Radboud University 

Design of Embedded System

Mars Rover Project Report

Luca Poli[1124244], Razvan Potcoveanu[1123076]

Version: 1.2.0

Table of Contents

1. Planning....3
1.1 Requirements....3
1.1.1 Definition....3
1.1.2 Timetable....4
1.1.3 Progress Timetable....6
1.2 Sensors/Actuators Mapping....7

1. Planning

1.1 Requirements

1.1.1 Definition

Group	Code	Priority	Description
Functionality	F_SFT	M	The rover shall perform any mission safely (without falling of the edge or getting stuck in the lakes)
	F_MSQ	M	The rover shall perform the missions in sequential order (from the first)
	F_COL	M	The rover shall be able to detect, find or avoid colors
	F_DIS	M	The rover shall be able to detect distant objects
	F_IMP	M	The rover shall be able to detect impact with other objects
	F_ARM	M	The rover shall be able to use the measurement arm
	F_FED	M	The rover shall give feedback (led or sound) on mission start, complete (mission was finished or it timed out) or errors
	F_POB	S	The rover shall be able to push another object
	F_PRK	C	The rover shall be able to park
	F_DNS	W	The rover shall be able to dance and sing
Usability	U_NHM	M	The rover shall perform any mission without human interaction
	U_ELG	M	The user shall be able to see the error log
	U_ALG	S	The rover shall keep an activity log
	U_TLG	C	The rover shall keep track of time required for a mission
Reliability	R_ESR	M	The rover shall stop and restart after a critical error
	R_BRC	M	After a Bluetooth connection lost, the rover shall try to reconnect
	R_SER	M	The rover shall be able to work normally even with some sensors read errors
Performance	P_CTM	M	The rover shall be able to perform the missions in the time specified by the user, otherwise it shall start the next mission
	P_BCM	M	The two bricks shall respond to each other's instructions within 1s
	P_TPR	M	The rover shall perform the defined actions when triggered by external factors based on their priority
	P_IEL	S	The rover shall log the errors immediately
	P_EFM	W	The rover shall perform the missions in an efficient way
Supportability	S_COL	M	The user shall be able to configure the colors to detect (and how to react to them)
	S_MSP	S	The user shall be able to configure the speed of the motor
	S_ARM	S	The user shall be able to configure on which objects the rover has to use the measurement arms

	S_MOV	C	The user shall be able to configure the movement actions parameters (angle to turn, distance, …)
	S_FED	C	The user shall be able to configure the missions feedback
	S_POB	C	The user shall be able to configure which objects the rover has to move
	S_FSN	W	The user shall be able to add and use custom sounds for feedback
DSL	D_MIS	M	The user shall be able to define missions using the DSL
	D_VAL	M	The user shall be able to give their own values to specific parameters
	D_BLU	M	The user shall be able to specify the MAC address for the server
	D_NEW	M	The user shall be able to define their own missions, to use every rover sensor and actuators as they want (as long as the instance is valid)
	D_ERR	M	The DSL shall check that the instance provided by the user is correct
	D_GEN	M	A valid DSL instance shall generate 2 correct Python files, one for the server and one for the client
	D_FED	M	The DSL shall give feedback regarding the correctness of the provided instance
	D_ESY	M	The DSL shall be easy to use and understand for the user

For example, the users shall be able to define the following missions with a DSL instance:

1. Exploring – The rover will move around the table and perform user-defined actions.
2. Obstacle avoidance – The rover will navigate on a board with obstacles while avoiding collisions.
3. Sample Collection – The rover will navigate and collect samples from the environment.

1.1.2 Timetable

Week number	Activities
0: 22/11 – 27/11	Requirements definition and planning; Sensor/actuators mapping; DSL TypePal refactor; Refine of DSL grammar
1: 29/11 – 05/12	Basic must have functionalities [F_SFT, F_COL, F_DIS, F_IMP, F_ARM, U_NHM, U_ELG, R_SER, S_COL]. DSL functionalities [D_MIS, D_VAL, D_BLU, D_NEW, D_ERR, D_GEN, D_FED, D_ESY]
2: 06/12 – 12/12	Test and refine of implementation of F_SFT; Finish implementation of previous functionalities; Implementation of [F_MSQ, F_POB, F_FED, R_ESR, R_BRC, P_CTM, P_BCM, U_ALG]
3: 13/12 – 19/12	Finish implementation of previous functionalities; Implementation of [P_TPR, P_IEL, S_MSP, S_ARM, S_MOV, F_PARK]
4: 20/12 – 26/12	Finish implementation of previous functionalities; Implementation of [P_IEL, S_MOV, S_FED, S_POB, U_TLG]

5: 27/12 – 02/01	Finish implementation of previous functionalities; Implementation of [S_FSN, P_EFM, F_DNS]
6: 03/01 – 09/01	Testing and refining

1.1.3 Progress Timetable

Week number	Activities
0: 22/11 – 27/11	Requirements definition and planning; Sensor/actuators mapping; TypePal learning; Refine of DSL grammar
1: 29/11 – 05/12	TypePal refactor for actions and triggers; F_SFT; Grammar update for [F_SFT, F_COL, F_DIS, F_IMP, U_NHM, U_ELG].
2: 06/12 – 12/12	Test and refine of implementation of F_SFT; Finish implementation of [F_SFT, F_COL, F_DIS, F_IMP, U_NHM, U_ELG].; Implementation of DSL generator prototype.
3: 13/12 – 19/12	Finished implementation of [F_MSQ, F_ARM, F_FED, R_SER, P_TPR, P_IEL]; Finish implementation of Typepal Typechecker; Starting implementation of final generator.
4: 20/12 – 26/12	
5: 27/12 – 02/01	
6: 03/01 – 09/01	

1.2 Sensors/Actuators Mapping

Port	EV3 brick (server / left)	Port	EV3 brick (client / right)
S1	CL	S1	TL
S2	CM	S2	TR
S3	CR	S3	TB
S4	UB	S4	UF
A	ML	A	
B	MR	B	
C	M	C	