



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 Micro:Bit & MakeCode

Difficulty level: medium

Reference age: from 14 to 18 years

Cost: currently quite low

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 Where other published and unpublished source materials have been used, these have been acknowledged.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 Micro:Bit

Contents

Guidelines for teachers.....	1
Introductory notes	4
Study and realization of the timing diagrams	5
Presentation of the Micro:Bit board and the MakeCode software	7
Connecting the Micro:Bit I/Os to external LED traffic lights	8
The breadboard	9
Let's create our traffic lights with Micro:Bit	10
Programming Micro:Bit Traffic Light using internal “A” pushbutton	11
Programming Micro:Bit Traffic Light using external input devices	13
Drawing and crafting the floor plan of a two ways junction	17
Notes about button and switch connections	21
Notes about pull-up and pull-down resistors	22
Supporting files	23
Micro:Bit guide	23
MakeCode guide and programs	23

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 Micro:Bit board, button and switch. The various devices will then be connected to the relative GPIOs of the Micro:Bit 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 Micro:Bit 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 Micro:Bit board GPIO, the I/O signals coming from the devices
- Develop, through the MakeCode 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 Micro:Bit board - 1 x Micro:Bit BreakOut board - 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 - MakeCode	- 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 Micro:Bit Board
5. Create the first operating program using the MakeCode software
6. Make all the necessary interconnections between the GPIOs of the Micro:Bit board and the various external components
7. Transfer the program from the PC to the Micro:Bit 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**

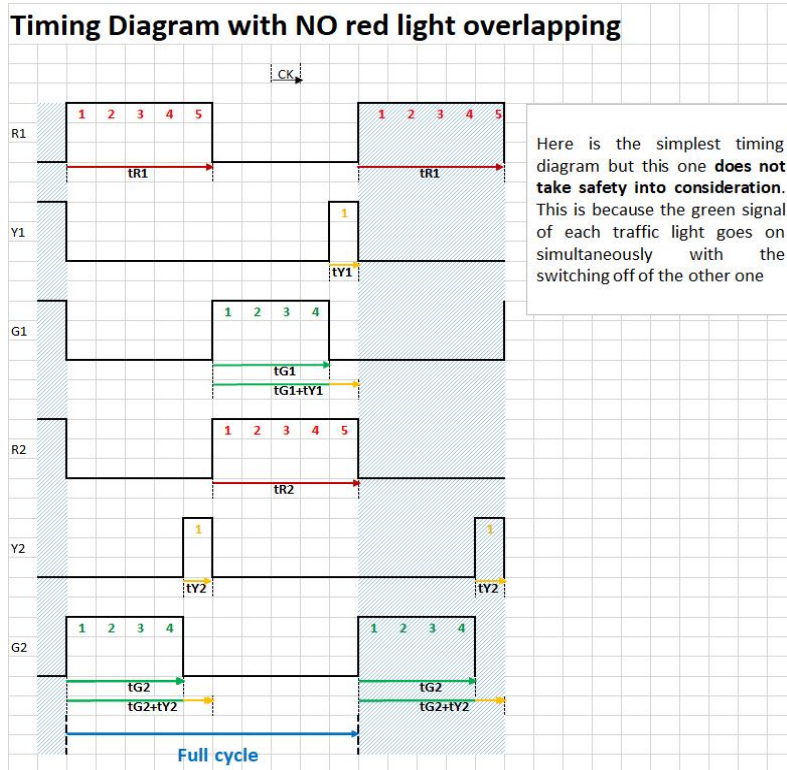


Figure 1

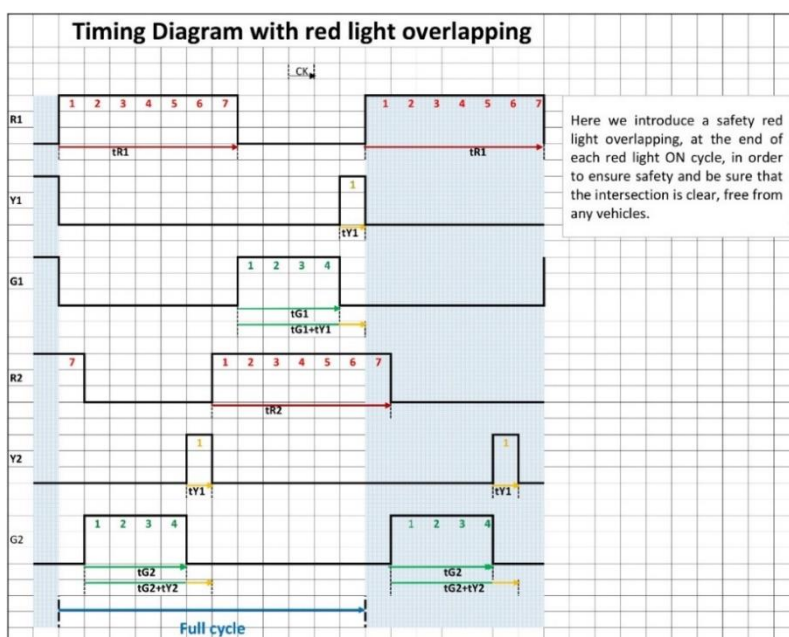


Figure 2

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 in 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](#)

Presentation of the Micro:Bit board and the MakeCode software

Micro:bit is a portable, user-programmable computer and has already integrated on board:

- 25 LEDs, which are individually programmable
- 2 programmable buttons labelled A and B
- Temperature and light sensors
- Motion sensors (accelerometer and compass)
- Wireless communication via radio and Bluetooth
- Connector with some GPIO interfaces for connecting external devices (via a breakout board)

Here is the Micro:Bit documentation: <https://microbit.org/get-started/first-steps/introduction/>

<https://microbit.org/get-started/user-guide/overview/>

It is possible to program the Micro:bit board through an online application, MakeCode, provided by Microsoft, which we find here:

<https://makecode.microbit.org> or you can download an offline application here:

<https://makecode.microbit.org/offline-app>

The offline app works the same as the Web application does in your browser but it's a stand-alone application that will work when a connection to the internet is restricted or not available.

