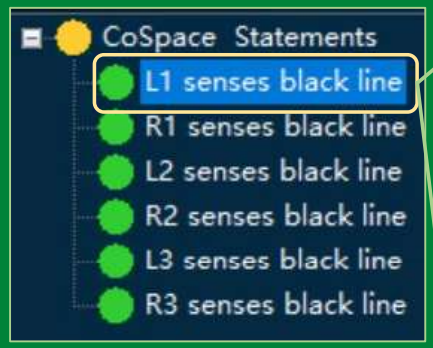
Sample Code:
U19_Using_Advanced_Condition

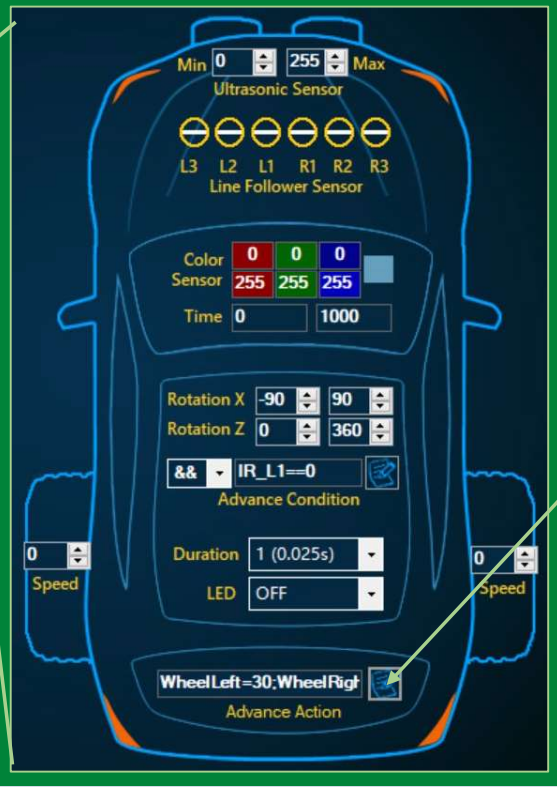
Note: IR_L1 == 0 means L1 senses black colour

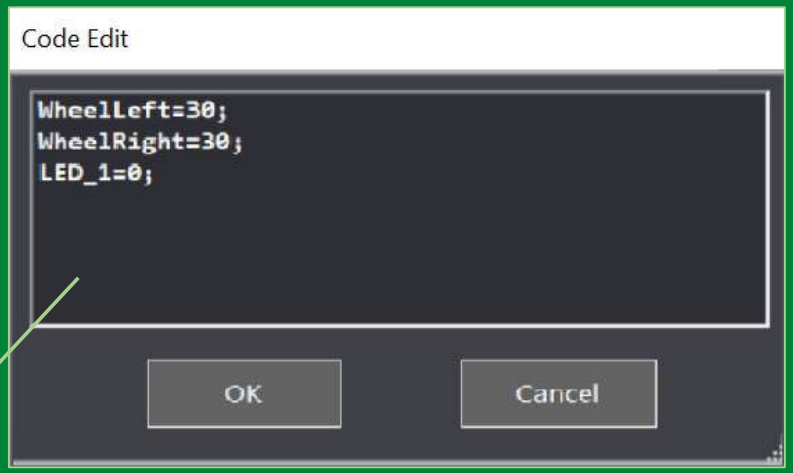
Using Advanced Action

U19

1) Open “Using_Advanced_Condition.smp” in AI
2) Select the 1st statement and change the original statement as follows:







Sample Code:
U19_Using_Advanced_Action

4) Complete the whole program.
5) Save the new program as “ **Using_Advanced_Action** ”.
6) Build and run the program.
7) Monitor the robot’s performance.

Note:
WheelLeft -> Left wheel; WheelRight -> Right wheel
LED_1 = 0 -> LED off
LED_1 = 1 -> LED flash
LED_1 = 2 -> LED on

Using printf

U19

A “printf” statement can be used to print out the live value of all variables, including system variables and self-defined variables, in the command window.

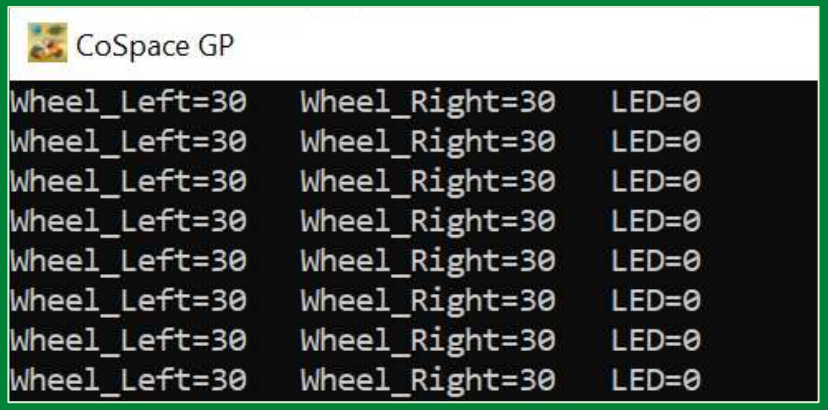
1) Open “Using_Advanced_Action.smp” in AI
2) Add the following to the advanced action of each statement

```
printf("Wheel_Left=%d    Wheel_Right=%d    LED=%d\n",WheelLeft,WheelRight,LED_1);
```

```
WheelLeft=30;
WheelRight=-10;
LED_1=2;
printf("Wheel_Left=%d   Wheel_Right=%d   LED=%d\n",WheelLeft,WheelRight,LED_1);
```

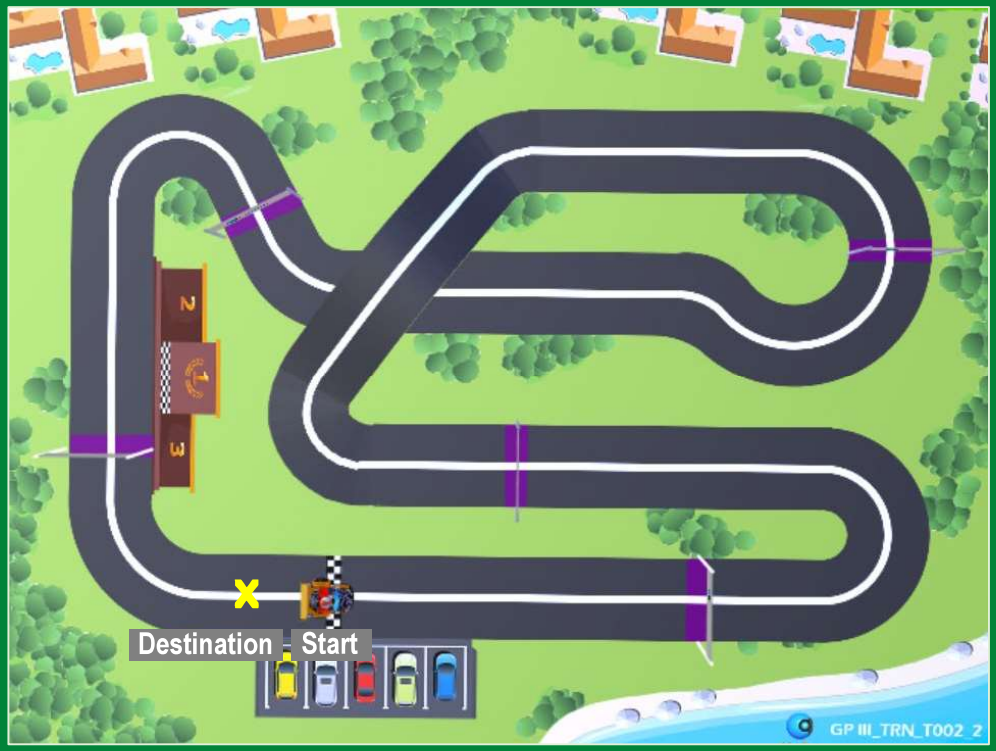
3) Save the new program as “ **Using_printf** ”.
4) Build and run the program.
5) Monitor the output in the command window when the robot is tracking the line.

Sample Code: U19_Using_printf



Challenge 1: Speedway

U19



: Stations

Task:

To program our robot to travel towards the destination. During its journey, It must stop at each pit station for 2 seconds with LED flashing, and then move away from the station autonomously.

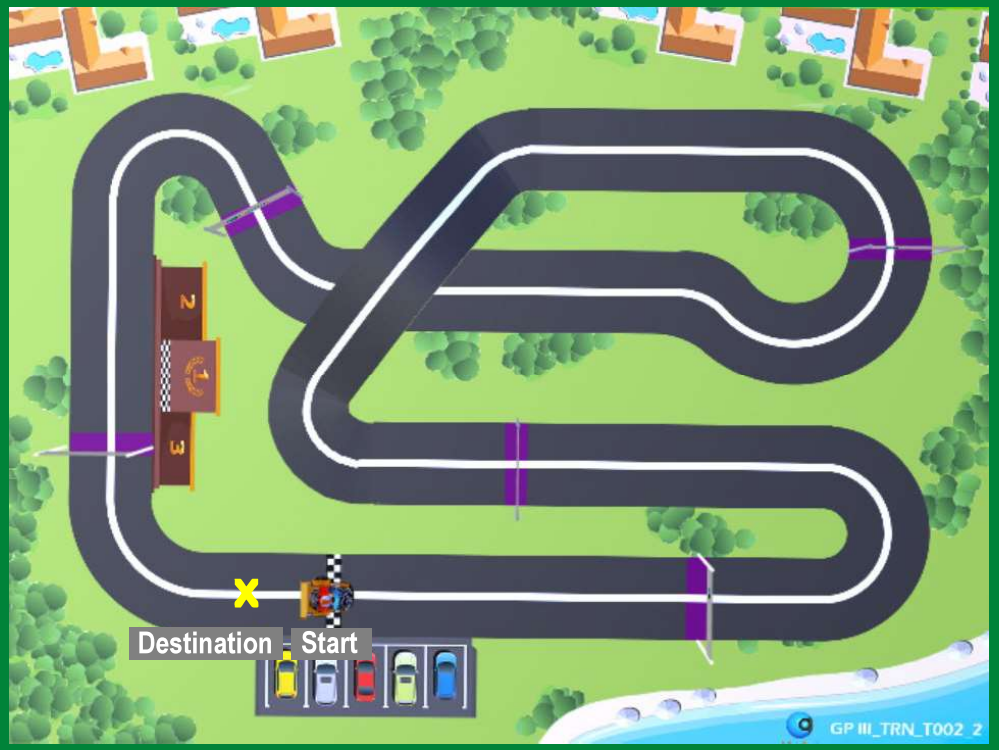
Virtual Map:

RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

Challenge 1: Speedway

Analysis U19

- Type of Task
  - This is a line following task.
- The key points of this task are:
  - The robot needs to detect the stations (using RGB sensors)
  - Stop at each station for 2 sec
  - Move away from the stations automatically afterwards.

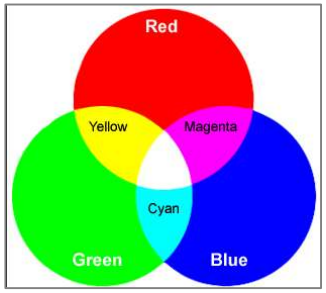


Challenge 1: Speedway

RGB Colour Sensor U19

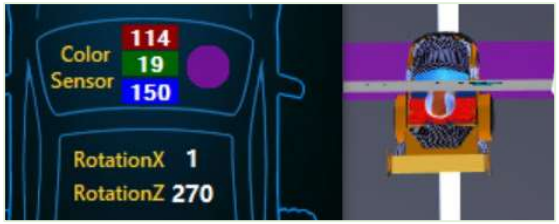
- The RGB sensor is mounted at the front right corner of the robot, facing downwards
- The RGB sensor detects the colour of the ground.

RGB Colour Sensor Working Principle:



- the RGB sensor detects colours. It outputs the exact colour it detects, in terms of the Red (**R**), Green (**G**) and Blue (**B**) light intensities. Each channel value can range from 0 to 255.

