# Real Time Systems

## Project 1: Cycling scheduling

Alberto Huertas Ramírez

Ramón Hernández León

# INDEX

# Section A

## Wagon controller (Arduino)

### Functions implemented

*gas_request()*

Processes a request related to the accelerometer, then sends the response.

*brake_request()*

Processes a request related to the brake, then sends the response.

*mixer_request()*

Processes a request related to the mixer, then sends the response.

*slope_request()*

Processes a request for reading the slope, then sends the response.

*speed_request()*

Processes a request for reading the current speed, then sends the response.

*computeSpeed()*

Computes the current speed based on the slope, gas, brake and the previous speed.

*readSlope(int status)*

Reads the current value of the slope: -1 upwards, 0 flat, 1 downwards.

*switchGas(int status)*

Switch the gas on or off according to the specified status.

*switchBrake(int status)*

Switch the brake on or off according to the specified status.

*switchMixer(int status)*

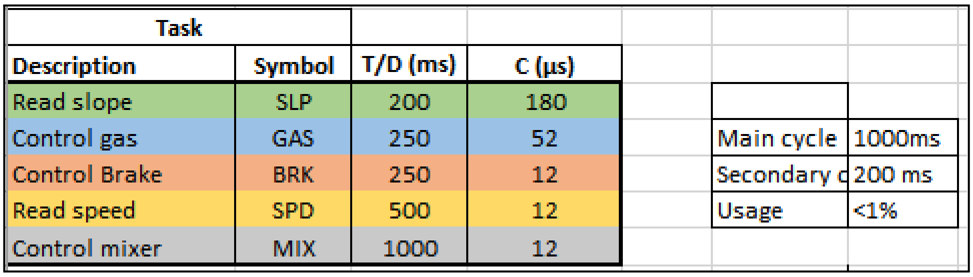
Switch the mixer on or off according to the specified status.

### Scheduling

For section A a total of 5 different tasks were identificated. The functions take very little time to execute (micro seconds) while the main controller will request information from seconds to second. For this reason, the Deadlines of each task are much bigger than their Computing time. The Deadlines are set based on the importance of the corresponding task. Thus the slope, gas and brake (fundamental to compute the speed) are the most frequent tasks. While switching the mixer is less frequent.

According to the information obtained about each task, the following Deadline has been set. Therefore, the main cycle will be 1000ms (mcm(Di). For the secondary cycles the duration chosen is 200ms. This is time enough for every function to be executed, and enough as well for the main controller to receive the answer (it takes 400ms to read data from the wire).

Therefore, the tasks are scheduled as follow:



| Task | | | |
|---|---|---|---|
| Description | Symbol | T/D (ms) | C (µs) |
| Read slope | SLP | 200 | 180 |
| Control gas | GAS | 250 | 52 |
| Control Brake | BRK | 250 | 12 |
| Read speed | SPD | 500 | 12 |
| Control mixer | MIX | 1000 | 12 |

| | |
|---|---|
| Main cycle | 1000ms |
| Secondary c | 200 ms |
| Usage | <1% |

Figure 1: tasks information, section A, wagon controller

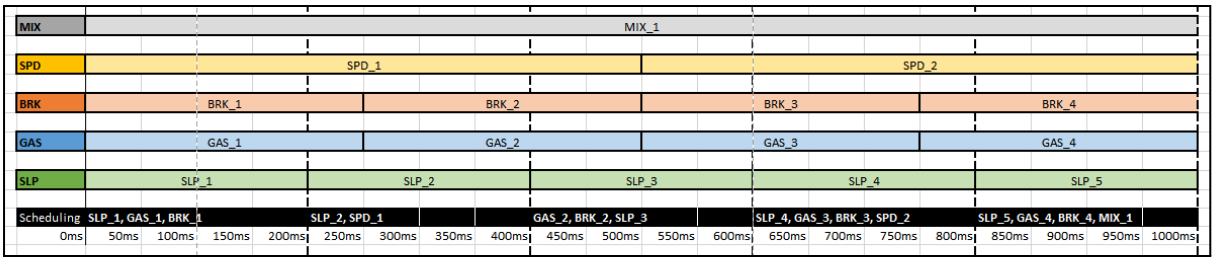


Figure 2: task scheduling, section A, wagon controller

# Main controller (Raspberry Pi)

## Functions implemented

*task_speed()*

Send a request asking for the speed, receive it back and display it.

*task_slope()*

Send a request asking for the slope, receive it back and display it.

*task_brake()*

Compute if the brake must be on/off, send the request depending on the computation result and display it.The computation compares if the speed is below or above the speed, and if it’s slowing or accelerating (going up or down).

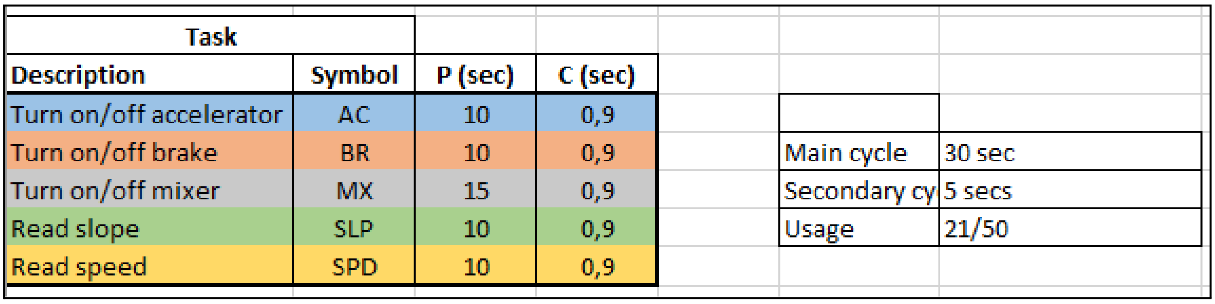
*task_accelerator()*

Compute if the accelerator must be on/off, send the request depending on the computation result and display it. The computation compares if the speed is below or above the speed, and if it’s slowing or accelerating (going up or down).