The layout of the board and the pinout of the Micro:Bit board are shown in the next **Figures 3-4**

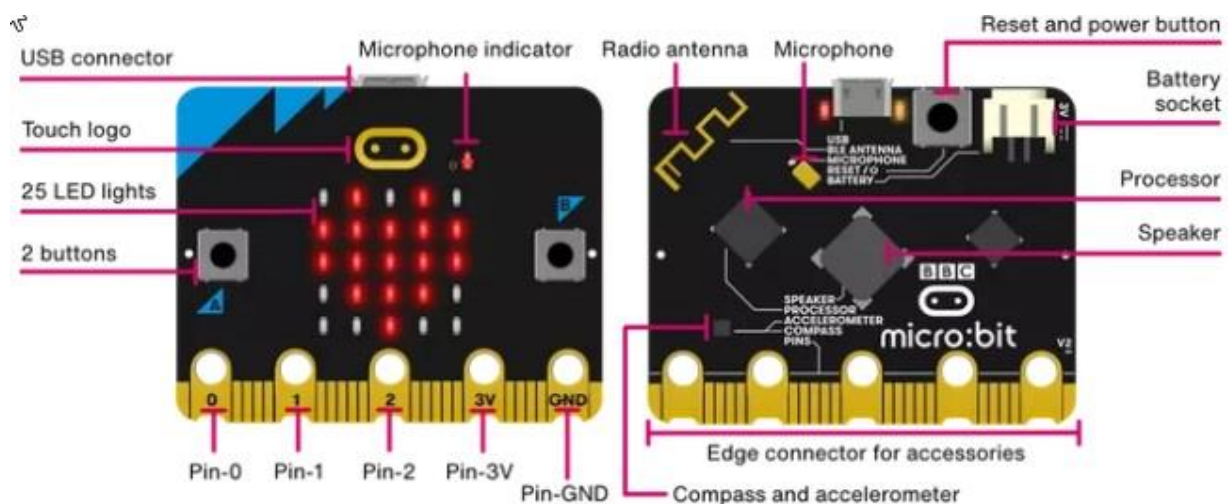
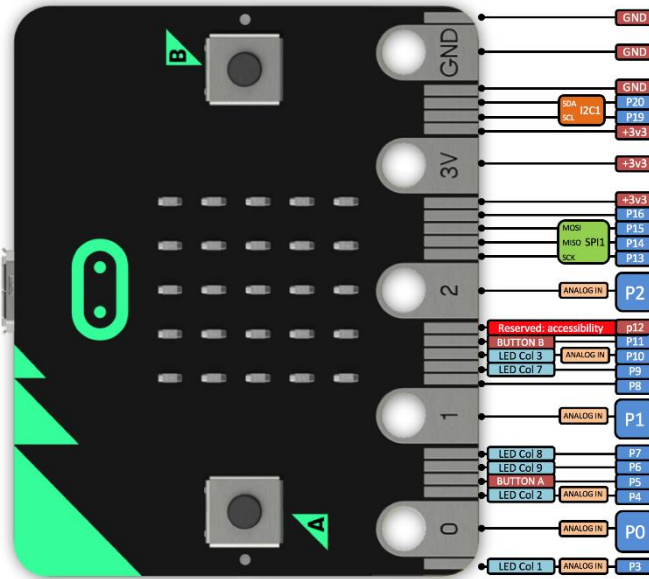


Figure 4



Connecting the Micro:Bit I/Os to external LED traffic lights

The next step is to connect the Micro:Bit I/Os to the bread boards. To do this it is necessary to use the breakout board for Micro:Bit and the jumper wires, included in the Mobile Maker Space. The Micro:Bit breakout board I/Os are shown in the following **Figures 5-6-7**

Figure 5: How to properly insert the Micro:Bit into the BreakOut Board

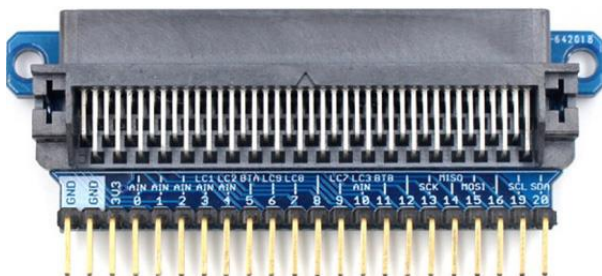
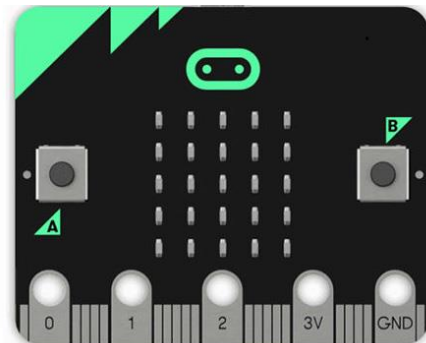