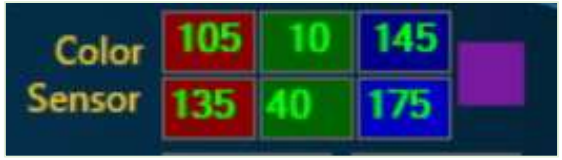
RGB Sensor Reading:



RGB sensor readings:

- Red Channel: 114
- Green Channel: 19
- Blue Channel: 150

RGB Sensor Setting:

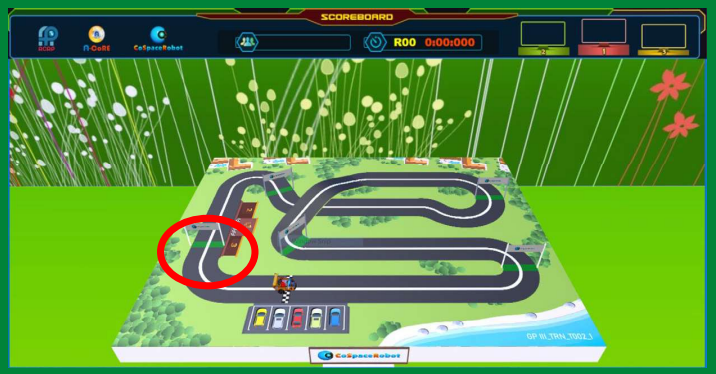


Just like programming with other sensors, the RGB sensor input values must be a range.

Challenge 1: Speedway

Analysis U19

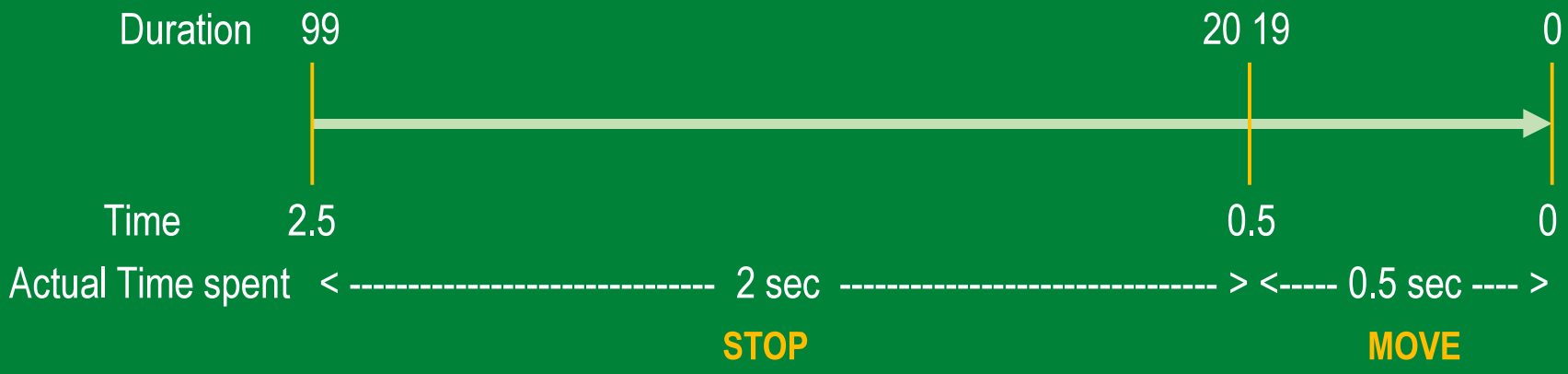
How to programme the robot to stop at the orange station for 2 sec and move away automatically afterwards:



We will specify a period of 2.5 sec – the robot will stop for 2 sec and then move forward for the next 0.5 sec

We can use Duration. The duration is a countdown, where 1 duration = 0.025 sec

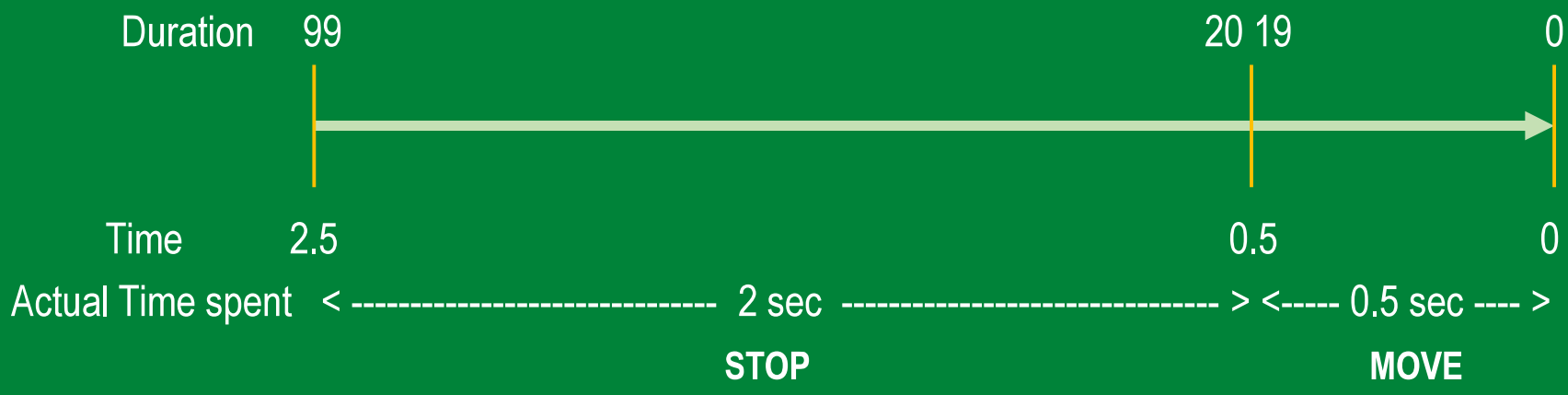
1 Duration = 25 ms, 2 sec = 80 Durations 2.5 sec = 100 Duration

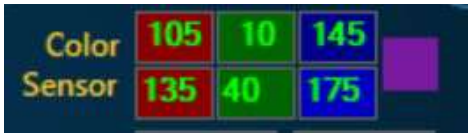


RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

Challenge 1: Speedway

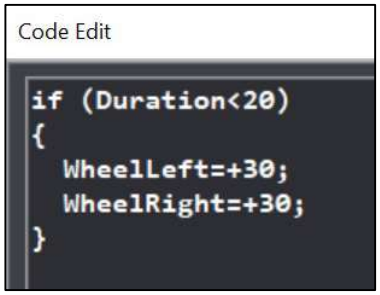
Analysis U19





Condition

Action



Advanced Action Code

Challenge 1: Speedway

Reference Code U19

S/N	Statement Name	Ultrasonic (cm)	IR	RGB	Time (Sec), RotationX, RotationZ	Left wheel	Right wheel	LED	Duration	Advanced Action

1	At Station	Default	Default	R: 105 – 135
				G: 10 – 40
				B: 145 – 175
					Default	0	0	FLASH	2.5 sec	`if (Duration<20)`
										`{`
										`  WheelLeft=+30;`
										`  WheelRight=+30;`
										`}`

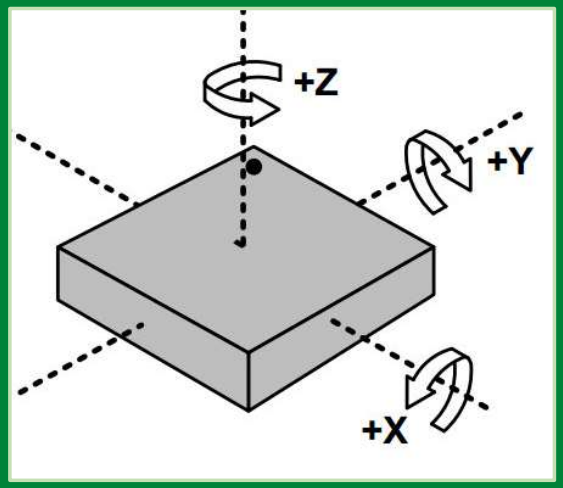
2	L1 senses white line	Default	⦸⦸O⦸⦸⦸	Default	Default	+30	+30	Default	0.025 s	-
3	R1 senses black line	Default	⦸⦸⦸O⦸⦸	Default	Default	+30	+30	Default	0.025 s	-
4	L2 senses black line	Default	⦸O⦸⦸⦸⦸	Default	Default	+10	+30	Default	0.025 s	-
5	R2 senses black line	Default	⦸⦸⦸⦸O⦸	Default	Default	+30	+10	Default	0.025 s	-
6	L3 senses black line	Default	O⦸⦸⦸⦸⦸	Default	Default	-10	+30	Default	0.025 s	-
7	R3 senses black line	Default	⦸⦸⦸⦸⦸O	Default	Default	+30	-10	Default	0.025 s	-

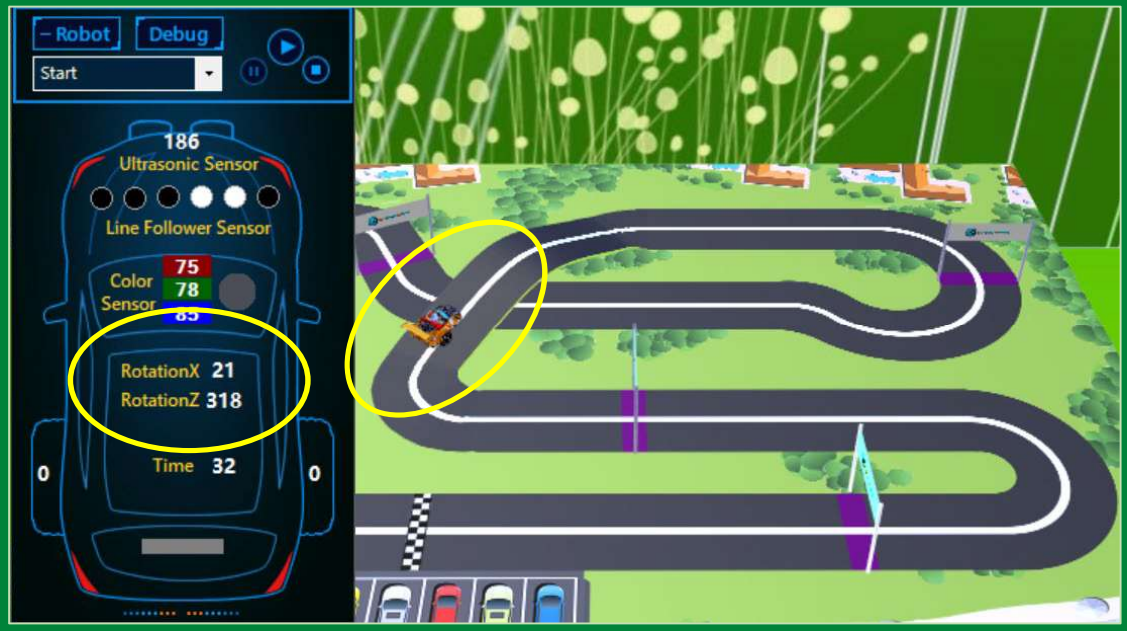
Sample Code: U19_Challenge_1

Challenge 1: Speedway

Gyro Sensor U19

In Grand Prix advanced, a Gyro sensor is used to detect the robot's orientation (direction and tilt).





