

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



Traffic light control system Raspberry Pico & Phyton

Renato Gatti – Loris Ranalli – Giovanni De Rita

international@isisdavinci.eu

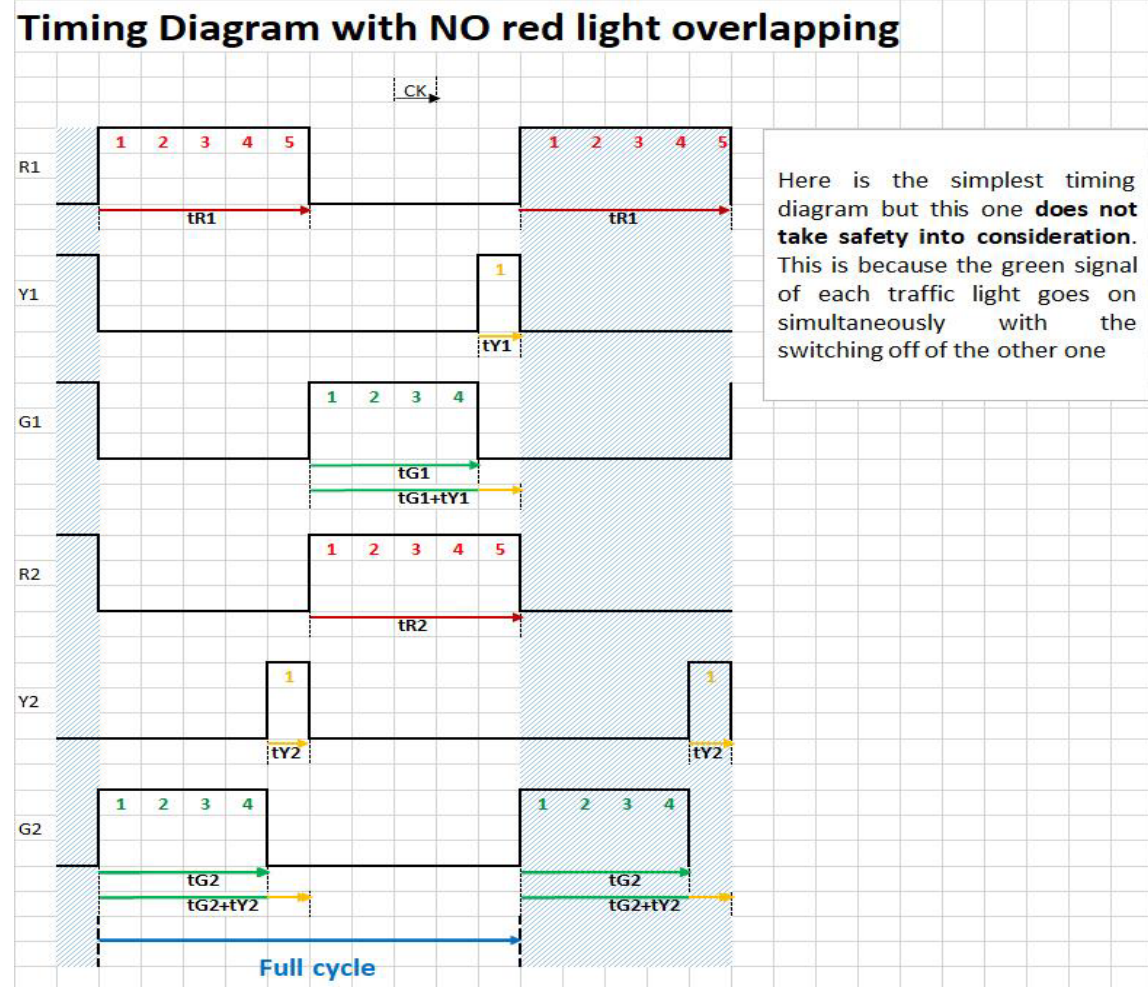


Erasmus+

Grant number: 2020-1-PL01-KA201-081698

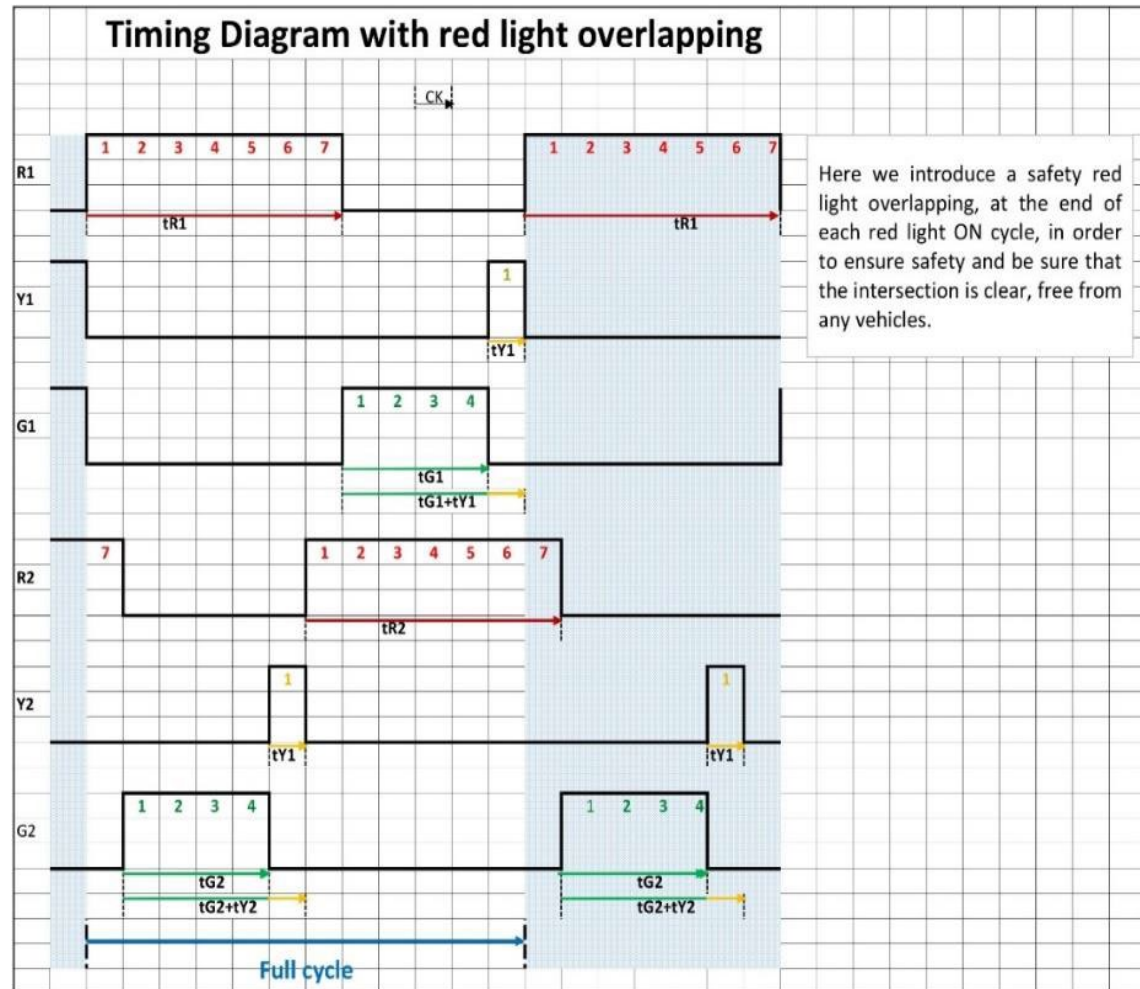
ABOUT THE PROJECT: REALIZATION OF THE TIMING DIAGRAMS

Simplest Timing Diagram with NO red light overlap



ABOUT THE PROJECT: REALIZATION OF THE TIMING DIAGRAMS

Timing diagram with red lights overlap



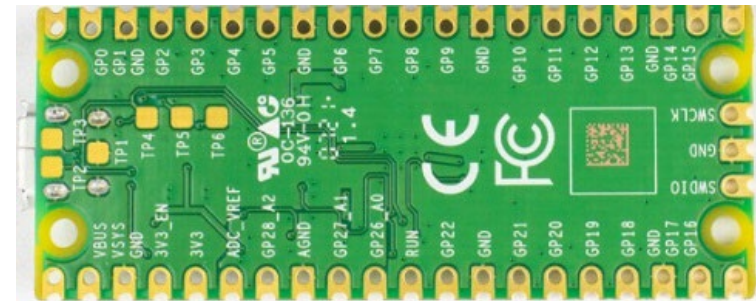
OUR "TRAFFIC LIGHTS" USING RASPBERRY PI PICO BOARD

Introducing the Raspberry Pi Pico board

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces.

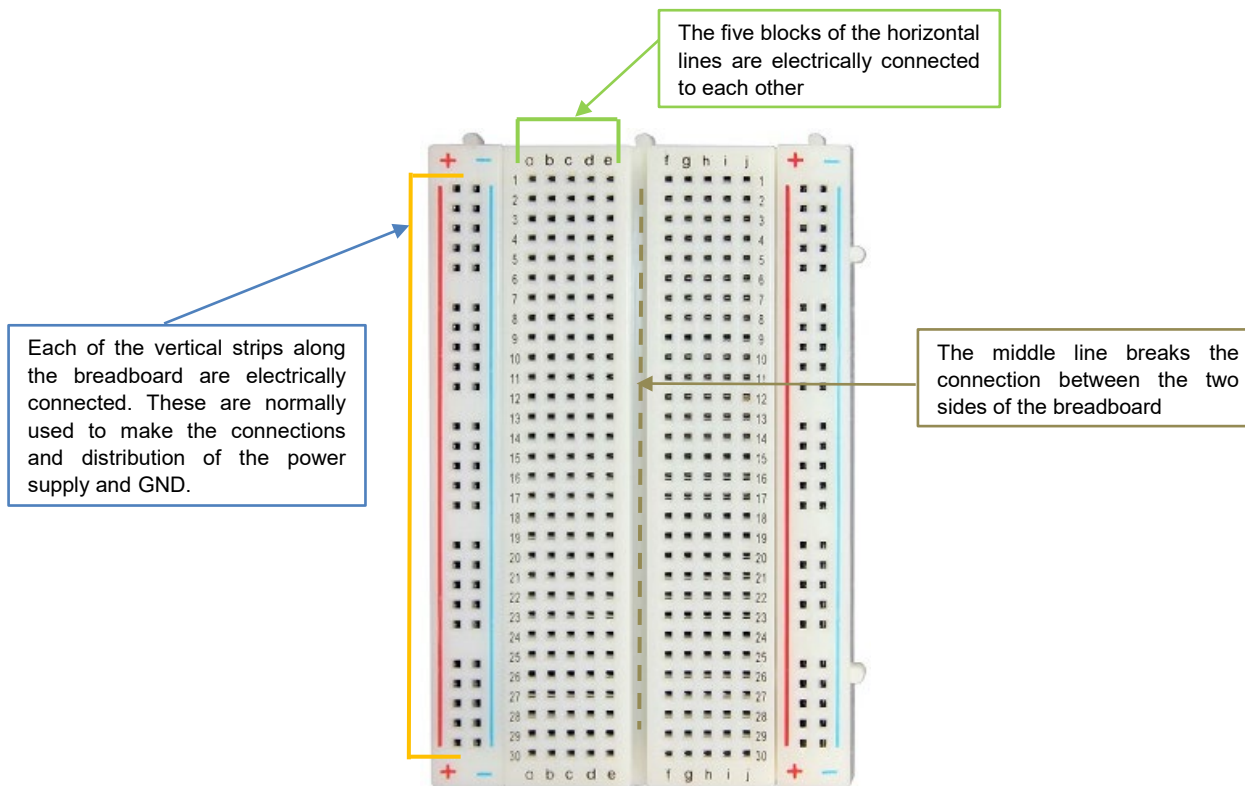
The Raspberry Pi Pico comes as a castellated module which allows soldering direct to carrier boards, while the Pico H comes with pre-soldered headers (recommended).