Figure 6

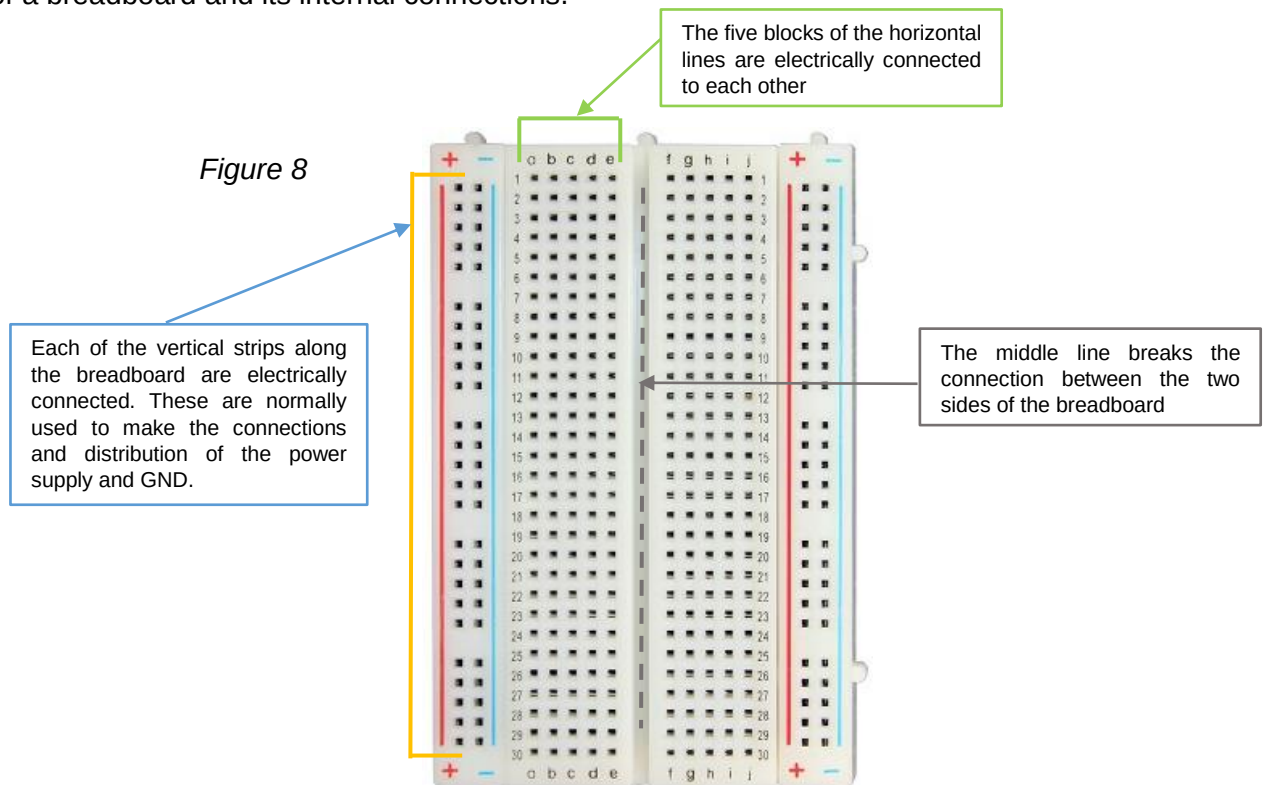


Figure 7

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 8** the layout of a breadboard and its internal connections.



In **Figure 9** an example of how to connect a LED with a battery and a button

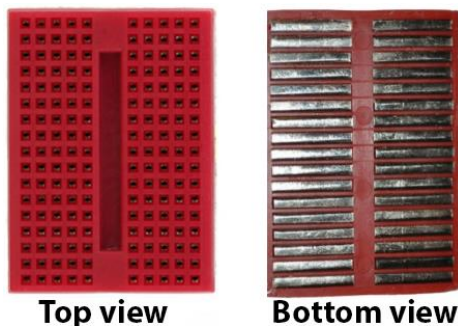
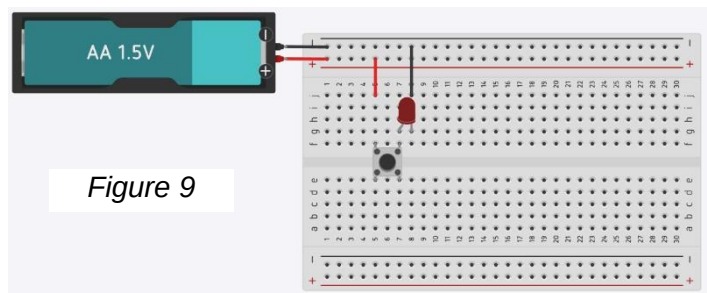


Figure 10

In **Figure 10** 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

For the correct assembly of the breakout board for Micro:Bit, the button and the switch it it's recommended to **check** the operation of the pins using the **digital multimeter** supplied with the Mobile Maker Space.

It is convenient to make a spreadsheet with the interconnections between the Micro:Bit I/Os and the traffic lights on the breadboards. Examples are in **Figures 11 - 12**

Micro:bit		Led Traffic Lights	
PIN	Traffic Light n.	Colour	Pin
16	1	Green 1	G
15	1	Yellow 1	Y
14	1	Red 1	R
GND	1	---	GND
13	2	Green 2	G
9	2	Yellow 2	Y
8	2	Red 2	R
14	2	---	GND

Figure 11

Micro:bit	Input device
PIN	Type
0	Start Push-button
2	Maintenance switch
3V3	+Vcc

Figure 12

Clicking on the links below you can download the connection tables in various formats:

Link to the connection table in various formats:

- [Connection TableTraffic Lights.xlsx](#)
- [Connection Table Traffic Lights.jpg](#)
- [Connection Table External Pushbuttons.xlsx](#)
- [Connection Table External Pushbuttons.jpg](#)

Let's create our traffic lights with Micro:Bit

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 13**)
- 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 14 - 15**)

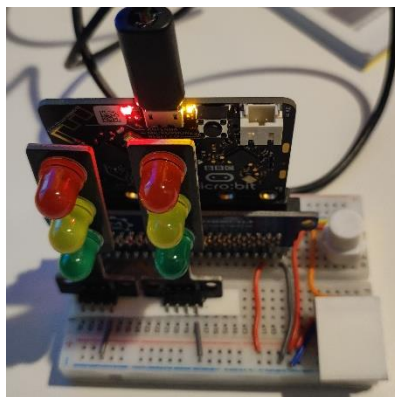


Figure 13

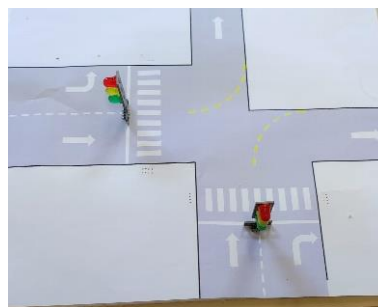


Figure 14

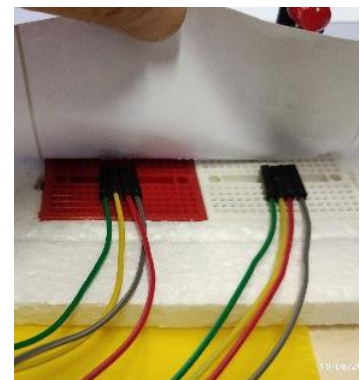


Figure 15

Now it is necessary to make all the connections between the Micro:Bit board and the breadboard(s) following the pinout indicated in **Figure 11**.

Once the connections are completed we can start programming our Micro:Bit board.

We have developed two pairs of projects, in progressive order of difficulty:

- the projects 1) and 2), the simplest ones, DO NOT use the external button and switch but the button (named A at the back of the board) integrated into the Micro:Bit board to control the "Start" of the system
- the projects 3) and 4), a little more complex, use the external button and switch to control the "Start" and "Maintenance" functions

Programming Micro:Bit Traffic Light using internal "A" pushbutton

IMPORTANT: the times in MakeCode are always expressed in **microseconds**.

Hereinafter are shown the first two projects that DON'T use external buttons:

- 1) TL-NO-overlap-NO-buttons
- 2) TL-overlap-NO-buttons

1) TL-NO-overlap-NO-buttons

Pay attention to the following indications:

- at the beginning of each sequence it is important to set all the I/Os outputs to "0", so that all the LEDs are turned off when the program starts
- every time you set a I/O 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.

