U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

LCOM - 2021/2022
Project Report

TypeRacer

T04G01

Members:
up202005358@edu.fe.up.pt - Alexandre Ferreira Nunes
up202006485@edu.fe.up.pt - José Miguel Isidro
up201800700@edu.fe.up.pt - Tomás Torres
up202007544@edu.fe.up.pt - Sérgio Carvalhais

Summary

User Instructions

Description:

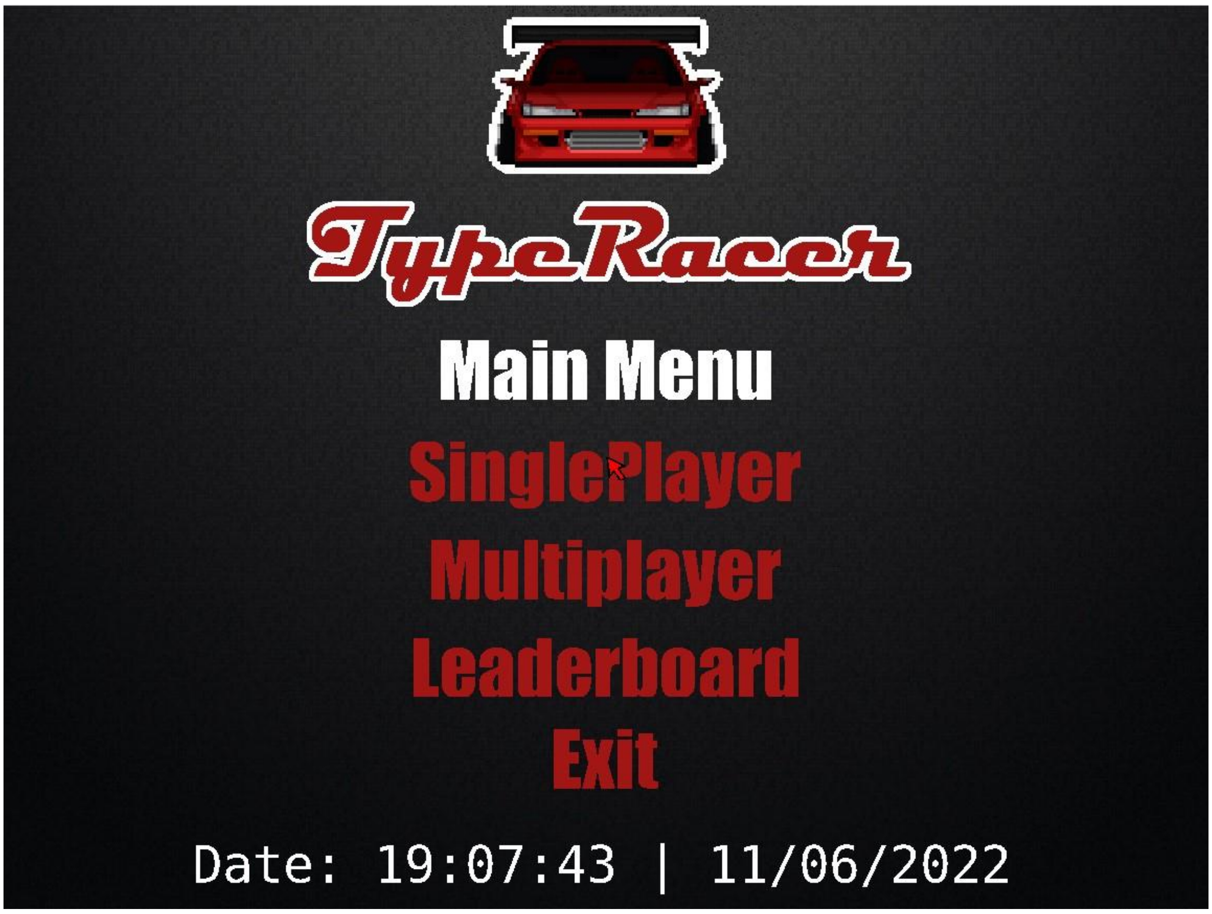
In TypeRacer, players complete typing tests of various texts as fast as possible, competing against themselves or with another player. We will add a mouse functionality where the player needs to drag and drop some words to the correct position before continuing the phrase.

How to play:

Keyboard: to type in words
Mouse Left-click: to drag words and select menu options.
ESC: to exit the game or return to the Main Menu.