- RotationX = > Tile sensor reading
- RotationZ => Compass sensor reading

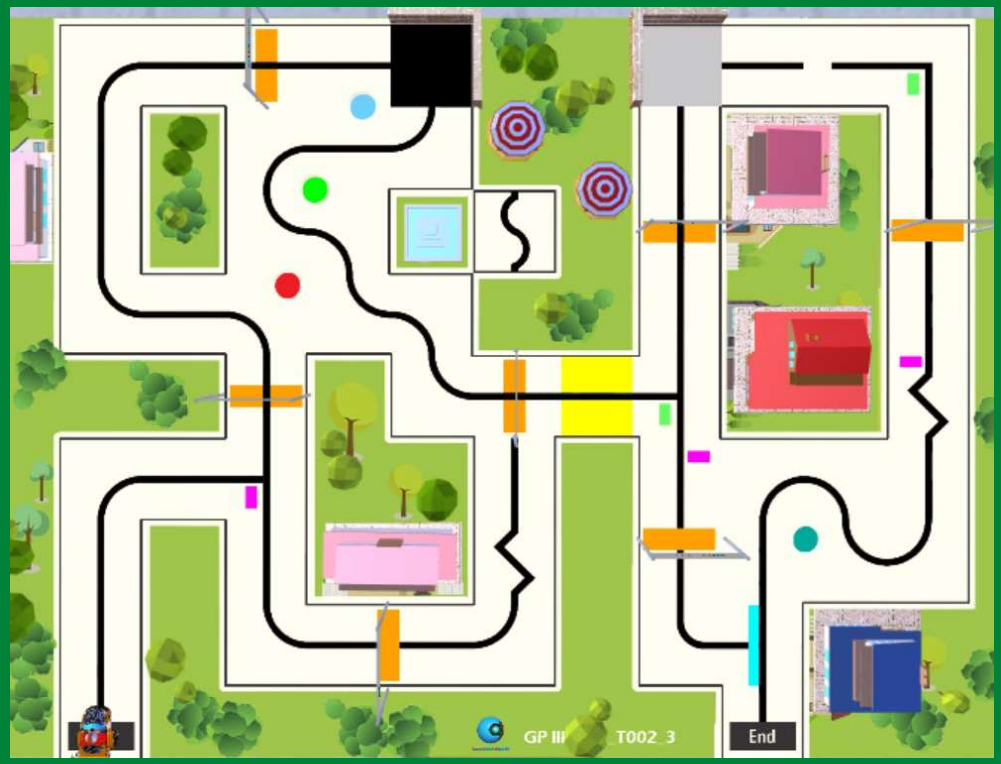
When the robot is on the bridge, we have

- Tilt sensor = 21 degrees
- Compass sensor = 318 degrees

RCAP RoboCup SINGAPORE CoSpaceRobot RMA ROBOTICS & MAKER ACADEMY Advanced Robotics & Intelligent Control Centre SINGAPORE POLYTECHNIC

Challenge 2: Smart City

U19



: Stations

Task:

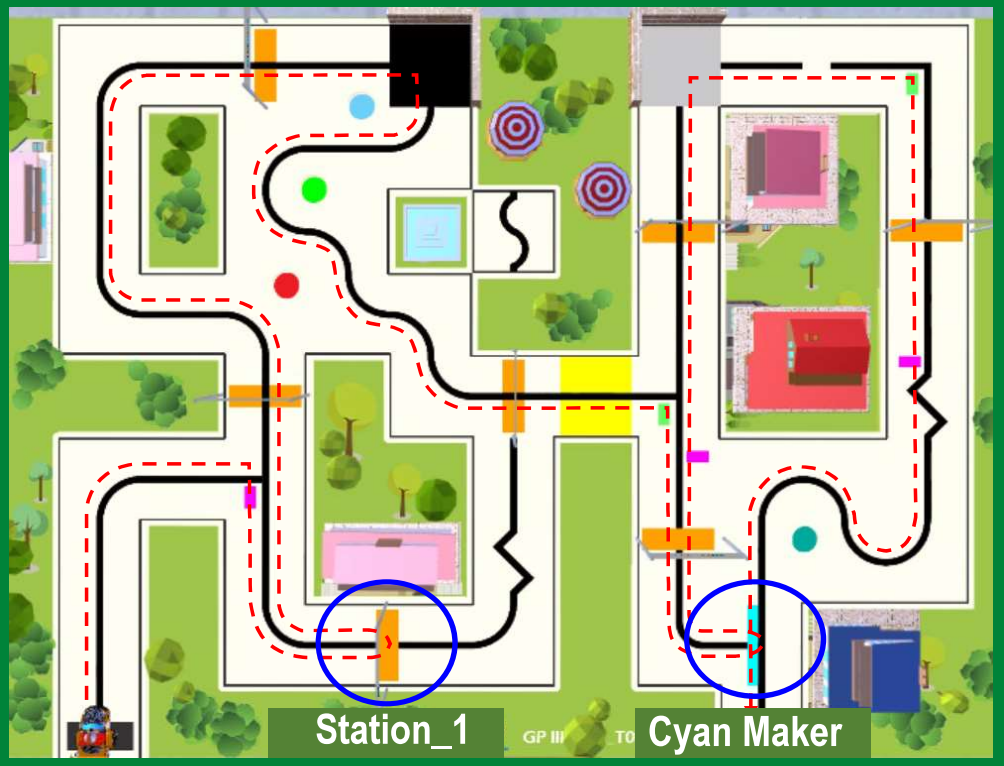
To program our robot to travel towards the destination. During its journey, It must stop at each pit station for 2 seconds with LED flashing, and then move away from the station autonomously.

Virtual Map:

Challenge 2: Smart City

Analysis U19

- Task
  - The robot needs to follow the black line to “END”
  - The robot needs pass all stations.
  - The robot must stop at each station for 2 seconds with LED flashing, and then move away from the station autonomously.
- Route
  - We have designed the route as indicated on the map.
  - In order to minimise the travelling time, the robot will make U-turn at Station_1 and the Cyan marker.

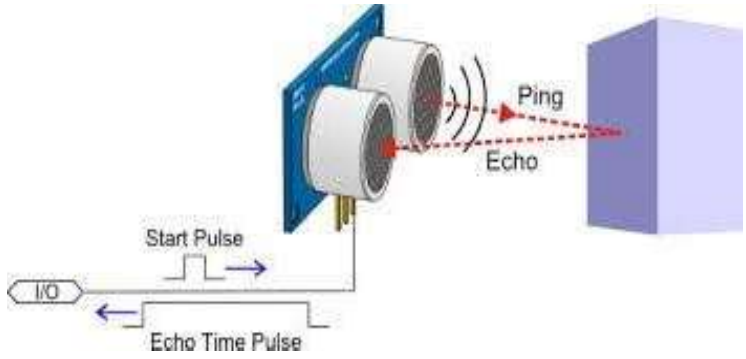


: Stations

Challenge 2: Smart City

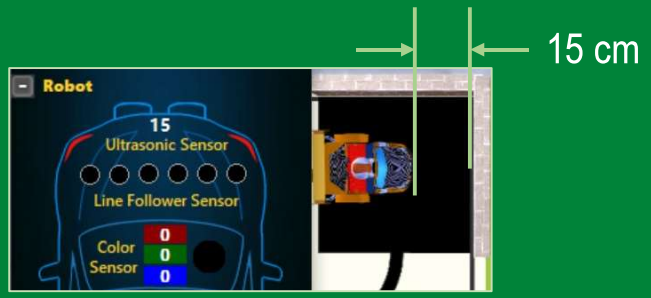
- The ultrasonic sensor is fitted at the front of the robot.
- It detects objects in front of the robot.

Ultrasonic Sensor Working Principle:

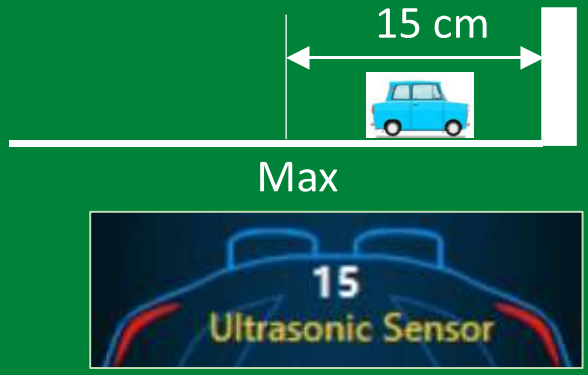


$$distance = \frac{speed\ of\ sound \times time\ taken}{2}$$

Ultrasonic Sensor Reading:



Ultrasonic Sensor Setting:

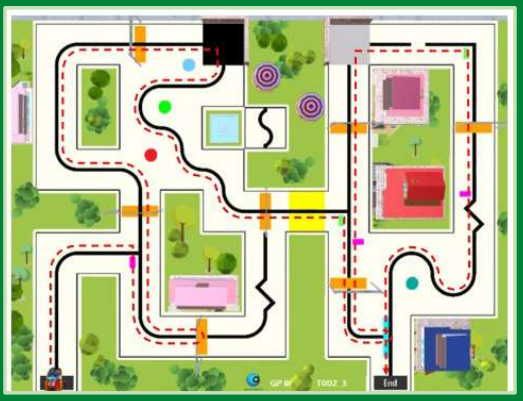


Challenge 2: Smart City

Analysis U19

- Deal with colour makers

When Robot senses …	Additional condition	Robot will …
Yellow patch	-	Move forward
Green marker	-	Turns right for 90 degree
Cyan marker	First time	Make U-Turn
	2nd time	Ignore Cyan and keep following line
Pink marker	Before yellow patch	Turn right for 90 degree
	After yellow patch	Move straight forward (to avoid Zig-Zag road)
Orange marker	At station 1	Stop for 2 sec with LED flashing, then U-turn and move away
Orange marker	At station 2 – 7	Stop for 2 sec with LED flashing, then move forward.
On black floor	10 cm away from wall	Move forward
On light grey floor	10 cm away from wall	Move forward



Challenge 2: Smart City

Analysis U19

Self-defined variables:

1. After_1st_Station
   - Before the 1st station -> After_1st_Station = 0
   - After the 1st station -> After_1st_Station = 1
   - When the robot senses the 1st orange -> set After_1st_Station = 1

2. RHS
   - Before yellow patch -> RHS = 0
   - After yellow patch -> RHS =1
   - When the robot senses yellow -> set RHS -> 1

3. U-Turn
   - Before it senses cyan colour patch -> U-Turn = 0
   - After it senses cyan colour patch -> U-Turn = 1
   - When the robot it senses cyan colour patch -> set U-Turn = 1

Initial value assigned to each variable = 0

Challenge 2: Smart City

Analysis U19

- Deal with colour makers

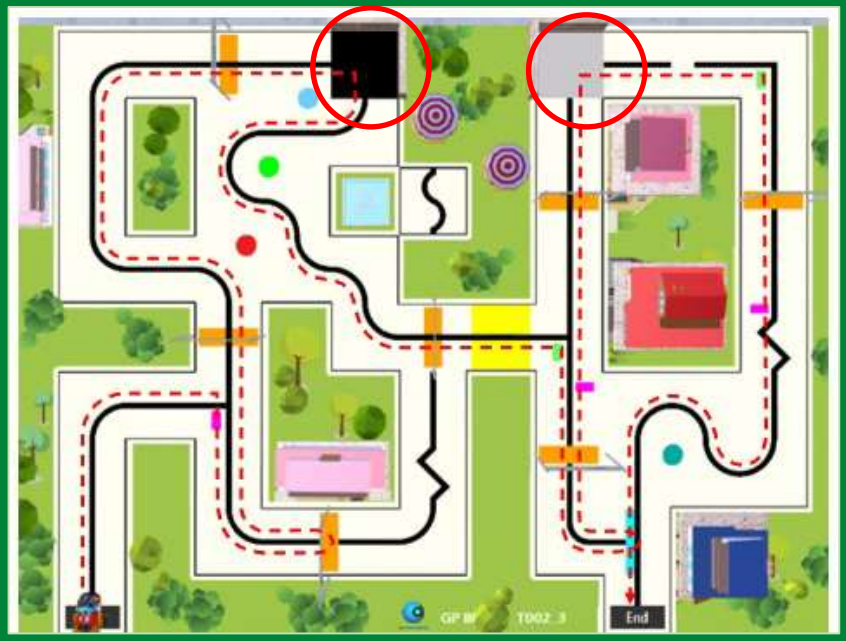
When Robot senses …	Additional condition	Robot will …
Yellow patch	-	Move forward
Green marker	-	Turns right for 90 degree
Cyan marker	U_turn_on_Cyan = 0	Make U-Turn
	U_turn_on_Cyan = 1	Ignore Cyan and keep following line
Pink marker	RHS = 0	Turn right for 90 degree
	RHS = 1	Move straight forward (to avoid Zig-Zag road)
Orange marker	After_1st_Station = 0	Stop for 2 sec with LED flashing, then U-turn and move away
Orange marker	After_1st_Station = 1	Stop for 2 sec with LED flashing, then move forward.
On black floor	Ultrasonic sensor: 10 – 255 cm	Move forward
On light grey floor		Move forward

Challenge 2: Smart City

From the diagram on the right, we can see that the robot has to turn 90 degrees to avoid the wall (obstacle).