When the "A" button is pressed, the sequence begins, making both yellow lights flash for 2 seconds.

Then the program enters the infinite working loop.

In this program, the timings, used as an example, are the following:

RED = 7 seconds

YELLOW = 2 second

GREEN = 5 seconds

Here you can download the MakeCode program:

[MakeCode Program: TL-NO-overlap-NO-buttons](#)

Figure 16 shows the TL-NO-overlap-NO-buttons program code.

Figure 17 shows the TL-overlap-NO-buttons program code.

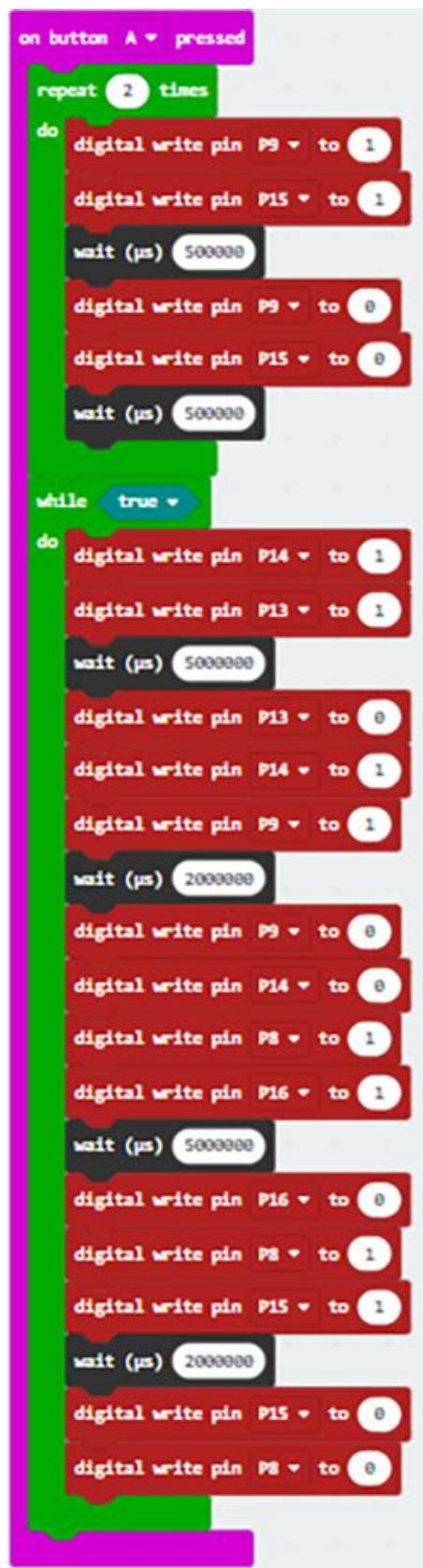
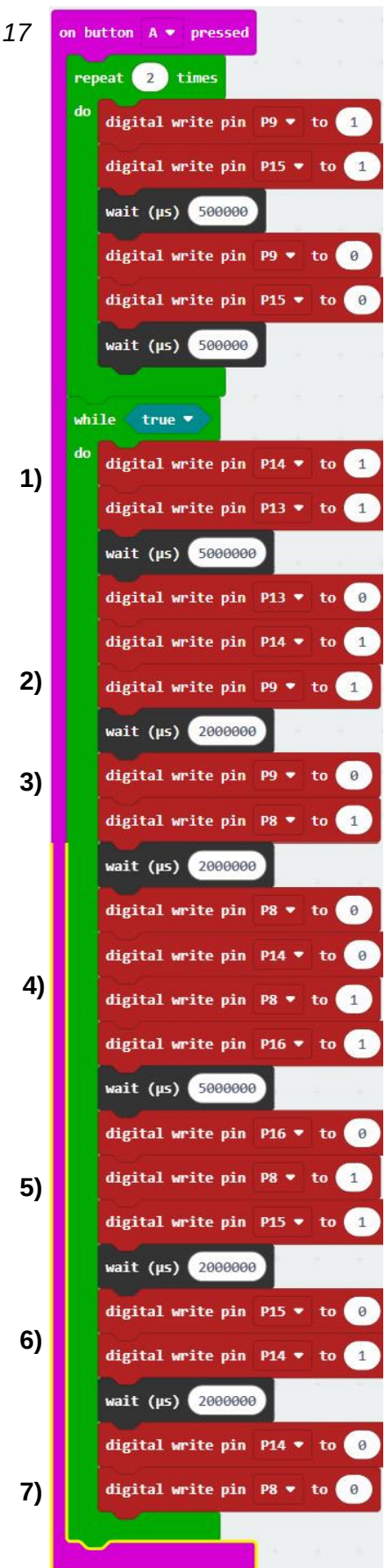


Figure 16

Figure 17



2. TL-overlap-NO-buttons program

The program is similar to the one already done (TL-NO-overlap-NO-buttons).

When the "A" button is pressed, the sequence begins, making both yellow lights flash two times for 0.5 seconds. The only difference is that the timings of the lights will have to take into account the overlap of the red lights. **Table 1** explains the functions present in the program (**Figure 17**).

In our case we have set:

1) RED_1 (P14) and GREEN_2 (P13) = "ON" for 5 seconds	Table 1
Then	
2) GREEN_2 (P13) = "OFF" and {RED_1 (P14) and YELLOW_2 (P9)} "ON" for 2 seconds	
Then	
3) YELLOW_2 (P9) = "OFF" and RED_2 (P8) = "ON" for 2 seconds (<i>here the 2 seconds red lights overlap</i>)	
Then	
4) RED_1 (P14) = "OFF" and {RED_2 (P8) and GREEN_1 (P16)} = "ON" for 5 seconds	
Then	
5) GREEN_1 (P16) = "OFF" and {RED_2 (P8) and YELLOW_1 (P15)} = "ON" for 2 seconds	
Then	
6) YELLOW_1 (P15) = "OFF" and RED_1 (P14) = "ON" for 2 seconds (<i>here the 2 seconds red lights overlap</i>)	
Then	
7) RED_1 (P14) and RED_2 (P8) = "OFF"	
Then	
The cycle restarts from step 1)	

Here you can download the code of the program created with MakeCode:

[MakeCode Program: TL-overlap-NO-buttons](#)

Programming Micro:Bit Traffic Light using external input devices

Hereinafter are shown two projects that use external buttons to activate the "Start" and "Maintenance" functions:

- 3) TL-NO-overlap-buttons
- 4) TL-overlap- buttons

Here we use two external input devices, a pushbutton to "Start" the operating sequence and a switch to enter in "Maintenance" mode. In "Maintenance" mode the yellow lights flash every 0,20 seconds (200.000 microseconds).