Layout of Raspberry Pi Pico is shown below.

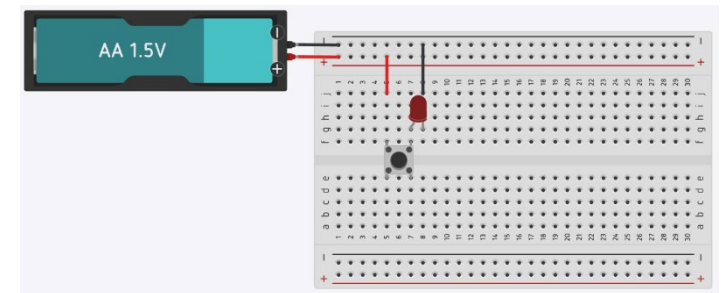


BUILD OUR "TRAFFIC LIGHTS" USING RASPBERRY Pi PICO I/Os AND THE BREAD BOARD

The 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



Below an example of how to connect a LED with a battery and a button



CREATE OUR "TRAFFIC LIGHTS" PROGRAM WITH RASPBERRY PI PICO

Programming Raspberry Pico board

First you need to download and install the “Thonny” IDE (Integrated Development Environment) software for Raspberry Pi Pico. An IDE is software that allows you to write code directly, compile it and download it to the Raspberry Pi Pico board to make it work. Often it also provides other features (such as debugger, help, keyword highlighting, etc.).

You can download the “Thonny” IDE and view all the features here: <https://thonny.org/>

