



Mobile maker spaces for promoting maker education in schools
democratizing STEAM education and innovation development for all the learners

IO2

Guidelines for teachers

Traffic light control system with Raspberry Pi and Scratch

Difficulty level: low

Reference age: from 14 to 18 years

Cost: currently quite high

Lead Partner: IIS Leonardo da Vinci

Authors: Renato Gatti & Loris Ranalli



Erasmus+

GRANT NUMBER: 2020-1-PL01-KA201-081698

Declaration

This file has been prepared in the context of the STEAM2GO project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

© Copyright 2020 - 2022 the [STEAM2GO](#) Consortium



All rights reserved.



This document is licensed to the public under a Creative Commons Attribution-Non Commercial - No Derivatives 4.0 International License.

Funding Disclaimer

This project has been funded with support from the European Commission. This report 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.

Guidelines for teachers

Traffic light control system with Raspberry Pi

Contents

Guidelines for teachers.....	1
Introductory notes.....	4
Study and realization of the timing diagrams	5
Set up the Raspberry Pi board.....	7
Let's create the basis of our first "Traffic Lights" program with Scratch..	8
Build our Traffic Lights using Scratch in “desktop mode” on the PC	10
Build our Traffic Lights program using external devices and GPIOs	14
The breadboard.....	15
Drawing and crafting the floor plan of two ways junction	20
Notes about button and switch connections	24
Notes about pull-up and pull-down resistors	25
Supporting files	26
Raspberry Pi guide	26

Introductory notes

About the project

We will start with the programming of a prototype built on a single bread board. Subsequently a model of the crossroad can be made on paper or cardboard, which will be placed on a sheet of polystyrene where rectangular holes - of suitable dimensions - will be made to house the four 170-points breadboards. These will then be connected to the breadboard with the Raspberry Pi board, button and switch. The various devices will then be connected to the relative GPIOs of the Raspberry Pi board and, finally, the entire system will be programmed.

Learning objectives

The students involved in the project will have the opportunities to:

- Understand and implement timing diagrams
- Understand an electronic circuit in detail
- Create circuits using the Raspberry Pi board and electronic components
- Be able to create a floor plan of a system by creating a paper model using everyday materials
- Learn the fundamentals of basic programming to implement a proposed model
- Analyze and identify the parameters and functions to be controlled in a complex system
- Interface, through the Raspberry Pi board GPIO, the I/O signals coming from the devices
- Develop, through the Scratch software, a complex control system
- Test the prototype
- Optimize and define the optimal work solution by sharing among the participants

Connection to other subject areas and/or specific STEM concepts

- Mathematics
- Technology
- Electronics
- Information Technology
- Realization of artifacts related to a technical-scientific project

Hardware and Software that you may need

Hardware	Software	Crafting material
- 3 x breadboards 170 points - https://amzn.eu/d/bcFYyKs - 1 x RPI 4 (or 3) - 1 x pushbutton - 1 x switch SPDT - Jumper wires kit - Wiring cables kit (M/M and M/F) - 2 x preassembled led traffic lights single kit: https://www.amazon.it/ZHITING-Display-semaforo-Mini-semaforo-elettronici/dp/B0859ZPM47	- Excel - Word - Scratch	- Some sheets of A4 (or A3) size white paper or paperboard - A sheet of expanded polystyrene - A sheet of alveolar polypropylene - 1 x Cutter - 1 x Scissor - 1 x Glue - 1 x Double-sided tape - 1 x Pencil - 1 x Pencil sharpener - Some fine permanent markers of various colors - Stationery equipment like pencils, pens, paper sheets

The working phases

1. The teacher may choose, based on the level of the students, one of the options presented both for the programming part and the realization of the plan of the road intersection.
2. The initial phase will be the identification of the correct functioning of a traffic light control. Particular attention must be paid to the timing of the signals and to safety measures to be adopted. This can be done through a quick Internet search guided by the teacher.
3. Subsequently the timing diagram of the system will have to be studied, initially in the simplest possible way and subsequently taking into account an overlap time of the red lights, in order to ensure that the intersection is free from any vehicles.
4. Setting up the Raspberry Pi Board
5. Create the first operating program using the Scratch software
6. Make all the necessary interconnections between the GPIOs of the Raspberry Pi board and the various external components
7. Transfer the program from the PC to the Raspberry Pi board
8. Design and create a plan of the road intersection where the bread boards and the various components will be positioned
9. Test the correct functioning of the system

Study and realization of the timing diagrams

The first step to carry out the project is to analyze and create a diagram with the timing of the various signals of each traffic light. For a correct understanding of each step, it's necessary to start with the simplest timing diagram without red lights overlapping. Once completed, the pros and cons are brought to the attention of the students.

The timing diagrams can be made on squared paper or using a spreadsheet or any graphics program.

a) Simplest Timing Diagram with NO red light overlap

The sequence of each traffic light will be, in the example shown in Fig.1:

- Green: 4 sec.
- Yellow: 1 sec.
- Red: 5 sec. (the time of the red signal must be equal to the sum of the green and yellow of the opposite traffic light)

In this case it is assumed that the green light of the traffic light [1], that we call G1, and the red light of the opposite traffic light [2], that we will call R2, light up simultaneously.

In turn, once the programmed time has elapsed, the green light of the traffic light [2], that we call G2, and the red light of the opposite traffic light [1], that we will call R1, light up simultaneously. Refer to **Figure 1**

b) Timing diagram with red lights overlap

In this case we considered the need to allow some time for the two red lights to overlap before triggering the green. This is to be sure that, once the green has been turned on, there is no longer any vehicle in the intersection.

This is the main safety criterion, required by road regulations and traffic law.

Now the sequence of each traffic light will be, in the example shown in Fig.2:

- Green: 4 sec.
- Yellow: 1 sec.
- Red: **7** sec. (the time of the red signal must be equal to the sum of the green and yellow of the opposite traffic light plus the overlapping time that we assumed as 1 sec each side). Refer to **Figure 2**

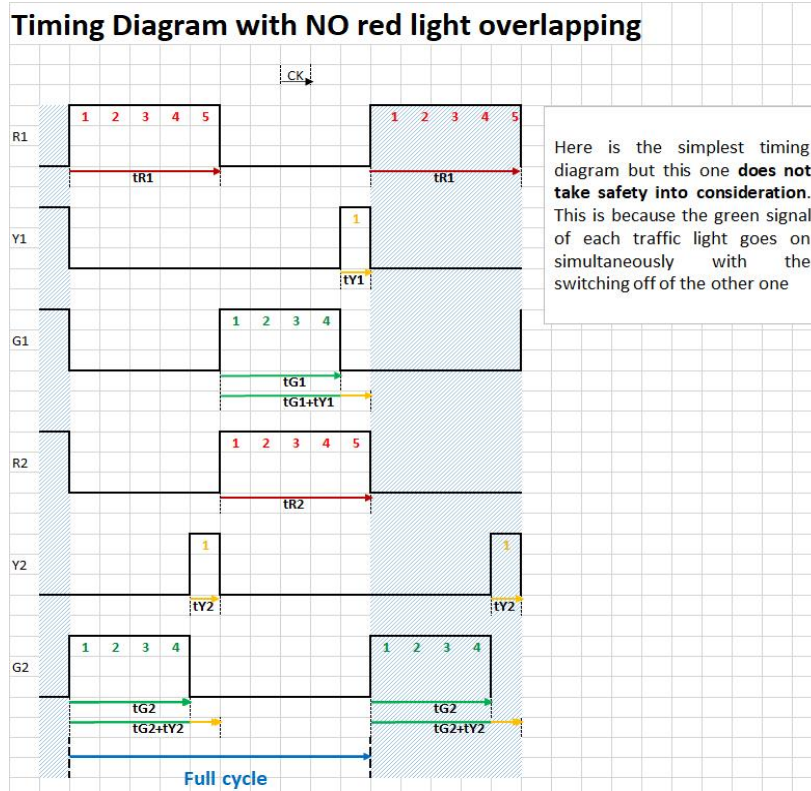


Figure 1

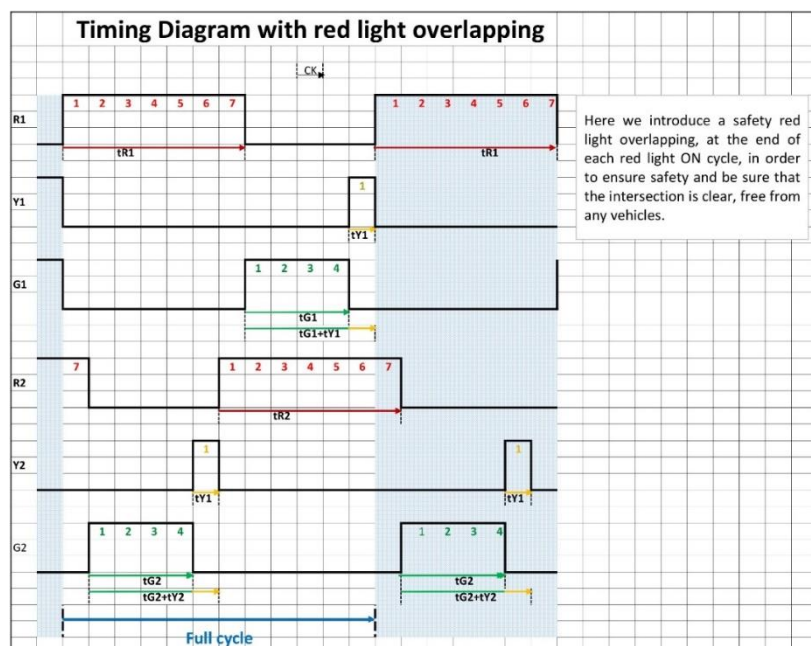


Figure 2

The teacher can intervene by raising safety issues, for example:

- Is an overlay of red lights necessary? If so, why?
- How long should the red lights overlap if necessary?
- On the basis of which parameters would you calculate the overlap time of the red lights?
- In the case of a pedestrian traffic light, how would you regulate the various times of the traffic light?
- Etc.