We have to create the three functions:

- 1) "forever" function that starts as soon as we turn on the device (**Figure 18**).

2) "maintenance" function which starts the maintenance sequence (the yellow lights alternately flash every 200 milliseconds) **Figure 19**

3) "on" function which starts the normal operating sequence **Figure 20**

The first two functions, 1) "forever" and 2) "maintenance", remain unchanged whether the program is implemented with or without overlapping of the red lights.

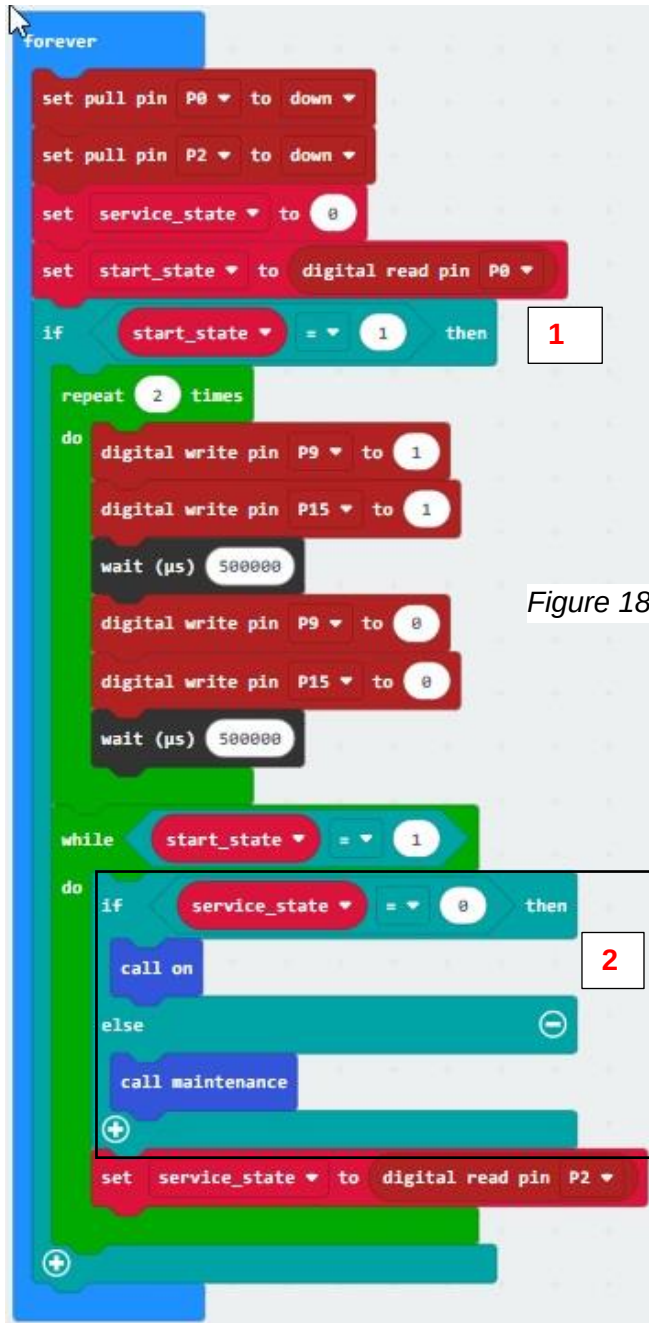


Figure 18

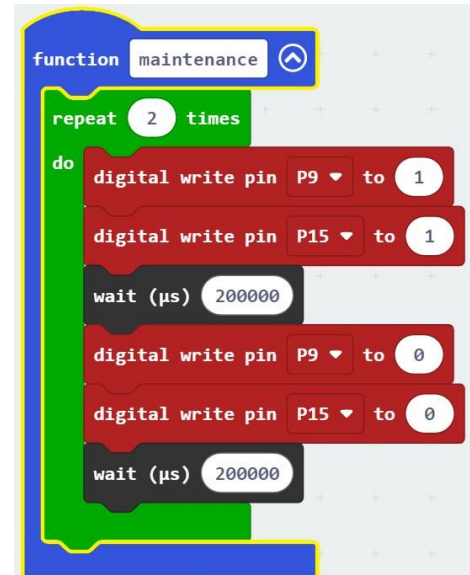


Figure 19

"If" sequence:

Here the pin P2 (maintenance switch) is read and sent to the "service_state" variable.

When the machine is powered up, the digital pins P0 (Start Push-button) and P2 (Maintenance switch) are put into pull-down mode (**Figure 18**)

Then the two variables are set:

- service_state: is initialized to 0 (OFF)

- **start_state**: this variable is assigned the value that is read from the **digital_pin_1**, which represents the status of the P0 “Start” Push-button (note **1** in Figure 18).

So, if the P0 (Start Push-button) is pressed the sequence starts, otherwise it remains in waiting state. Once the sequence has started, the yellow lights blink twice for 0.5 seconds (“repeat/do”) and then enters in the “while/do” condition. Here we found the logic block “if/else” which checks the value of the “service_state” variable, which depends on the “Maintenance_switch” status: (note **2** in Figure 18)

- if “Maintenance_switch” is OFF (0) the function “on” is called and the system enters normal operation
- else “Maintenance_switch” is ON (1) the function “maintenance” is called and the system enters the maintenance mode, making the yellow lights flash “n” times (in our case 2) before checking again the else function “Maintenance_switch”
- once the maintenance switch is turned off again, the control sequence returns the program to the “on” function

The third function 3) “on” for TL-NO-overlap-buttons program



Figure 20

In **Figure 20** the “on” function with no overlap is explained.

It is exactly the same as we saw in the first exercise TL-NO-overlap-NO-buttons. It’s the part contained in the “while true” “do” loop in **Figure 16**

[Click here to watch the video of the working prototype](#)

Here you can download the MakeCode program using external pushbuttons::

[MakeCode Program: TL-NO-overlap-buttons](#)

[MakeCode Program: TL-overlap- buttons](#)

4. TL-overlap-buttons “on” function (see Worksheet point 3-2 pag. 5)

Based on what has been explained so far, the development of this function could be a valid exercise to propose to the students, as indicated in the **Worksheet point 3-2 pag. 5**.

