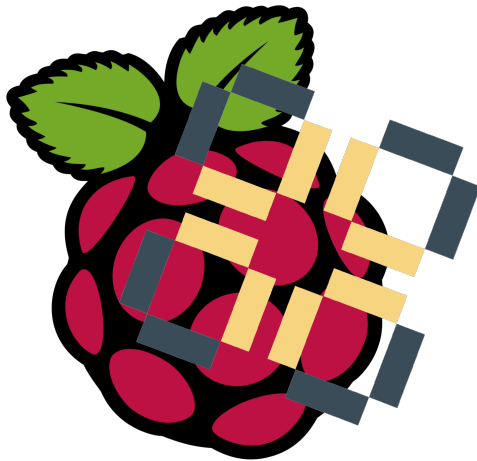


# HepiaLight3

## Raspberry Pi Pico

Université d'Été 2024



Michael Divià (N° 22649552)  
Alejandro Escribano Martín (N° 15315914)  
Gaspard Le Gouic (N° 19816289)

---

vendredi 28 juin 2024  
Informatique et Systèmes de Communication



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Répartition des tâches initiales</b>	<b>3</b>
<b>3</b>	<b>Librairie hepialight3</b>	<b>4</b>
3.1	Connections . . . . .	4
3.2	Display . . . . .	5
3.2.1	Predefined colors . . . . .	5
3.2.2	Clear the matrix . . . . .	6
3.2.3	Set a line of the matrix . . . . .	6
3.2.4	Set a column of the matrix . . . . .	6
3.2.5	Set a LED of the matrix . . . . .	7
3.2.6	Get an LED color . . . . .	7
3.2.7	Display text . . . . .	8
3.2.8	Display image . . . . .	8
3.3	Communication . . . . .	9
3.3.1	UART . . . . .	9
3.4	External Sensors . . . . .	10
3.4.1	I2C . . . . .	10
3.4.2	SPI . . . . .	12
<b>4</b>	<b>Conclusion</b>	<b>13</b>
	<b>Bibliographie</b>	<b>14</b>

## Table des figures

1	Raspberry Pi Pico Pinout for HepiaLight3 . . . . .	4
2	UART communication between Raspberry Pi Pico Pinout for HepiaLight3 . . . . .	4

## 1 Introduction

Dans le cadre de l'**Université d'Été 2024**, il nous a été proposé de réaliser un projet basé sur **HepiaLight 2**. Dans ce projet, nous devons évaluer le portage de ce projet sur différentes nouvelles architectures afin de démontrer la possibilité de développement d'une nouvelle carte **HepiaLight 3**. Pour notre groupe, nous nous sommes vus attribués un **RP2040-PICO-HDR** ainsi qu'une matrice de LED numérique **Adafruit NeoPixel NeoMatrix 8x8 - 64 RGB LED Pixel Matrix**. Nous allons donc devoir implémenter, en **MicroPython**, toutes les fonctionnalités de base du projet **HepiaLight 2** en nous adaptant aux différentes limitations qu'apporte la nouvelle architecture **RP2040**.

## 2 Répartition des tâches initiales

Ne connaissant pas les contraintes de codage avec **MicroPython** et sur le **RP2040-PICO-HDR**, nous nous sommes initialement réparti les tâches du cahier des charges comme suit :

**Set matrix** : Michael Divià

**Set pixel** : Michael Divià

**Set texte** : Gaspard Le Gouic

**Scroll text** : Gaspard Le Gouic

**init UART** : Alejandro Escribano

**Write UART** : Alejandro Escribano

**Read UART** : Alejandro Escribano

Nous avons pour but initial de terminer toutes ces tâches avant la fin de la première semaine afin de pouvoir, par la suite, implémenter des fonctionnalités supplémentaires.

Cependant, dû à des limitations de temps externes, entre autres, cette répartition à été modifiée de jour en jour afin de pouvoir avancer au mieux sur ce projet.

### 3 Librairie hepialight3

This section will be in English. As this is the most technical section of the document and the one that, in case this is the selected architecture, will be used for the development of the new HepiaLight model.