We have two methods to do it:

- Method 1 – Working with both wheel speeds and timing.
- Method 2 – Using compass sensor

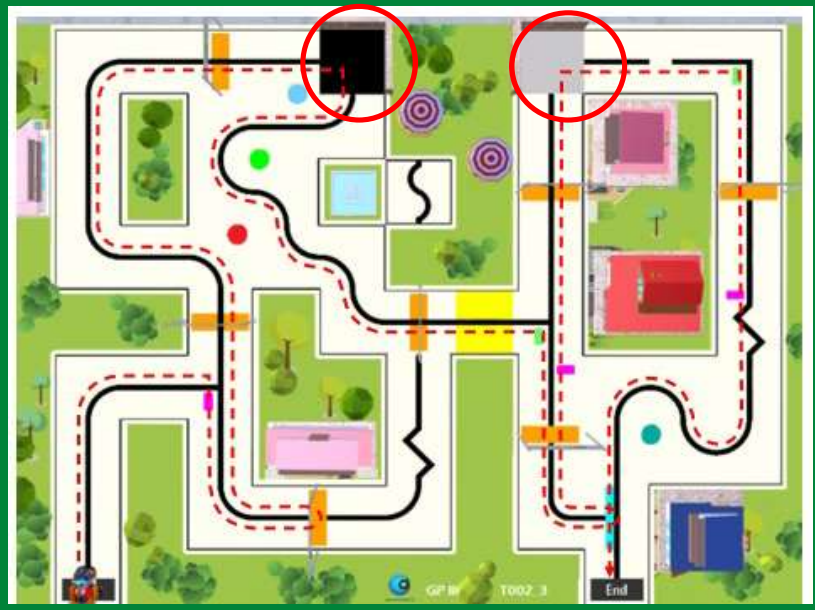


Challenge 2: Smart City

90 Degree Turn U19

- **Method 1 – Working with both wheel speeds and timing.**

In this case, the robot needs to continuously turn 90 degrees without any interruption. This is controlled by both **wheel speeds** and action **Duration**.



Action Duration – Defines how long the action will run for. Within this period, the robot will ignore all other statements and continue running the specified action.

The minimum duration is 0.025 sec.

In this method, we need to

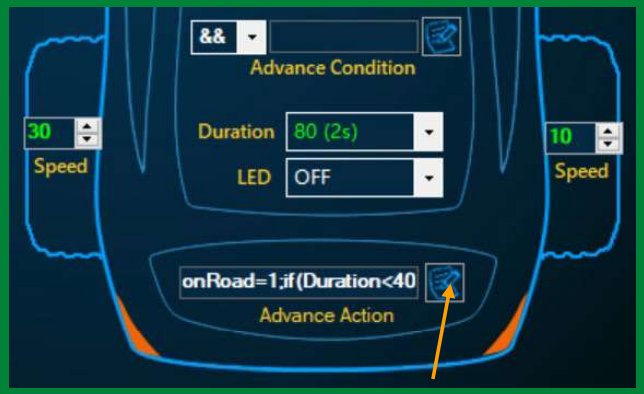
1) Decide both wheel speeds at the turning point
2) Decide on the correct duration in order to have a proper turn to avoid the wall in front.

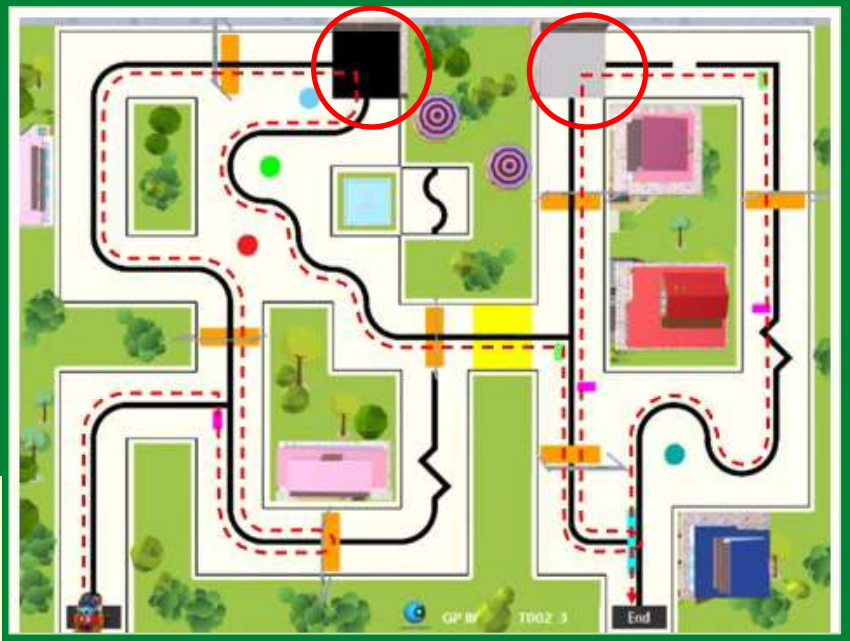
Challenge 2: Smart City

90 Degree Turn **U19**

- **Method 2 – Working with compass sensor**

An exact 90° turn can be made by using compass sensor






```
if(Duration<20 && (RotationZ%90 > 88 || RotationZ%90 <2))
{
   Duration =0;
}
```

Note:

- **RotationZ** indicates the compass' live data
- **'%' (modulus operator)** computes the remainder that results from performing an integer division.

  Example: 5 % 2 gives **1** because when we divide 5 by 2 we get 2 as quotient and 1 as the remainder.

Challenge 2: Smart City

Reference Code (1) **U19**

1 : Changing state

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	RHS==0
	RGB	R: 230 – 255 G: 230 – 255 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	RHS==1;
	LED	OFF	
	Duration	1 (0.025 s)	

2 : At 1st station

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	After_1st_Station==0
	RGB	R: 230 – 255 G: 150 – 180 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	0	Advanced Action

	Right wheel	0	if (Duration<20)
			{
			WheelLeft=50;
			WheelRight=-50;
			After_1st_Station=1;
			}

	LED	FLASH	
	Duration	100 (2.5 s)	

Challenge 2: Smart City

Reference Code (2) **U19**

3 : At station 2 - 7

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	After_1st_Station==1
	RGB	R: 230 – 255 G: 150 – 180 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	0	Advanced Action

	Right wheel	0	if (Duration<20)
			{
			WheelLeft=30;
			WheelRight=30;
			}

	LED	FLASH	
	Duration	100 (2.5 s)	

4 : Turn right before wall

Condition	Ultrasonic	0 – 10	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 230 – 255 G: 230 – 255 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	15	Advanced Action

	Right wheel	-15	if ((Duration<40) &&
			(RotationZ%90>88)
			|| (RotationZ%90<2))
			
			{
			Duration=0;
			}

	LED	OFF	
	Duration	80 (2 s)	

Challenge 2: Smart City

Reference Code (3) **U19**

5 : U-Turn on cyan

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	U_Turn_on_Cyan==0
	RGB	R: 0 – 30 G: 230 – 255 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	50	Advanced Action
	Right wheel	-50	U_Turn_on_Cyan=1;
	LED	OFF	
	Duration	20 (0.5 s)	

6 : Forward on pink (RHS)

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	RHS==1
	RGB	R: 230 – 255 G: 0 – 30 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	–
	LED	OFF	
	Duration	60 (1.5 s)	

Challenge 2: Smart City

Reference Code (4) **U19**

7 : Turn right on pink (LHS)

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 230 – 255 G: 0 – 30 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action

	Right wheel	10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2))
			
			{
			Duration=0;
			}

	LED	OFF	
	Duration	80 (2 s)	

8 : Turn right on green

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 80 – 120 G: 230 – 255 B: 80 – 120	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action

	Right wheel	10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2))
			
			{
			Duration=0;
			}

	LED	OFF	
	Duration	80 (2 s)	

Challenge 2: Smart City

Reference Code (5) **U19**

9 : Forward on light grey floor

Condition	Ultrasonic	10 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 185 – 215 G: 185 – 215 B: 185 – 215 	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	–
	LED	OFF	
	Duration	1 (0.025 s)	

10 : Forward on black floor

Condition	Ultrasonic	10 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 0 – 30 G: 0 – 30 B: 0 – 30 	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	–
	LED	OFF	
	Duration	1 (0.025 s)	

Challenge 2: Smart City

Reference Code (6) U19

11 : Tracking black line			
Condition	Ultrasonic	Using "my_First_Program" as a reference, write 6 statements for tracking the black line You may need to adjust the wheel speeds to achieve better performance	Advanced Condition
	IR		–
	RGB		
	Time		
	RotationX		
	RotationZ		
Action	Left wheel		Advanced Action
	Right wheel		–
	LED		
	Duration		

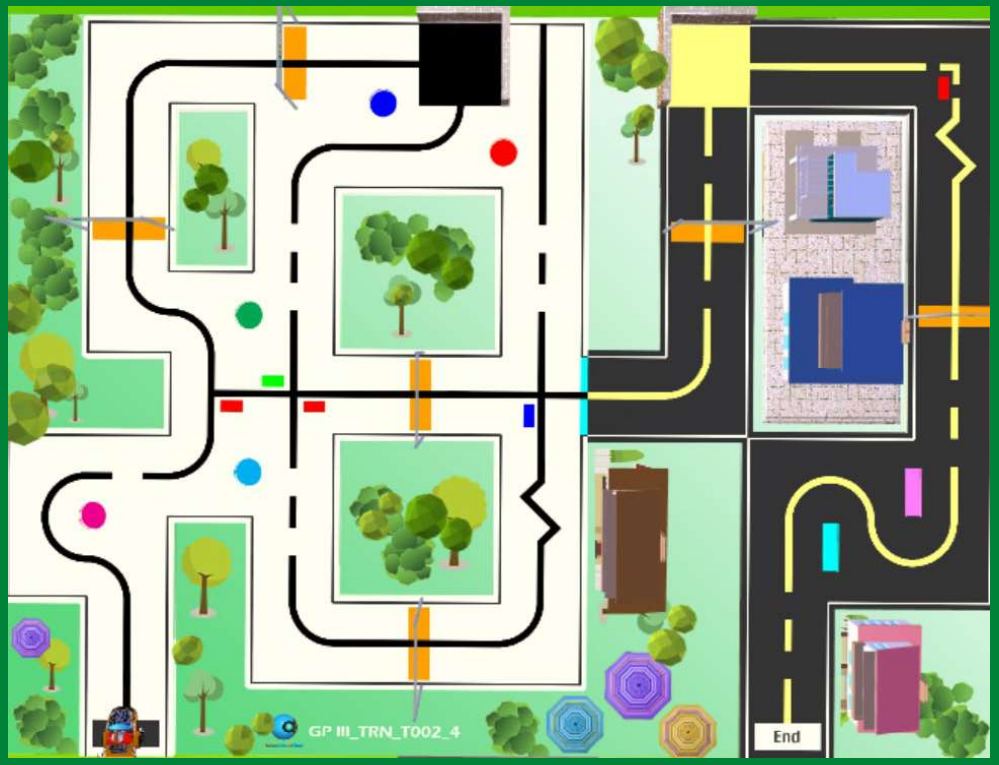
Note:

There are many different methods and logics to program the robot to complete the assigned task. Please free to explore!