*task_mixer()*

Compare how much time the mixer has been off or on and if necessary to change it, send the request and display it.

## Scheduling



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off accelerator | AC | 10 | 0,9 |
| Turn on/off brake | BR | 10 | 0,9 |
| Turn on/off mixer | MX | 15 | 0,9 |
| Read slope | SLP | 10 | 0,9 |
| Read speed | SPD | 10 | 0,9 |

| | |
|---|---|
| Main cycle | 30 sec |
| Secondary cy | 5 secs |
| Usage | 21/50 |

*Figure 3: task information, section A, Raspberry Pi controller*

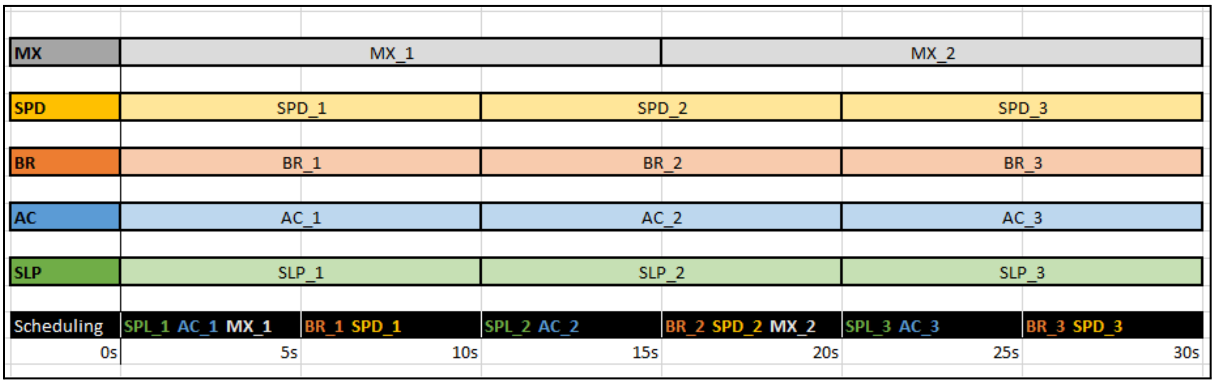


*Figure 4: task scheduling, section A, Raspberry Pi controller*

# Section B

## Wagon controller (Arduino)

### Functions implemented

*light_request()*

Proces a “LIT” request, reads the luminosity sensor, then sends the response.

*lamp_request()*

Process a “LAM” request. Switches the lamp sensor and finally sends the response back to the main controller.

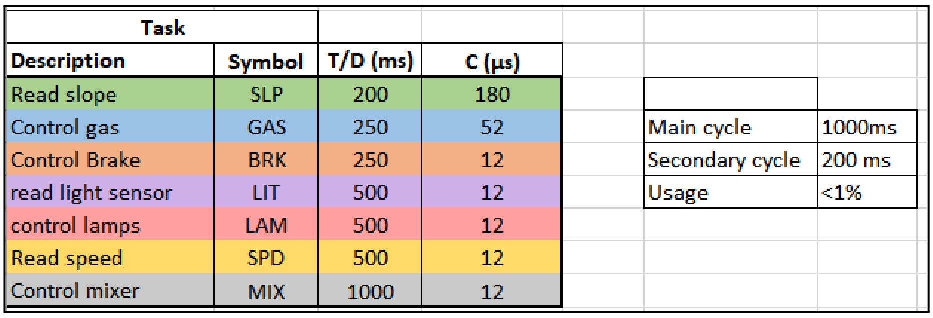
*switchLamp(int status)*

Switches the lamp on or off according to the status specified in the parameters

### Scheduling

For section B new tasks were added to the system, therefore the scheduling needs to be planned again. The new tasks are placed in between the previous tasks: not the biggest frequency, nor the smaller. The reason is that the lamps and sensors are not critical for speed control, that is our main goal in this mode.

The scheduling design is detailed in figures 5 and 6.



| Task | | | |
|---|---|---|---|
| Description | Symbol | T/D (ms) | C (µs) |
| Read slope | SLP | 200 | 180 |
| Control gas | GAS | 250 | 52 |
| Control Brake | BRK | 250 | 12 |
| read light sensor | LIT | 500 | 12 |
| control lamps | LAM | 500 | 12 |
| Read speed | SPD | 500 | 12 |
| Control mixer | MIX | 1000 | 12 |

| | |
|---|---|
| Main cycle | 1000ms |
| Secondary cycle | 200 ms |
| Usage | <1% |