#### 3.1 Connections

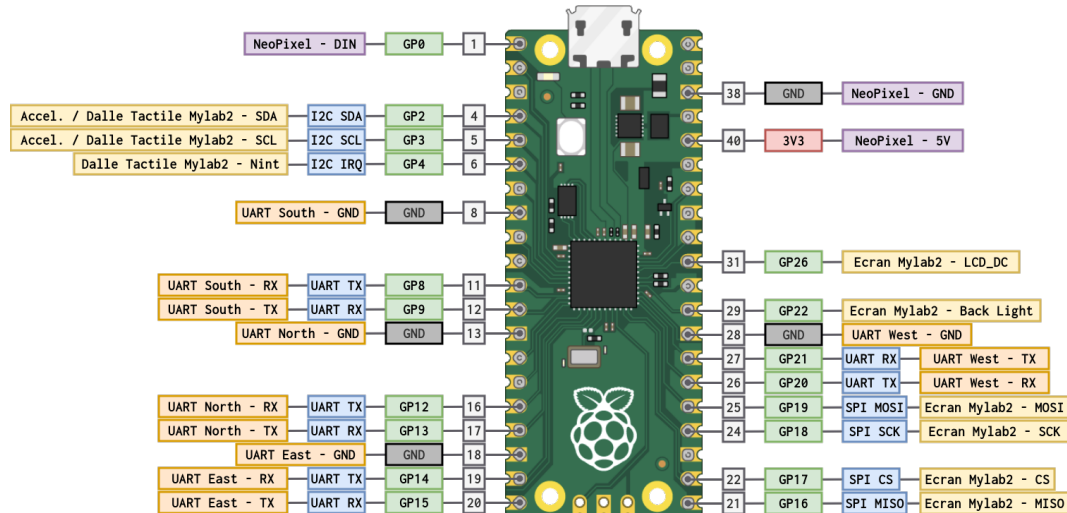


FIGURE 1 – Raspberry Pi Pico Pinout for HepiaLight3

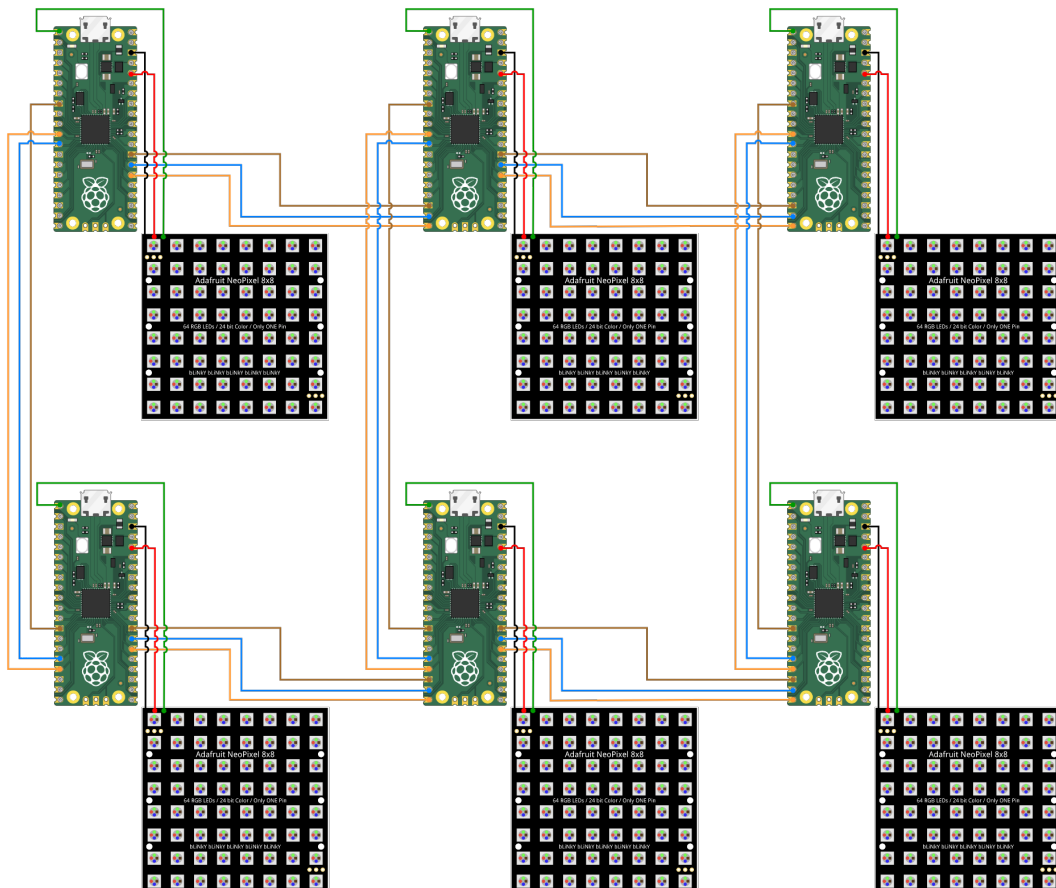


FIGURE 2 – UART communication between Raspberry Pi Pico Pinout for HepiaLight3

## 3.2 Display

### 3.2.1 Predefined colors

Here are all the predefined colors that come in the class Color with there RGB values :

```
class Color :
```

```
Color.BLACK = (0, 0, 0)
Color.BLUE = (0, 0, 255)
Color.CYAN = (0, 255, 255)
Color.GREEN = (0, 255, 0)
Color.MAGENTA = (255, 0, 255)
Color.RED = (255, 0, 0)
Color.YELLOW = (255, 255, 0)
Color.WHITE = (255, 255, 255)
Color.RED_DARKER = (153, 0, 0)
Color.RED_DARK = (204, 0, 0)
Color.RED_LIGHT = (255, 102, 102)
Color.RED_LIGHTER = (255, 153, 153)
Color.BLUE_DARKER = (0, 0, 153)
Color.BLUE_DARK = (0, 0, 204)
Color.BLUE_LIGHT = (102, 102, 255)
Color.BLUE_LIGHTER = (153, 153, 255)
Color.GREEN_DARKER = (0, 153, 0)
Color.GREEN_DARK = (0, 204, 0)
Color.GREEN_LIGHT = (102, 255, 102)
Color.CYAN_DARK = (0, 204, 204)
Color.CYAN_LIGHT = (102, 255, 255)
Color.MAGENTA_DARKER = (153, 0, 153)
Color.MAGENTA_DARK = (204, 0, 204)
Color.MAGENTA_LIGHT = (255, 102, 255)
Color.YELLOW_DARKER = (153, 153, 0)
