ARCHITENTERPRISES

Team 20

Lab number 2

Debugging Specification

February 24th, 2013

Version 4.0 (Final)

Version 1.0 (initial shell) updated on February 4th

Version 2.0 (rough draft) updated on February 11th

Version 3.0 (initial editing) updated on February 23rd

By signing below, each group member approves of this document and contributed fairly to its completion.

Raymond Tang, Andrew McMillion, Archit Rupakhetee, Tyler Lenig

On our honors, as students of the University of Virginia, we have neither given nor received unauthorized aid on this assignment.

Raymond Tang, Andrew McMillion, Archit Rupakhetee, Tyler Lenig

Contents

Preamble

The following document is our specification for the NXT robot's onboard debugging system. The system is intended to be used exclusively by the engineering team when there is a problem with the onboard software. It is part of the actual control interface that can be brought up when there is an error or system failure. This system is invoked when the onboard system fails, and it is manipulated by only the engineers, not the operator.

The debugging system allows for the engineering team to have in depth access to the robots variables and state. The main function of this system is to get the robot back up and running, so it is purposefully intended to give the engineers full control. This requires software components to be on the robot itself and in the base station. The base station holds the interface for interacting with the debugging software on the robot. These components in conjunction are used to return the robot to its operational state and allow the operator to continue to use the robot. The debugging systems largest restriction is if communication with the robot is cut off then there is not much the system can do until connection is established.

This document contains a list of modes, conditions, and other variables that the debugger can perform or check with respect to the robot. Furthermore, it also

outlines how various errors can occur and how they are displayed to the engineering team. The goal is to provide a clear and concise definitions of all elements contained in the document to aid in outlining the capabilities of the debugger.

Assumptions

Our mode table consists of three modes without any data initialization. This is because we are the debug system and in normal mode, our software simply monitors the system. Our event table and mode transition table displays and creates output after which brings the system back to the normal mode.

Since our specification is exclusively for the debugging side, we assume that there are no input data items for our side of the GUI, all input comes from the base control software. The input data items that we have included in this specification are utilized by our software, but the actual implementation and manipulation of the input data items is exclusively controlled by the main control software.

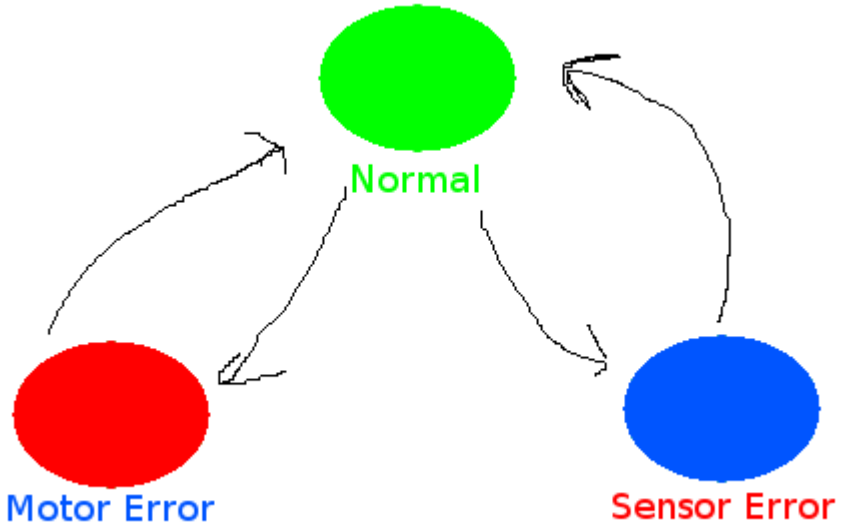
Modes

Mode	Definition
normal	Normal robot operation (i.e. no errors detected)
motor_error	Some error with a motor has been determined, this mode is where it is debugged. (i.e. motor is running too fast in the forward or backward direction)
sensor_error	Some sensor error has been determined, this mode is where it is debugged. (i.e. sensor is not transmitting data)

Mode Transition Table

Mode	*normal*	*motor_error*	*sensor_error*
normal	Robot is operating as expected (no motor or sensor error, i.e. @T(%no_error%))	any event that causes an error with a motor (i.e. @T(%motor_1_overclock_forward%))	any event that causes an error with a sensor (i.e. @F(%sensor_mic_running%))
motor_error	Action that fixes a motor error (i.e. !motor_1_speed! := $max_forward_speed$ //ERROR// shows $motor_1_error$)	-	-
sensor_error	Action that fixes a sensor error (i.e. //ERROR// shows $sensor_mic_error$)	-	-

Mode Transition Diagram



Our mode transition diagram shows the various modes our system can be in and how you get to and from each one. Our main mode is called *normal* and it is the mode in which the robot is operating under conditions that are making an errors. From this mode, our system goes into either *motor error* or *sensor error* depending on the type of error detected (all problems with motors go force the system into *motor error* and all problems with sensors force the system into *sensor error*). After the error is rectified, the system returns to normal mode, waiting on another error to surface.