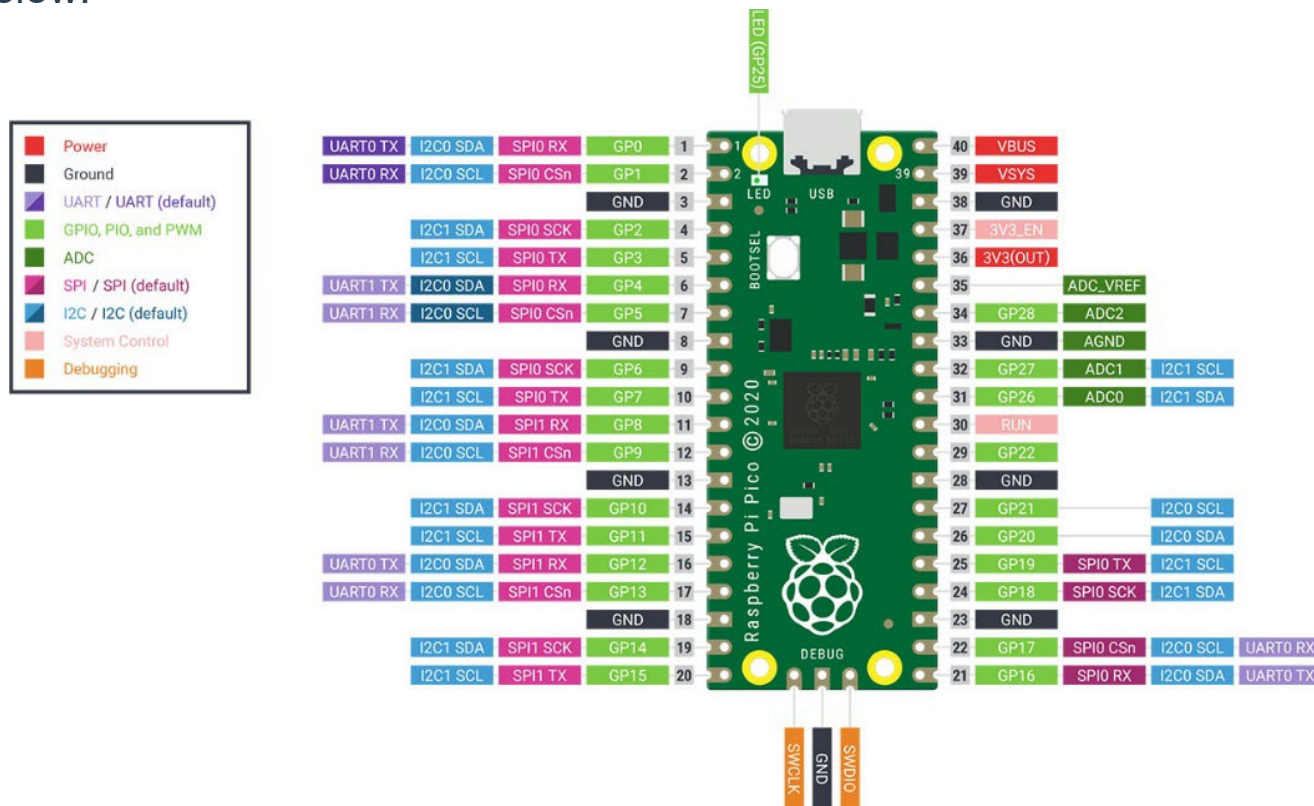
Create our "Traffic Lights" program with Raspberry Pi Pico

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

CREATE OUR "TRAFFIC LIGHTS" PROGRAM WITH RASPBERRY PI PICO

Create our "Traffic Lights" program with Raspberry Pi Pico

The next step is to connect the Raspberry Pico GPIOs to the bread boards. To do this it is necessary to use the jumper wires, included in the Mobile Maker Space. The Raspberry Pi Pico GPIOs are shown on the picture below.



CREATE OUR "TRAFFIC LIGHTS" PROGRAM WITH RASPBERRY PI PICO

Make a connection table with all the interconnections.

It is convenient to make a spreadsheet with the interconnections between the Raspberry Pi Pico GPIOs and the various traffic lights present in the respective breadboards. Examples are in the following figures.

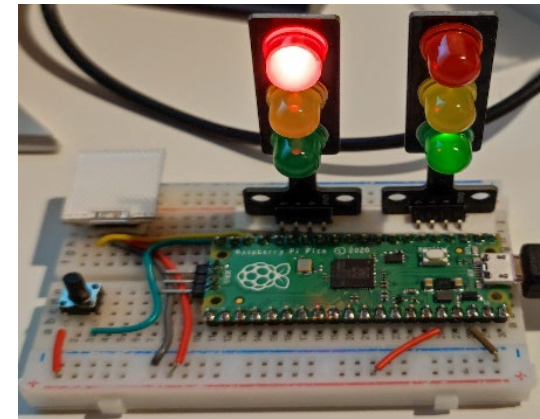
Raspberry Pi pico		Led Traffic Lights		
GP #	PIN	Traffic Light n.	Colour	Pin
GP 12	16	1	Green 1	G
GP 11	15	1	Yellow 1	Y
GP 10	14	1	Red 1	R
Ground	13	1	---	GND
GP 4	6	2	Green 2	G
GP 3	5	2	Yellow 2	Y
GP 2	4	2	Red 2	R
Ground	3	2	---	GND

Raspberry Pi pico		Input device
GP #	PIN	Type
GP 15	20	Start Push-button
GP 14	19	Maintenance switch
3V3	36	VCC

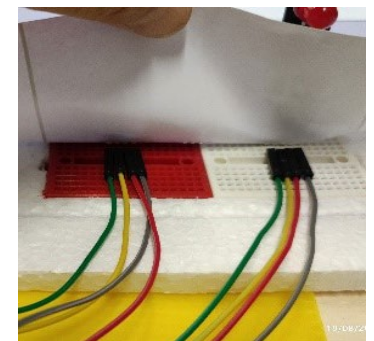
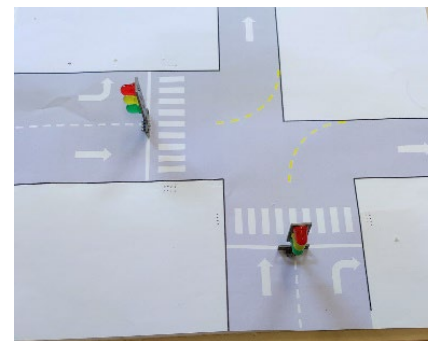
BUILD OUR "TRAFFIC LIGHTS" WITH RASPBERRY PI PICO

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.



- made street layout and crafting 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.



BUILD OUR "TRAFFIC LIGHTS" WITH RASPBERRY PI PICO

Connecting all the components

Now it is necessary to make all the connections between the Raspberry Pico board and the breadboard(s) following the PinOut table indicated before.

Once the connections are completed we can start programming our Raspberry Pi Pico using the IDE software "Thonny". Four different programs can be created, in order of increasing difficulty:

1. with NO external buttons and NO overlap of red lights
2. with NO external buttons - with overlap of red lights
3. with external buttons and NO overlap of red lights
4. with external buttons - with overlap of red lights

The first two programs are made, for simplicity, without the button and switch inputs. The traffic light sequence will automatically start the sequence without the intervention of external devices ("Start" button and "Maintenance" switch).

The third and fourth are a little more complex as they will also use the "Start" button and "Maintenance" switch to activate the various program phases.