Clicking on the links below you can download the Timing Diagrams in various format:

Link to the Timing Diagrams realized using MS Excel:

- [Simplest Timing Diagram with NO red light overlap](#)
- [Timing diagram with red lights overlap](#)

Link to the Timing Diagrams in PDF format:

- [Simplest Timing Diagram with NO red light overlap](#)
- [Timing diagram with red lights overlap](#)

Link to the Timing Diagrams in JPG format:

- [Simplest Timing Diagram with NO red light overlap](#)
- [Timing diagram with red lights overlap](#)

Set up the Raspberry Pi board

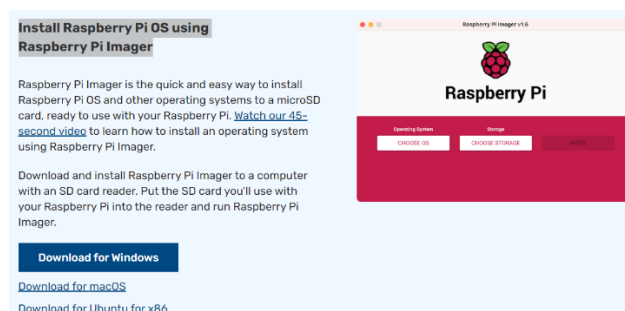
In order to make the Raspberry board work, you must first transfer the Raspbian operating system to an SD card of at least 16 GB.

It is possible to install Raspberry Pi OS using Raspberry Pi Imager, downloadable on the official Raspberry website at the following link:

<https://www.raspberrypi.com/software>

and following the step-by-step instructions therein (**Figure 3**)

Figure 3



With the same tool it will then be possible to make multiple copies of the SD card, once fully configured, if more cards are to be used with the identical setup.

Once Raspbian has been installed on the microSD card, you can proceed to insert it into the appropriate slot on the Raspberry board (**Figure 4**). Components layout is shown in **Figure 5**.

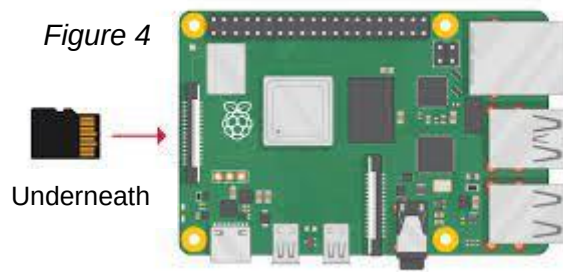
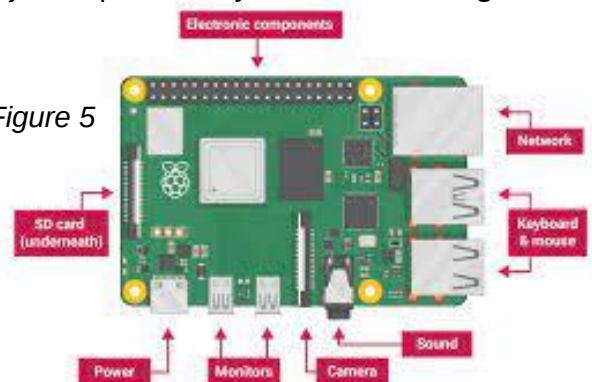


Figure 5



[Here](#) you will find a complete image of Raspberry with Scratch 3 pre-installed and the Scratch programs already on board (14 GB).

After inserting the microSD card in the appropriate slot, you can turn on the Raspberry board and connect it to the Internet using a network cable (recommended) or WiFi. The operating system will start and you can install “Scratch 3” on the board. The easier way to do it is to use the “Recommended Software app”:

Open up the menu, click on “Preferences” > “Recommended Software”, and then select “Scratch 3” and click on OK.

Here you will find the complete tutorial of Scratch for Raspberry:

<https://www.raspberrypi.org/blog/scratch-3-desktop-for-raspbian-on-raspberry-pi>

where you can find all the instructions and some programming examples in desktop mode.

Let's create the basis of our first "Traffic Lights" program with Scratch

Initially I recommend creating one or more programs in desktop mode on a PC in order to become familiar with the system.

The download of the Scratch environment for PC or Mac can be found at this link:

<https://scratch.mit.edu/download>

The first thing to do is to create a new file with Scratch and save it in a specific folder where we will save all the progress of the work.

Once we have created our file we can start by importing the drawing previously made on Scratch, which is called "**Backdrop**" and will represent our background on which we will place the two traffic lights.

You can create your own design or you can easily use this one already made one:

[Traffic light layout Backdrop.jpg](#)

As shown in **Figures 6 - 7 - 8**

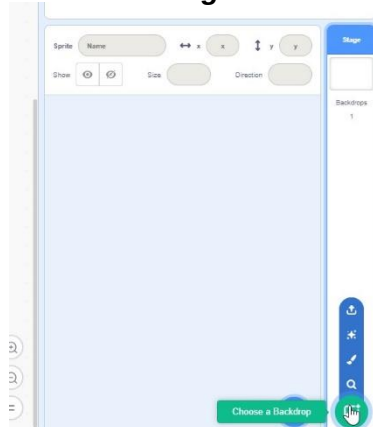


Figure 6

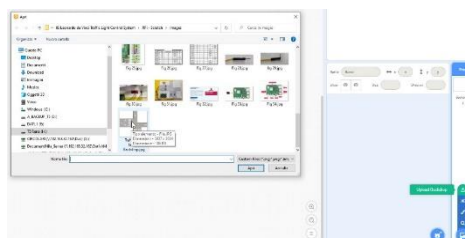


Figure 7

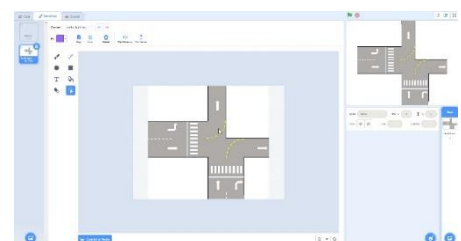


Figure 8

Here is a video explaining how to import the Backdrop:

[Video #1: How to import the backdrop](#)

After importing our Backdrop we will have to build the traffic light shape, that in Scratch is called **"Sprite"**. This will be duplicated as you need one for each of the two street of the intersection.

Each traffic light must be represented in the three different situations, so we have three situations for each Sprite: Red, Yellow and Green. These situations are represented by what Scratch calls **"Costumes"**.

Now we can create the graphic part relating to the Sprites and Costumes on Scratch.

To do this we can use two methods:

1. if we want to save time in the graphic construction part to invest more time to the programming part, you can use the Sprites already created for this project, as shown in **Figures 9 - 10 - 11 - 12**.

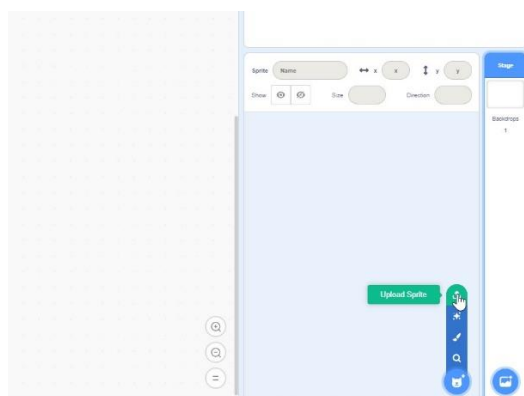


Figure 9

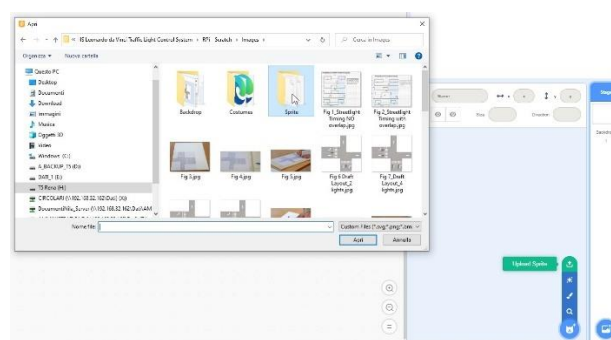


Figure 10

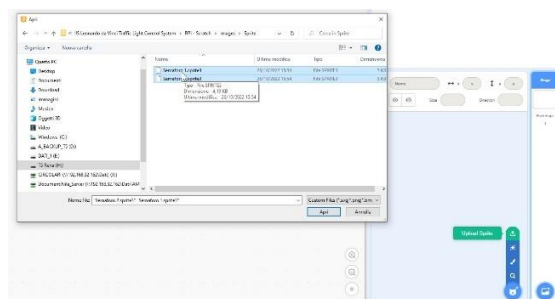


Figure 11

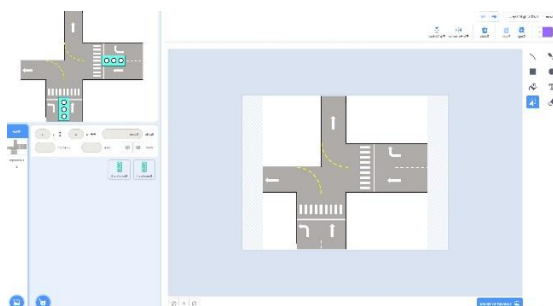


Figure 12

Here are the preassembled Sprites:

[Sprite TL 1](#) and [Sprite TL 2](#)

or following the video tutorial here:

[Video #2: How to import sprites and costumes](#)

2. on the other hand, if we want to invest a bit of time to the graphics design, we can draw the Sprites and Costumes manually, within the Scratch development environment. Here you will find a video tutorial on how to make them manually:

[Video #3: How to manually make Sprite and Costumes](#)

Once the graphic part has been created, we can start programming our traffic light with Scratch.

Build our Traffic Lights using Scratch in “desktop mode” on the PC

This phase is useful for those who are not familiar with Scratch programming, i.e. those who are approaching Scratch for the first time.