The two functions, 1) “forever” (**Figure 18**) and 2) “maintenance” (**Figure 19**), remain unchanged whether the program is implemented with or without overlapping of the red lights. Please refer to the functions already explained at page 14.

This function is exactly the same as we saw in the first exercise TL-overlap-NO-buttons. It’s the part contained in the “while true” “do” loop in **Figure 17**.

The TL-overlap-buttons “on” function is shown in **Figure 21**.



Figure 21 TL-overlap-buttons “on” function

Once the programming is complete we can move on to the test phase, using the prototype with a single breadboard (**Figure 13**) or create the street layout drawings and the construction of the platform (**Figure 14 - 15**) as indicated below.

Drawing and crafting the floor plan of a 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 22 - 23 - 24**

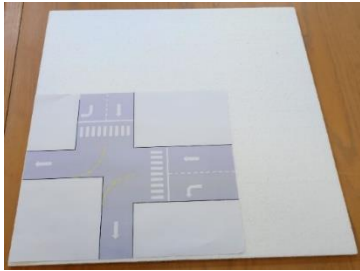


Figure 22

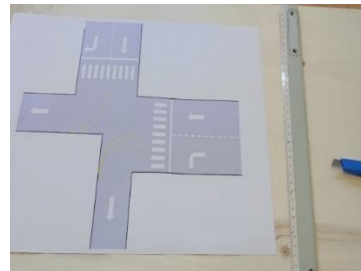


Figure 23



Figure 24

The plan could be created by hand with felt-tip pens or coloured 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 25**
- “Draft Layout_4 lights” **Figure 26**
- “Traffic Light Layout for Assembly” **Figure 27**

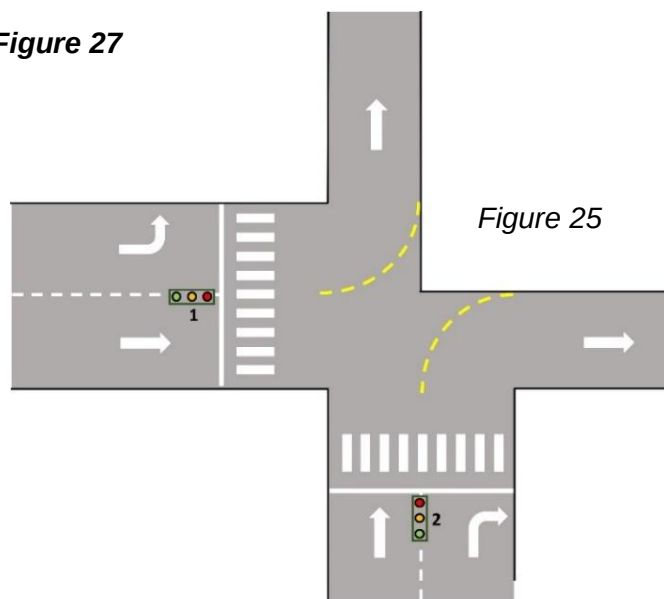


Figure 25

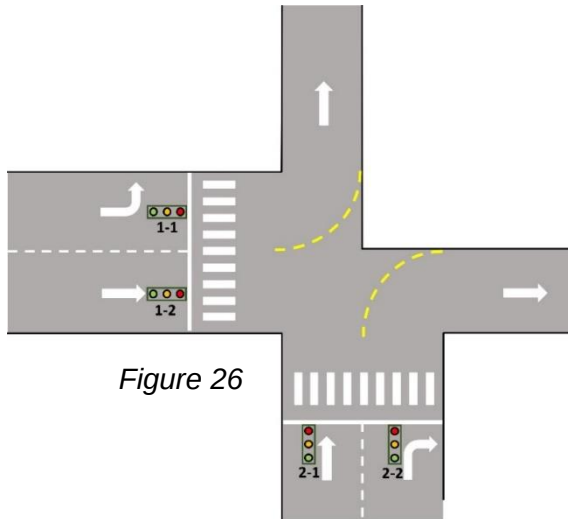


Figure 26

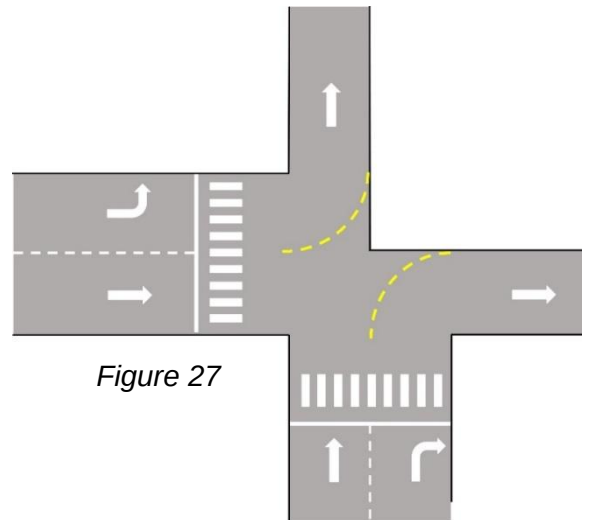


Figure 27

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 **Figure 28 - 29**



Figure 28

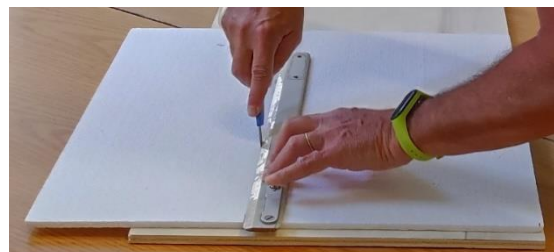


Figure 29

The layout and the styrofoam sheets cut to size are shown in **Figure 30**

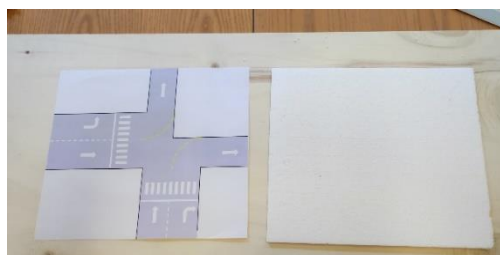


Figure 30

The next step will be to mark the exact position of the breadboards on a drawing of the intersection using a previously created layout. **Figure 31 - 32**