LET'S GO MAKE OUR PROGRAMS

1. with NO external buttons and NO overlap of red lights

TL NO overlap_NO Buttons.py

```
1 #Importing libraries
2 from machine import Pin
3 from utime import sleep
4
5 #Definitions
6 led_red_1 = Pin(2, Pin.OUT) #Assigns the variable "led_red_1" the GPIO 2 (pin 1 on the board)
7 led_yellow_1 = Pin(3, Pin.OUT) #Assigns the variable "led_yellow_1" the GPIO 3 (pin 2 on the board)
8 led_green_1 = Pin(4, Pin.OUT) #Assigns the variable "led_green_1" the GPIO 4 (pin 4 on the board)
9 led_red_2 = Pin(10, Pin.OUT) #Assigns the variable "led_red_2" the GPIO 10 (pin 5 on the board)
10 led_yellow_2 = Pin(11, Pin.OUT) #Assigns the variable "led_yellow_2" the GPIO 11 (pin 6 on the board)
11 led_green_2 = Pin(12, Pin.OUT) #Assigns the variable "led_green_2" the GPIO 12 (pin 7 on the board)
12
13 #Light Initialization
14 #All the LEDs are turned off during the initialization phase to be sure the sequence starts with all lights off
15 led_red_1.value(0) #Turn OFF light RED_1
16 led_red_2.value(0) #Turn OFF light RED_2
17 led_yellow_1.value(0) #Turn OFF light YELLOW_1
18 led_yellow_2.value(0) #Turn OFF light YELLOW_2
19 led_green_1.value(0) #Turn OFF light GREEN_1
20 led_green_2.value(0) #Turn OFF light GREEN_2
21 sleep(1) #Wait a second
22
23 for i in range(1,3):
24     led_yellow_1.value(1) #Turn ON light YELLOW_1
25     led_yellow_2.value(1) #Turn ON light YELLOW_2
26     sleep(0.5) #For 0.25 sec
27     led_yellow_1.value(0) #Turn OFF light YELLOW_1
28     led_yellow_2.value(0) #Turn OFF light YELLOW_2
29     sleep(0.5) #For 0.25 sec
30
```

```
30
31 while True: # Infinite Loop
32     #Phase 1
33     led_red_1.value(1) #Turn ON light RED_1
34     led_green_2.value(1) #Turn ON light GREEN_2
35     sleep(5) #For 5 sec
36     led_green_2.value(0) #Turn OFF light GREEN_2
37     led_red_1.value(1) #Turn ON light RED_1 while YELLOW_2 is ON
38     led_yellow_2.value(1) #Turn ON light YELLOW_2
39     sleep(2) #For 2 sec
40     led_yellow_2.value(0) #Turn OFF light YELLOW_2
41     led_red_1.value(0) #Turn ON light RED_1
42     sleep(0) #For 2 sec overlap
43     #Phase 2
44     led_red_2.value(1) #Turn ON light RED_2
45     led_green_1.value(1) #Turn ON light GREEN_1
46     sleep(5) #For 5 secs.
47     led_green_1.value(0) #Turn OFF light GREEN_1
48     led_red_2.value(1) #Turn ON light RED_2 while YELLOW_1 is ON
49     led_yellow_1.value(1) #Turn ON light YELLOW_1
50     sleep(2) #For 2 sec
51     led_yellow_1.value(0) #Turn OFF light YELLOW_2
52     led_red_2.value(0) #Turn ON light RED_1
53     sleep(0) #Immediately
```

LET'S GO MAKE OUR PROGRAMS

2. with NO external buttons - with overlap of red lights

TL overlap_NO Buttons.py x

```
1 #Importing libraries
2 from machine import Pin
3 from utime import sleep
4
5 #Definitions
6 led_red_1 = Pin(2, Pin.OUT) #Assigns the variable "led_red_1" the GPIO 2 (pin 1 on the board)
7 led_yellow_1 = Pin(3, Pin.OUT) #Assigns the variable "led_yellow_1" the GPIO 3 (pin 2 on the board)
8 led_green_1 = Pin(4, Pin.OUT) #Assigns the variable "led_green_1" the GPIO 4 (pin 4 on the board)
9 led_red_2 = Pin(10, Pin.OUT) #Assigns the variable "led_red_2" the GPIO 10 (pin 5 on the board)
10 led_yellow_2 = Pin(11, Pin.OUT) #Assigns the variable "led_yellow_2" the GPIO 11 (pin 6 on the board)
11 led_green_2 = Pin(12, Pin.OUT) #Assigns the variable "led_green_2" the GPIO 12 (pin 7 on the board)
12
13 #Light Initialization
14 #All the LEDs are turned off during the initialization phase to be sure the sequence starts with all lights off
15 led_red_1.value(0) #Turn OFF light RED_1
16 led_red_2.value(0) #Turn OFF light RED_2
17 led_yellow_1.value(0) #Turn OFF light YELLOW_1
18 led_yellow_2.value(0) #Turn OFF light YELLOW_2
19 led_green_1.value(0) #Turn OFF light GREEN_1
20 led_green_2.value(0) #Turn OFF light GREEN_2
21 sleep(1) #Wait a second
22
23 for i in range(1,3):
24     led_yellow_1.value(1) #Turn ON light YELLOW_1
25     led_yellow_2.value(1) #Turn ON light YELLOW_2
26     sleep(0.5) #For 0.25 sec
27     led_yellow_1.value(0) #Turn OFF light YELLOW_1
28     led_yellow_2.value(0) #Turn OFF light YELLOW_2
29     sleep(0.5) #For 0.25 sec
30
```

```
30
31 while True: # Infinite Loop
32     #Phase 1
33     led_red_1.value(1) #Turn ON light RED_1
34     led_green_2.value(1) #Turn ON light GREEN_2
35     sleep(5) #For 5 sec
36     led_green_2.value(0) #Turn OFF light GREEN_2
37     led_red_1.value(1) #Turn ON light RED_1 while YELLOW_2 is ON
38     led_yellow_2.value(1) #Turn ON light YELLOW_2
39     sleep(2) #For 2 sec
40     led_yellow_2.value(0) #Turn OFF light YELLOW_2
41     led_red_2.value(1) #Turn ON light RED_1
42     sleep(2) #For 2 sec overlap
43     led_red_1.value(0) #Turn OFF light RED_1
44     led_red_2.value(0) #Turn OFF light RED_2
45     sleep(0) #Immediately
46     #Phase 2
47     led_red_2.value(1) #Turn ON light RED_2
48     led_green_1.value(1) #Turn ON light GREEN_1
49     sleep(5) #For 7 secs.
50     led_green_1.value(0) #Turn OFF light GREEN_1
51     led_red_2.value(1) #Turn ON light RED_2 while YELLOW_1 is ON
52     led_yellow_1.value(1) #Turn ON light YELLOW_1
53     sleep(2) #For 2 sec
54     led_yellow_1.value(0) #Turn OFF light YELLOW_2
55     led_red_1.value(1) #Turn ON light RED_1
56     sleep(2) #For 2 sec overlap
57     led_red_1.value(0) #Turn OFF light RED_1
58     led_red_2.value(0) #Turn OFF light RED_2
59     sleep(0) #Immediately
```