Various traffic light control programs for Scratch in desktop mode will be shown in ascending order of difficulty:

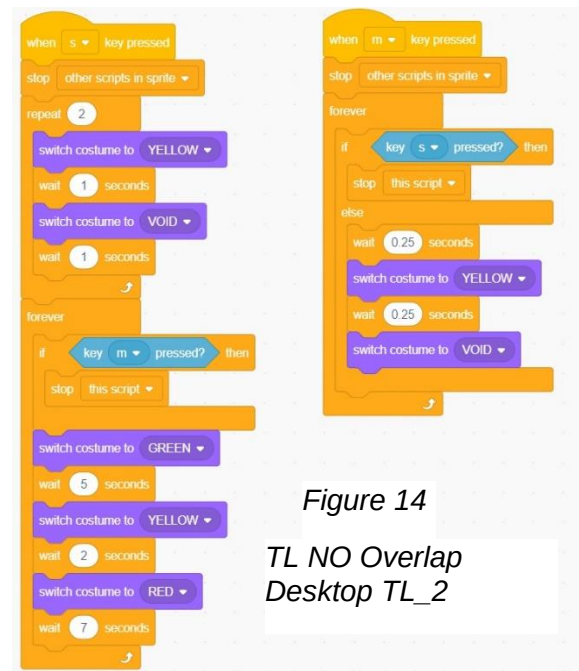
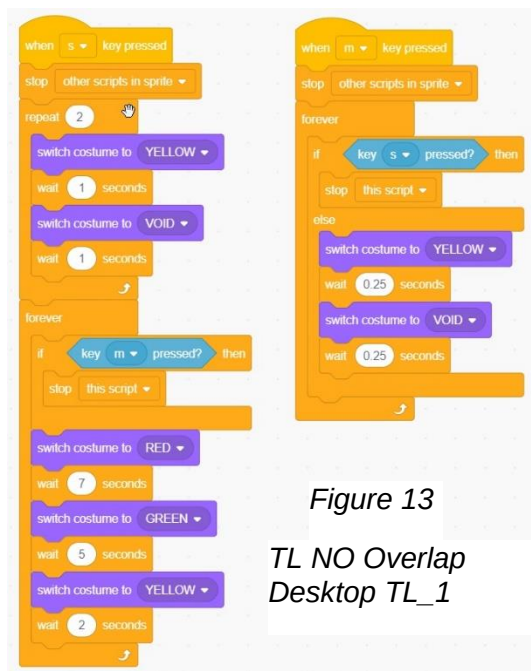
1. TL NO Overlap Desktop
2. TL Overlap Desktop
3. TL Overlap Desktop_Set timing
4. TL Overlap Desktop_Set timing_with Alert

1. TL NO Overlap Desktop

Here the first program which is very simple, controls the two traffic lights using two keys on the PC keyboard (“s” start and “m” maintenance) to activate the two scripts based on the button pressed:

- pressing the “s” key starts the sequence (in our example 2 “only initial” flashes of the yellow lights for 1 second then, continuously in loop, 7 seconds of red, 2 seconds of yellow and 5 seconds of green). Pay attention that the red time must be equal to the sum of the yellow time plus the green time in each Sprite
- pressing the “m” key you enter the “maintenance” mode which will make all the yellow lights flash for 0.25 seconds.
- pressing again the “s” key the sequence restarts from the beginning and so on.

In order to activate one or other function it is necessary to interrupt the currently running script. This is done using the “stop other script in sprite” instruction, as highlighted in the example program shown here:



[Program: TL NO Overlap Desktop](#)

[Screenshot TL_1](#)

[Screenshot TL_2](#)

[Video #4: TL NO Overlap Desktop](#)

2. TL Overlap Desktop

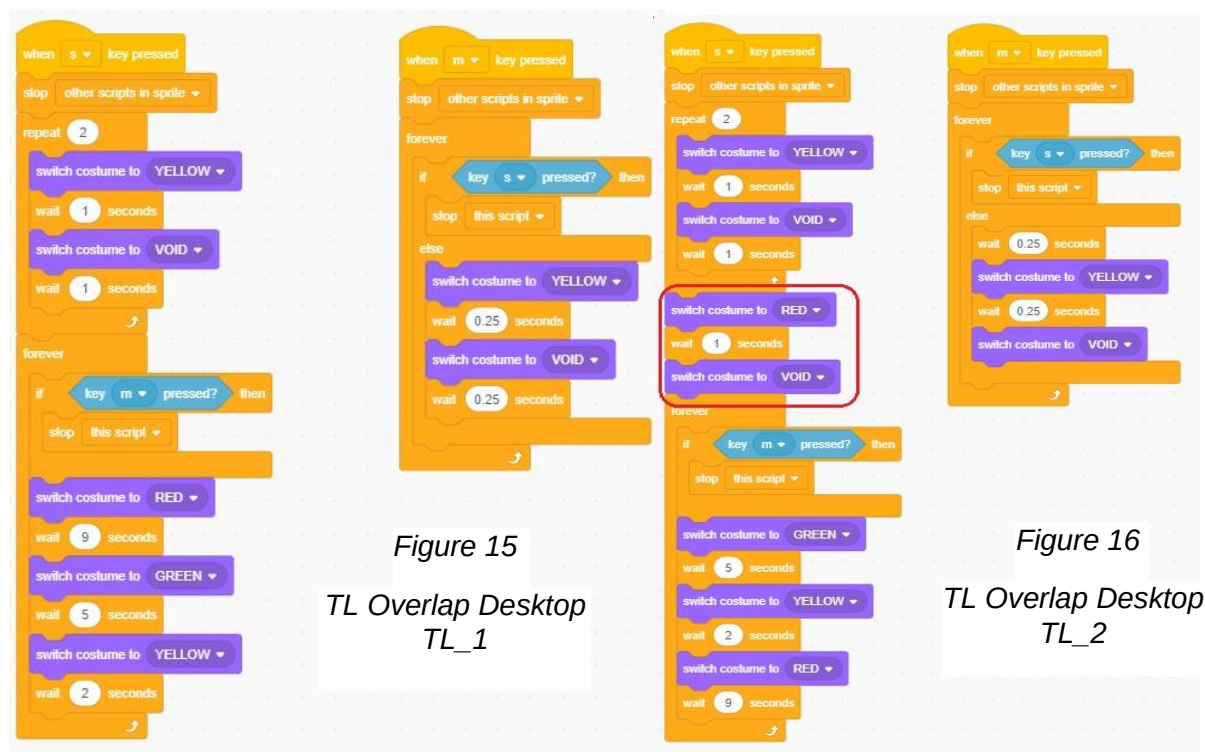
This program is the same as the previous one but we have simply inserted the overlap of the two red lights for a second.

This is achieved by keeping the red lights on in both Sprites for longer than the sum of yellow and green. In our example we have put green=5 seconds, yellow=2 seconds and red=9 seconds, therefore - in the previous example - 2 seconds higher than the 7. This results in a 1 second overlap of the red lights.

Pay attention to these notes:

- *the red time must be equal to the sum of green and yellow ones plus the overlap time*
- *at the beginning of the sequence the Sprite that will start with the green will have to start with a delay equal to the time of overlap (in our case 1 seconds).*

Here is the Scratch Code and some screenshots:



[Program: TL Overlap Desktop](#)

[Screenshot TL_1](#)

[Screenshot TL_2](#)

3. TL Overlap Desktop Set timing

Now we introduce to the program the possibility of being able to define, by the operator, the duration of the individual colours. To do this it is necessary to insert the "ask" and "answer" functions that we find under the "Sensing" group.

We have to create as many "ask" functions as the number of colours we want to program, that is three, RED, YELLOW and GREEN. The red lights in both Sprites have to remain "on" for longer than the sum of yellow and green.

Here are the Scratch Code and some screenshots of instructions for the new functions that will be used:

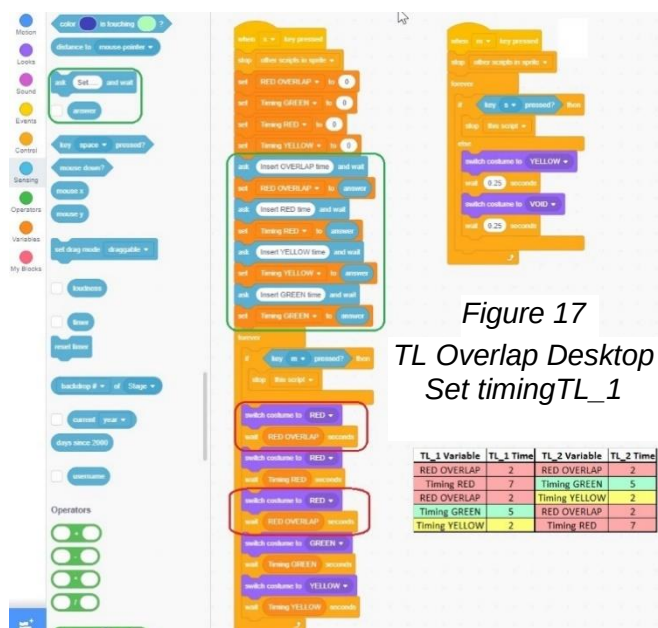


Figure 17

TL Overlap Desktop
Set timing TL_1



Figure 18

TL Overlap Desktop
Set timing TL_2

[Program: TL Overlap Desktop Set timing](#)

[Screenshot TL_1](#)

[Screenshot TL_2](#)