Color.YELLOW_DARK = (204, 204, 0)
Color.YELLOW_LIGHT = (255, 255, 102)
Color.GRAY_DARK = (64, 64, 64)
Color.GRAY = (128, 128, 128)
Color.GRAY_LIGHT = (192, 192, 192)
Color.ORANGE_DARK = (204, 102, 0)
Color.ORANGE = (255, 128, 0)
Color.ORANGE_YELLOW = (255, 204, 0)
```

### 3.2.2 Clear the matrix

Set the whole LED matrix to the specified color.

#### Usage :

```
Matrix.clear(Color)
```

#### Example :

```
Matrix.clear(Color.RED)  
Matrix.clear((255, 0, 0))  
Matrix.clear(0xFF0000)
```

Clearing the matrix in red.

#### Special case :

```
Matrix.clear(0)
```

Turn off the whole matrix.

### 3.2.3 Set a line of the matrix

Set a specific line of the LED matrix to the specified color.

#### Usage :

```
Matrix.set_line(line, Color)
```

Line must be between 0 and 7.

#### Example :

```
Matrix.set_line(5, Color.GREEN)
```

Setting line 5 (the 6th line) in green.

### 3.2.4 Set a column of the matrix

Set a specific column of the LED matrix to the specified color.

#### Usage :

```
Matrix.set_column(line, Color)
```

Column must be between 0 and 7.

**Example :**

```
Matrix.set_column(0, Color.YELLOW)
```

Setting line 0 (the 1st column) in yellow.