LET'S GO MAKE OUR PROGRAMS

3. with external buttons and NO overlap of red lights

TL NO overlap Buttons.py ×

```
1 #Importing libraries
2 from machine import Pin
3 from utime import sleep
4
5 #Definitions
6 led_red_1 = Pin(2, Pin.OUT) #Assigns the variable "led_red_1" the GPIO 2 (pin 1 on the board)
7 led_yellow_1 = Pin(3, Pin.OUT) #Assigns the variable "led_yellow_1" the GPIO 3 (pin 2 on the board)
8 led_green_1 = Pin(4, Pin.OUT) #Assigns the variable "led_green_1" the GPIO 4 (pin 4 on the board)
9 led_red_2 = Pin(10, Pin.OUT) #Assigns the variable "led_red_2" the GPIO 10 (pin 5 on the board)
10 led_yellow_2 = Pin(11, Pin.OUT) #Assigns the variable "led_yellow_2" the GPIO 11 (pin 6 on the board)
11 led_green_2 = Pin(12, Pin.OUT) #Assigns the variable "led_green_2" the GPIO 12 (pin 7 on the board)
12 start = Pin(15, Pin.IN, machine.Pin.PULL_DOWN) #Assigns the variable "start" the GPIO 15 (pin 20 on the board)
13 service = Pin(14, Pin.IN, machine.Pin.PULL_DOWN) #Assigns the variable "service" the GPIO 14 (pin 19 on the board)
14 START_STATE = 0 #Set START_STATE variable to "0"
15 SERVICE_STATE = 0 #Set SERVICE_STATE variable to "0"
16
17 #Light Initialization
18 #All the LEDs are turned off during the initialization phase to be sure the sequence starts with all lights off
19 led_red_1.value(0) #Turn OFF light RED_1
20 led_red_2.value(0) #Turn OFF light RED_2
21 led_yellow_1.value(0) #Turn OFF light YELLOW_1
22 led_yellow_2.value(0) #Turn OFF light YELLOW_2
23 led_green_1.value(0) #Turn OFF light GREEN_1
24 led_green_2.value(0) #Turn OFF light GREEN_2
25 sleep(1) #Wait a second
26
27 def on():
28 #Phase 1
29 led_red_1.value(1) #Turn ON light RED_1
30 led_green_2.value(1) #Turn ON light GREEN_2
31 sleep(5) #For 5 sec
32 led_green_2.value(0) #Turn OFF light GREEN_2
33 led_red_1.value(1) #Turn ON light RED_1 while YELLOW_2 is ON
34 led_yellow_2.value(1) #Turn ON light YELLOW_2
35 sleep(2) #For 2 sec
36 led_yellow_2.value(0) #Turn OFF light YELLOW_2
37 led_red_1.value(0) #Turn ON light RED_1
38 sleep(0) #For 2 sec overlap
```

```
39 #Phase 2
40 led_red_2.value(1) #Turn ON light RED_2
41 led_green_1.value(1) #Turn ON light GREEN_1
42 sleep(5) #For 5 secs.
43 led_green_1.value(0) #Turn OFF light GREEN_1
44 led_red_2.value(1) #Turn ON light RED_2 while YELLOW_1 is ON
45 led_yellow_1.value(1) #Turn ON light YELLOW_1
46 sleep(2) #For 2 sec
47 led_yellow_1.value(0) #Turn OFF light YELLOW_2
48 led_red_2.value(0) #Turn ON light RED_1
49 sleep(0) #Immediately
50
51 def maintenance():
52 led_yellow_1.value(1) #Turn ON light YELLOW_1
53 led_yellow_2.value(1) #Turn ON light YELLOW_2
54 sleep(.25) #For 0.25 sec
55 led_yellow_1.value(0) #Turn OFF light YELLOW_1
56 led_yellow_2.value(0) #Turn OFF light YELLOW_2
57 sleep(0.25) #For 0.25 sec
58
59 while True: # Infinite Loop
60 START_STATE=start.value()
61 if START_STATE==1:
62 for i in range(1,3):
63 led_yellow_1.value(1) #Turn ON light YELLOW_1
64 led_yellow_2.value(1) #Turn ON light YELLOW_2
65 sleep(0.5) #For 0.25 sec
66 led_yellow_1.value(0) #Turn OFF light YELLOW_1
67 led_yellow_2.value(0) #Turn OFF light YELLOW_2
68 sleep(0.5) #For 0.25 sec
69 while START_STATE==1: #If the state of the START_STATE variable is at level 1 ("start" button pressed), the following instructions are executed
70 print("START %s - SERVICE %d" % (START_STATE,SERVICE_STATE)) #This is a check that can be useful during the programming phase to see the status of the variables
71 if SERVICE_STATE==0: #If the state of the SERVICE_STATE variable is at level 0 ("service" switch OFF), the program hands the control to the "on()" function:
72 on() #Here the "on()" function is executed if the "service" switch was at level 0 (OFF) in the previous instruction
73 else: #Otherwise, if the status of the SERVICE_STATE variable is at level 1 ("service" switch ON), the program hands the control to the "maintenance()" function:
74 maintenance() #Here the "maintenance()" function is executed if the "service" switch was pressed in the previous instruction
75 SERVICE_STATE=service.value() #Here, at the end of the while loop, the status of the "service" switch (Pin 15) is read and sent to the "SERVICE_STATE" variable.
76 #This variable is evaluated in the "if" statement in line 71 and if its value is at 0 then the "on" function is called.
77 #Otherwise if its value is at 1, the "maintenance" function is called.
78 #Note that in the meantime START_STATE variable remained at 1.
79
80 #Keep in mind that Python does not use curly brackets to contain instructions or functions but uses "indents".
81 #i.e. it reads the position on the listing with respect to the left margin.
```

LET'S GO MAKE OUR PROGRAMS

4. with external buttons - with overlap of red lights

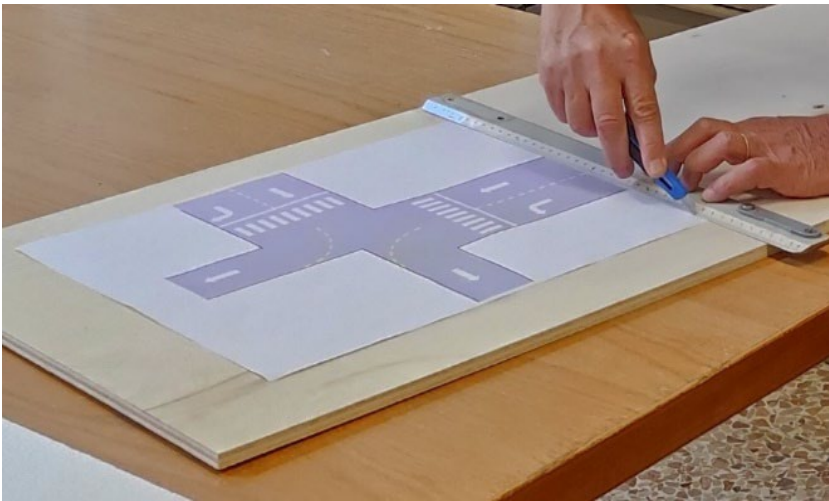
TL overlap Buttons.py x

```
1 #Importing libraries
2 from machine import Pin
3 from utime import sleep
4
5 #Definitions
6 led_red_1 = Pin(2, Pin.OUT) #Assigns the variable "led_red_1" the GPIO 2 (pin 1 on the board)
7 led_yellow_1 = Pin(3, Pin.OUT) #Assigns the variable "led_yellow_1" the GPIO 3 (pin 2 on the board)
8 led_green_1 = Pin(4, Pin.OUT) #Assigns the variable "led_green_1" the GPIO 4 (pin 4 on the board)
9 led_red_2 = Pin(10, Pin.OUT) #Assigns the variable "led_red_2" the GPIO 10 (pin 5 on the board)
10 led_yellow_2 = Pin(11, Pin.OUT) #Assigns the variable "led_yellow_2" the GPIO 11 (pin 6 on the board)
11 led_green_2 = Pin(12, Pin.OUT) #Assigns the variable "led_green_2" the GPIO 12 (pin 7 on the board)
12 start = Pin(15, Pin.IN, machine.Pin.PULL_DOWN) #Assigns the variable "start" the GPIO 15 (pin 20 on the board)
13 service = Pin(14, Pin.IN, machine.Pin.PULL_DOWN) #Assigns the variable "service" the GPIO 14 (pin 19 on the board)
14 START_STATE = 0 #Set START_STATE variable to "0"
15 SERVICE_STATE = 0 #Set SERVICE_STATE variable to "0"
16
17 #Light Initialization
18 #All the LEDs are turned off during the initialization phase to be sure the sequence starts with all lights off
19 led_red_1.value(0) #Turn OFF light RED_1
20 led_red_2.value(0) #Turn OFF light RED_2
21 led_yellow_1.value(0) #Turn OFF light YELLOW_1
22 led_yellow_2.value(0) #Turn OFF light YELLOW_2
23 led_green_1.value(0) #Turn OFF light GREEN_1
24 led_green_2.value(0) #Turn OFF light GREEN_2
25 sleep(1) #Wait a second
26
27 def on():
28 #Phase 1
29 led_red_1.value(1) #Turn ON light RED_1
30 led_green_2.value(1) #Turn ON light GREEN_2
31 sleep(5) #For 5 sec
32 led_green_2.value(0) #Turn OFF light GREEN_2
33 led_red_1.value(1) #Turn ON light RED_1 while YELLOW_2 is ON
34 led_yellow_2.value(1) #Turn ON light YELLOW_2
35 sleep(2) #For 2 sec
36 led_yellow_2.value(0) #Turn OFF light YELLOW_2
37 led_red_2.value(1) #Turn ON light RED_2
38 sleep(2) #For 2 sec overlap
39 led_red_1.value(0) #Turn OFF light RED_1
40 led_red_2.value(0) #Turn OFF light RED_2
41 sleep(0) #Immediately
```