4. TL Overlap Desktop Set timing with Alert

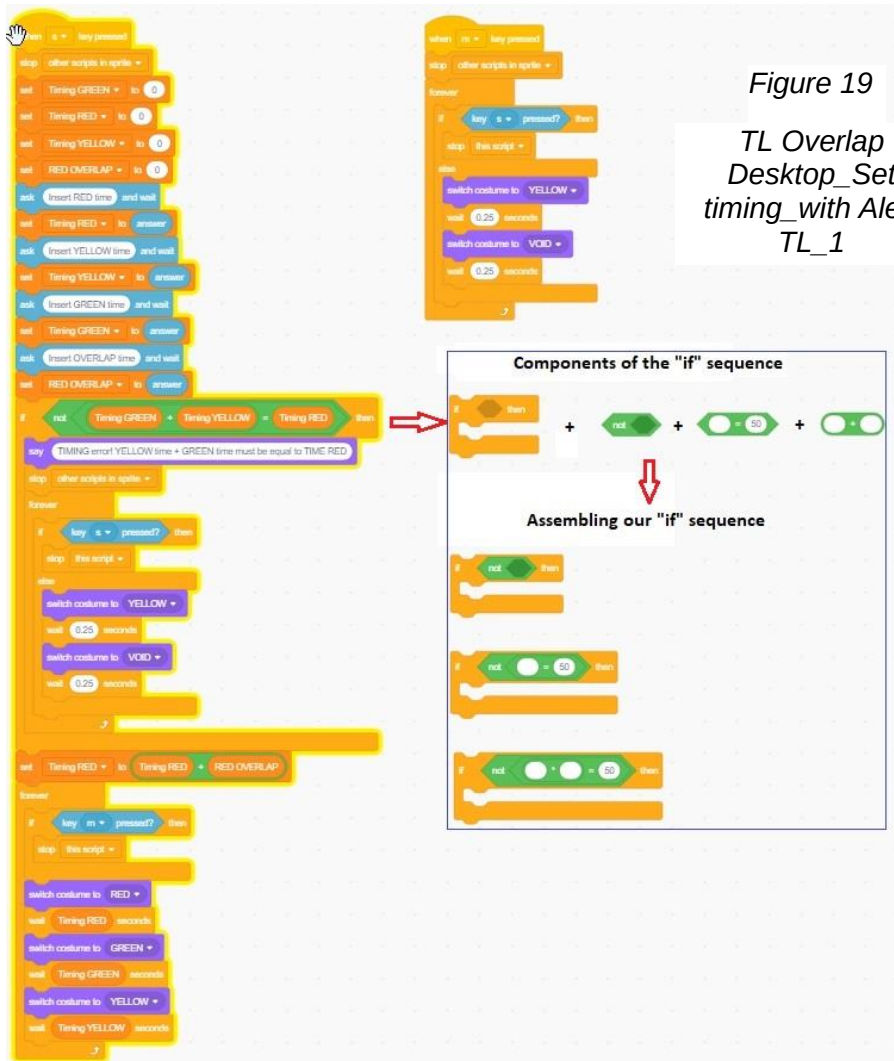
Another improvement can be made introducing an "alert" - just in case if the sum of the Green+Yellow times is not equal to the Red time. This is achieved using the "say" function belonging to the "Looks" group, as shown in the following screenshots and video.

[Program: TL Overlap Desktop Set timing with Alert](#)

[Screenshot TL_1](#)

[Screenshot TL_2](#)

[Video #5: TL Overlap Desktop Set timing with Alert](#)



Build our Traffic Lights program using external devices and GPIOs

Now we are going to connect the GPIOs of the Raspberry board to the external devices, in our case two pre-assembled traffic lights, a button and a switch.

Connecting the Raspberry GPIOs to external devices

The next step is to connect the Raspberry Pi GPIOs to the breadboards. To do this it is necessary to use the jumper wires, included in the STEAM2GO Mobile Maker Space. The Raspberry Pi GPIOs are shown in the following **Figure 21**

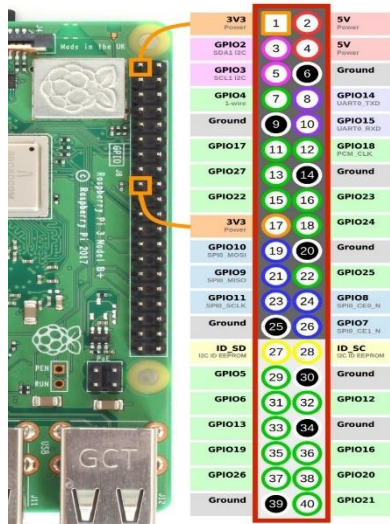


Figure 21 Raspberry Pi PinOut

It is convenient to make a spreadsheet with the interconnections between the Raspberry Pi GPIOs and the various traffic lights present in the respective breadboards. Examples are in **Figure 23** (two LEDs traffic lights) and **Figure 24** (four LEDs traffic lights). Note that in the example “four LEDs traffic lights” we simply duplicate the single ones. In **Figure 22** the input devices PinOut.

For the correct assembly of the button and the switch it is necessary to **check** the operation of the pins using the **tester** supplied with the Mobile Maker Space. See Notes about button and switch at [page 24](#).

Raspberry Pi		Input devices
GPIO	PIN	Type
GPIO17	11	Start Push-button
GPIO27	13	Maintenance switch
3V3	1	+Vcc

Figure 22
Input devices
pinout

Raspberry Pi		Led Traffic Lights		
GPIO	PIN	Traffic Light n.	Colour	Pin
GPIO10	19	1	Green 1	G
GPIO9	21	1	Yellow 1	Y
GPIO11	23	1	Red 1	R
Ground	25	1	---	GND
GPIO2	3	2	Green 2	G
GPIO3	5	2	Yellow 2	Y
GPIO4	7	2	Red 2	R
Ground	9	2	---	GND

Figure 23 two LEDs traffic lights Connection Table

Raspberry Pi		Led Traffic Lights		
GPIO	PIN	Traffic Light n.	Colour	Pin
GPIO10	19	1-1	Green 1-1	G
GPIO9	21	1-1	Yellow 1-1	Y
GPIO11	23	1-1	Red 1-1	R
Ground	25	1-1	---	GND
Ground	20	1-2	---	GND
GPIO25	22	1-2	Red 1-1	R
GPIO8	24	1-2	Yellow 1-1	Y
GPIO7	26	1-2	Green 1-1	G
GPIO2	3	2-1	Green 2-1	G
GPIO3	5	2-1	Yellow 2-1	Y
GPIO4	7	2-1	Red 2-1	R
Ground	9	2-1	---	GND
Ground	6	2-2	---	GND
GPIO14	8	2-2	Red 2-2	R
GPIO15	10	2-2	Yellow 2-2	Y
GPIO18	12	2-2	Green 2-2	G

Figure 24 four LEDs traffic lights Connection Table

Clicking on the links below you can download the Connection Table in various formats:

Link to the Connection Table in MS Excel format:

- [Connection Table Traffic Light 2 lights.xlsx](#)
- [Connection Table Traffic Light 4 lights - 2 x street.xlsx](#)

Link to the Connection Table in PDF format:

- [Connection Table Traffic Light 2 lights.pdf](#)
- [Connection Table Traffic Light 4 lights - 2 x street.pdf](#)

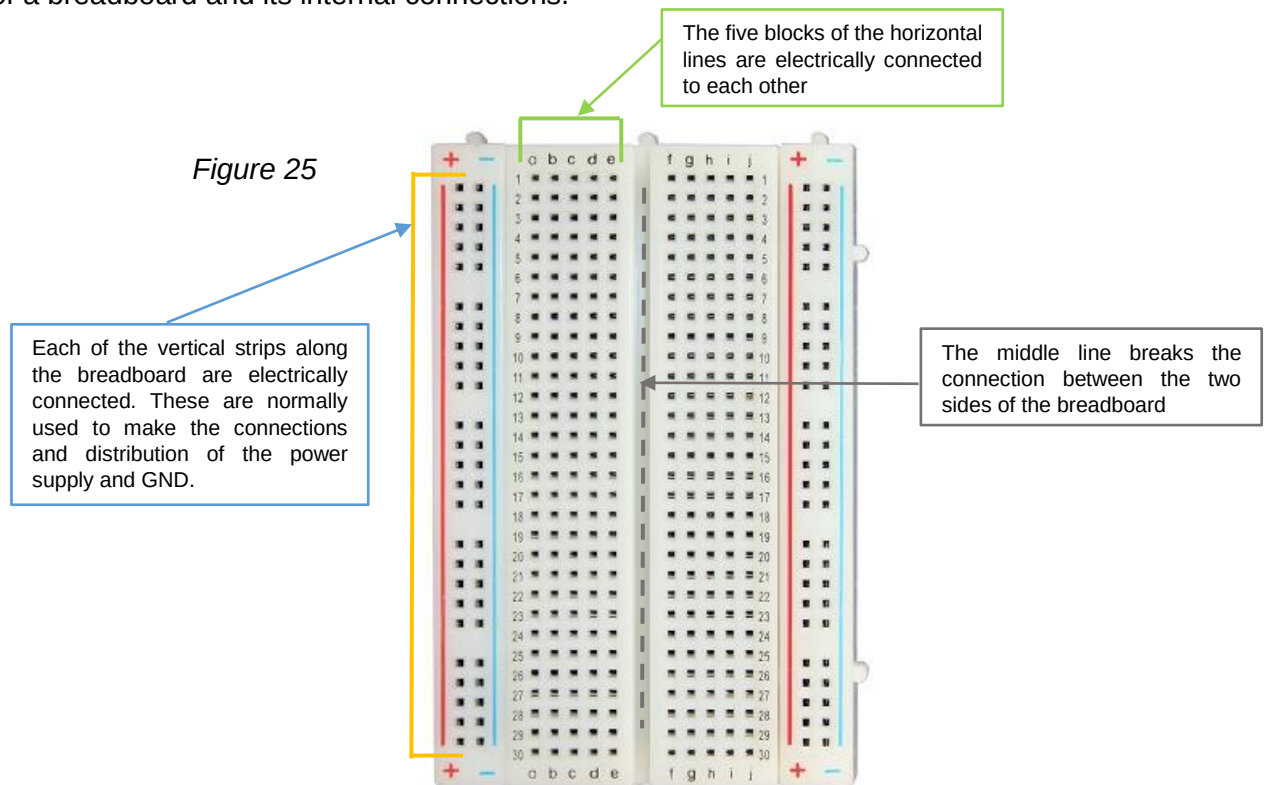
Link to the Connection Table in JPG format:

- [Connection Table Traffic Light 2 lights.jpg](#)
- [Connection Table Traffic Light 4 lights - 2 x street.jpg](#)

The breadboard

Breadboard is a way of constructing electronics without having to use a soldering iron. Components are pushed into the sockets (holes) on the breadboard and then extra 'jumper' wires are used to make connections between the various components.

Breadboards are made for prototyping doing quick experiments. They are not suitable for keeping circuits together for a long time. The connections are not permanent, so it is easy to remove a component if you make a mistake, or just start over and do a new project. In **Figure 25** the layout of a breadboard and its internal connections.