Figure 5: tasks information, section B, wagon controller

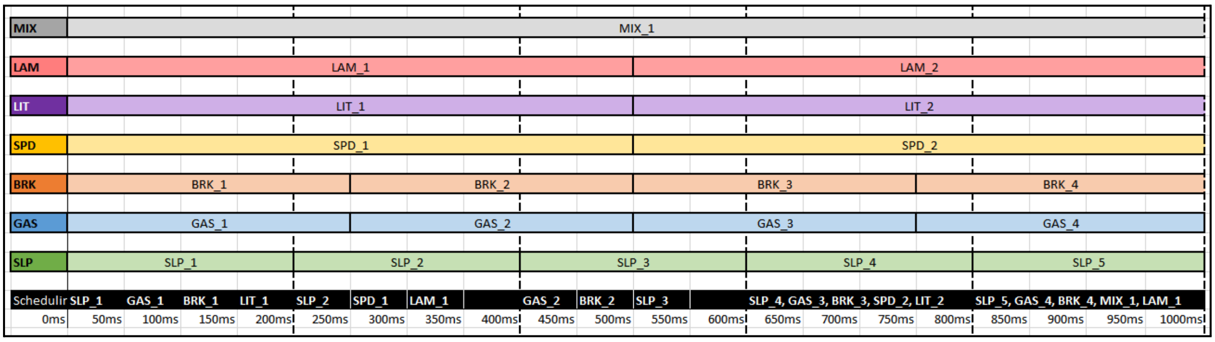


Figure 6: task scheduling, section B, wagon controller

# Main controller (Raspberry Pi)

## Functions implemented

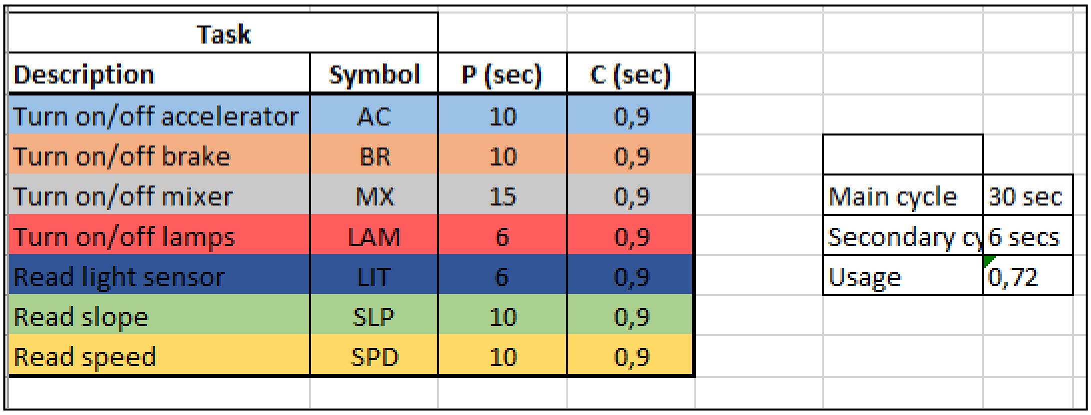
*task_light()*

Send a request asking for the light sensor, receive the response and display it.

*task_lamps()*

Turn off or on the lamps depending if we are in a dark zone (light below 50%) or not.

## Scheduling



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off accelerator | AC | 10 | 0,9 |
| Turn on/off brake | BR | 10 | 0,9 |
| Turn on/off mixer | MX | 15 | 0,9 |
| Turn on/off lamps | LAM | 6 | 0,9 |
| Read light sensor | LIT | 6 | 0,9 |
| Read slope | SLP | 10 | 0,9 |
| Read speed | SPD | 10 | 0,9 |

| | |
|---|---|
| Main cycle | 30 sec |
| Secondary cy | 6 secs |
| Usage | 0,72 |