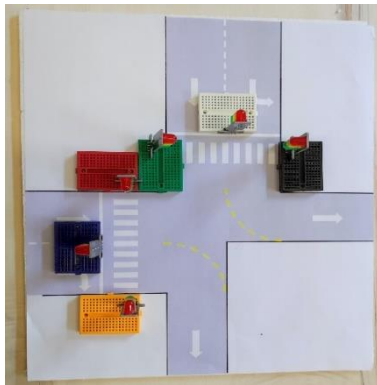


Figure 31

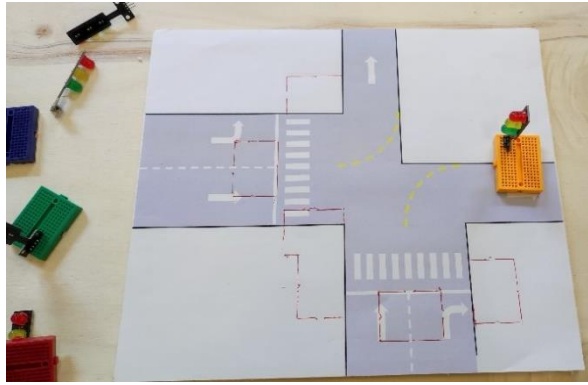


Figure 32

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 33**

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 34**

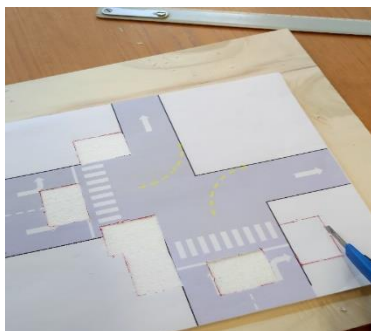


Figure 33

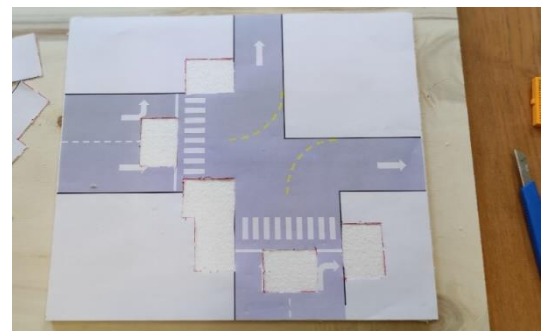


Figure 34

Now we are ready to cut out the breadboard slots in the styrofoam sheet where we will insert the breadboards. **Figures 35 - 36 - 37**

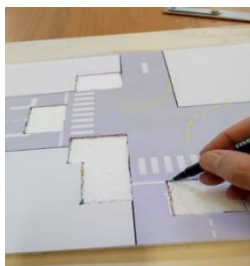


Figure 35

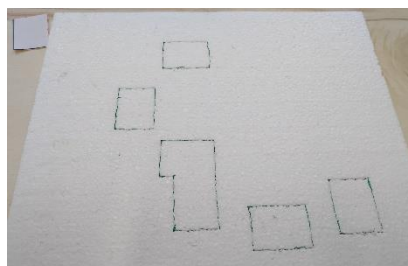


Figure 36

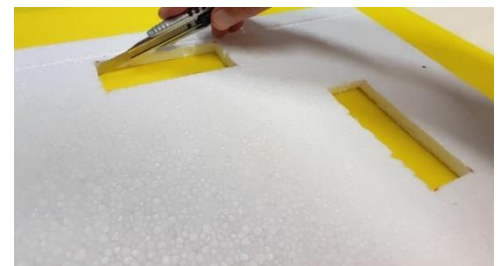


Figure 37

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 38**.

The result of this intermediate step is shown on **Figure 39**.



Figure 38

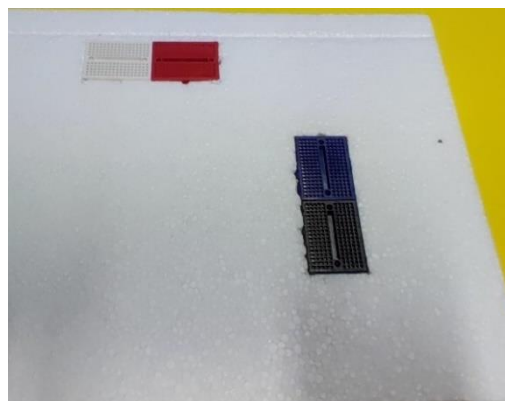


Figure 39

We can now place the paper layout previously made on the polystyrene base and proceed to insert the mini LED traffic lights on the relative breadboard, following the steps shown in **Figures 40 - 41**

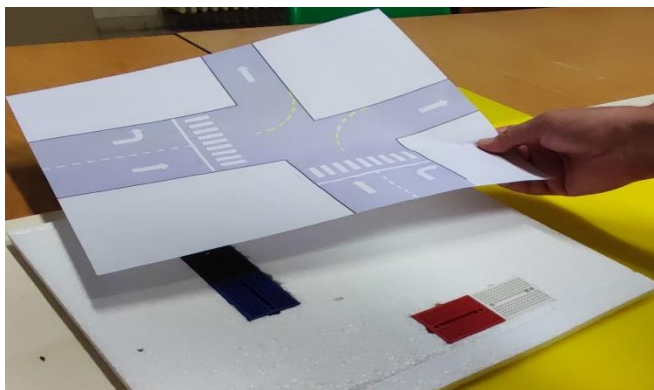


Figure 40

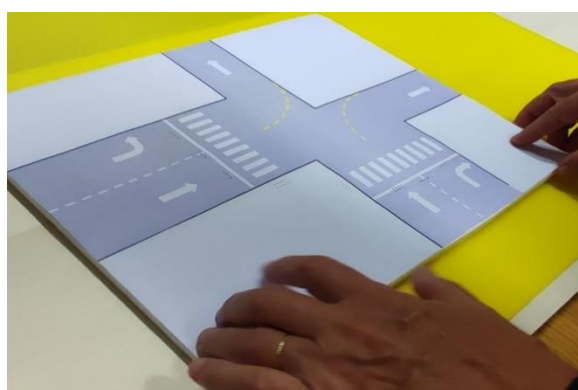


Figure 41

Below the layout of the complete traffic light with the two LEDs traffic lights inserted in the breadboard **Figure 42**