In **Figure 26** an example of how to connect a LED with a battery and a button

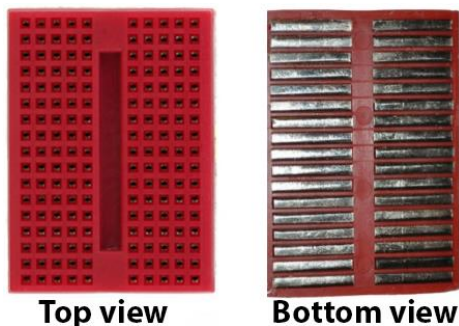
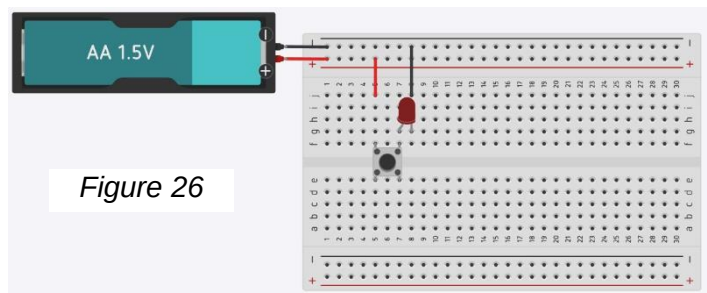


Figure 27

In **Figure 27** another type of breadboard, simpler and smaller, which does not have the vertical strips for the connections and distribution of the power supply and GND

The system can be implemented in two different ways:

- the fastest is to build a prototype with a single breadboard on which to assemble all the components. This system does not provide for the creation of street layout drawings and the construction of the platform on which to place the various components (**Figure 28**)
- realization of the street layout drawings and the construction of the platform on which to place the various components. This system requires more time because it also involves some graphic realization and construction of the panel on which to insert the various elements (**Figures 29 - 30**)



Figure 28

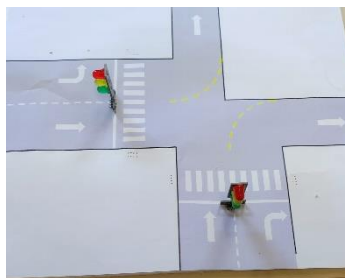


Figure 29

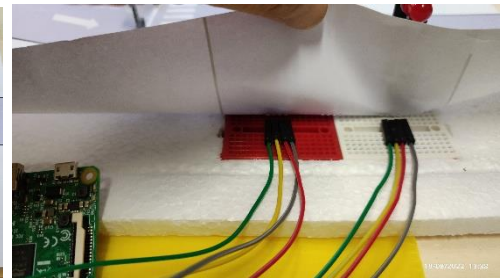


Figure 30

Now it is necessary to make all the connections between the Raspberry board and the breadboard(s) following the pinout indicated in **Figures 21 - 22 - 23**.

Once the connections are completed we can start programming our Raspberry using the GPIO.

Please note that in order to work with the GPIOs it is necessary to use the Raspberry board, as the GPIOs modules cannot be loaded on a PC.

Here are shown two examples:

1. TL NO Overlap GPIO
2. TL Overlap GPIO

1. TL NO Overlap GPIO

This program is similar to the one already done in the Desktop version (TL NO Overlap Desktop) with the difference that instead of the "Costumes" we will have to directly program the various GPIOs that control the various LEDs of the two traffic lights.

Pay attention to the following indications:

- at the beginning of each sequence it is important to set all the GPIO outputs to "0", so that all the LEDs are turned off when the program starts
- every time you set a GPIO to "1" and program the time in which that LED will remain on, you will also need to turn it off by bringing it back to "0" before turning on the next one and vice versa.

Figures 31 and 32 show the TL_NO Overlap GPIO program for the two Sprites TL_1 and TL_2 using the GPIOs indicated in **Figures 21 - 22 - 23**

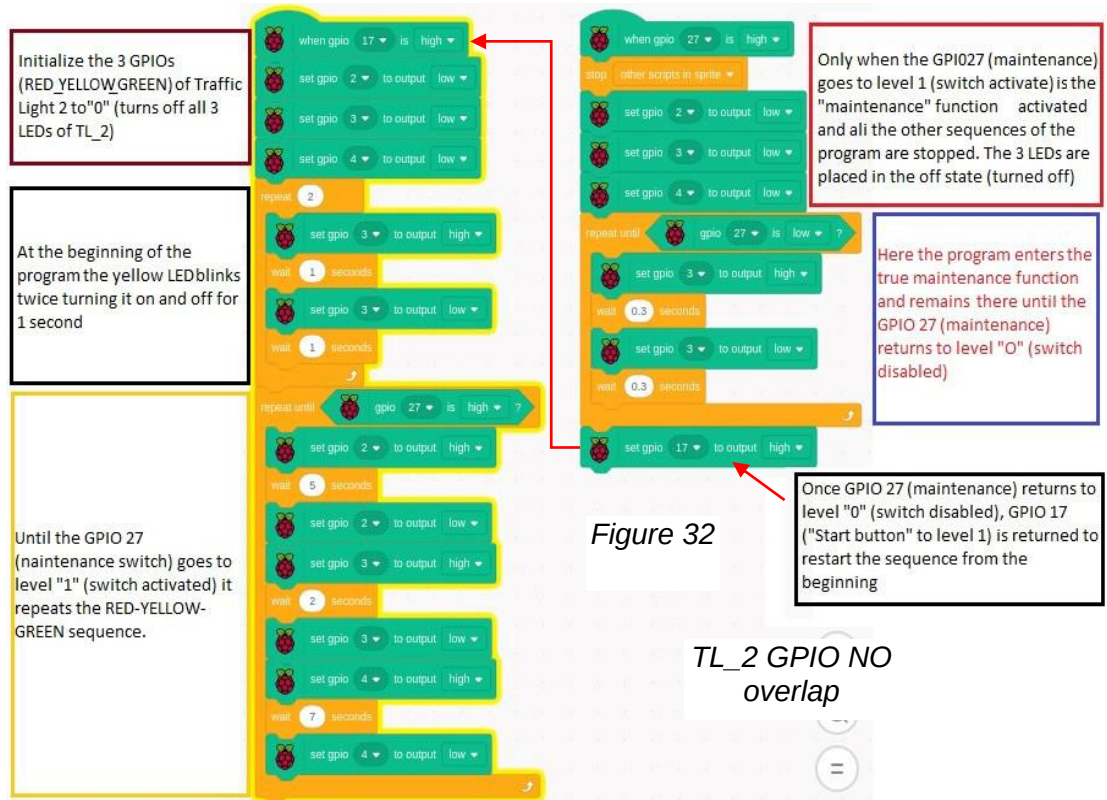
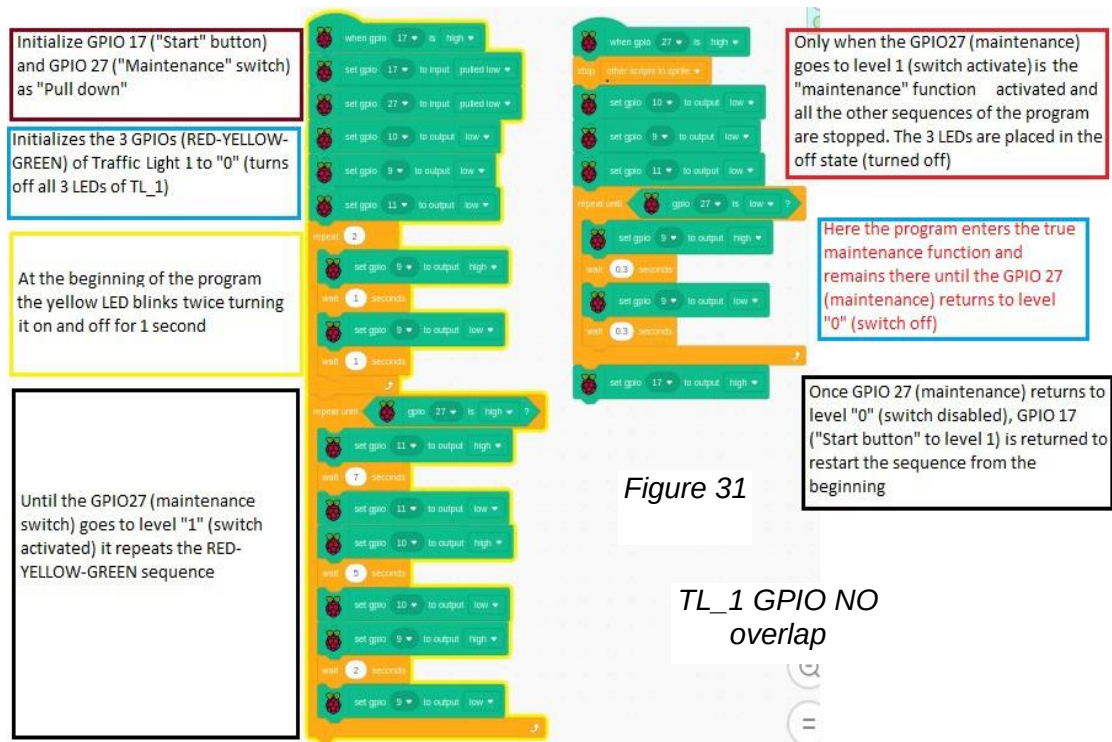
Here you can download the Scratch Code, some screenshot and a video:

[Program: TL NO Overlap GPIO](#)

[Screenshot TL_1](#)

Screenshot TL 2

Video #6: TL NO Overlap GPIO



2. TL Overlap GPIO

The program is very similar to the one already done (TL_NO Overlap GPIO). The only difference is that the timings of the lights will have to take into account the overlap of the red lights. In our case we have set:

RED = 7 seconds

YELLOW = 1 second

GREEN = 4 seconds

As you can see, the sum of the GREEN (4 sec.) + YELLOW (1 sec.) is set to 5 seconds but the light will remain "on" for 7 seconds, therefore 2 seconds more than the sum of GREEN + YELLOW. This causes the two RED lights to overlap for 2 seconds before the next GREEN light activates.

Note that an "If" instruction has been added in TL_2 which keeps TL_2's red light on for 1 second in order to create the first red light overlap at the beginning of the sequence, which works as follows:

- *If GPIO 27 (Maintenance switch) is at level "0" (it is not active)*
- *then*
- *set GPIO 4 (RED) to "1" (turn it on)*
- *wait 1 second (for 1 s.)*
- *set GPIO 4 (RED) to "0" (turn it off)*
- *wait 1 second (for 1 s.)*

and then the program moves on to the next instructions.