User Interface

This is the basic design for the interface of the debugger. The debugger is integrated into the actual interface and the user types in a command and an access code that opens up the debugger. The console demonstrates this as the user is running program and an error is encountered. The command for the debugger is then typed, followed by the access code which starts the debugger. The debugger opens up on a separate window in the same interface with its own console. The debugger has direct access to sensor and motor values and it can further test and check values through the console.

Input Data Items

Input Data	Definition
/user_motor_1_speed/	The desired speed of motor 1 inputed by the user.
/user_motor_2_speed/	The desired speed of motor 2 inputed by the user.
/user_motor_3_speed/	The desired speed of motor 3 inputed by the user.

Input data item: motor 1 speed
Acronym: /user_motor_1_speed/
Class: Integer
Data representation: I/O port 1
Description: This field governs the speed of motor 1, it must be greater than or equal to 0 and less than or equal to $max_forward_speed$ or greater than or equal to $max_backward_speed$ and less than or equal to 0.
Value encoding: user defined integer value.
Timing characteristics: updated once user strikes enter key

Input data item: motor 2 speed
Acronym: /user_motor_2_speed/
Class: Integer
Data representation: I/O port 2
Description: This field governs the speed of motor 2, it must be greater than or equal to 0 and less than or equal to $max_forward_speed$ or greater than or equal to $max_backward_speed$ and less than or equal to 0.
Value encoding: user defined integer value.
Timing characteristics: updated once user strikes enter key

Input data item: motor 3 speed
Acronym: /user_motor_3_speed/
Class: Integer
Data representation: I/O port 3
Description: This field governs the speed of motor 3, it must be greater than or

equal to 0 and less than or equal to $max_forward_speed$ or greater than or equal to $max_backward_speed$ and less than or equal to 0.
Value encoding: user defined integer value.
Timing characteristics: updated once user strikes enter key

Output Data Items

Output Data	Definition
//motor_1_speed//	The angular velocity (degrees per second) of motor 1 (left motor).
//motor_2_speed//	The angular velocity (degrees per second) of motor 2 (right motor).
//motor_3_speed//	The angular velocity (degrees per second) of motor 3 (rear motor).
//ERROR//	Displays error output. This is an actual console displaying these errors.

Output data item: motor 1 speed
Acronym: //motor_1_speed//
Class: Integer
Data representation: I/O Port 1
Description: Displays current speed of motor 1
Value encoding: Some integer x such that:
0<= x <= $max_forward_speed$
` or
$max_backward_speed$ <= x <= 0
Timing characteristics: Update in background every 1 ms, displayed to user in console upon request

Output data item: motor 2 speed
Acronym: //motor_2_speed//
Class: Integer
Data representation: I/O Port 2
Description: Displays current speed of motor 2
Value encoding: Some integer x such that:
0<= x <= $max_forward_speed$
` or
$max_backward_speed$ <= x <= 0
Timing characteristics: Update in background every 1 ms, displayed to user in

console upon request

Output data item: motor 3 speed
Acronym: //motor_3_speed//
Class: Integer
Data representation: I/O Port 3
Description: Displays current speed of motor 3
Value encoding: Some integer x such that:
0<= x <= $max_forward_speed$
` or
$max_backward_speed$ <= x <= 0
Timing characteristics: Update in background every 1 ms, displayed to user in console upon request

Output data item: error message
Acronym: //ERROR//
Class: String
Data representation: String printed out to debugger console
Description: Error message that displays than an error code in execution and where the error occurred (i.e. motor or sensor)
Value encoding: String that comes from a predefined table of basic errors
Timing characteristics: Error message is printed to the console once it arrive in the interface within 1 ms of its arrival.

Event Table

Mode	Definition
normal ACTION	@T(%no_error%) //ERROR// shows no_error
motor_error ACTION	@T(%motor_1_overclock_forward%) !motor_1_speed! := $max_forward_speed$ //ERROR// shows $motor_1_error$
motor_error ACTION	@T(%motor_2_overclock_forward%) !motor_1_speed! := $max_backward_speed$ //ERROR// shows $motor_1_error$
motor_error ACTION	@T(%motor_3_overclock_forward%) !motor_3_speed! := $max_forward_speed$ //ERROR// shows $motor_3_error$
motor_error ACTION	@T(%motor_1_overclock_backward%) !motor_2_speed! := $max_backward_speed$ //ERROR// shows $motor_2_error$
motor_error ACTION	@T(%motor_2_overclock_backward%) !motor_2_speed! := $max_forward_speed$ //ERROR// shows $motor_2_error$
motor_error ACTION	@T(%motor_3_overclock_backward%) !motor_3_speed! := $max_backward_speed$ //ERROR// shows $motor_3_error$