**3.2.5 Set a LED of the matrix**

Set a specific LED of the matrix to the specified color.

**Usage :**

```
Matrix.set_led(column, line, Color)
```

Position is represented by a column and line, where both must be between 0 and 7.

**Example :**

```
Matrix.set_led(3, 5, Color.ORANGE)
```

Setting LED at position column 3, line 5 in orange.

**3.2.6 Get an LED color**

Get the current color of a specific LED of the matrix.

**Usage :**

```
Matrix.get_led(column, line)
```

Position is represented by a column and line, where both must be between 0 and 7.

**Example :**

```
color = Matrix.get_led(7, 0)
```

Getting the color of the LED at column 7, line 0.

### 3.2.7 Display text

Display a scrolling message of the specified color using the specified delay in seconds.

**Usage :**

```
show_text(text, Color, speed)
```

Speed is in seconds.

**Example :**

```
show_text("PiPo", Color.RED, 0.1)  
show_text("PiPo", Color.RED, 1/10)
```

scrolling «Pipo» message in red at a delay of 0.1 second.

### 3.2.8 Display image

Display a full matrix of colors.

**Usage :**

```
set_img(matrix)
```

**Example :**

```
set_img(""" .RR..RR.  
            RRRRRRRR  
            RRRRRRRR  
            RRRRRRRR  
            RRRRRRRR  
            .RRRRRR.  
            ..RRRR..  
            ...RR...""")
```

**Available colors :**

```
R : red  
G : green  
B : blue  
C : cyan  
V : violet  
Y : yellow  
W : white  
. : black
```

### 3.3 Communication

#### 3.3.1 UART

Transmit or receive data via UART with up to 4 other cards.

##### Usage :

```
Uart(Direction, baudrate, parity, bits, stop)
```

Create a UART communication.

```
x.send(string)
```

Send data

```
x.sendline(string)
```

Send line of data. '\n' will be added at the end of the string automatically.

```
x.receive(length)
```

Wait until you received the ask length of data.

```
x.receive_line(length)
```

Wait until you received the full line of data (ending with a '\n').

##### Example :

```
uart_north = Uart(Direction.NORTH)
uart_south = Uart(Direction.SOUTH)
uart_east = Uart(Direction.EAST)
uart_west = Uart(Direction.WEST)

uart_east.sendline("123")
uart_west.send("45")
data_north = uart_north.receive(3)
data_south = uart_south.receive_line()
print(f"Data received from North: {data_north}")
print(f"Data received from South: {data_south}")
```

Default values are :

**baudrate** : 9'600

**parity** : None

**bits** : 8

**stop** : 1

Each Direction Pin Out can be found on figure 1

### 3.4 External Sensors

#### 3.4.1 I2C

Receive data from MyLab2 accelerometer.

##### Usage :

```
Accel.init()
```

Initialize MyLab2 accelerometer.

```
Accel.get_x()
```

Returns X value of accelerometer in g.

```
Accel.get_y()
```

Returns Y value of accelerometer in g.

```
Accel.get_z()
```

Returns Z value of accelerometer in g.

```
Accel.facing(side)
```

Returns True/False if facing in the asked side.

```
Accel.tilting(dir)
```

Returns True/False if tilting the asked direction.

```
accel_test()
```

Print in terminal the facing direction, the tilting direction and the acceleration values for the 3 axis

##### Example :

```
Accel.init()

x_value = Accel.get_x()
print(f"Acc. axe X: {x_value}")

facing_down = Accel.facing(Direction.BACK)
print(f"MyLab2 facing down ? {facing_down}")
```

Get the X value of accelerometer and check if it is facing down.

Receive data from MyLab2 touch screen.