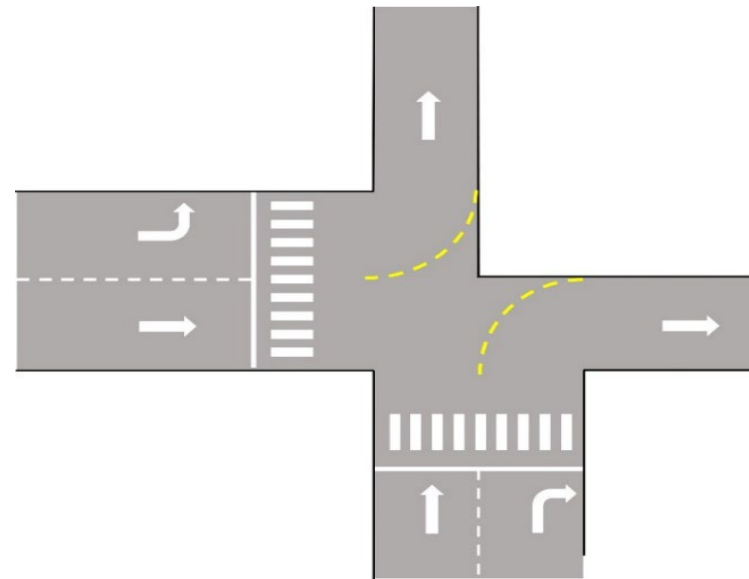
```
42 #Phase 2
43 led_red_2.value(1) #Turn ON light RED_2
44 led_green_1.value(1) #Turn ON light GREEN_1
45 sleep(5) #For 7 secs.
46 led_green_1.value(0) #Turn OFF light GREEN_1
47 led_red_2.value(1) #Turn ON light RED_2 while YELLOW_1 is ON
48 led_yellow_1.value(1) #Turn ON light YELLOW_1
49 sleep(2) #For 2 sec
50 led_yellow_1.value(0) #Turn OFF light YELLOW_2
51 led_red_1.value(1) #Turn ON light RED_1
52 sleep(2) #For 2 sec overlap
53 led_red_1.value(0) #Turn OFF light RED_1
54 led_red_2.value(0) #Turn OFF light RED_2
55 sleep(0) #Immediately
56
57 def maintenance():
58 led_yellow_1.value(1) #Turn ON light YELLOW_1
59 led_yellow_2.value(1) #Turn ON light YELLOW_2
60 sleep(.25) #For 0.25 sec
61 led_yellow_1.value(0) #Turn OFF light YELLOW_1
62 led_yellow_2.value(0) #Turn OFF light YELLOW_2
63 sleep(0.25) #For 0.25 sec
64
65 while True: # Infinite Loop
66 START_STATE=start.value()
67 if START_STATE==1:
68 for i in range(1,3):
69 led_yellow_1.value(1) #Turn ON light YELLOW_1
70 led_yellow_2.value(1) #Turn ON light YELLOW_2
71 sleep(0.5) #For 0.25 sec
72 led_yellow_1.value(0) #Turn OFF light YELLOW_1
73 led_yellow_2.value(0) #Turn OFF light YELLOW_2
74 sleep(0.5) #For 0.25 sec
75 while START_STATE==1: #If the state of the START_STATE variable is at level 1 ("start" button pressed), the following instructions are executed
76 print("START %s - SERVICE %d" %(START_STATE,SERVICE_STATE)) #This is a check that can be useful during the programming phase to see the status of the variables
77 if SERVICE_STATE==0: #If the state of the SERVICE_STATE variable is at level 0 ("service" switch OFF), the program hands the control to the "on()" function:
78 on() #Here the "on()" function is executed if the "service" switch was at level 0 (OFF) in the previous instruction
79 else: #Otherwise, if the status of the SERVICE_STATE variable is at level 1 ("service" switch ON), the program hands the control to the "maintenance()" function:
80 maintenance() #Here the "maintenance()" function is executed if the "service" switch was pressed in the previous instruction
81 SERVICE_STATE=service.value() #Here, at the end of the while loop, the status of the "service" switch (Pin 15) is read and sent to the "SERVICE_STATE" variable.
82 #This variable is evaluated in the "if" statement in line 71 and if its value is at 0 then the "on" function is called.
83 #Otherwise if its value is at 1, the "maintenance" function is called.
84 #Note that in the meantime START_STATE" variable remained at 1.
85 #Keep in mind that Python does not use curly brackets to contain instructions or functions but uses "indents".
86 #i.e. it reads the position on the listing with respect to the left margin.
```

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



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 as shown below

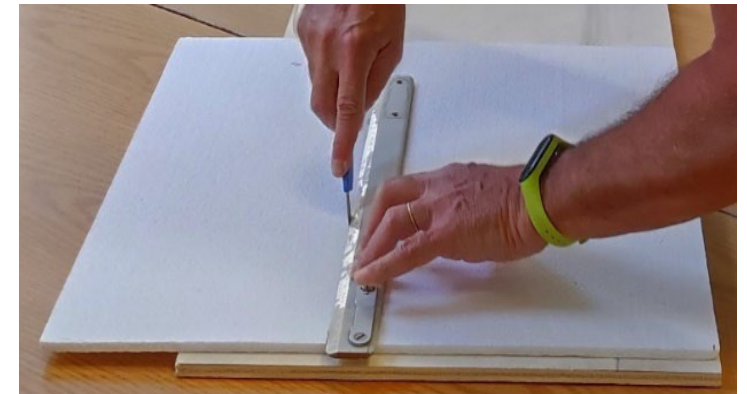


DRAWING AND CRAFTING THE FLOOR PLAN OF TWO WAYS JUNCTION

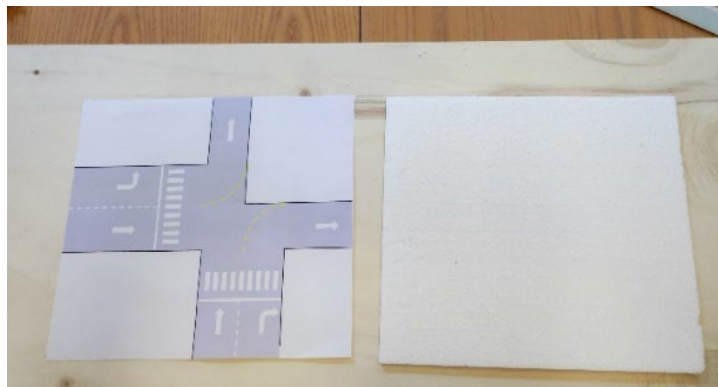
After creating the paper layout it will be necessary to create a 1 cm high styrofoam base where the breadboards will be inserted