Figure 7: task information, section B, Raspberry Pi controller

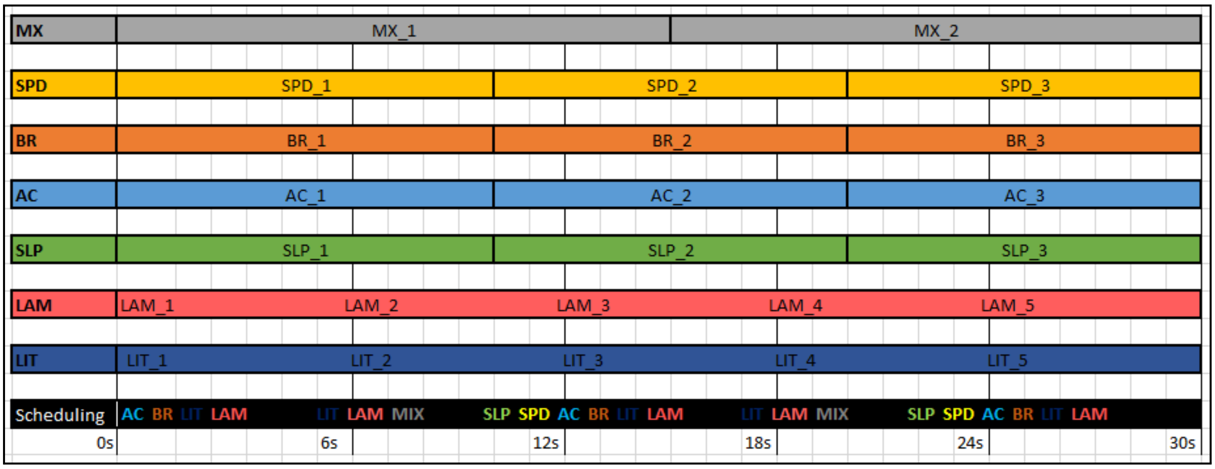


*Figure 8: task scheduling, section B, Raspberry Pi controller*

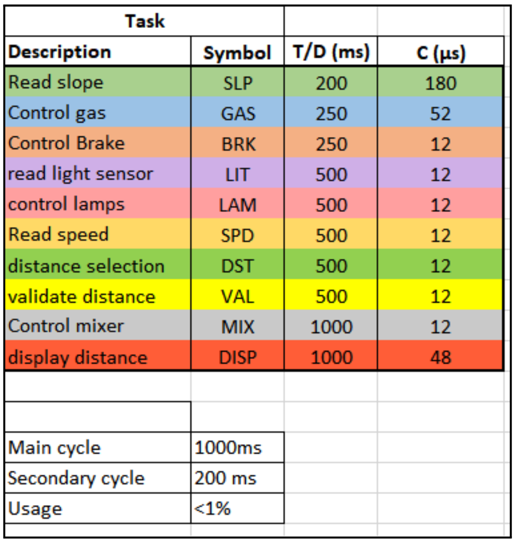
# Section C

## Wagon controller (Arduino)

### Functions implemented

### Scheduling

Several new tasks are added in this section. Additionally 3 operation modes are defined. Each operation mode has different requirements, so that the scheduling is adjusted to meet those requirements.



| Task | | | |
|---|---|---|---|
| Description | Symbol | T/D (ms) | C (µs) |
| Read slope | SLP | 200 | 180 |
| Control gas | GAS | 250 | 52 |
| Control Brake | BRK | 250 | 12 |
| read light sensor | LIT | 500 | 12 |
| control lamps | LAM | 500 | 12 |
| Read speed | SPD | 500 | 12 |
| distance selection | DST | 500 | 12 |
| validate distance | VAL | 500 | 12 |
| Control mixer | MIX | 1000 | 12 |
| display distance | DISP | 1000 | 48 |
| | | | |
| Main cycle | 1000ms | | |
| Secondary cycle | 200 ms | | |
| Usage | <1% | | |

*Figure 9: tasks information, section C, distance mode, wagon controller*

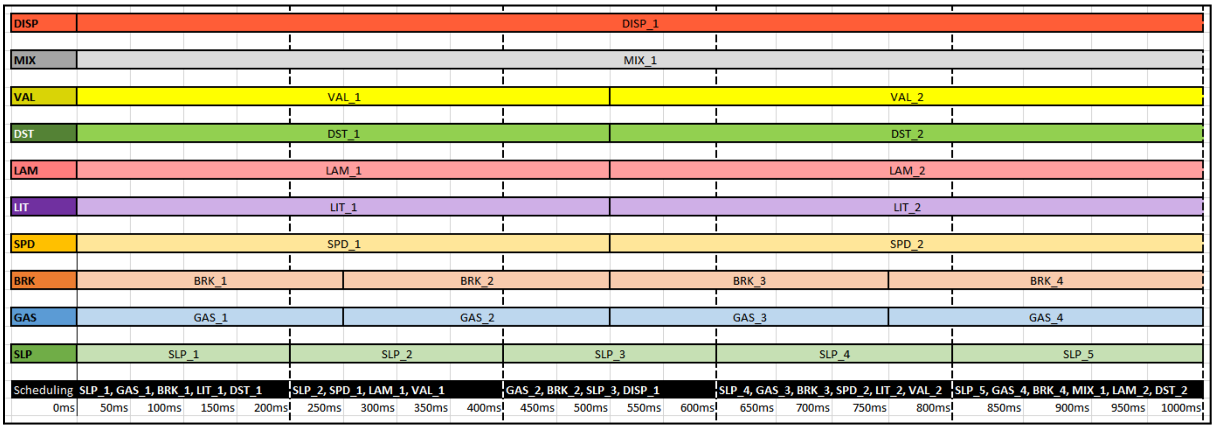
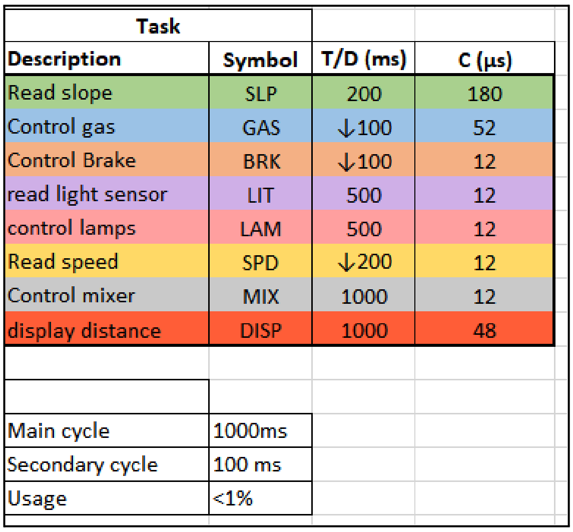


*Figure 10: tasks scheduling, section C, distance mode, wagon controller*



| Task | | | |
|---|---|---|---|
| **Description** | **Symbol** | **T/D (ms)** | **C (μs)** |
| Read slope | SLP | 200 | 180 |
| Control gas | GAS | ↓100 | 52 |
| Control Brake | BRK | ↓100 | 12 |
| read light sensor | LIT | 500 | 12 |
| control lamps | LAM | 500 | 12 |
| Read speed | SPD | ↓200 | 12 |
| Control mixer | MIX | 1000 | 12 |
| display distance | DISP | 1000 | 48 |
| | | | |
| | | | |
| Main cycle | 1000ms | | |
| Secondary cycle | 100 ms | | |
| Usage | <1% | | |