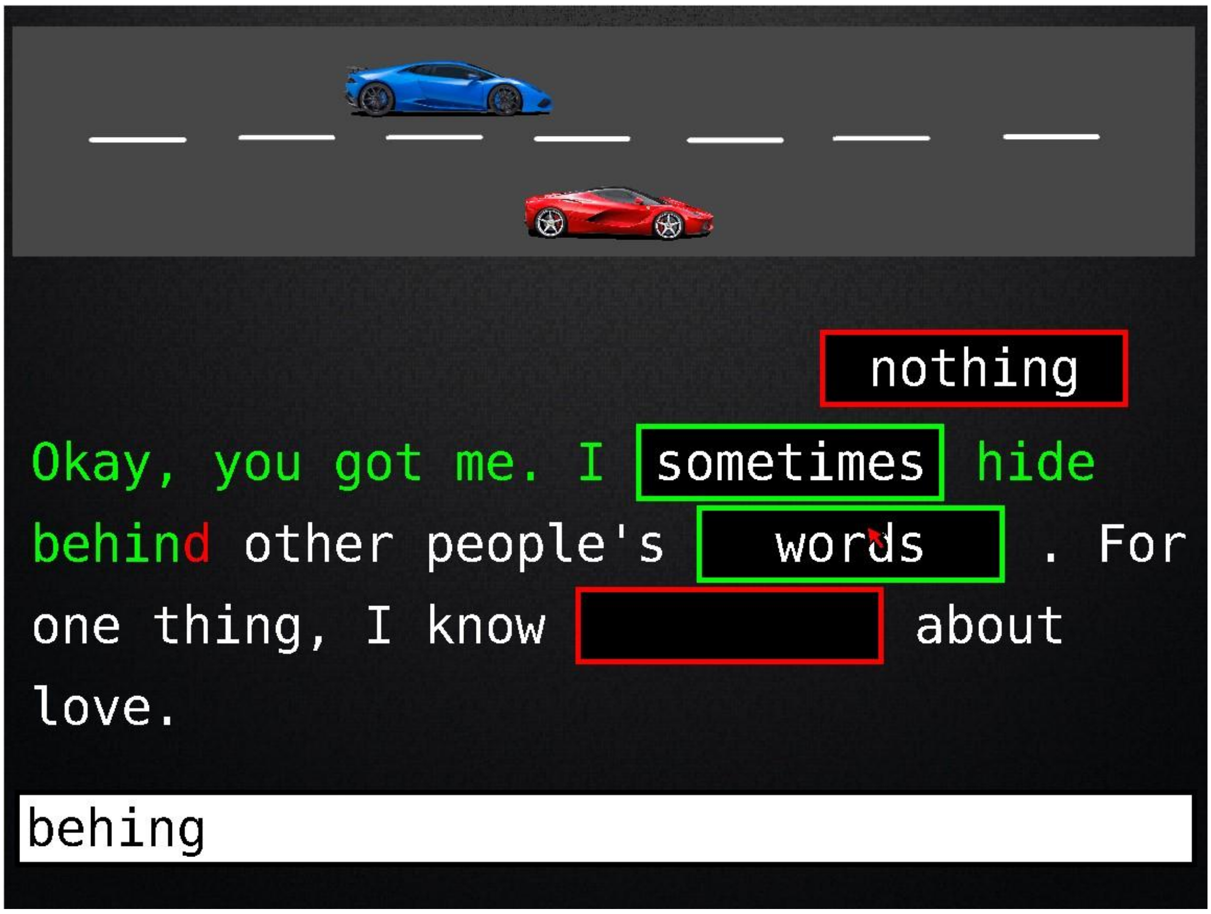
Game Functionalities:

1. **Main menu**



Date: 19:07:43 | 11/06/2022

2. SinglePlayer

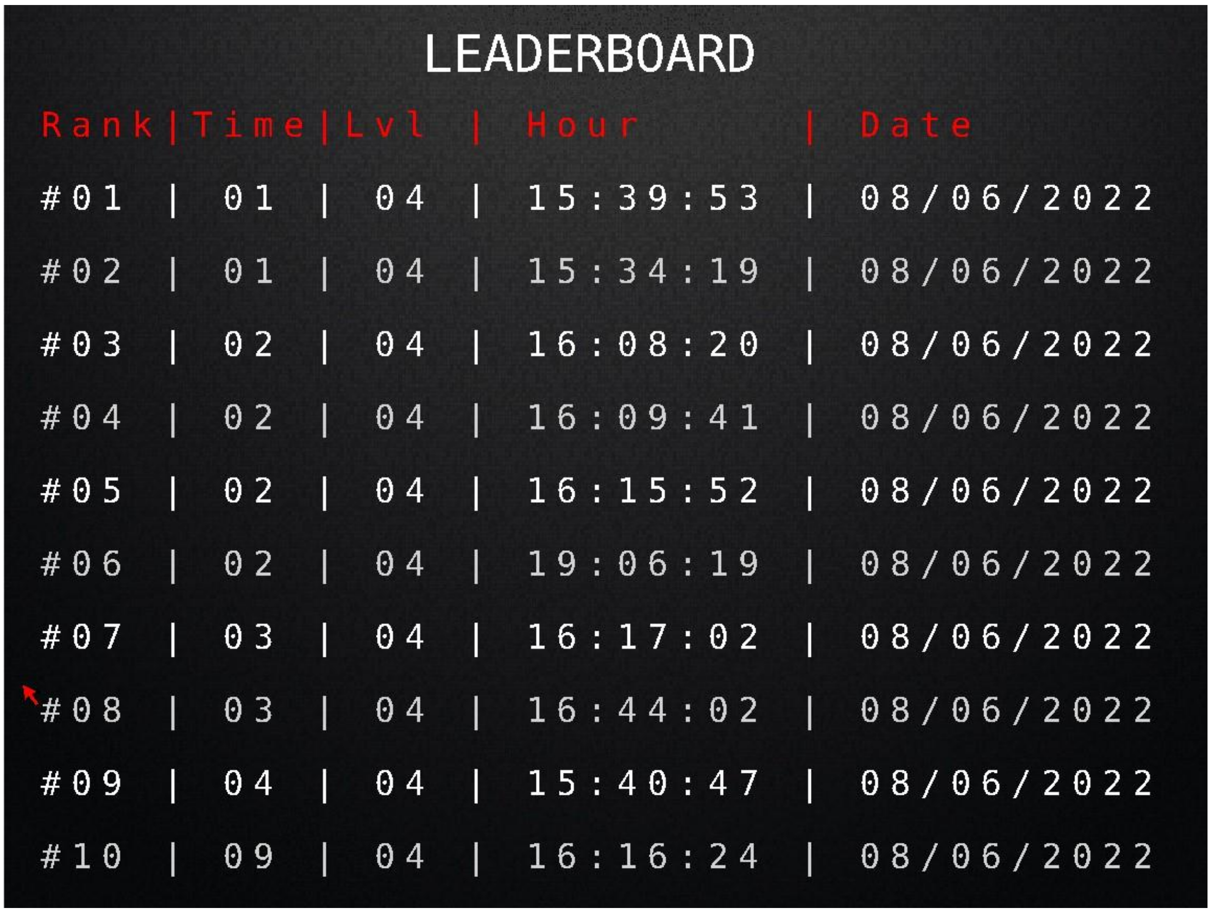

nothing

Okay, you got me. I sometimes hide behind other people's words . For one thing, I know about love.

behing

As the user types, the interactive text will change colors depending on the user input: the text's color will change to green if the input matches the words (and letters), otherwise the text's color will change to red.
This also happens with the blank spaced boxes, which the user has to complete by dragging the correct word into the respective place in the sentence. If the word is placed correctly, it will remain there and the border color will change to green.

3. Leaderboard



LEADERBOARD

Rank	Time	Lvl	Hour	Date
#01	01	04	15:39:53	08/06/2022
#02	01	04	15:34:19	08/06/2022
#03	02	04	16:08:20	08/06/2022
#04	02	04	16:09:41	08/06/2022
#05	02	04	16:15:52	08/06/2022
#06	02	04	19:06:19	08/06/2022
#07	03	04	16:17:02	08/06/2022
#08	03	04	16:44:02	08/06/2022
#09	04	04	15:40:47	08/06/2022
#10	09	04	16:16:24	08/06/2022

The leaderboard shows the top 10 scores. Each entry displays the time reached, the level played and information about the date/hour.

Project Status

Functionalities implemented:

- Main Menu
- Singleplayer
- Leaderboard
- Drag And Drop with mouse
- Keyboard typing
- Actual time and date display

Functionalities not implemented:

- Multiplayer

I/O devices used in this project:

Device	What for	Interrupt/pooling
Timer	Frame rate and game time	Interrupts
KBD	Words typing	Interrupts
Mouse	Buttons and drag and drop	Interrupts
Video Card	Game drawing	None
RTC	Leaderboards, date/time display and game countdown	Update-ended Interrupts

Timer:

We used the Timer 0 to obtain a fixed 60 FPS.

```
int(timer_subscribe_int)(uint8_t *bit_no);
int(timer_unsubscribe_int)();
void(timer_int_handler)();
```

For Application Dependent interrupt handling we use this function:

```
static void timer_handler();
```

KBD:

For interrupt handling we use the following functions, where the ih is Application Independent.

```
int(kbc_subscribe_int)(uint8_t *bit_no);
int(kbc_unsubscribe_int)();
void(kbc_ih)();
```

For Application Dependent handling we use this function:

```
static void kbd_handler();
```

For word typing we use this function to translate scan codes to a char.

```
char kbd_get_key();
```

Mouse:

For interrupt handling we use the following functions, where the ih is Application Independent.

```
int(mouse_subscribe_int)(uint8_t *bit_no);
int(mouse_unsubscribe_int)();
void(mouse_ih)();
```

For Application Dependent handling we use this function:

```
static void mouse_handler();
```

For drag and drop we use the following functions:

```
void resetDraggablePosition(Draggable* draggable);
void handleDrag(Draggable* draggable, uint16_t mouse_delta_x, uint16_t
mouse_delta_y);
bool checkBounds(Draggable* dragabble, uint16_t mouse_x, uint16_t
mouse_y);
void checkTarget(Draggable* draggable, DragTarget* targets);
```

Video Card:

- **Video Mode used:** 0x14C
- **Color Model:** Direct color
- **Resolution:** 1152x864 pixels
- **Color:** 32 bits for each color ((8:)8:8:8)

We use the **double buffering** technique:

```
void vg refresh();
void vg_clear_next_frame();
```

We utilize a special function to draw **font** xpms with different colors and scales.

```
int vg_draw_font_xpm(uint16_t x, uint16_t y, xpm_image_t xpm, uint8_t
scale, uint32_t color);
```

For efficiency we only draw the mouse when the rest of the screen doesn't change:

```
void vg_draw_mouse(uint16_t x, uint16_t y, xpm_image_t xpm);
void vg_restore_behind_mouse();
```

Other functions for drawing:

```
int (vg_draw_xpm)(uint16_t x, uint16_t y, xpm_image_t xpm);
int (vg_draw_scaled_xpm)(uint16_t x, uint16_t y, xpm_image_t xpm,
uint8_t scale);
void(vg_draw_background)(uint32_t color);
```

RTC:

For interrupt handling we use the following functions, where the ih is Application Independent.

```
int(rtc_subscribe_int)(uint8_t *bit_no);
int(rtc_unsubscribe_int)();
void rtc_ih();
void rtc_toggle_int(bool enable);
```

For Application Dependent handling we use this function:

```
void rtc handler();
```

To get the current date/hour we use the following function: (Fills a global variable)

```
void rtc_get_date();
```

Code Organization/Structure

data/

Contains the stored leaderboard.

libs/

Contains the device's modules.

- **graphics/ -** Video card (include **fonts/** and **xpms/**)
- **kbc/ -** Keyboard and Mouse
- **rtc/ -** RTC
- **timer/ -** Timer

project/

- **levels/ -** Contains all stored level files (levelX.txt corresponds to the main paragraph and answersX.txt corresponds to the answers for blank spaces in that level)