The styrofoam sheet must have the size of the paper sheet (you can use a A4 sheet) used to create the graphic layout

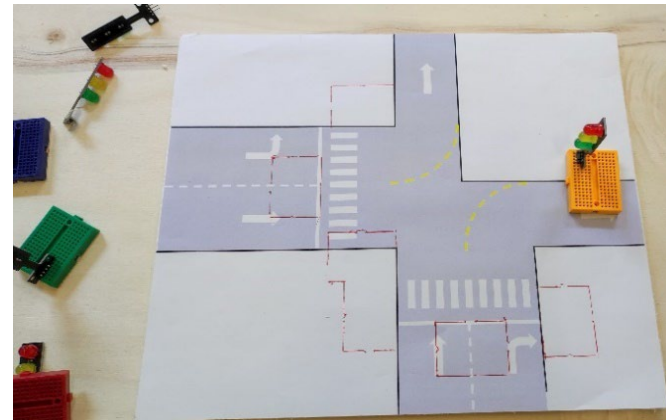
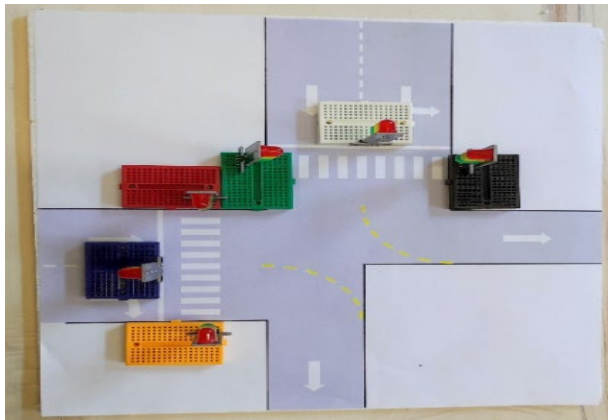


The layout and the styrofoam sheets cut to size are shown below

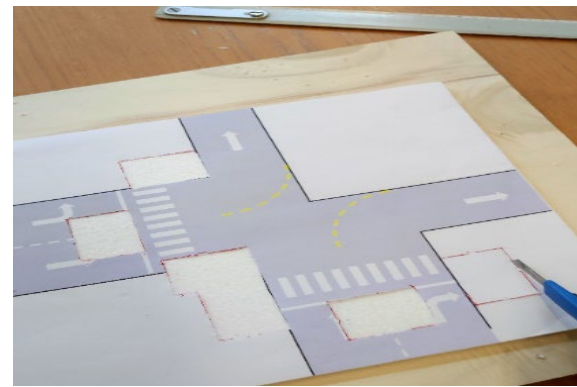
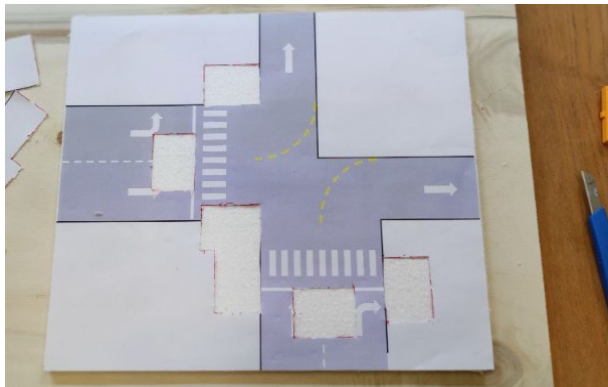


DRAWING AND CRAFTING THE FLOOR PLAN OF TWO WAYS JUNCTION

The next step will be to mark the exact position of the breadboards on a drawing of the intersection using a previously created layout

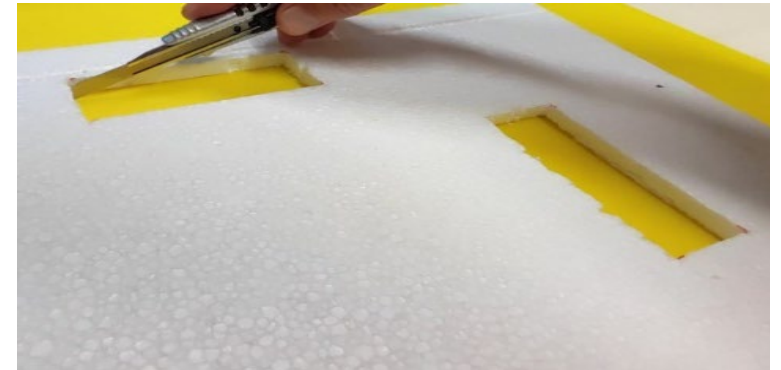
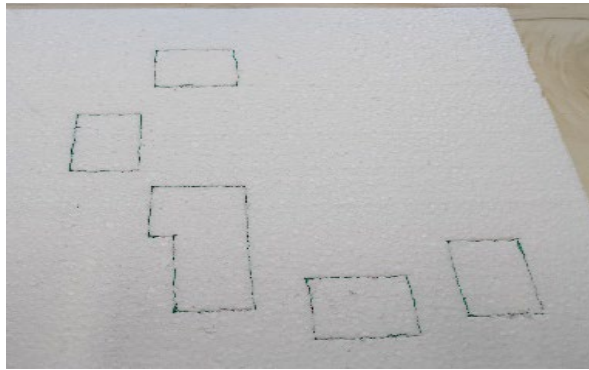
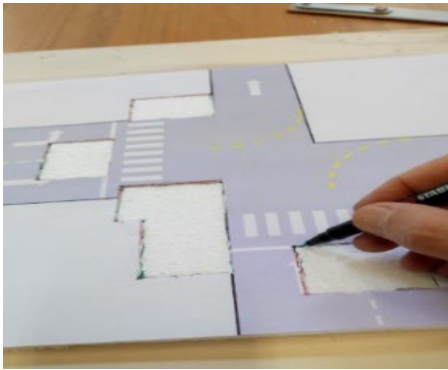


Now we have to cut out the breadboard slots in the styrofoam sheet to insert the breadboards in