Sample Code:
U19_Challenge_2

Challenge 3: Eco-Garden

U19



: Stations

Task:

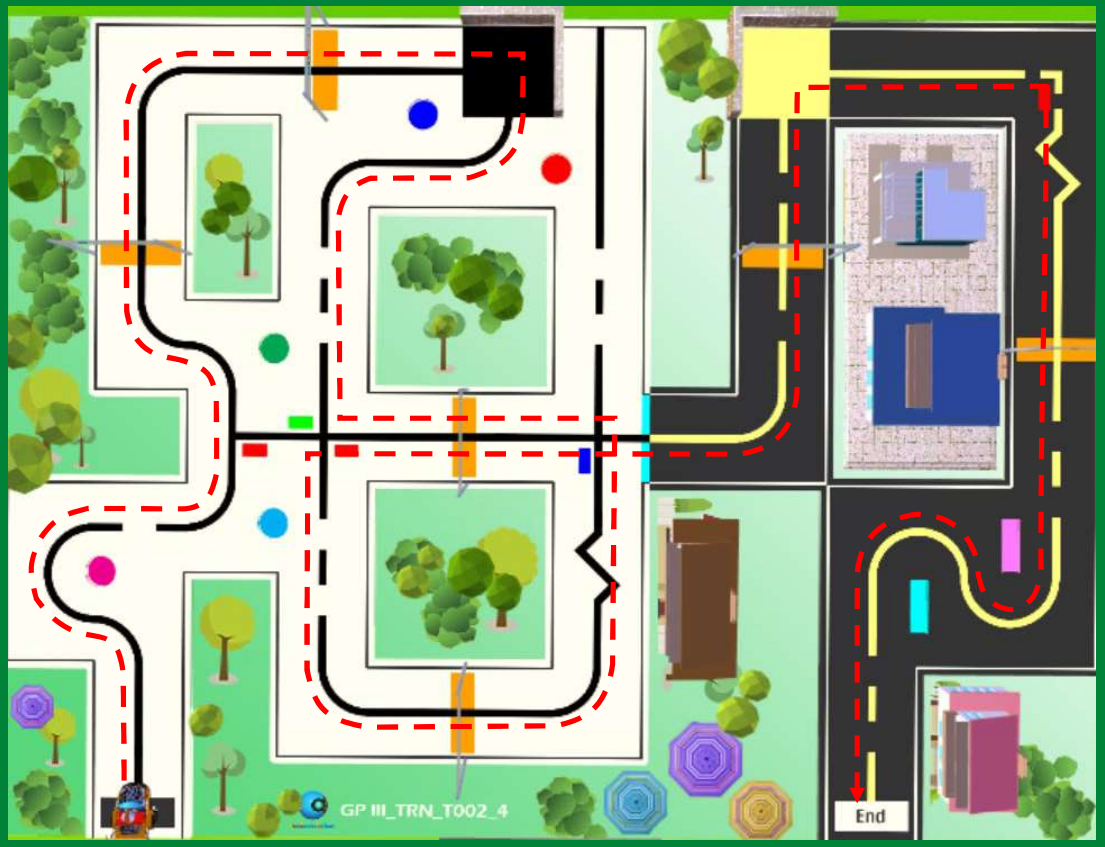
To program our robot to travel towards the destination. During its journey, It must stop at each pit station for 2 seconds with LED flashing, and then move away from the station autonomously.

Virtual Map:

Challenge 3: Eco-Garden

Analysis U19

- Task
  - The robot needs to follow the black line to “END”
  - The robot needs pass all stations.
  - The robot must stop at each station for 2 seconds with LED flashing, and then move away from the station autonomously.
- Route
  - We have designed the route as indicated.



Challenge 3: Eco-Garden

Analysis U19

Self-defined variables:

1. After_1st_Red
   - On start -> After_1st_Red = 0
   - After the 1st red marker -> After_1st_Red = 1
   - When the robot senses the 1st red marker -> set After_1st_Red = 1

2. After_Blue
   - Before blue marker -> After_Blue = 0
   - After passing the blue marker (1st time) -> After_Blue =1
   - When the robot senses the 1st blue marker -> set After_Blue =1

Self-defined variables:

3. OnBlackRoad
   - Before cyan marker -> OnBlackRoad = 0
   - After cyan marker -> OnBlackRoad = 1
   - When the robot senses the cyan marker -> set OnBlackRoad = 1

4. GoStraight
   - When GoStraight = 1, robot will move straight forward for 1.5 sec. GoStraight will be set to 1 when robot passes the 1st blue marker, and 3rd red marker. This is to avoid the zig-zag road. The GoStraight will be reset to 0 at the end of 1.5 sec.

Initial value assigned to each variable = 0

Challenge 3: Eco-Garden

Reference Code (1) **U19**

1 : At all stations

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 230 – 255 G: 150 – 180 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	0	Advanced Action
	Right wheel	0	GoStraight=0;
	LED	FLASH	

	Duration	100 (2.5 s)	if (Duration<20)
			{
			WheelLeft=30;
			WheelRight=30;
			}

2 : Change state

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	OnBlackRoad==0
	RGB	R: 0 – 30 G: 230 – 255 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	OnBlackRoad=1;
	LED	OFF	
	Duration	100 (2.5 s)	

Challenge 3: Eco-Garden

Reference Code (2) **U19**

3 : Turn right before wall

Condition	Ultrasonic	0 – 10	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	_
	RGB	R: 0 – 255 G: 0 – 255 B: 0 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	10	Advanced Action

	Right wheel	-10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2))
			{
			Duration=0;
			}

	LED	OFF	
	Duration	100 (2.5 s)	

4 : Forward on 1st red

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	(After_1st_Red==0)&& (OnBlackRoad==0)
	RGB	R: 230 – 255 G: 0 – 30 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	After_1st_Red=1;
	LED	OFF	
	Duration	20 (0.5 s)	

Challenge 3: Eco-Garden

Reference Code (3) U19

5 : Turn right on 2nd red

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	(After_Blue==1)&& (OnBlackRoad==0)
	RGB	R: 230 – 255  G: 0 – 30 B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2)) { Duration=0; }
	LED	OFF	
	Duration	100 (2.5 s)	

6 : Turn left on green

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	–
	RGB	R: 0 – 30 G: 230 – 255  B: 0 – 30	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	10	Advanced Action
	Right wheel	30	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2)) { Duration=0; }
	LED	OFF	
	Duration	100 (2.5 s)	

Challenge 3: Eco-Garden

Reference Code (4) **U19**

7 : Turn right on blue

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	(After_Blue==0)&& (OnBlackRoad==0)
	RGB	R: 0 – 30 G: 0 – 30 B: 230 – 255 	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2)) { Duration=0; After_Blue=1; GoStraight=1; }
	LED	OFF	
	Duration	100 (2.5 s)	

8 : Turn right on 3rd red

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	OnBlackRoad==1
	RGB	R: 230 – 255 G: 0 – 30 B: 0 – 30 	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	10	if ((Duration<40) && (RotationZ%90>88) || (RotationZ%90<2)) { Duration=0; GoStraight=1; }
	LED	OFF	
	Duration	100 (2.5 s)	

Challenge 3: Eco-Garden

Reference Code (5) **U19**

9 : Forward on blue again

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	(After_Blue==1)&& (OnBlackRoad==0)
	RGB	R: 0 – 30 G: 0 – 30 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action
	Right wheel	30	–
	LED	OFF	
	Duration	20 (0.5 s)	

10 : Move straight forward

Condition	Ultrasonic	0 – 255	Advanced Condition
	IR	⦸⦸⦸⦸⦸⦸	GoStraight==1
	RGB	R: 230 – 255 G: 230 – 255 B: 230 – 255	
	Time	0 – 1000	
	RotationX	- 90 – 90	
	RotationZ	0 – 360	
Action	Left wheel	30	Advanced Action

	Right wheel	30	if (Duration<1)
			{
			GoStraight=0;
			}

	LED	OFF	
	Duration	80 (2 s)	

Challenge 3: Eco-Garden

Reference Code (6) **U19**

11 – 16 : Tracking black line

Condition	Ultrasonic	Using "my_First_Program" as a reference, write 6 statements for tracking the black line	Advanced Condition
	IR		OnBlackRoad==0
	RGB		
	Time		
	RotationX		
	RotationZ		
Action	Left wheel		Advanced Action
	Right wheel		–
	LED		
	Duration		

17 – 22 : Tracking white line

Using "step 11 – 16" as a reference, write 6 statements for tracking the white line. We need to set "**OnBlackRoad==1**" as the advanced condition for all 6 statements.

23 : Forward

The robot will move forward at "all default sensor values". Both wheels' speeds can be set as +30.

Note:

There are many different methods and logics to program the robot to complete the assigned task. Please explore by yourselves.

Sample Code: U19_Challenge_3

4

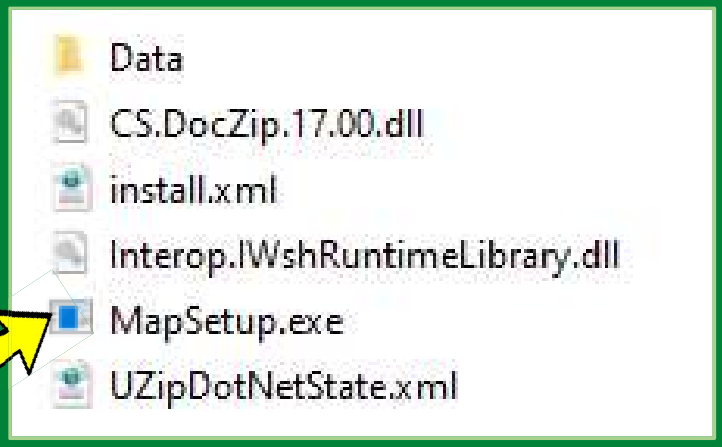
CoSpace Online Challenge

(iCooLChallenge)

iCooL Challenge Procedure

U19

- Download the challenge map (a zip file) from the “iCooL server” as provided by the organizer.
- Extract the downloaded zip file .
- Double click on the .exe file in the unzipped map folder



iCooLChallenge Procedure

Launch iCooLChallenge U19

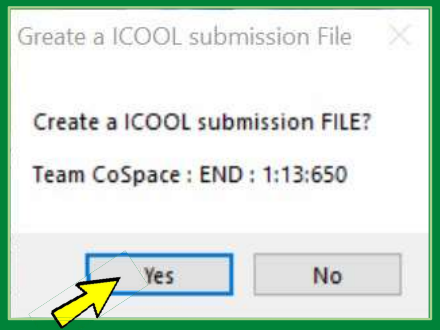
iCooLChallenge Procedure

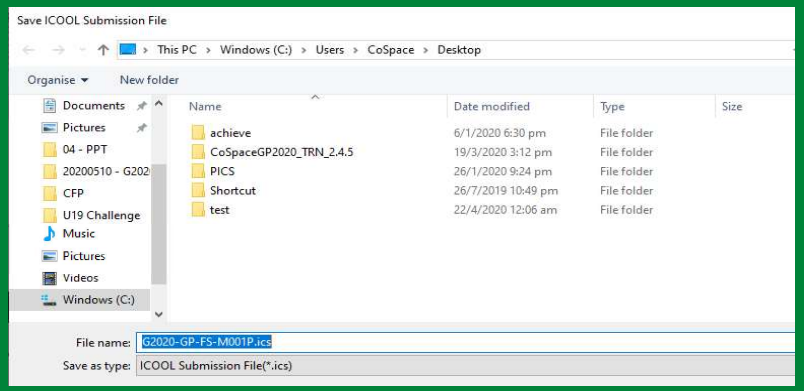
U19

- Program
  - Program your robot according to the task given
- Load the program ".dll" file in control panel

- Select "Virtual" Mode.

- Execute the program.
- Create a " .ics" file when the robot reaches the finish line or when you click "STOP" button.





- Send the " . ics" to the iCooL server.