*Figure 11: tasks information, section C, approach  mode, wagon controller*

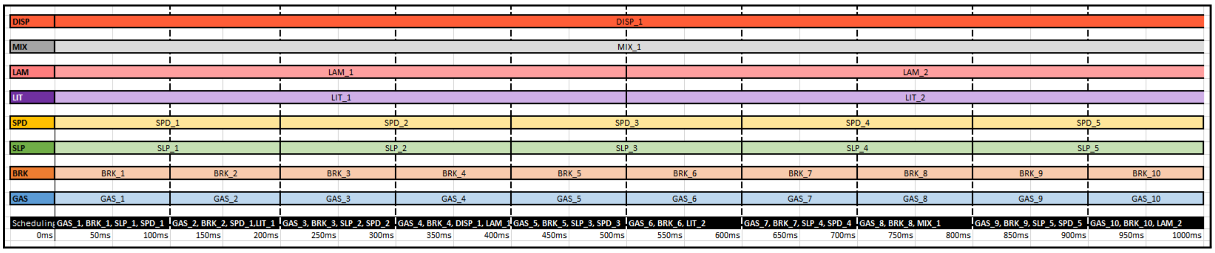
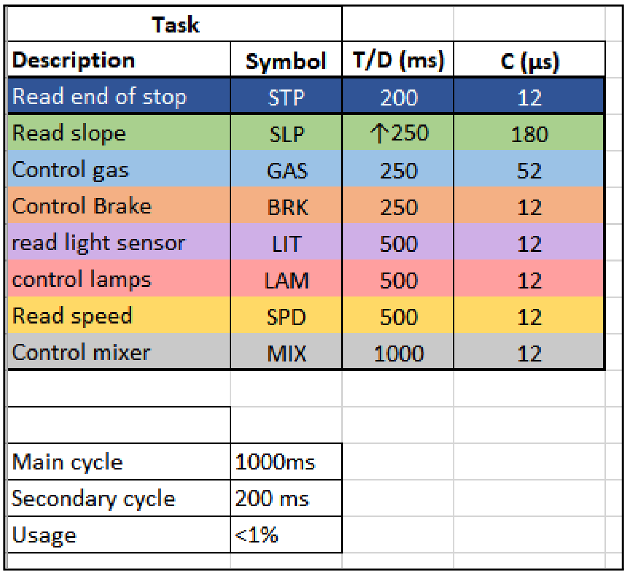


*Figure 12: tasks scheduling, section C, approach  mode, wagon controller*

For the approach mode, the tasks related to the accelerometer, the brake and the speed computation have their frequencies doubled. To meet the new Deadlines, a shorter Secondary Cycle is defined: 100ms. The frequency of the slope task is reduced as well in order to give more space for the speed control tasks.



| Task | | | |
|---|---|---|---|
| Description | Symbol | T/D (ms) | C (µs) |
| Read end of stop | STP | 200 | 12 |
| Read slope | SLP | ↑250 | 180 |
| Control gas | GAS | 250 | 52 |
| Control Brake | BRK | 250 | 12 |
| read light sensor | LIT | 500 | 12 |
| control lamps | LAM | 500 | 12 |
| Read speed | SPD | 500 | 12 |
| Control mixer | MIX | 1000 | 12 |

| | |
|---|---|
| Main cycle | 1000ms |
| Secondary cycle | 200 ms |
| Usage | <1% |

*Figure 13: tasks information, section C, stop mode, wagon controller*

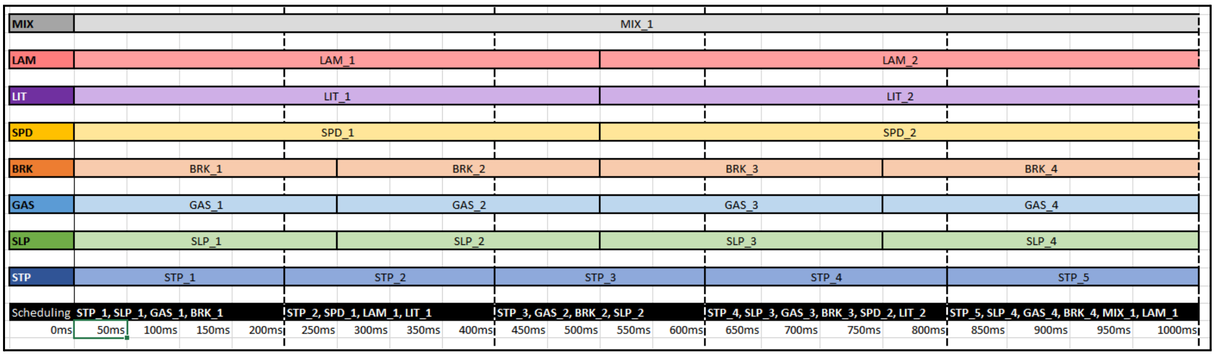


*Figure 14: tasks scheduling, section C, stop mode, wagon controller*

# Main controller (Raspberry Pi)

## Functions implemented

*task_distance()*

Send a request asking for the distance to the next download location, compute if we are still far enough or we need to start braking (change to braking mode), or if we are already on the download location and we have to stop (change to stop mode)

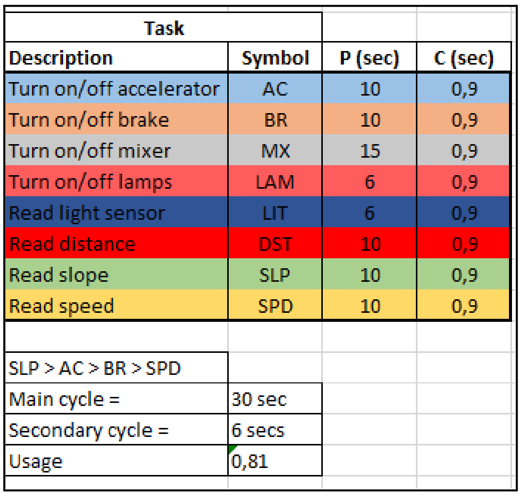
task_load()

Send a request to check if the download is complete and check if the execution mode can be changed to normal mode again.