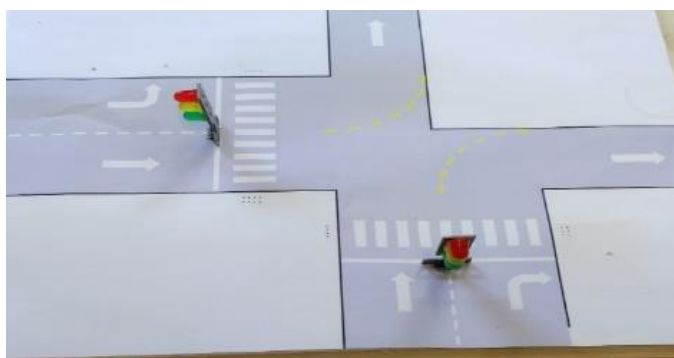


Figure 42

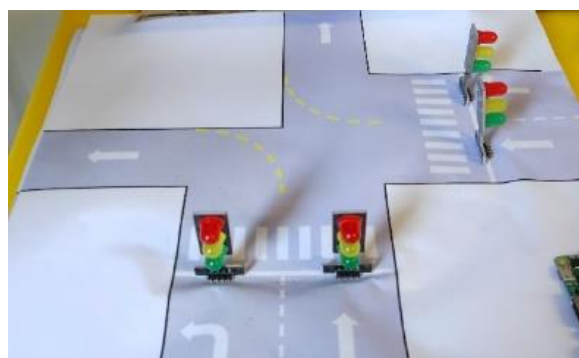


Figure 43

Figure 43 shows another example of the same crossroad but with two traffic lights per street.

Now we can connect the LED devices to the I/Os of the Micro:Bit board.

It is necessary to bend the pins of the jumper wires that are to be plugged into the breadboard by 90°, in order to be able to cover them with the planimetry drawing. See **Figures 44 - 45 - 46**



Figure 44



Figure 45

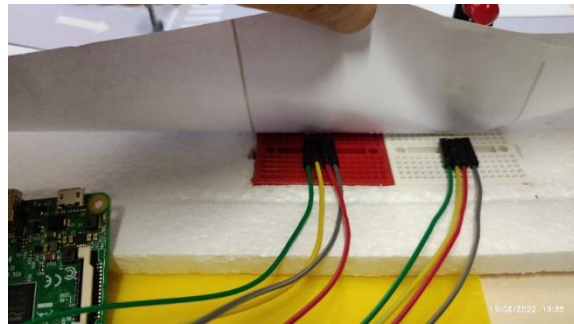


Figure 46

In **Figure 47** an example of the final complete system.

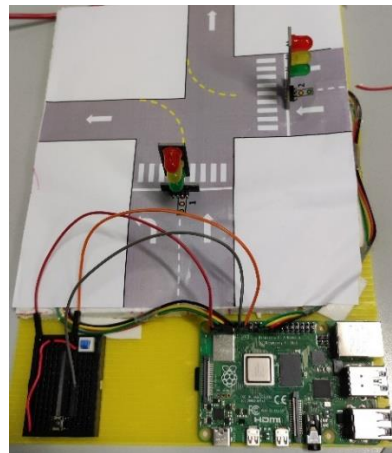


Figure 47

Notes about button and switch connections

The **pushbutton** pinout used in the prototype is shown below (**Figure 48**):



Figure 48

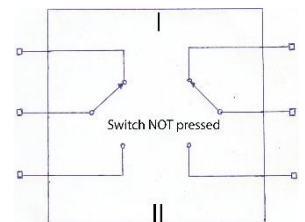
Note that pins 1-3 (blue in the figure) are connected to each other, as are pins 2-4 (red in the figure). The connection between pins 1-2 is **NORMALLY OPEN** and will only close when the button is pressed. Same for pins 3-4

The **switch** pinout used in the prototype is shown below (**Figure 49**):

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



Figure 49



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.

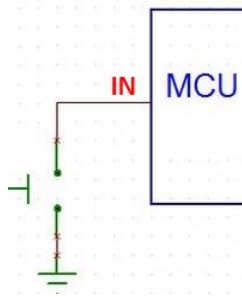


Figure 50
No resistor

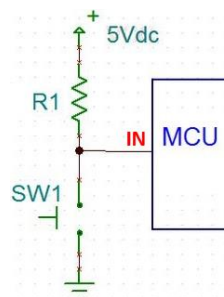


Figure 51
Pull-Up resistor
Switch unpressed

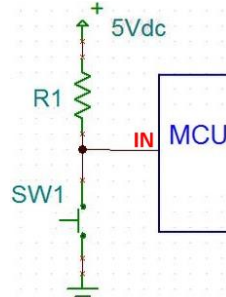


Figure 52
Pull-Up resistor
Switch pressed

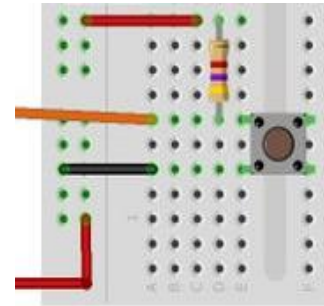


Figure 53
Pull-Up resistor on
the breadboard

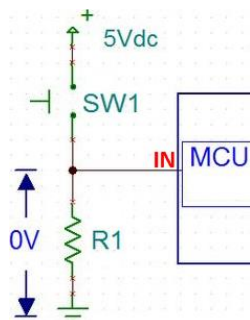


Figure 54
Pull-Down resistor
Switch unpressed

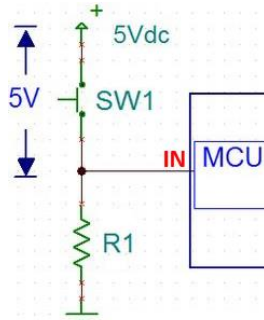


Figure 55
Pull-Down resistor
Switch pressed

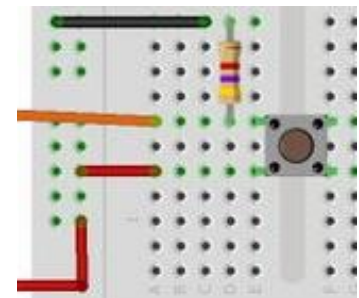


Figure 56
Pull-Down resistor
on the breadboard

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 Micro:Bit guides for further information.



Pull-down example:

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](#)

Micro:Bit guide

Getting started with Micro:Bit:

<https://microbit.org/get-started/first-steps/introduction/>

Micro:Bit documentation:

<https://microbit.org/get-started/user-guide/overview/>

MakeCode guide and programs

<https://makecode.microbit.org>

<https://makecode.microbit.org/offline-app>