5

Appendix 1:

Introduction to CoSpace Grand Prix Challenge Platform

Real Robot vs Virtual Robot

Digital Twins

Real Robot ⟷ Virtual Robot

The real and virtual robots are equipped with the same sensors
They have the same physical models (except for the floor friction and lighting conditions)
The same program works for both the virtual and real robots

Robot Sensors

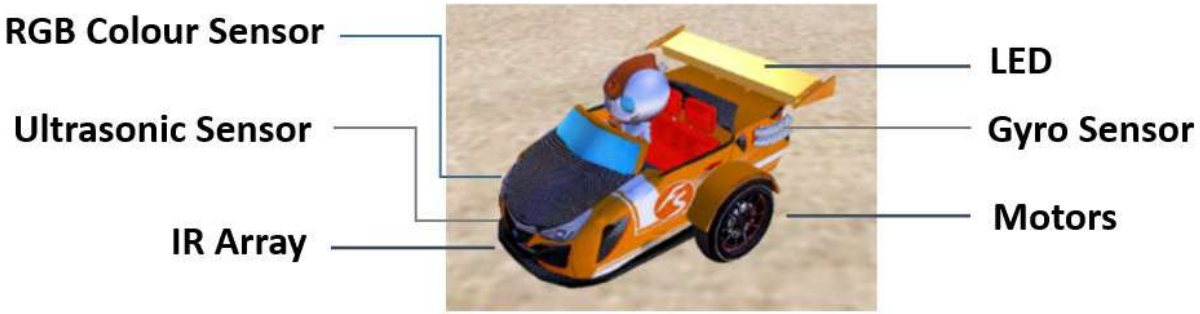
U19

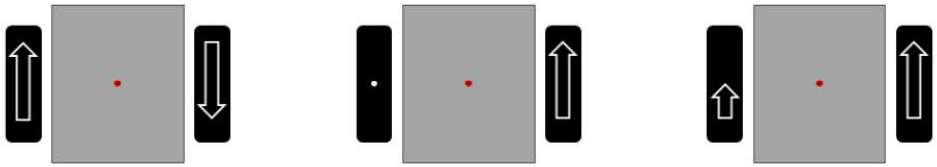
Sensors

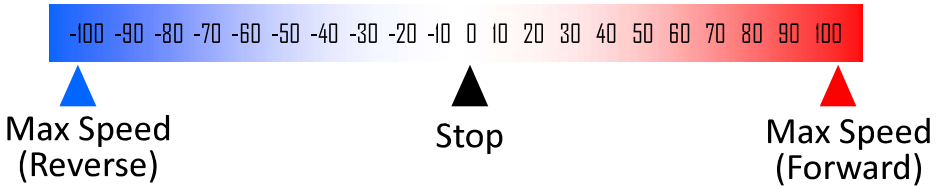
- 1 ultrasonic sensor is fitted at the front of the robot. The ultrasonic sensor detects objects in front of the robot. If it detects any object, it will output the distance to the nearest object. Otherwise, it will output the maximum range of the sensor. Unit used is in cm.
- 1 line-following sensor array, comprising 6 IR sensors, is mounted at the bottom of the robot, towards the front. These sensors are used to detect the colour of the ground beneath the robot. For each sensor, if the ground is light-coloured, it will output WHITE; otherwise, it will output BLACK.
- 1 RGB colour sensor, is mounted at the front right corner of the robot, facing downwards. Like the line sensors, the RGB sensor detects the colour of the ground. However, unlike the line sensors, it outputs the exact colour it detects, in terms of the Red (R), Green (G) and Blue (B) light intensities. Each value can range from 0 to 255.
- 1 Gyro sensor is used to detect the robot's orientation (direction and tilt). In the RCAP CoSpace Grand Prix Challenge, the gyro sensor is only available to U19 category participants.

Robot Actuators

U19







LED (Output)

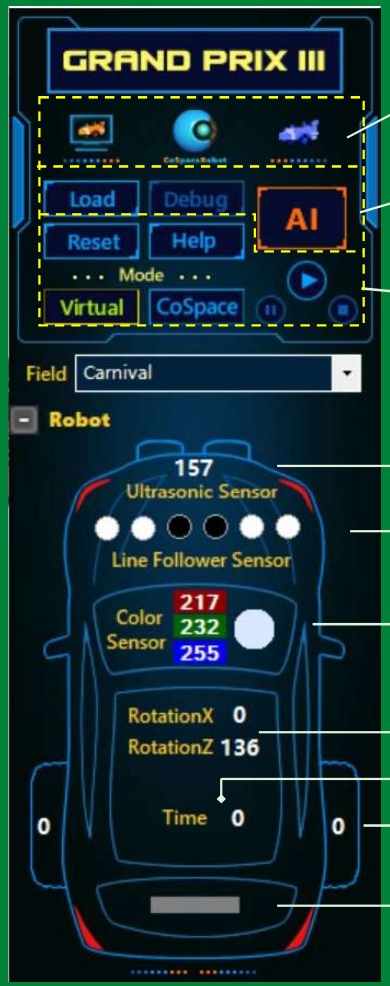
- 1 LED is used to provide visual indication of the robot's status when it is in action.
- This does not have any direct impact on the robot's performance; its main purpose is to facilitate program testing.

Motors (Output)

- 2 independently controlled motors provide power to the robot. These are simulated direct-drive, fixed-gear DC motors. Each motor can be assigned its own power setting, ranging from -100 (full reverse) to 100 (full forward). At a power setting of 0, the motor will be idle.

Control Panel

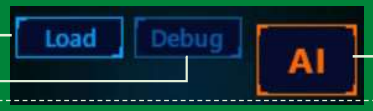
U19



Robot selection (Virtual/Real)

Load a virtual program

Launch debug panel



Launch AI panel

Reset Robot

Launch Help File

Challenge Modes

Pause/Run/Stop

Ultrasonic sensor's live data (cm)

IR sensors' live data (Black or White)

RGB sensor's live data (Red, Green, and Blue Channel values: 0 - 255)

Gyro sensor's live data. RotationX – Tilt (-90° = +90°), RotationZ – Compass (0° – 359°)

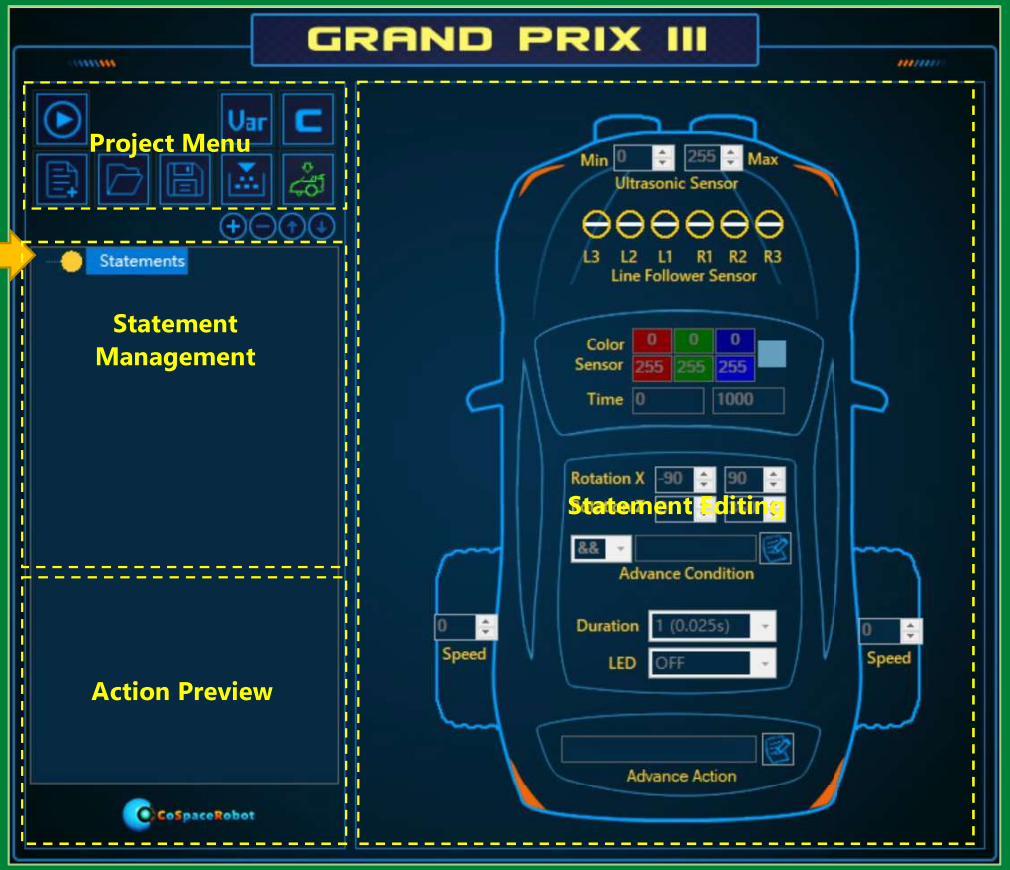
Timer (0 ~ 1000)

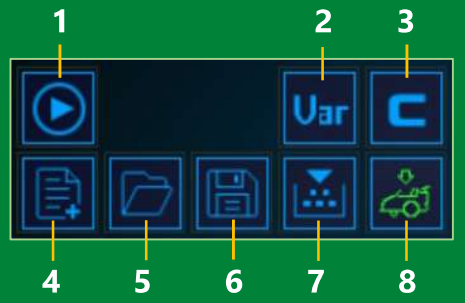
Motor speed (-100 ~ +100)

LED

AI Panel

U19

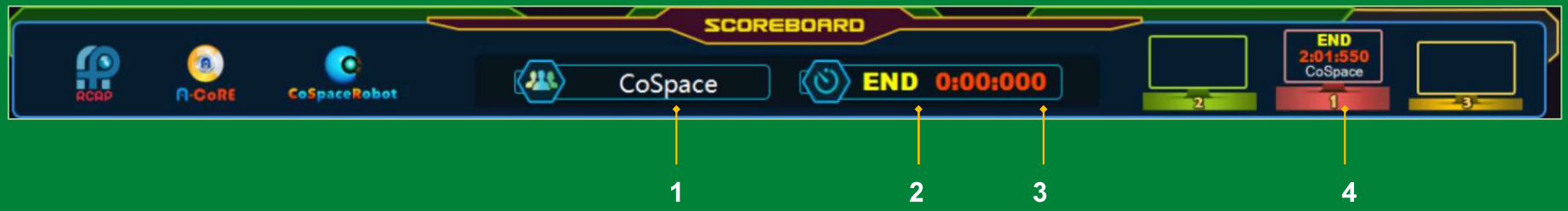




1. Trial RUN
2. Define new variables
3. View C code
4. Create a new project
5. Open an existing project
6. Save the current project
7. Save and build the current project
8. Upload to real robot

Score Board

U19



Team Name: Displays the team name

Zone: Displays the virtual zones travelled.

Time: Records the overall robot movement time since start of the race, including that of the real robot if a real robot started the race

Rank: Only results of the top three teams will be recorded.

6

Appendix 2:

CoSpace Grand Prix

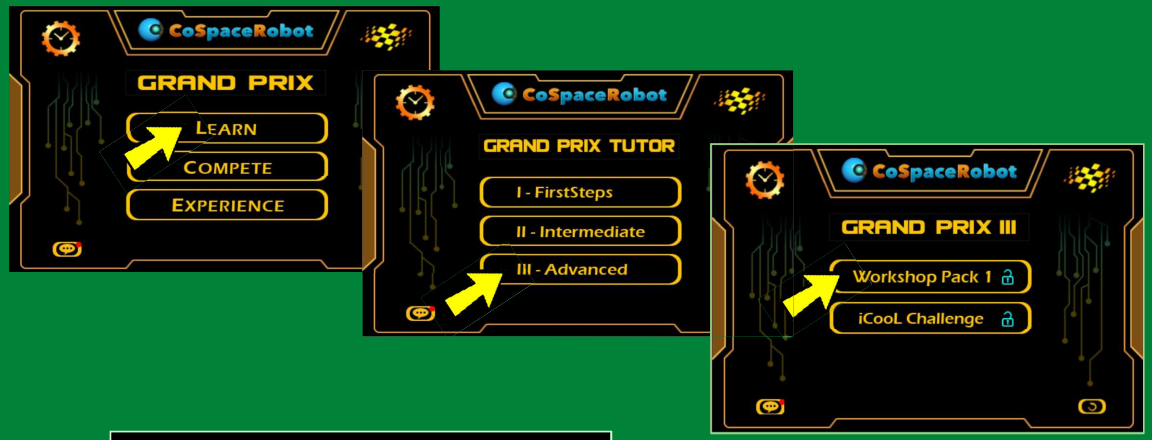
&

C Code

Graphic UI Coding and C Code

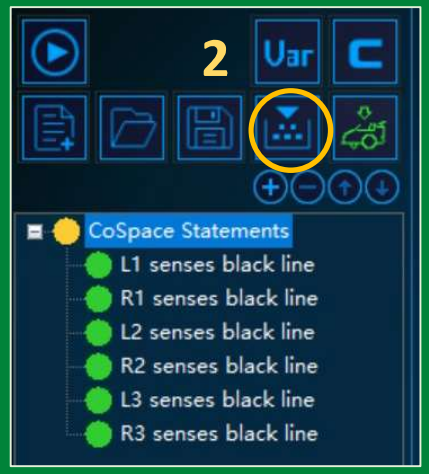
U19

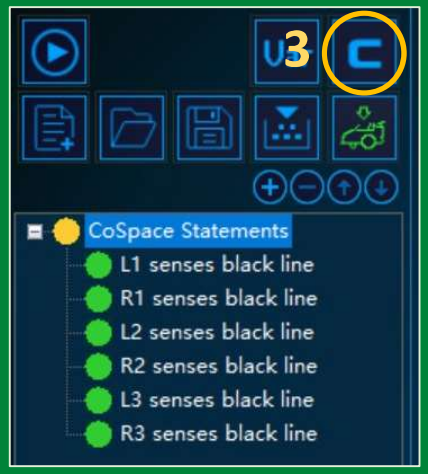
- Review My First Program:



Graphic UI Coding and C Code

1) Open “My_First_Program.smp” in AI
2) Click on “**build**”
3) View the corresponding “C” code






```
1  //////////////////////////////////////
2  //
3  //  File : ai.c
4  //  CoSpace Robot
5  //  Version 1.0.0
6  //  Jan 1 2016
7  //  Copyright (C) 2016 CoSpace Robot. All Rights Reserved
8  //
9  //////////////////////////////////////
10 //
11 // ONLY C Code can be compiled.
12 //
13 //////////////////////////////////////
14
15 #define CsBot_AI_H//DO NOT delete this line
16 #ifndef CSBOT_REAL
17 #include <windows.h>
18 #include <stdio.h>
19 #include <math.h>
20 #define DLL_EXPORT extern __declspec(dllexport)
21 #define false 0
22 #define true 1
23 #endif
24 //The robot ID : six chars unique CID.
25 //Find it from your CoSpace Robot label or CoSpace program download GUI.
26 //Don't write the below line into two lines or more lines.
27 char AI_MyID[6] = {'1','2','3','4','5','6'};
28
29 int Duration = 0;
30 int CurAction = -1;
31 int CurGame = 0;
32 int MyState_1 = 0;
33 int US_Front = 0;
34 int IR_L3 = 0;
35 int IR_L2 = 0;
36 int IR_L1 = 0;
37 int IR_R1 = 0;
```

Ln 1 : 209 Col 1 Sel 0 4.73 KB ANSI CR+LF INS C/C++

C Code

U19

If the C code file is not able to be open by notepad or word pad automatically and the there are some red txt appearing in the command window, it means the c programming file is not associated with notepad or WordPad.