As indicated in the previous example as well you also have to pay attention to the following indications:

- at the beginning of each sequence it is important to set all the GPIO outputs to "0", so that all the LEDs are turned off when the program start
- every time you set a GPIO to "1" and program the time in which that LED will remain on, you will also need to turn it off by bringing it back to "0" before turning on the next one and vice versa.

Figures 33 and 34 show the "TL_Overlap GPIO" program for the two Sprites TL_1 and TL_2 using the GPIOs as indicated in **Figures 21 - 22 - 23**.

Here you can download the Scratch Code, some screenshot and a video:

[Program: TL Overlap GPIO](#)

[Screenshot TL_1](#)

[Screenshot TL_2](#)

[Video #7: TL Overlap GPIO](#)

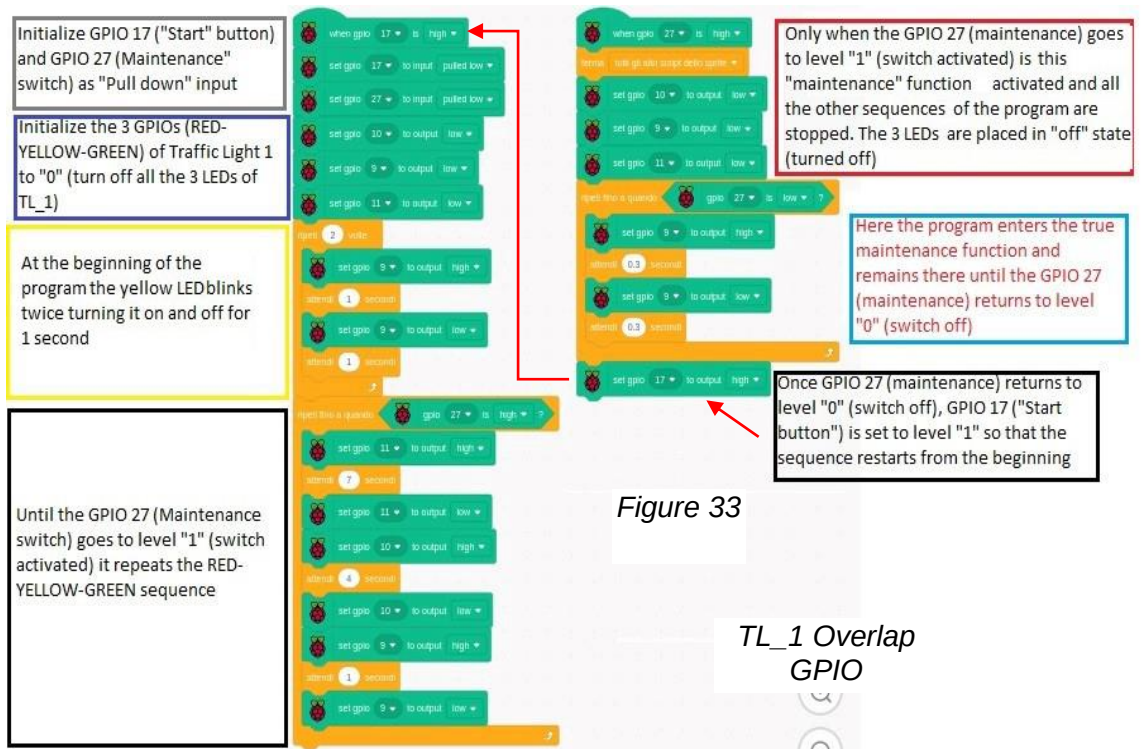


Figure 33

TL_1 Overlap
GPIO

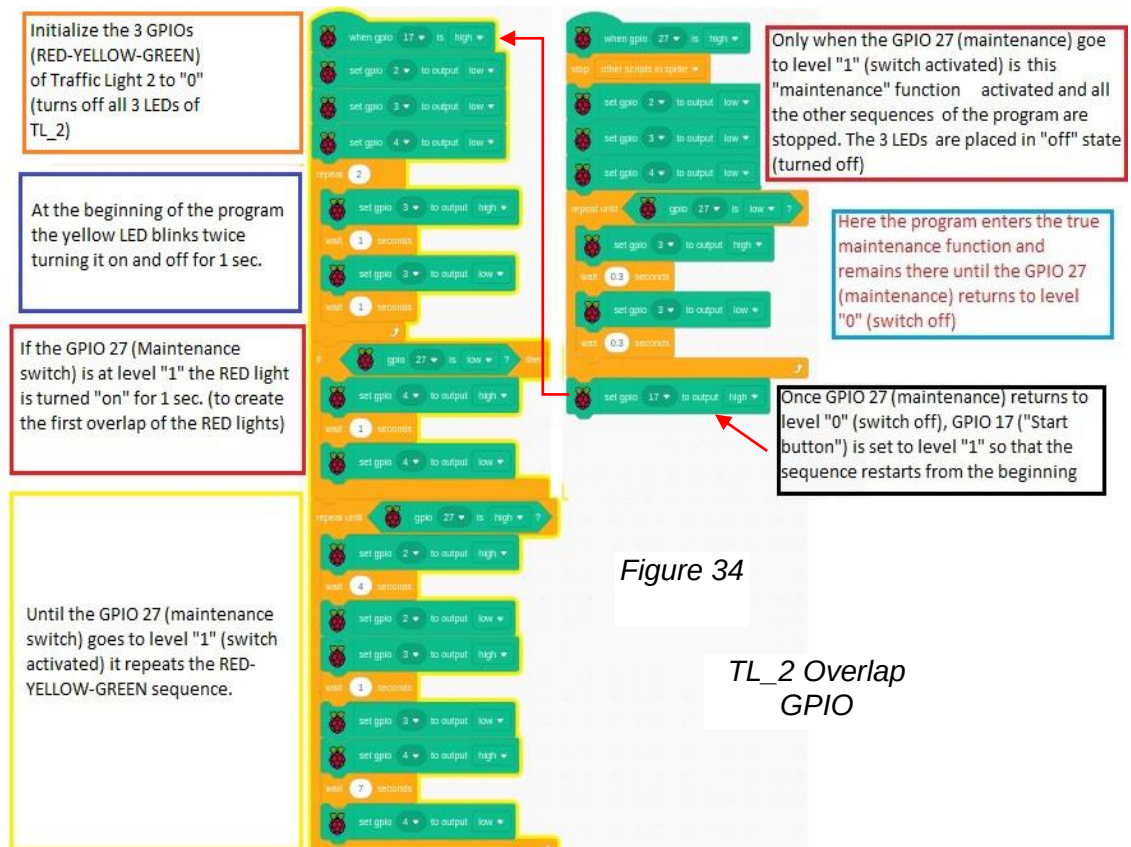


Figure 34

TL_2 Overlap
GPIO

Once the programming is complete we can move on to the test phase, using the prototype with a single breadboard (**Figure 28**) or create the street layout drawings and the construction of the platform (**Figures 29 - 30**) as indicated below

Drawing and crafting the floor plan of two ways junction

The plan of the traffic light intersection can be made with a sheet of paper of the size, as in our case, 28x34 cm., cutting out an A3 sheet or using a simple A4 sheet. (**Figures 35 - 36 – 37**)

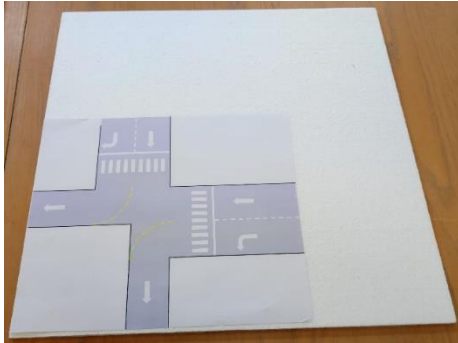


Figure 35

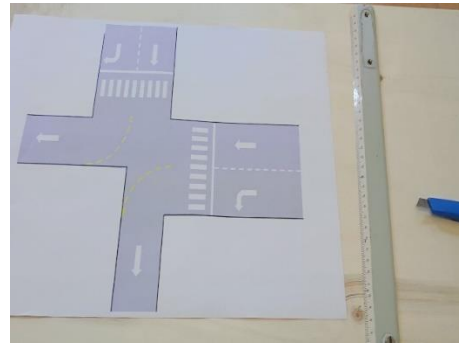


Figure 36

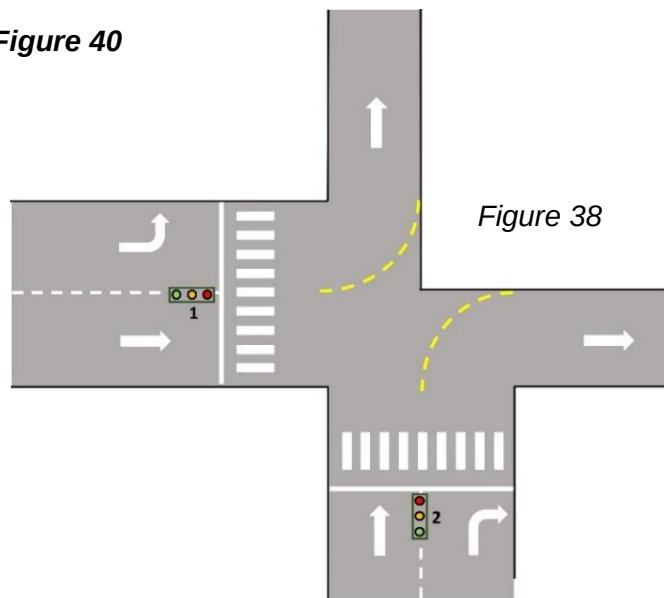


Figure 37

The plan could be created by hand with felt-tip pens or colored pencils or by using a drawing program and printing it; in our case we used MS Word.