Also some of the previously implemented functions are changed as the functionalities change depending on the execution mode we are in.
task_accelerator() and task_break() have different speed target depending on the execution mode
task_lamps() change depending on execution mode too, as in braking and stop mode, lamps are on all the time.

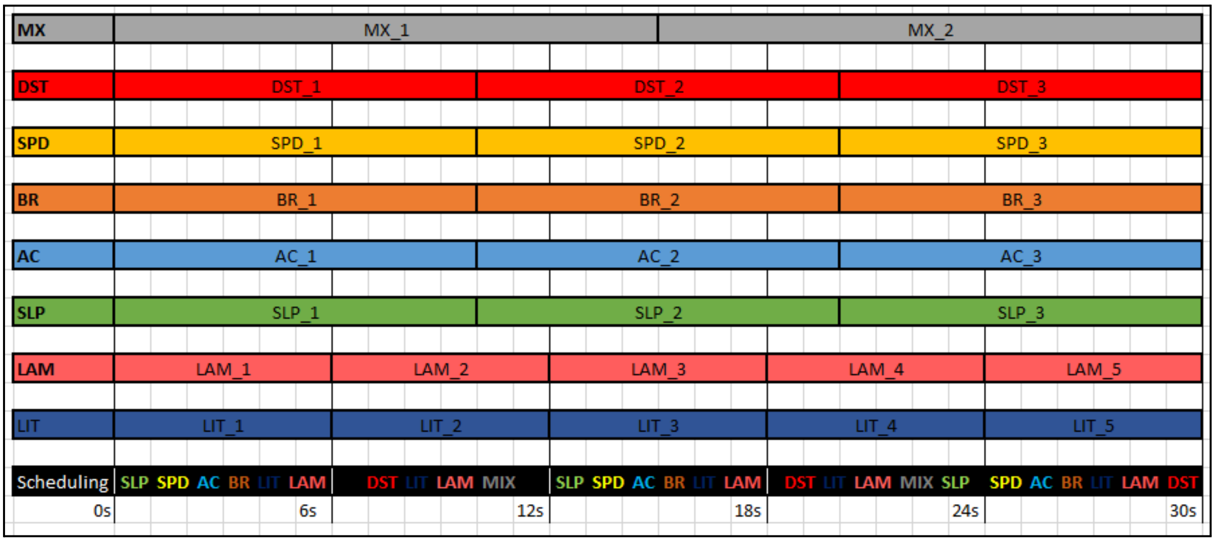
## Scheduling

### Normal mode



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off accelerator | AC | 10 | 0,9 |
| Turn on/off brake | BR | 10 | 0,9 |
| Turn on/off mixer | MX | 15 | 0,9 |
| Turn on/off lamps | LAM | 6 | 0,9 |
| Read light sensor | LIT | 6 | 0,9 |
| Read distance | DST | 10 | 0,9 |
| Read slope | SLP | 10 | 0,9 |
| Read speed | SPD | 10 | 0,9 |
| | | | |
| SLP > AC > BR > SPD | | | |
| Main cycle = | 30 sec | | |
| Secondary cycle = | 6 secs | | |
| Usage | 0,81 | | |

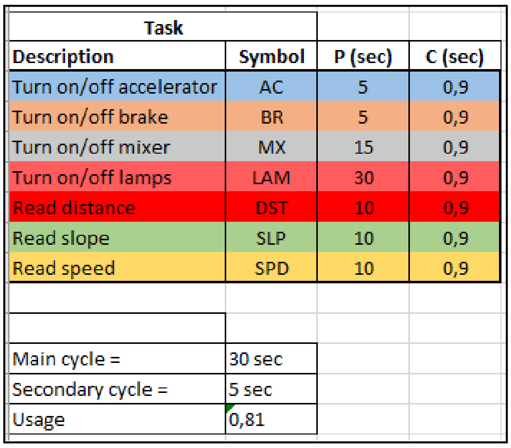
*Figure 15: task information, section C, normal mode Raspberry Pi controller*



<table>
<tr><td>MX</td><td colspan="4">MX_1</td><td colspan="4">MX_2</td></tr>
<tr><td>DST</td><td colspan="2">DST_1</td><td colspan="4">DST_2</td><td colspan="2">DST_3</td></tr>
<tr><td>SPD</td><td colspan="2">SPD_1</td><td colspan="4">SPD_2</td><td colspan="2">SPD_3</td></tr>
<tr><td>BR</td><td colspan="2">BR_1</td><td colspan="4">BR_2</td><td colspan="2">BR_3</td></tr>
<tr><td>AC</td><td colspan="2">AC_1</td><td colspan="4">AC_2</td><td colspan="2">AC_3</td></tr>
<tr><td>SLP</td><td colspan="2">SLP_1</td><td colspan="4">SLP_2</td><td colspan="2">SLP_3</td></tr>
<tr><td>LAM</td><td>LAM_1</td><td colspan="2">LAM_2</td><td colspan="2">LAM_3</td><td colspan="2">LAM_4</td><td>LAM_5</td></tr>
<tr><td>LIT</td><td>LIT_1</td><td colspan="2">LIT_2</td><td colspan="2">LIT_3</td><td colspan="2">LIT_4</td><td>LIT_5</td></tr>
<tr><td>Scheduling</td><td>SLP SPD AC BR LIT LAM</td><td colspan="2">DST LIT LAM MIX</td><td colspan="2">SLP SPD AC BR LIT LAM</td><td colspan="2">DST LIT LAM MIX SLP</td><td>SPD AC BR LIT LAM DST</td></tr>
<tr><td>0s</td><td>6s</td><td colspan="2">12s</td><td colspan="2">18s</td><td colspan="2">24s</td><td>30s</td></tr>
</table>