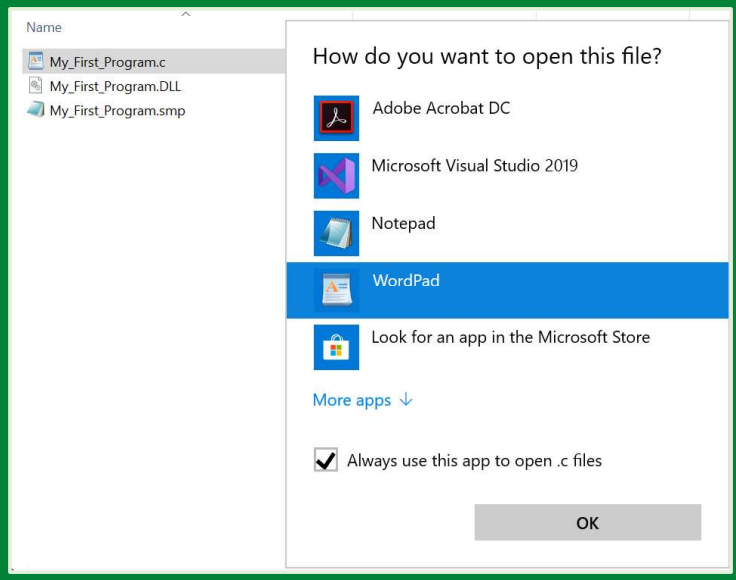
Below are the steps to solve the issue:

1. Locate the My_First_Program.c

   Most likely, the file is in:-

   C:\Microsoft Robotics Dev Studio 4\CS.C\User\GP

2. Open the file using WordPad



3. Now, you will see that the file is open.

4. The C program file is now successfully associated with the WordPad.

C Code

U19

You can use “My_First_Program.c” as a template to write your own C code.

The C code is structured as follows:

- Section 1: Header file
- Section 2: Variable definition
- Section 3: Interface with virtual robot (cannot be changed)
- Section 4: Game 0 ()

```
int Duration = 0;
int CurAction = -1;
int CurGame = 0;
int MyState_1 = 0;
int US_Front = 0;
int IR_L3 = 0;
int IR_L2 = 0;
int IR_L1 = 0;
int IR_R1 = 0;
int IR_R2 = 0;
int IR_R3 = 0;
int CS_R = 0;
int CS_G = 0;
int CS_B = 0;
int RotationX = 0;
int RotationY = 0;
int RotationZ = 0;
int Time = 0;
int WheelLeft = 0;
int WheelRight = 0;
int LED_1 = 0;
int AI_SensorNum = 14;
int AI_TeamID = 1; //Robot Team ID. 1:Blue Ream; 2:Red Team.
```

You can add more variables here

```
void Game0()
{

    if(Duration>0)
    {
        Duration--;
    }
    else if(IR_L2>=0 && IR_L2<=0)
    {
        Duration = 0;
        CurAction =1;
    }
    else if(IR_R1>=0 && IR_R1<=0)
    {
        Duration = 0;
        CurAction =2;
    }
    else if(IR_L2>=0 && IR_L2<=0)
    {
        Duration = 0;
        CurAction =3;
    }
    else if(IR_R2>=0 && IR_R2<=0)
    {
        Duration = 0;
        CurAction =4;
```

You can write you own code in this section

C Operators (1)

U19

CoSpace Grand Prix accept all C programming syntax. Below are some C operators frequently used.

Arithmetic Operators

Operator	Description	Example
+	Adds two operands.	A + B = 30
–	Subtracts second operand from the first.	A – B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Relational Operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

C Operators (2)

U19

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

If you need to know more C programing concepts, there are plenty of materials online and offline.

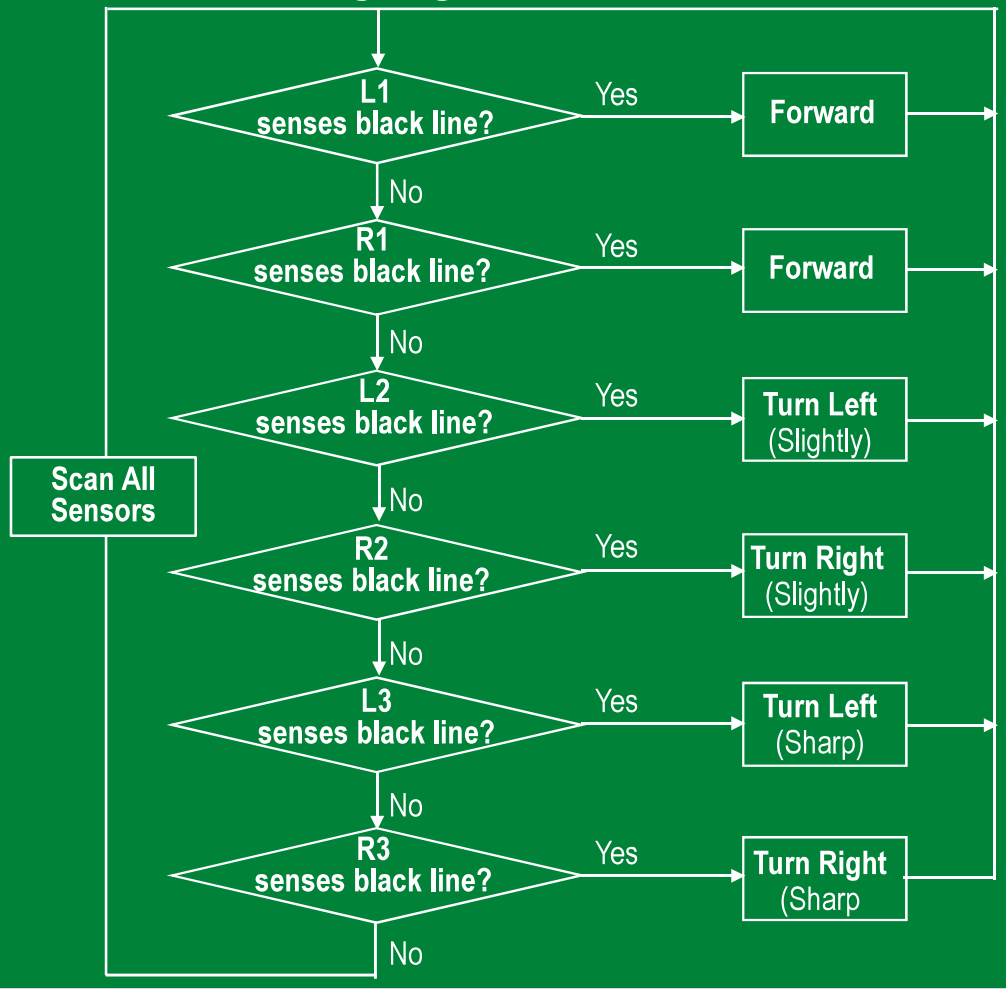
Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A

Using if-else statement (1)

U19

- Line Following Logic:



- The corresponding if-else statement:

```
if (IR_L1 == 0)
{
  WheelLeft = 30;
  WheelRight = 30;
}
else if (IR_R1 == 0)
{
  WheelLeft = 30;
  WheelRight = 30;
}
else if (IR_L2 == 0)
{
  WheelLeft = 10;
  WheelRight = 30;
}
```

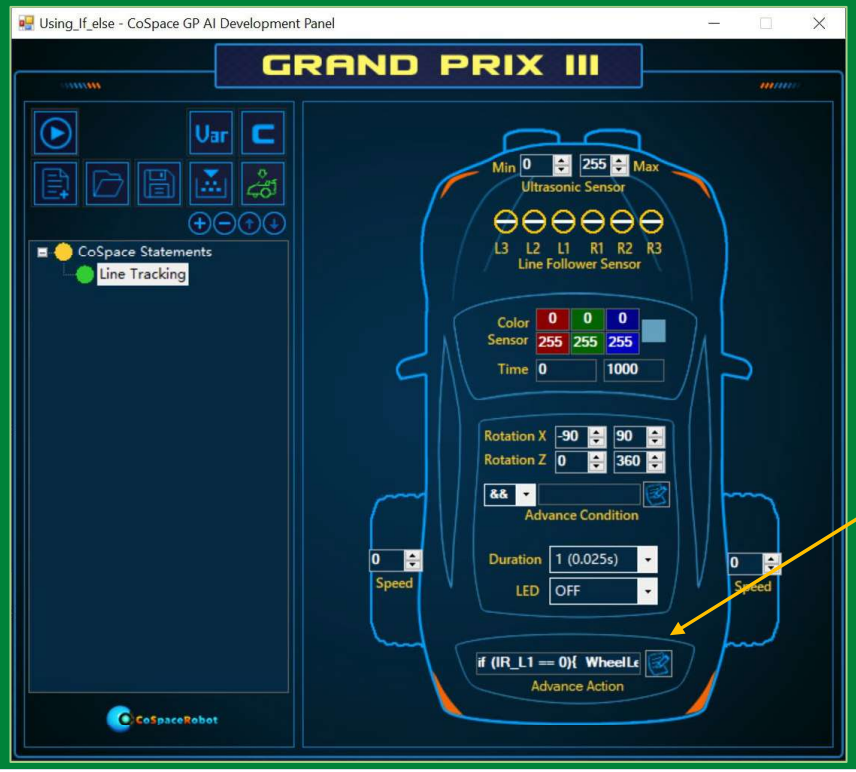
```
else if (IR_R2 == 0)
{
  WheelLeft = 30;
  WheelRight = 10;
}
else if (IR_L3 == 0)
{
  WheelLeft = -10;
  WheelRight = 30;
}
else if (IR_R3 == 0)
{
  WheelLeft = 30;
  WheelRight = -10;
}
```

Using if-else statement (2)

U19

- Create a new project: "**Using_if_else**"
- Add a new statement name: "**Line tracking**"
- Add the if-else statement in the advanced action code editor.