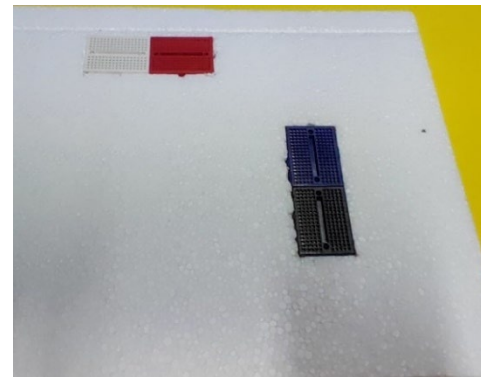
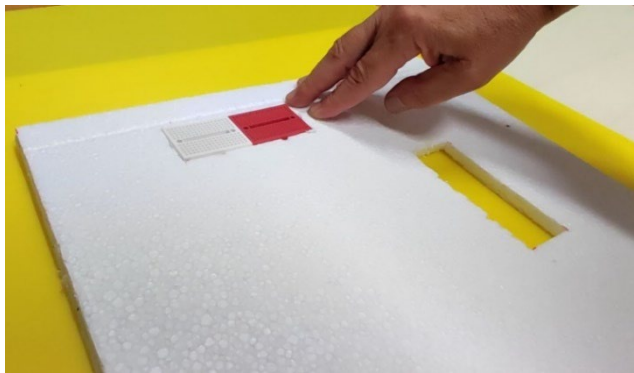


DRAWING AND CRAFTING THE FLOOR PLAN OF TWO WAYS JUNCTION

Now we will cut out the breadboard slots in the styrofoam sheet to insert the breadboards in

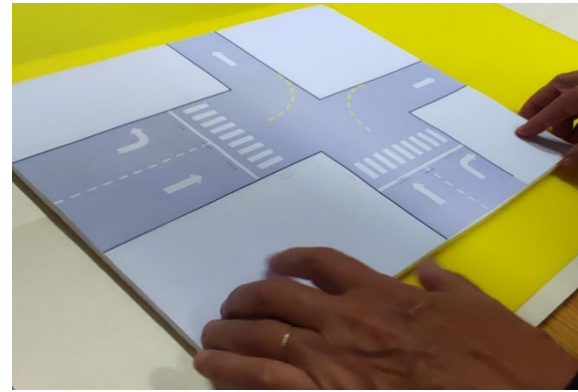
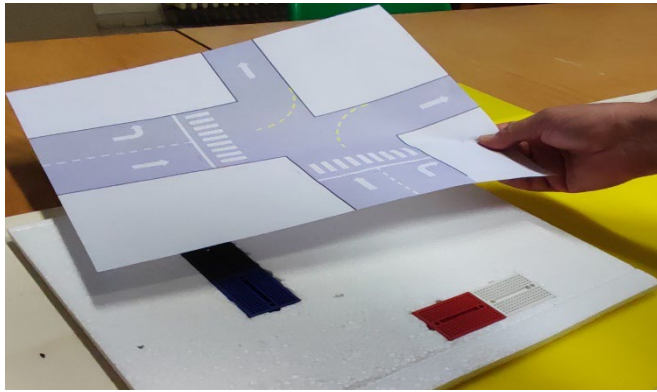


Now we have to cut out the breadboard slots in the styrofoam sheet to insert the breadboards in

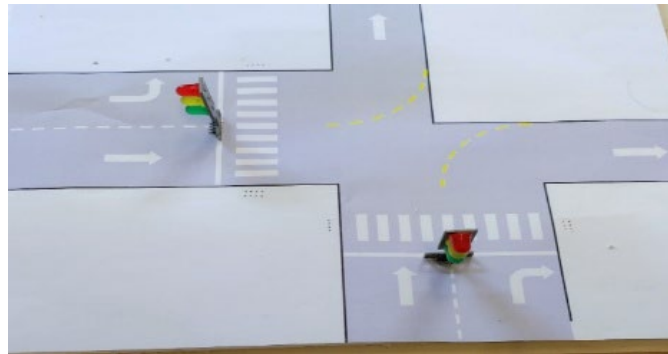


DRAWING AND CRAFTING THE FLOOR PLAN OF TWO WAYS JUNCTION

We can now put the plan on a alveolar polypropylene substrate and proceed to insert the mini led traffic lights on the relative breadboard

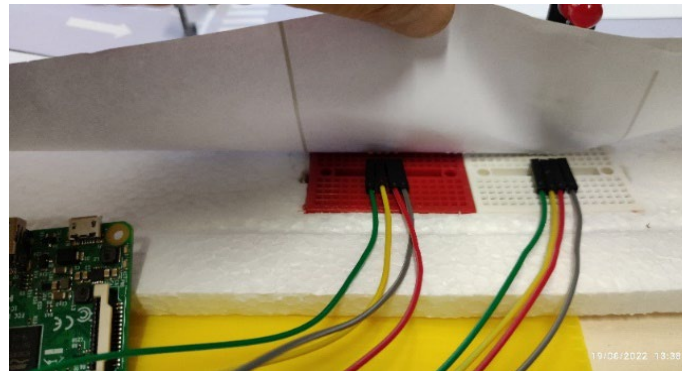


Below the layout of the complete traffic light with the two LEDs traffic lights inserted in the breadboard

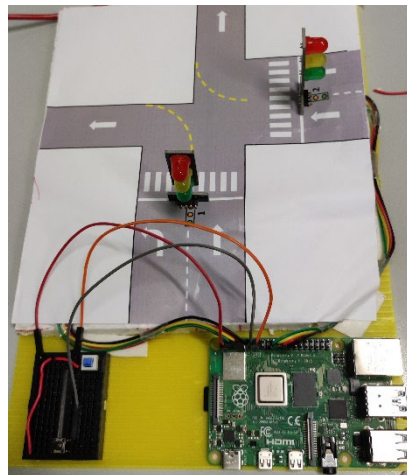


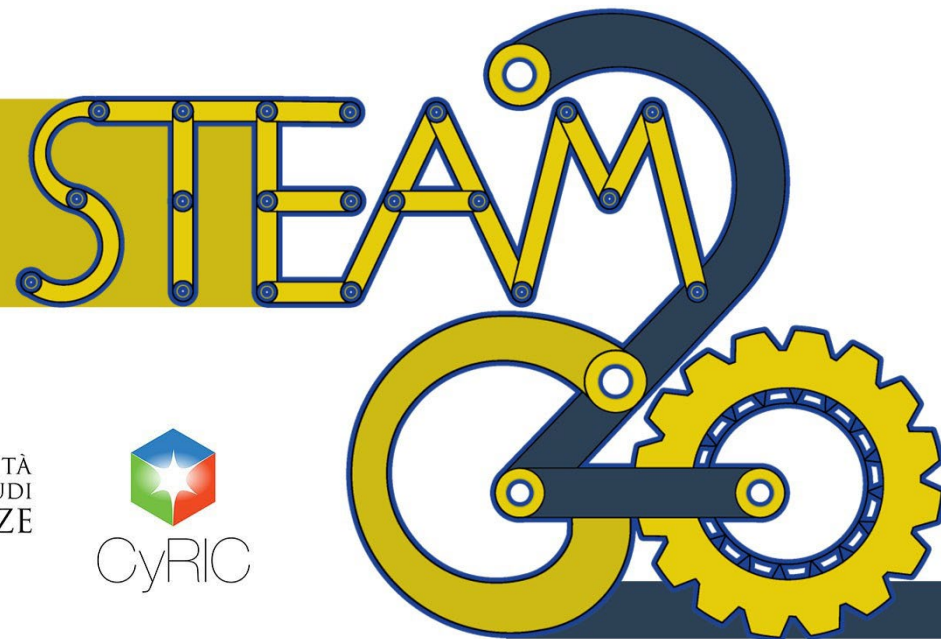
DRAWING AND CRAFTING THE FLOOR PLAN OF TWO WAYS JUNCTION

Last step: connect the LED devices to the Raspberry Pi Pico board



Here an example of the complete system





Erasmus+