motor_error ACTION	@T(!motor_1_speed! != 0 && //motor_1_speed// = 0) !motor_1_speed!:= 0 //ERROR// shows $motor_1_error$
motor_error ACTION	@T(!motor_2_speed! != 0 && //motor_2_speed// = 0) !motor_2_speed! := 0 //ERROR// shows $motor_2_error$
motor_error ACTION	@T(!motor_3_speed! != 0 && //motor_3_speed// = 0) !motor_3_speed! := 0 //ERROR// shows $motor_3_error$
sensor_error ACTION	@F(%sensor_mic_running%) !sensor_mic! := 0 //ERROR// shows $sensor_mic_error$
sensor_error ACTION	@F(%sensor_ultrasonic_running%) !sensor_ultra! := 0 //ERROR// shows $sensor_ultrasonic_error$
sensor_error ACTION	@F(%sensor_touch_running%) !sensor_touch! := 0 //ERROR// shows $sensor_touch_error$
sensor_error ACTION	@F(%sensor_light_running%) !sensor_light! := 0 //ERROR// shows $sensor_light_error$
sensor_error ACTION	@F(%bluetooth_sensor_running%) !motor_1_speed! := 0 !motor_2_speed! := 0

	!motor_3_speed! := 0 //ERROR// shows $connection_error$
sensor_error ACTION	@F(!time_robot_sent! - !time_base_received! <= $timeout$) !motor_1_speed! := 0 !motor_2_speed! := 0 !motor_3_speed! := 0 //ERROR// shows $connection_error$
sensor_error ACTION	@F(!time_base_sent! - !time_robot_received! <= $timeout$) !motor_1_speed! := 0 !motor_2_speed! := 0 !motor_3_speed! := 0 //ERROR// shows $connection_error$
sensor_error ACTION	@T(!poll_time! > $poll_interval$!motor_1_speed! := 0 !motor_2_speed! := 0 !motor_3_speed! := 0 //ERROR// shows $connection_error$

Glossary

Symbolic Constants

Name	Definition	Value
no_error	String	“Running well”
$motor_1_error$	String	“ERM0000001”
$motor_2_error$	String	“ERM0000002”
$motor_3_error$	String	“ERM0000003”
$sensor_mic_error$	String	“ERS0000001”
$sensor_ultrasonic_error$	String	“ERS0000002”
$sensor_touch_error$	String	“ERS0000003”
$sensor_light_error$	String	“ERS0000004”
$connection_error$	String	“Connection to the robot has been interrupted.”
$poll_interval$	Time (in ms) between base-station polls to determine Bluetooth connectivity.	1000ms
$timeout$	Time in ms that the !message_received! signal has to be received between !message_sent! before *connection_lost* is declared.	5000ms
$max_forward_speed$	Maximum speed at which motor can operate without reaching an !unsafe_speed!	TBD

$max_backward_speed$	Maximum speed at which motor can operate without reaching !unsafe_speed!	TBD

Text Macros

Name	Definition
!motor_1_speed!	Integer to contain motor 1 speed
!motor_2_speed!	Integer to contain motor 2 speed
!motor_3_speed!	Integer to contain motor 3 speed
!sensor_light!	Value to hold output from light sensor
!sensor_mic!	Value to hold output from microphone
!sensor_ultra!	Value to hold output from ultrasonic sensor
!sensor_touch!	Value to hold output from touch sensor
!poll_time!	Amount of time required to poll from base station from robot.

Conditions

Condition	Definition
%Sent_Message%	A message has been sent.
%Receive_Message%	A message has been received.
%Await_Message%	A response is needed.
%Check_Format%	A message is verified to be in the correct format.
%no_error%	Not in an error mode.
%sensor_mic_running%	Microphone sensor is returning data.
%sensor_ultrasonic_running%	Ultrasonic sensor is returning data.
%sensor_touch_running%	Touch sensor is returning data.
%sensor_light_running%	Light sensor is returning data.
%bluetooth_sensor_running%	Bluetooth connection active
%motor_1_overclock_forward%	!motor_1_speed! >= $max_forward_speed$
%motor_2_overclock_forward%	!motor_2_speed! >= $max_forward_speed$
%motor_3_overclock_forward%	!motor_3_speed! >= $max_forward_speed$
%motor_1_overclock_backward%	!motor_1_speed! >= $max_backward_speed$
%motor_2_overclock_backward%	!motor_2_speed! >= $max_backward_speed$
%motor_3_overclock_backward%	!motor_3_speed! >= $max_backward_speed$