	Statement Name	All sensors	Advanced Condition	Left Wheel / Right Wheel / LED	Duration	Advanced Action
1	Line Tracking	Default	-	Default	0.025 s	



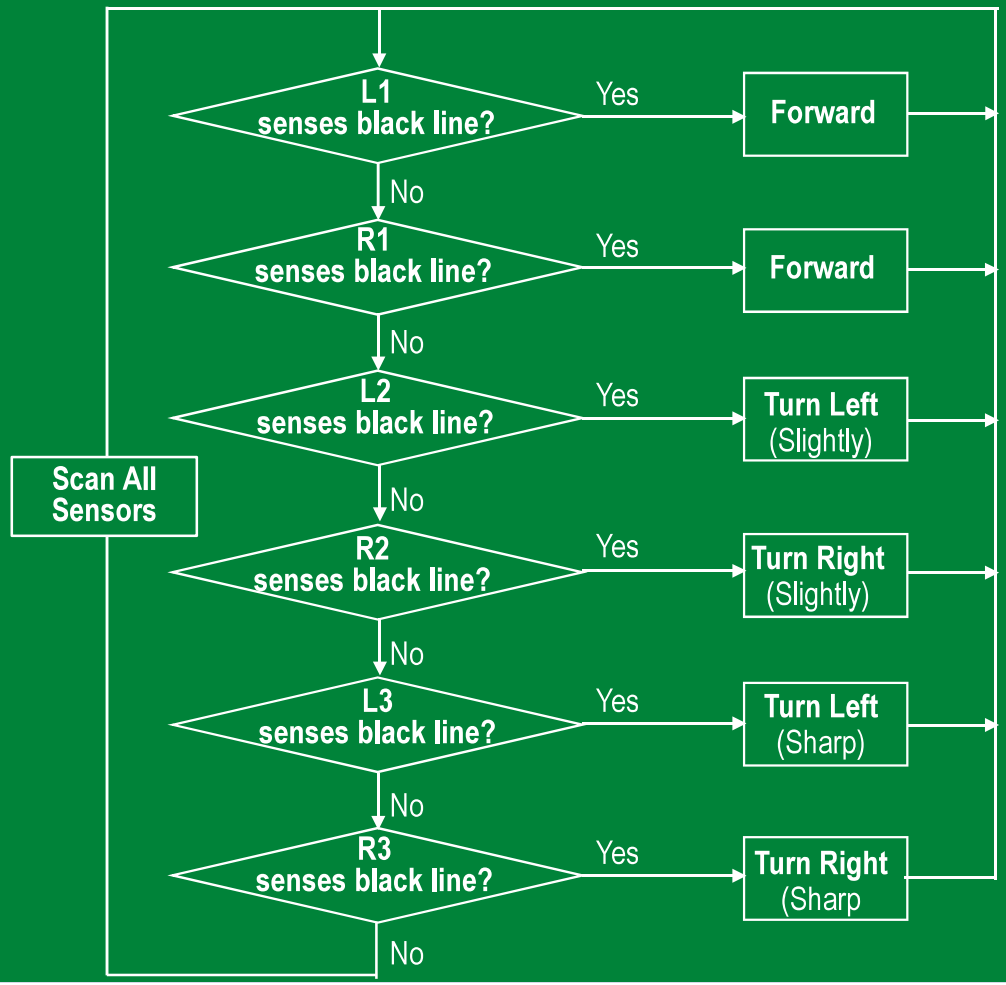

```
if (IR_L1 == 0)
{
  WheelLeft = 30;
  WheelRight = 30;
}
else if (IR_R1 == 0)
{
  WheelLeft = 30;
  WheelRight = 30;
}
else if (IR_L2 == 0)
{
  WheelLeft = 10;
  WheelRight = 30;
}
```

```
else if (IR_L2 == 0)
{
  WheelLeft = 10;
  WheelRight = 30;
}
else if (IR_R2 == 0)
{
  WheelLeft = 30;
  WheelRight = 10;
}
else if (IR_L3 == 0)
{
  WheelLeft = 30;
  WheelRight = -10;
}
```

Sample Code: U19_Using_If_else

Using switch-case (1)

- Line Following Logic:



- The corresponding switch-case statement:

```
if (IR_L1 == 0)
{
   Action = 1;
}
else if (IR_R1 == 0)
{
   Action = 2;
}
else if (IR_L2 == 0)
{
   Action = 3;
}
else if (IR_R2 == 0)
{
   Action = 4;
}
else if (IR_L3 == 0)
{
   Action = 5;
}
else if (IR_R3 == 0)
{
   Action = 6;
}
```

```
switch (Action)
{
  case 1:
      WheelLeft = 30;
      WheelRight = 30;
     break;
  case 2:
      WheelLeft = 30;
      WheelRight = 30;
      break;
  case 3:
      WheelLeft = 10;
      WheelRight = 30;
      break;
  case 4:
     WheelLeft = 30;
      WheelRight = 10;
      break;
  case 5:
      WheelLeft = -10;
      WheelRight = 30;
      break;
  case 6:
      WheelLeft = 30;
      WheelRight = -10;
      break;
  default:
      break;
}
```

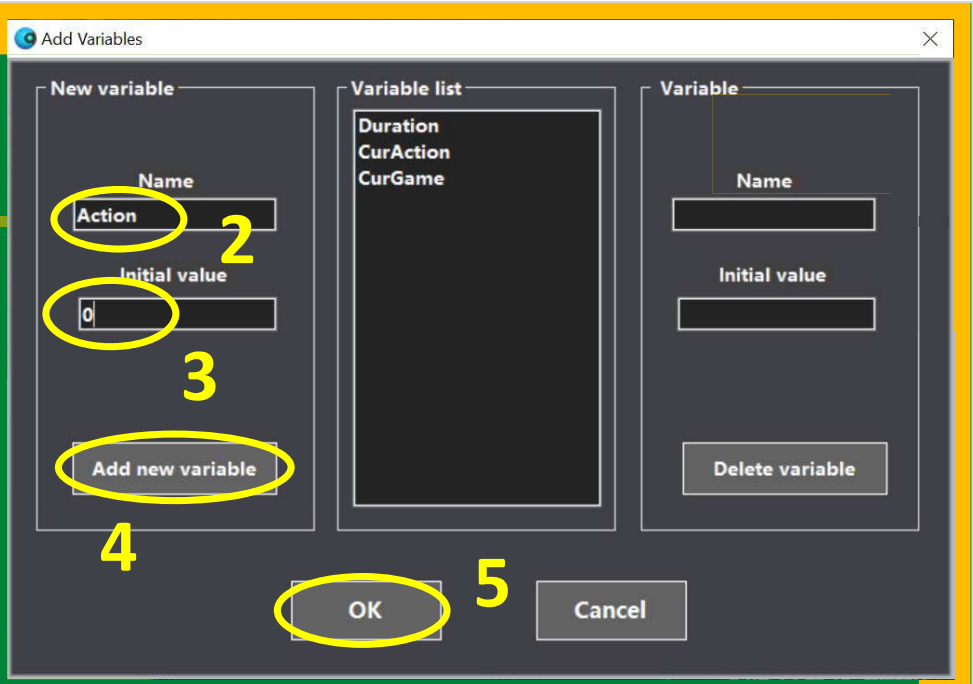
Using switch-case (2)

In the switch-case statement, we need to have a new variable: **Action**

- Procedure of adding new variable: **Action**

1. Select "**Var**" in AI panel

2. Define a new variable: Action
3. Assign initial value to variable "Action" created
4. Click on "Add new variable"
5. Confirm.



This is important. It will clear all hidden unused variables

If you wish to create a new program, you need follow the following procedure:

1. Select a field

2. Open AI

3. Create a new project

4. Add new variables.

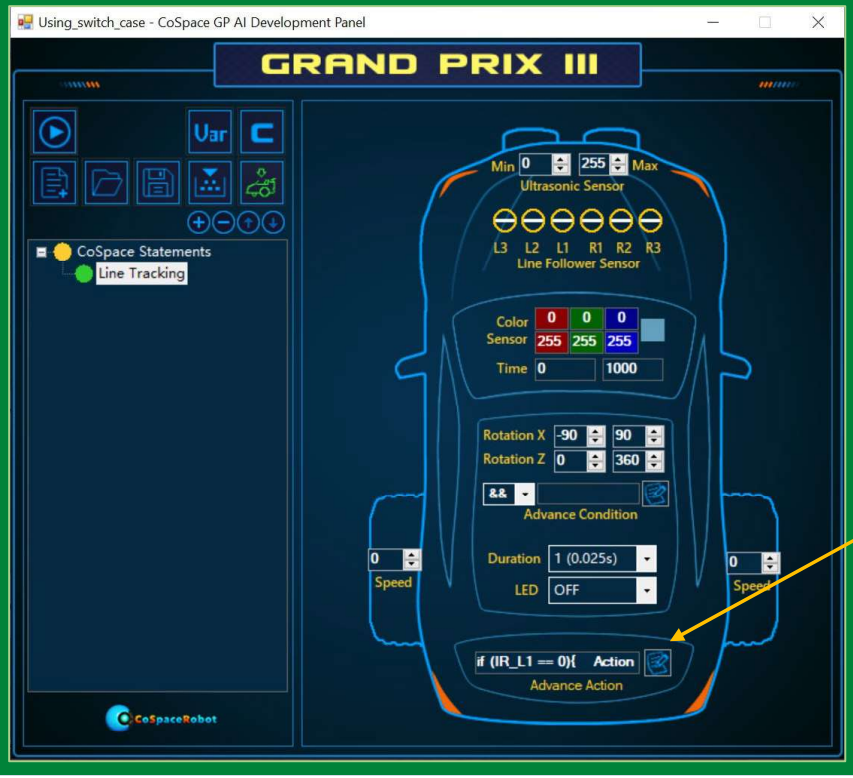
5. Work on the coding

Using switch-case (3)

Sample Code: U19_Using_switch_case

U19

- Create a new project: "**Using_switch-case**"
- Add a new statement name: "**Line tracking**"
- Add the switch-case statement in the advanced action code editor.



	Statement Name	All sensors	Advanced Condition	Left Wheel / Right Wheel / LED	Duration	Advanced Action
1	Line Tracking	Default	-	Default	0.025 s	

```
if (IR_L1 == 0)
{
   Action = 1;
}
else if (IR_R1 == 0)
{
   Action = 2;
}
else if (IR_L2 == 0)
{
   Action = 3;
}
else if (IR_R2 == 0)
{
   Action = 4;
}
else if (IR_L3 == 0)
{
   Action = 5;
}
else if (IR_R3 == 0)
{
   Action = 6;
}
```

```
switch (Action)
{
   case 1:
     WheelLeft = 30;
     WheelRight = 30;
     break;
   case 2:
     WheelLeft = 30;
     WheelRight = 30;
     break;
   case 3:
     WheelLeft = 10;
     WheelRight = 30;
     break;
   case 4:
     WheelLeft = 30;
     WheelRight = 10;
     break;
   case 5:
     WheelLeft = -10;
     WheelRight = 30;
     break;
   case 6:
     WheelLeft = 30;
     WheelRight = -10;
     break;
   default:
     break;
}
```

7

Appendix 3:

CoSpace Grand Prix, U19 Online Challenge (iCooLChallenge) Rules in Brief

CoSpace Grand Prix Rules in Brief

iCooLChallenge U19

- In the CoSpace Grand Prix Challenge (iCooL), students are only required code a virtual robot to complete the route in the virtual world.
- The maximum time the robot can stay in the virtual world is 8 min.

Virtual World

Game Process:

1. The virtual robot is placed at the “START” station at beginning of the game.
2. The robot must pass all pit stations in the virtual world in any sequence during its journey to the destination. It must stop at each pit station for 2 seconds with LED flashing, and then move away from the station autonomously.
3. The game ends when the virtual robot arrives at the finish line in the virtual world.

CoSpace Grand Prix Rules in Brief

- Ranking

	Situation	Rank
Tier 1	○ VIRTUAL_ROBOT passes all Pit Stops and reaches the Finish Line	○ The team rank is determined by the race time at the Finish Line in the VIRTUAL_WORLD.
Tier 2	○ VIRTUAL_ROBOT is not able to passes all Pit Stops (regardless whether it reaches the Finish Line or not)	○ The race time for VIRTUAL_ROBOT to reach the last Pit Stop will be recorded. ○ The team rank will be determined based on the number of Pit Stops passed, followed by the race time.

Well done!
Welcome to the CoSpace Robot Family!

CoSpaceRobot ® is an official platform for both RoboCupJunior Rescue Simulation & RoboCup Asia-Pacific (RCAP) CoSpace Challenge.

http://robocupap.org/index.php/rcap-cospace-challenges/

Please follow our official social media pages for updates

facebook.com/cospacerobot
facebook.com/robocupap
facebook.com/robocupsg

instagram.com/robocupap

Email: CoSpace@robocupsingapore.org

RoboCup SINGAPORE