Three different layouts are shown below:

- “Draft Layout_2 lights” **Figure 38**
- “Draft Layout_4 lights” **Figure 39**
- “Traffic Light Layout for Assembly” **Figure 40**



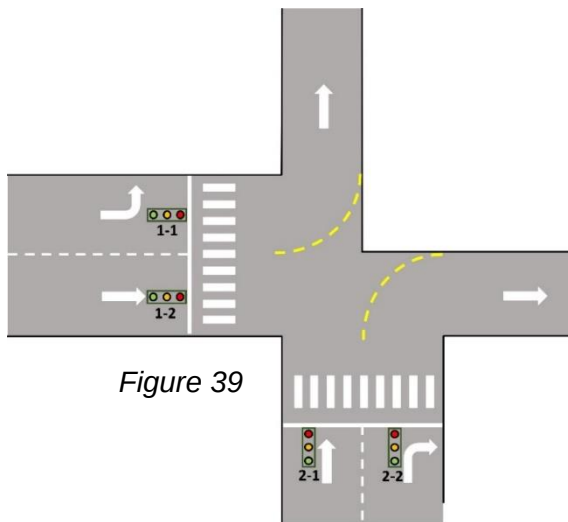


Figure 39

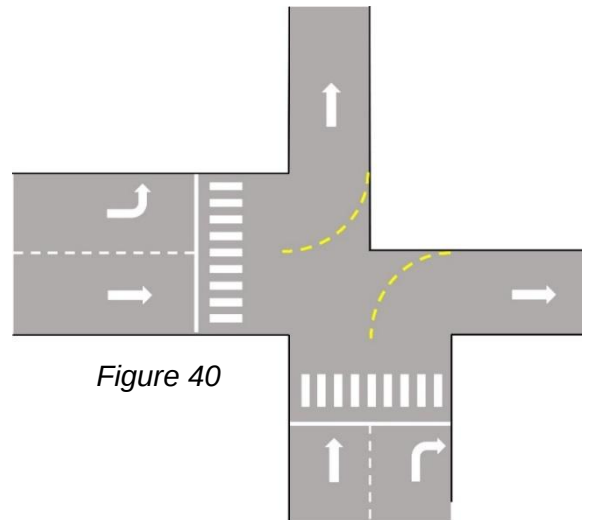


Figure 40

Clicking on the links below you can download the Layout in various format:

Link to the Layout realized using MS Word:

- [Draft Layout 2 lights](#)
- [Draft Layout 4 lights](#)
- [Traffic Light Layout for Assembly](#)

Link to the Layout realized in PDF format:

- [Draft Layout 2 lights](#)
- [Draft Layout 4 lights](#)
- [Traffic Light Layout for Assembly](#)

Link to the Layout realized in JPG format:

- [Draft Layout 2 lights](#)
- [Draft Layout 4 lights](#)
- [Traffic Light Layout for Assembly](#)

After creating the paper layout it will be necessary to create a 1 cm high styrofoam base where the breadboards will be inserted. The styrofoam sheet must have the size of the paper sheet (you can use a A4 sheet) used to create the graphic layout or slightly larger (**Figures 41 - 42**)



Figure 41

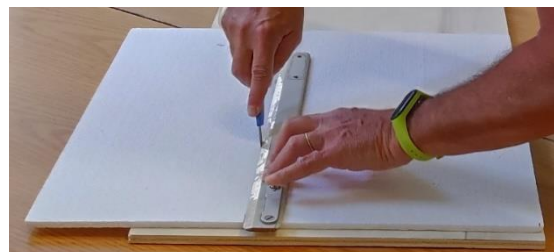


Figure 42

The layout and the styrofoam sheets cut to size are shown in **Figure 43**

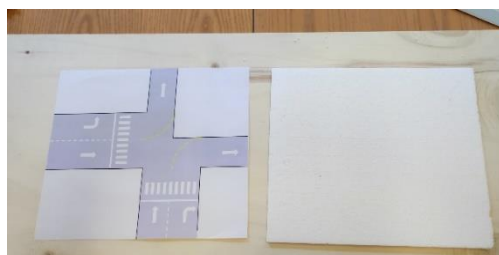


Figure 43

The next step will be to mark the exact position of the breadboards on a drawing of the intersection using a previously created layout. **Figures 44 - 45**

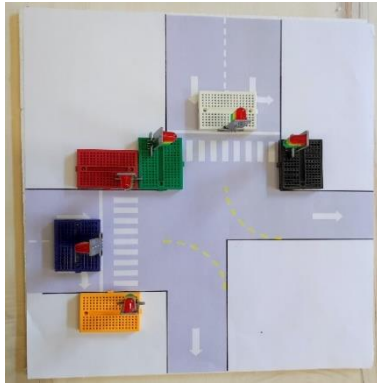


Figure 44

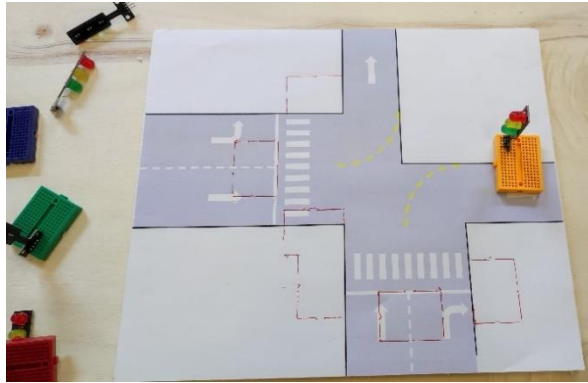


Figure 45

Once we have marked the exact outlines of the breadboards on the drawing, we will need to cut out the outlines using a cutter **Figure 46**

Once this is done, we need to draw the outlines on the styrofoam plate using the design we have just shaped as a reference and guide **Figure 47**

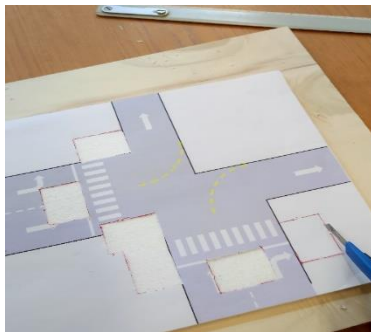


Figure 46

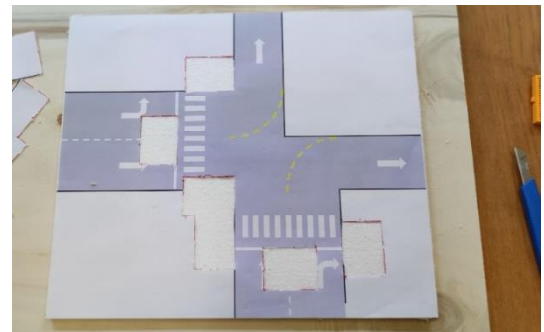


Figure 47

Now we are ready to cut out the breadboard slots in the styrofoam sheet where we will insert the breadboards. **Figures 48 - 49 - 50**

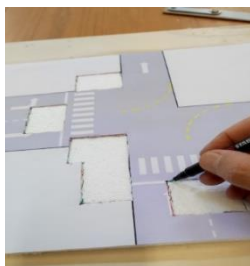


Figure 48

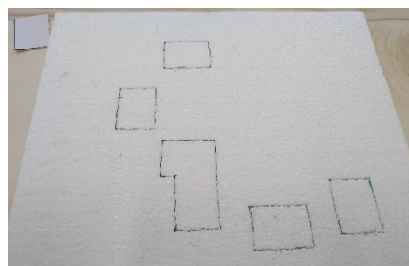


Figure 49

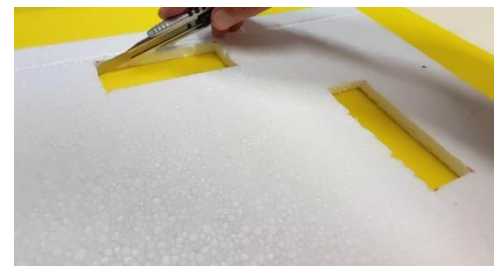


Figure 50

It will be necessary to pay a little attention in shaping perfectly the marked slots in the styrofoam so that the breadboards fit perfectly into the styrofoam plate.

Once you have cut the slots in the styrofoam sheets, you will have to insert the bread boards, each one in its housing, as shown in **Figure 51**.

The result of this intermediate step has shown on **Figure 52**.

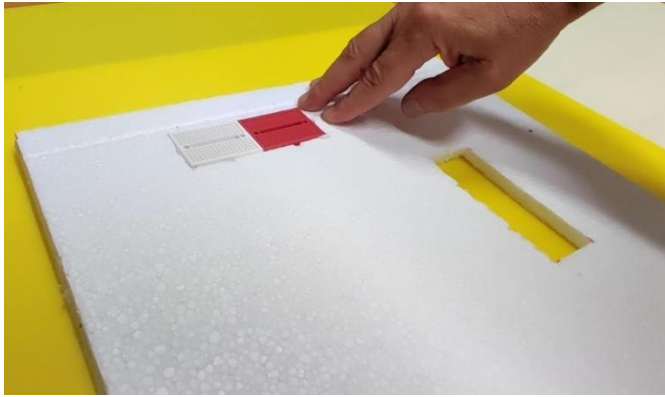


Figure 51

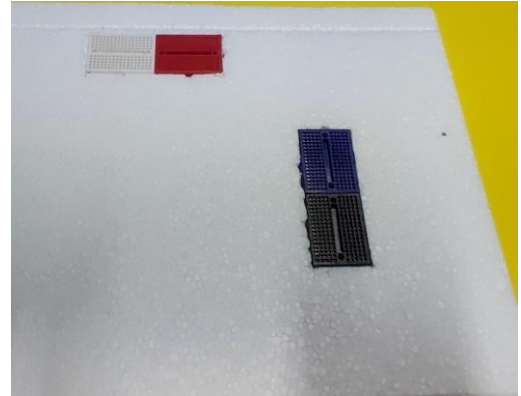


Figure 52

We can now put the plan on the alveolar polypropylene substrate and proceed to insert the mini led traffic lights on the relative breadboard, following the steps shown in **Figures 53 - 54**