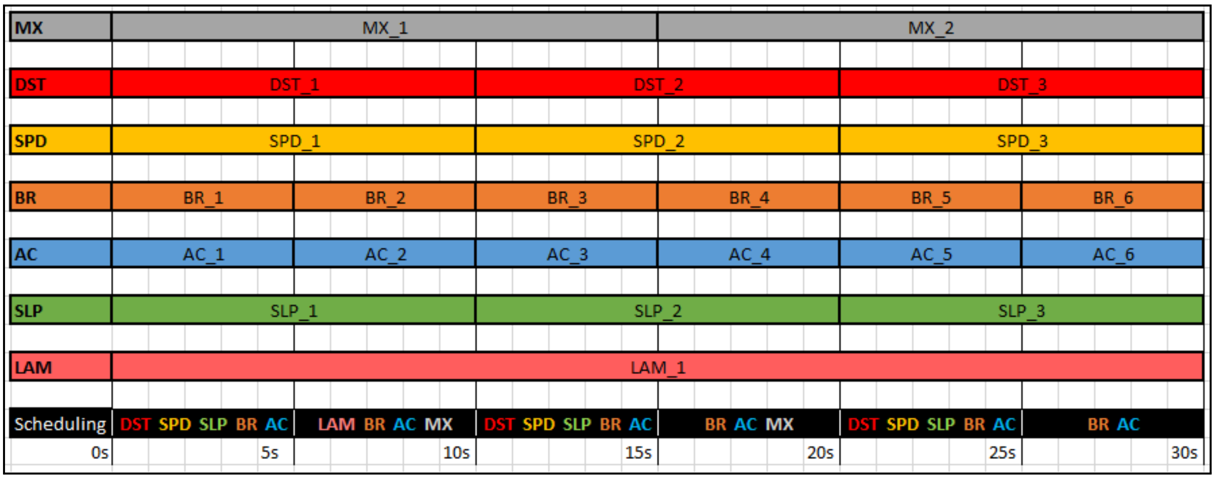
*Figure 16: task scheduling, section C, normal mode, Raspberry Pi controller*

### Braking mode



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off accelerator | AC | 5 | 0,9 |
| Turn on/off brake | BR | 5 | 0,9 |
| Turn on/off mixer | MX | 15 | 0,9 |
| Turn on/off lamps | LAM | 30 | 0,9 |
| Read distance | DST | 10 | 0,9 |
| Read slope | SLP | 10 | 0,9 |
| Read speed | SPD | 10 | 0,9 |
| | | | |
| | | | |
| Main cycle = | 30 sec | | |
| Secondary cycle = | 5 sec | | |
| Usage | 0,81 | | |

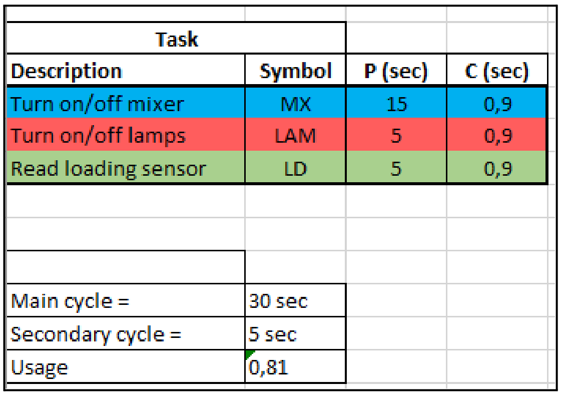
*Figure 17: task information, section C, braking mode Raspberry Pi controller*



| MX | MX_1 | | | MX_2 | | |
|---|---|---|---|---|---|---|
| DST | DST_1 | | DST_2 | | DST_3 | |
| SPD | SPD_1 | | SPD_2 | | SPD_3 | |
| BR | BR_1 | BR_2 | BR_3 | BR_4 | BR_5 | BR_6 |
| AC | AC_1 | AC_2 | AC_3 | AC_4 | AC_5 | AC_6 |
| SLP | SLP_1 | | SLP_2 | | SLP_3 | |
| LAM | LAM_1 | | | | | |
| Scheduling | DST SPD SLP BR AC | LAM BR AC MX | DST SPD SLP BR AC | BR AC MX | DST SPD SLP BR AC | BR AC |
| 0s | 5s | 10s | 15s | 20s | 25s | 30s |