#### Usage :

```
Touch.attach(cb)
```

Attach callback to touchscreen interruption.

```
Touch.detach()
```

Detach callback.

```
Touch.callback_do()
```

Execute the callback attached.

```
Touch.init()
```

Initialize MyLab2 touchscreen.

```
Touch.read(pos)
```

Return True/False if touching the asked dial.

```
Touch.compute_pos(x, y)
```

Convert screen position to dial position.

```
Touch.read_pos()
```

Return the position of the touch on screen.

```
Touch.touch_cb(pos)
```

Callback function to display corresponding color on matrix from touched screen dial.

```
Touch.handler()
```

Touchscreen IRQ handler.

```
touch_test()
```

Print the current touched dial position.

```
touch_irq_test()
```

Setup the callback and the IRQ.

**Example :**

```
touch_irq_test()
while True:
    touch_test()
    utime.sleep(1)
```

Print the current touched dial position and display color on matrix depending on touched dial position.

**3.4.2 SPI**

Display pixels on MyLab2 screen.

**Usage :**

```
Lcd.write_cmd(cmd)
```

Write command "cmd" to MyLab2 LCD.

```
Lcd.write_data()
```

Write data "data" to MyLab2 LCD.

```
Lcd.init()
```

Initialize MyLab2 LCD.

```
Lcd.set_window()
```

Set a window on MyLab2 screen.

```
lcd_test()
```

Display 4 dials on MyLab2 screen with colors white, red, green and blue.

**Example :**

```
lcd_test()
while True:
    utime.sleep(1)
```

Display 4 dials on MyLab2 screen with colors white, red, green and blue.

## 4 Conclusion

L'adoption du Raspberry Pi Pico comme architecture principale pour le projet **HepiaLight 3** est particulièrement pertinente et viable. Le Raspberry Pi Pico offre une solution économique et efficace, parfaitement adaptée à la gestion des LED RGB et au contrôle multiplexé nécessaire pour la carte **HepiaLight 2**. Avec son microcontrôleur **RP2040**, le Raspberry Pi Pico combine une bonne puissance de calcul avec des capacités de gestion des entrées/sorties robustes, ce qui est essentiel pour le contrôle précis des matrices de LED et la prise en charge des périphériques comme l'accéléromètre et les zones tactiles de la carte d'extension hypothétique (démontré avec une MyLab2). De plus, la communauté active et la documentation abondante autour du Raspberry Pi Pico facilitent grandement le développement et la programmation, permettant ainsi une évolution continue du projet **HepiaLight 3** tout en assurant une maintenance aisée à long terme.

L'utilisation du Raspberry Pi Pico comme architecture pour **HepiaLight 3** garantit donc non seulement la modernisation de la carte tout en conservant ses fonctionnalités clés, mais aussi une évolutivité et une compatibilité avec les besoins pédagogiques et les futurs développements du projet.

Pour conclure, le projet **HepiaLight 3** basé sur le Raspberry Pi Pico a représenté un défi stimulant pour notre équipe durant l'Université d'Été 2024. À travers l'implémentation en MicroPython des fonctionnalités de **HepiaLight 2** sur cette nouvelle architecture, nous avons non seulement démontré la faisabilité technique du portage, mais aussi surmonté les défis posés par les limitations spécifiques du **RP2040-PICO-HDR**. Malgré les ajustements nécessaires dans la répartition des tâches en raison de contraintes de temps, notre collaboration a permis de consolider nos compétences en programmation embarquée et en gestion de projet. Ce projet nous a également familiarisés avec les bibliothèques et les capacités de communication **UART**, enrichissant ainsi notre expérience dans le domaine des systèmes embarqués. Enfin, nous sommes fiers du résultat atteint et de la manière dont notre équipe a su s'adapter pour répondre aux objectifs fixés.

## Bibliographie

- [1] *HepiaLight 2*. Github. 2024. URL : <https://gitedu.hesge.ch/cores/projects/hepialight2>.
- [2] *MicroPython documentation*. MicroPython. 2024. URL : <https://docs.micropython.org/en/latest/>.
- [3] *MyLab2*. hepialsn - HESO-SO. 2024. URL : <https://hepialsn.hesge.ch/myLab2/>.
- [4] *Raspberry Pi Pico documentation*. Raspberrypi. 2024. URL : <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>.
- [5] *Thonny IDE*. Thonny. 2024. URL : <https://github.com/thonny/thonny/>.