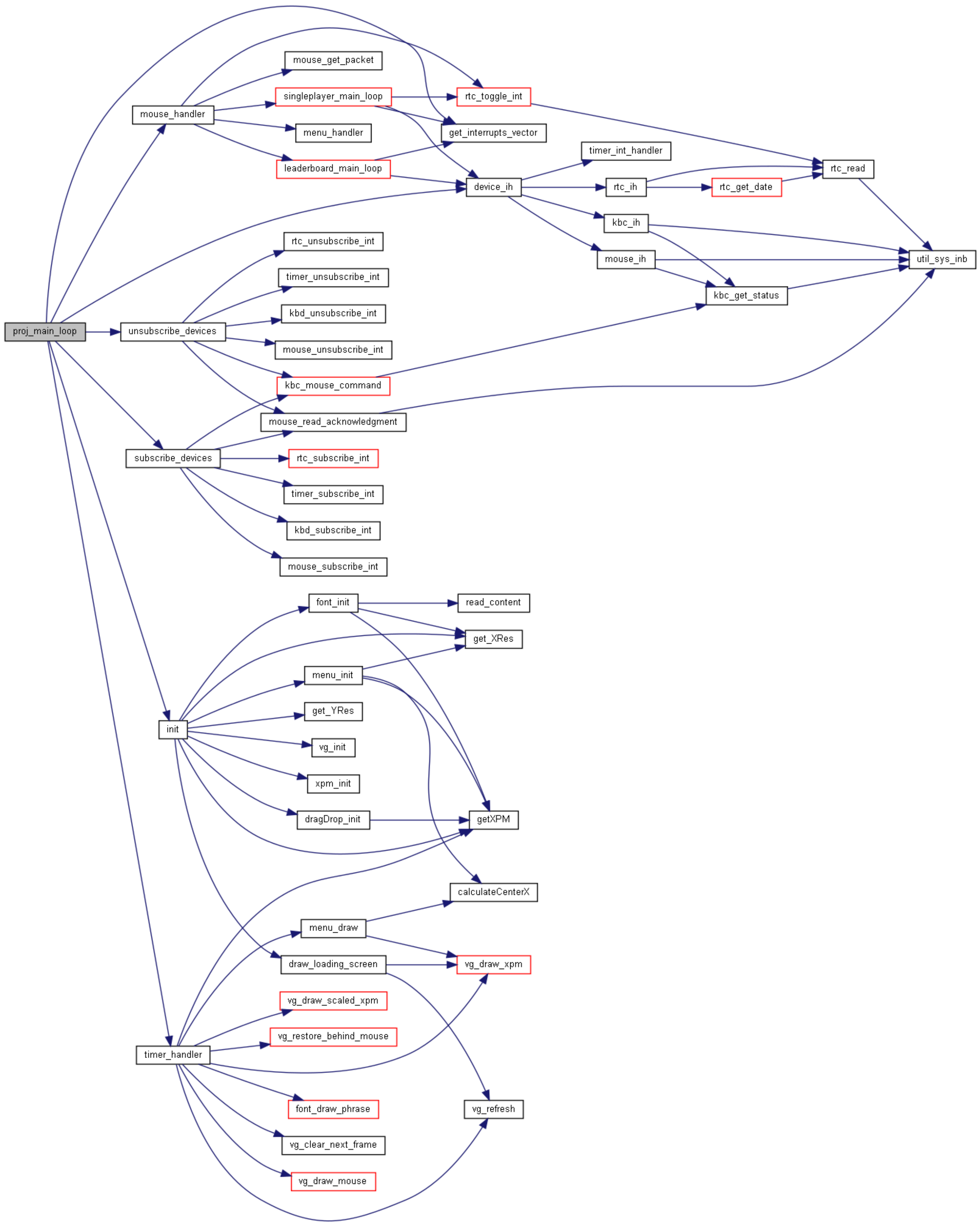
- **singleplayer -** Module for singleplayer mode
- **config -** Config file (graphics mode and game settings)
- **devices -** Module for handling all devices, subscriptions and interrupts
- **dragDrop -** Module to handle mouse and drop functionality.
- **endGame -** Module containing the end game screen.
- **leaderboard -** Module containing the leaderboard functionality.
- **level -** Module for level reading
- **menu -** Module handling the menu functionality.

Relative Weight (%) and Contributions

		Weight	Done by
libs	graphics	10%	Labs, Alexandre
	graphics/font	6%	Alexandre e José
	kbc/mouse	5%	Labs
	kbc/kbd	5%	Labs
	rtc	4%	Everyone
	timer	7%	Labs
project	singleplayer	25%	Alexandre, José e Tomás
	config	2%	Tomás
	devices	7%	Alexandre
	dragDrop	8%	Alexandre
	endGame	2%	Alexandre
	leaderboard	7%	Alexandre
	level	7%	José
	menu	5%	Alexandre e José

Functions calls graph

We generated various function calls with doxygen, most of the functions have smaller graphs, the graph below shows the root of the function calls.



Implementation Details

- The code developed follows an **Event Driven Design (EDD)** where the flow control is determined by the events (environment) rather than the program itself.
- To get the highest possible graphics video **efficiency**, whenever possible we just draw/redraw the mouse. We only redraw the entire screen when it is needed. We accomplish this by using a boolean variable.
- The **RTC** is used with **UIE** interrupts that correspond to every second.
- The **drag and drop** of the mouse is implemented by listening to the **left click** of the mouse. If we are dragging and we detect that the button is not pressed, we check if there is a target near.
- We separated the interrupt handlers: **Application Dependent** and **Application Independent**.
- To use a **font**, it is necessary to load **a lot of xpms**. To facilitate the process, the xpms of the letters are in **xpm2** format where we need to **parse the file**. This way we can load everything using a *for loop*.

Conclusions

We initially hoped to implement the multiplayer feature using the *serial port*, however we didn't have time for that. We have begun to implement this feature but with no success.

Main achievements of the project:

- Use of almost every device (except serial port)
- Good code structure, modulary and readability
- Documentation with Doxygen
- Menu Buttons
- Mouse drag and drop
- Keyboard typing
- Very good correct/incorrect input display (gameplay)
- Leaderboard
- Use of RTC interrupts and data
- Really efficient graphics
- Moving cars
- Simple BOT to play against