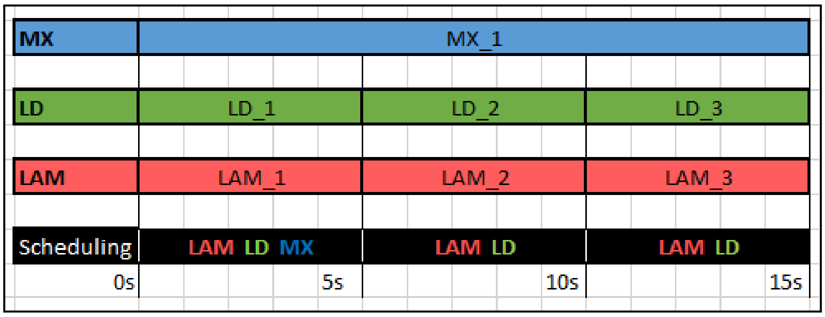
Figure 18: task scheduling, section C,  braking mode, Raspberry Pi controller

## Stop mode



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off mixer | MX | 15 | 0,9 |
| Turn on/off lamps | LAM | 5 | 0,9 |
| Read loading sensor | LD | 5 | 0,9 |
| | | | |
| Main cycle = | 30 sec | | |
| Secondary cycle = | 5 sec | | |
| Usage | 0,81 | | |

Figure 19: task information, section C, stop mode Raspberry Pi controller



| MX | MX_1 | | |
|---|---|---|---|
| LD | LD_1 | LD_2 | LD_3 |
| LAM | LAM_1 | LAM_2 | LAM_3 |
| Scheduling | LAM LD MX | LAM LD | LAM LD |
| 0s | 5s | 10s | 15s |

Figure 20: task scheduling, section C,  stop mode, Raspberry Pi controller

# Section D

## Wagon controller (Arduino)

## Scheduling

# Main controller (Raspberry Pi)

## Functions implemented

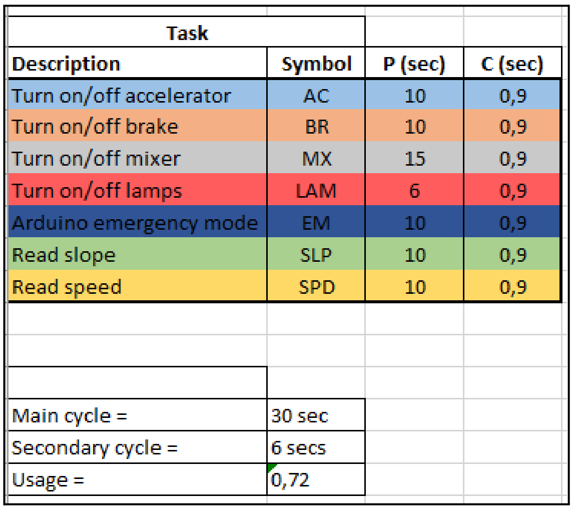
*task_arduino()*

Send a request to the arduino to change to emergency mode.

## Scheduling

Normal, braking and stop mode are the same as in section C

<u>Emergency mode</u>



| Task | | | |
|---|---|---|---|
| Description | Symbol | P (sec) | C (sec) |
| Turn on/off accelerator | AC | 10 | 0,9 |
| Turn on/off brake | BR | 10 | 0,9 |
| Turn on/off mixer | MX | 15 | 0,9 |
| Turn on/off lamps | LAM | 6 | 0,9 |
| Arduino emergency mode | EM | 10 | 0,9 |
| Read slope | SLP | 10 | 0,9 |
| Read speed | SPD | 10 | 0,9 |
| | | | |
| | | | |
| Main cycle = | 30 sec | | |
| Secondary cycle = | 6 secs | | |
| Usage = | 0,72 | | |

*Figure 21: task information, section D, emergency mode Raspberry Pi controller*

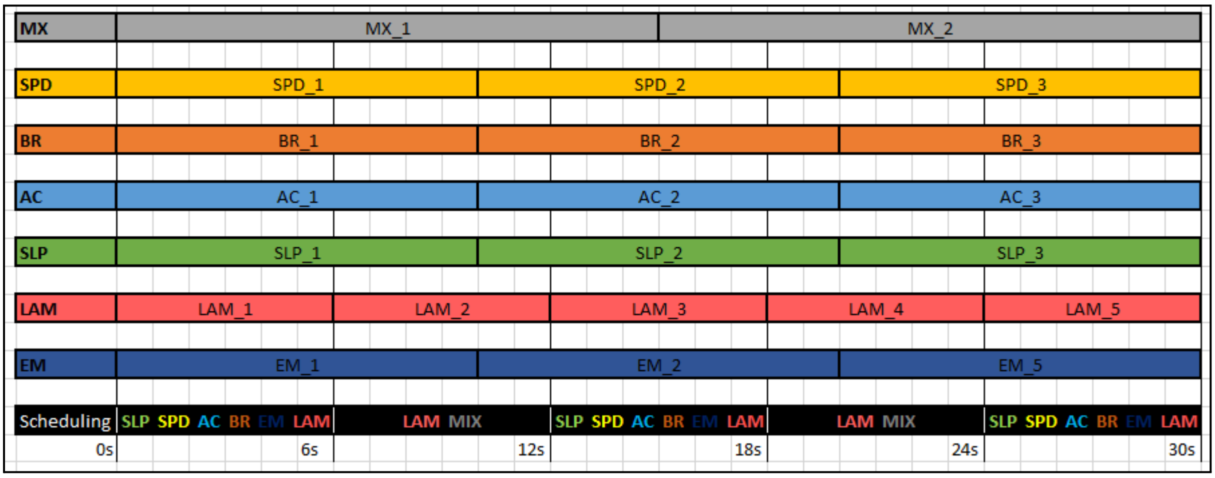


*Figure 22: task scheduling, section D,  emergency mode, Raspberry Pi controller*