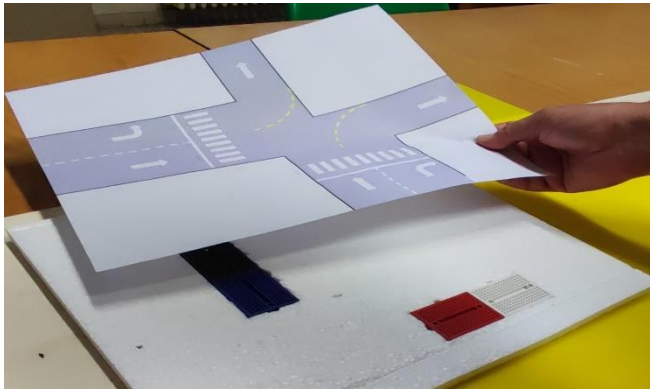


Figure 53

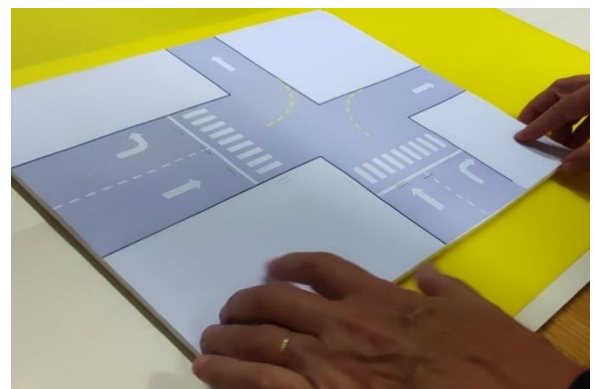


Figure 54

Below the layout of the complete traffic light with the two LEDs traffic lights inserted in the breadboard **Figure 55**

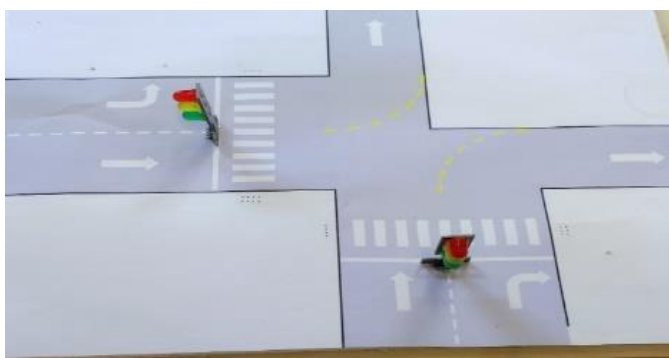


Figure 55

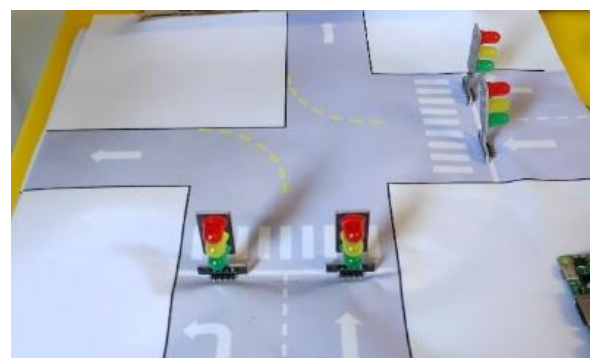


Figure 56

Figure 56 shows another example of the same crossroad but with two traffic lights per street.

Now we can connect the LED devices to the GPIO of the Raspberry Pi.

It is necessary to bend the pins of the jumper wires that are to be plugged in the breadboard by 90°, in order to be able to cover them with the planimetry drawing. See **Figures 57 - 58 - 59**



Figure 57



Figure 58

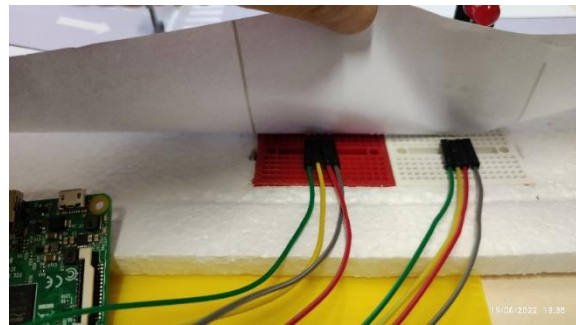


Figure 59

In the end we obtain the complete system as shown in **Figure 60**

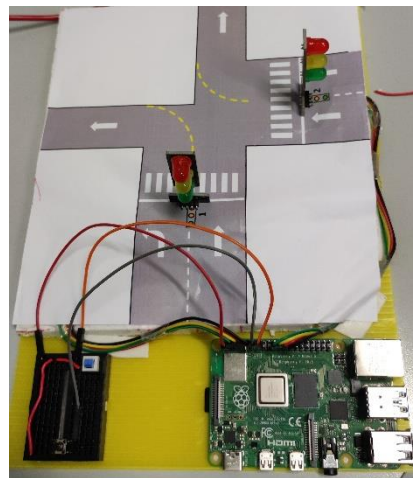


Figure 60

Notes about button and switch connections

The **pushbutton** pinout used in the prototype is shown below (**Figure 61**):



Figure 61

Note that pins 1-3 (blue in the figure) are connected to each other, as are pins 2-4 (red in the figure). The connection between pins 1-2 is **NORMALLY OPEN** and will only close when the button is pressed. Same for pins 3-4

The **switch** pinout used in the prototype is shown below (**Figure 62**):

Use the vertical lines on both sides as a reference (one side has only one bar, the other side has two bars)

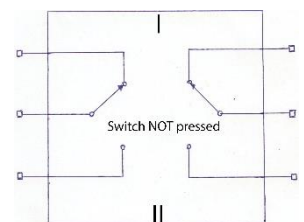


Figure 62

Notes about pull-up and pull-down resistors

Pull-up (or pull-down) resistors are fixed value resistors used between the connection of a voltage supply (or ground) and a particular pin in a digital logic circuit. Commonly paired with switches, its purpose is to ensure the voltage (or ground) between Ground and Vcc is actively controlled when the switch is open. Without the pull-up resistor the logic static of the open switch is floating so it could be either a '0' or a '1', because, not being connected to a stable potential, it behaves like an antenna and the voltage across it is fluctuating, indeterminate.

Here below some examples of how to connect the pull-up resistor.

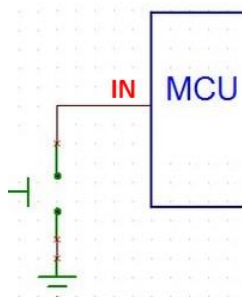


Figure 63
No resistor

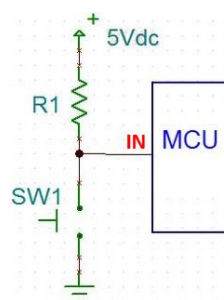


Figure 64
Pull-Up resistor
Switch unpressed

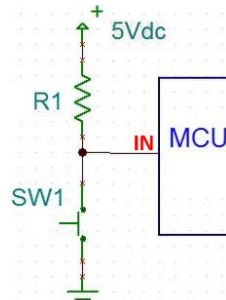


Figure 65
Pull-Up resistor
Switch pressed

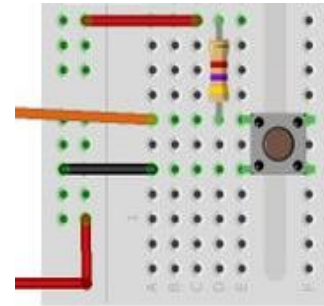


Figure 66
Pull-Up resistor on
the breadboard

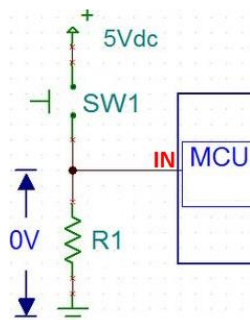


Figure 67
Pull-Down resistor
Switch unpressed

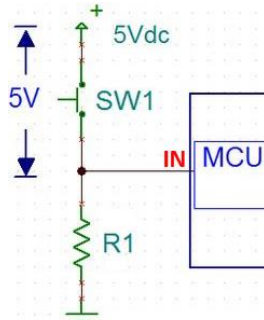


Figure 68
Pull-Down resistor
Switch pressed

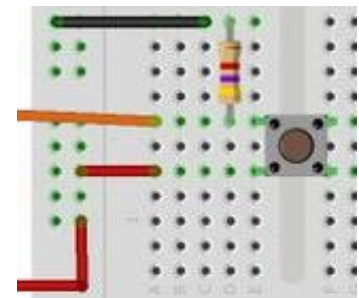


Figure 69
Pull-Down resistor
on the breadboard

What should be the value of the resistor?

Lets assume we want to limit the current to 1mA. For example if $V_{cc} = 5V$, using Ohms law: $R = V/I$ we obtain $R = 5V/1mA$ (or $R = 5000mV/1mA$) so $R = 5000\Omega = 5k$. In this case we can use a resistor of $4.7k\Omega$. Normally it's possible to use, in general, a $10k\Omega$ resistor, this will need only 0.5mA

Normally in all the microcontroller systems, the pull-up and pull-down resistors are already built inside the board itself. The input pins can be programmed to be configured as pull-up or pull-down inputs with simple instructions.

See the Raspberry Pi guides for further information.

Pull-down example:



Using the right arrow you can choose pull-down or pull-up mode.

Supporting files

Worksheet for students

[The worksheet for student](#)

Programs

[The folder with all the programs](#)

Figures

[The folder with all the figures](#)

Videos

[The folder with all the videos](#)

Some YouTube Videos with subtitles

Title	YouTube link	Subtitles
Video #1 How to import the backdrop	https://youtu.be/50UOzYD03b8	Italian - Polish - Greek - English
Video #2 How to import sprites and costumes	https://youtu.be/oZUyCjgZfFc	Italian - Polish - Greek - English
Video #3 How to manually make Sprite and Costumes	https://youtu.be/lb0bh-uLU6Y	Italian - Polish - Greek - English
Video #6 TL NO Overlap GPIO	https://youtu.be/JE-h8yClbG0	Italian - Polish - Greek - English
Video #7 TL Overlap GPIO	https://youtu.be/TXQW1D1XOGI	Italian - Polish - Greek - English

Raspberry Pi guide

Raspberry Pi documentation:

<https://www.raspberrypi.com/documentation>

Getting started with Raspberry Pi:

<https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started>

Here you can download Scratch environment:

<https://scratch.mit.edu/download>