1. **Software**

**Functional summary**

When the autonomous buggy is switched on it will keep still until it detects a white line, then it will immediately turn on the motor and start rotating its wheels. Once it had started, it will be capable of autonomously follow the track without losing its stability and direction even when it goes through a ramp up to 18 degrees of inclination. If there is an end in the track, not a small gap, the buggy will instantly stop, if not, it will keep going [?].

However, if a rotation of 180 degrees of the buggy is desired, then there will be a system to perform this task. A Bluetooth Low Energy (BLE) will intervene to make the buggy turn around and return all the way back, of course, while a white line is present [?].

**List of possible software constraints**

* Lack of experience:

From the beginning of this project each member has been showing their strengths and weaknesses. Everyone has difficulties for programming, lack of experience using C++ and/or using mbed/low-level libraries, which could probably end with a buggy performing in a different way to the desired.

* Memory capacity:

The device used to execute the program will be the STM32F401RE which has a Flash memory of up to 512 Kbytes and a SRAM of up to 96 Kbytes. The code should not exceed those limits [?].

* Time response:

Another characteristic of the device mentioned before is its frequency. It can work with a frequency of up to 84Mhz according with its datasheet [?].

* Program not loaded correctly:

Some problems may be found when the program is tried to load to the STM32 board, this can be, it is no longer available to download the software to the board or the microcontroller not being found.

**Context diagram**

* SPT: Sensors for Perceiving the Track.
* SC: Sensors for Curving
* PRL: Position with respect to the line.
* PWM: Pulse wave modulation.
* M (R, L): Right Motor, Left Motor.
* BT: Battery Monitor

**SC**

**SC**

**SPT**

**SC**

**SC**



**PRL**

**PRL**

**Encoder R**

**MR**

**ML**

**Encoder L**

**SPT**

**BLE**

**PRL**

**PRL**

**PRL**

**PRL**

**Command**

**PWM**

**PWM**

Autonomous Buggy

**BM**



**Remaining Power**

**Angular Speed**

**Angular Speed**

**Figure 1.1** Interaction between the external objects and the system.

**Table of messages**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Source | Destination | Interface | Size in bits | Arrival pattern | Comments |
| Position with respect to the line | Detects the changes between black and white | SPTs & SCs | System | SPI | TBD | TBD | Sensors still to be selected |
| Command | Receive the command to turn around | BLE | System | SPI | TBD | TBD | None |
| Remaining Power | Estimate remaining run time | MB | System | SPI | TBD | TBD | None |
| Angular Speed | Speed of the rotation about vertical axis | Encoders | System | I2C | TBD | TBD | Sensor still to be selected |
| Pulse wave modulation | Pulse generated for rotating the motor | System | M (L, R) | SPI | TBD | TBD | None |

**Table 1.1** Capture information about messages.

**Case Diagram**

**SPT**

**SPT**

**Encoder R**

**Encoder L**

**SC**

**SC**

**SC**

**SPT**

**SPT**

<<include>>

<<include>>

**SC**

**BM**

**BLE**

**ML**

**MR**

**Figure 1.2** Message exchanges between actors and system.

**Case Descriptions**

**INITIALISATION**

**Main Flow of events**

Wait for **detection of white line** from **SPTs**

MOVE selected use case.

**Alternative flow**

IF **detection of white line** is not recived from **SPTs**

DO nothing.

**MOVE**

**Main Flow of events**

Wait for **position with respect to the line** from **SCs**

Wait for **speed of rotation** from **Encoders.**

Wait for **level of power** from **BM**

IF **command** is received from **BLE**

Start sending same **PWM** to **ML** and **MR**

**Alternative flow 1**

IF **position with respect to the line** is received from **SCs**

Increment and/or decrease the **PWM** of **ML** or **MR** depending of the **position with** **respect to the line**

**Alternative flow 2**

IF **command** is received from **BLE**

Deactivate **SCs**

Increment and/or decrease the **PWM** of **ML** or **MR** for turning around

INITIALISATION selected use case

**Alternative flow 3**

IF **level of power** is received from **BM**

Deactivate **SCs** and **SPTs**

**Object specifications**

WHILE NOT finished

Wait for detection of white line

While (detection of white line) {

Input:

Position with respect to the line from SPTs

Position with respect to the line from SCs

Angular speed from Encoders

Command from BLE

Remaining power from BM

Execute control algorithm

Output:

PWM to MR

PWM to ML

}

ELSE

Finished=TRUE