UML

Self-Balancing App-Controlled Robot

Abstract

This document includes diagrams and descriptions on how SAR systems are going to work.

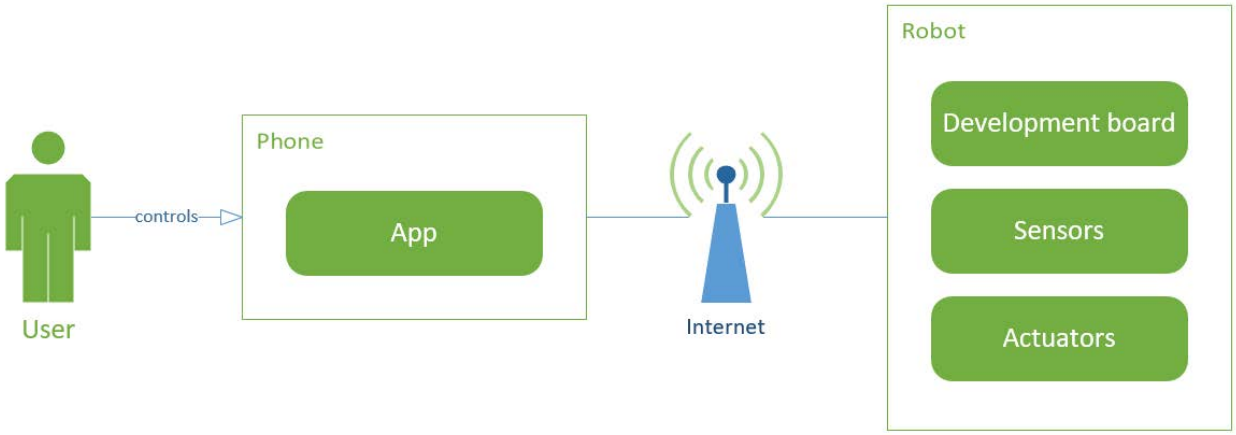
Konghon Choo and Eveline Ververgaert

11-09-2017

Version	Description	Date
0.1	Setup	08-09-2017
0.2	Use case, class diagram, state diagram, protocols	09-11-2017

Table of content

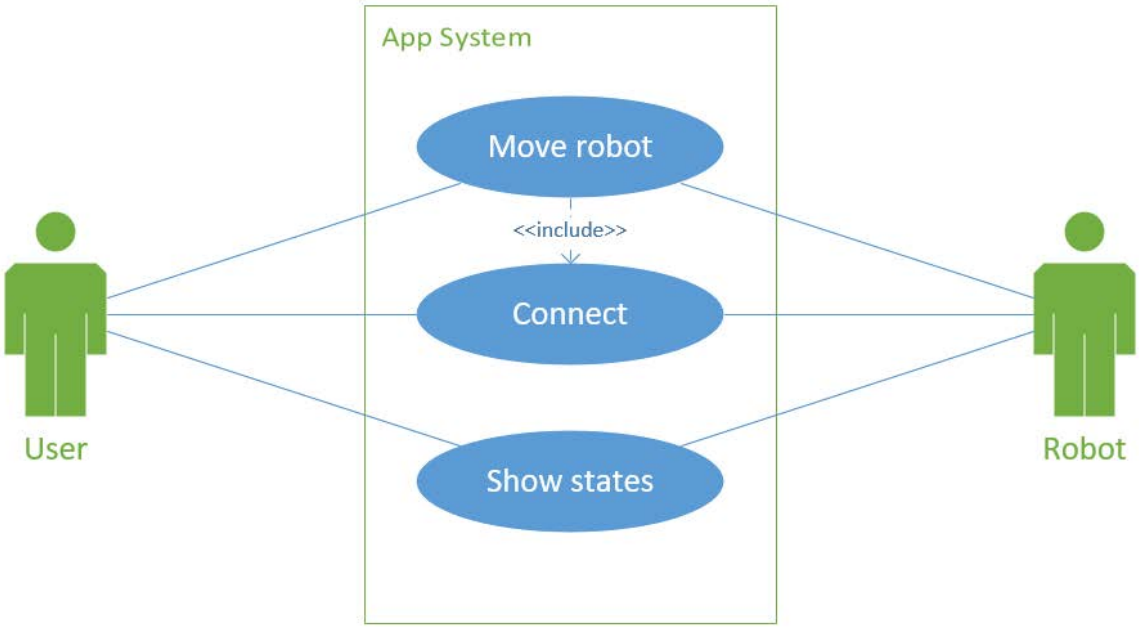
System architecture



This diagram displays a rough sketch of what SAR looks like on the outside. The SAR system consists of 2 parts, the App and the Robot. The user controls the robot through the app. The app and robot communicate through a WiFi module.

Use cases

Diagram



Description tables

UC1	**Move Robot**
Goal in context	The robot is moved into requested direction.
Preconditions	The robot is on and connected (UC2).
Successful end condition	The robot moves into requested direction.
Failed end condition	The robot moves into wrong direction or doesn't move at all.
Actors	User, robot and app
Trigger	User moved the joystick forward (or another direction).

	Steps	**Action**
Main flow:	1	App sends command (e.g. direction forward) to the robot.
	2	Robot detects if there are no obstacles.
	3	Robot moves into requested direction.
Exception flow:	2.1	Robot detects an obstacle.
	2.2	Robot does not move.
	3.1	Robot moves into an undesired direction.

UC2	Connect
Goal in context	The robot and app are connected.
Preconditions	The robot is on.
Successful end condition	The robot successfully connects with the app.
Failed end condition	The robot fails to connect with the app.
Actors	User, robot and app
Trigger	User pressed “connect”.

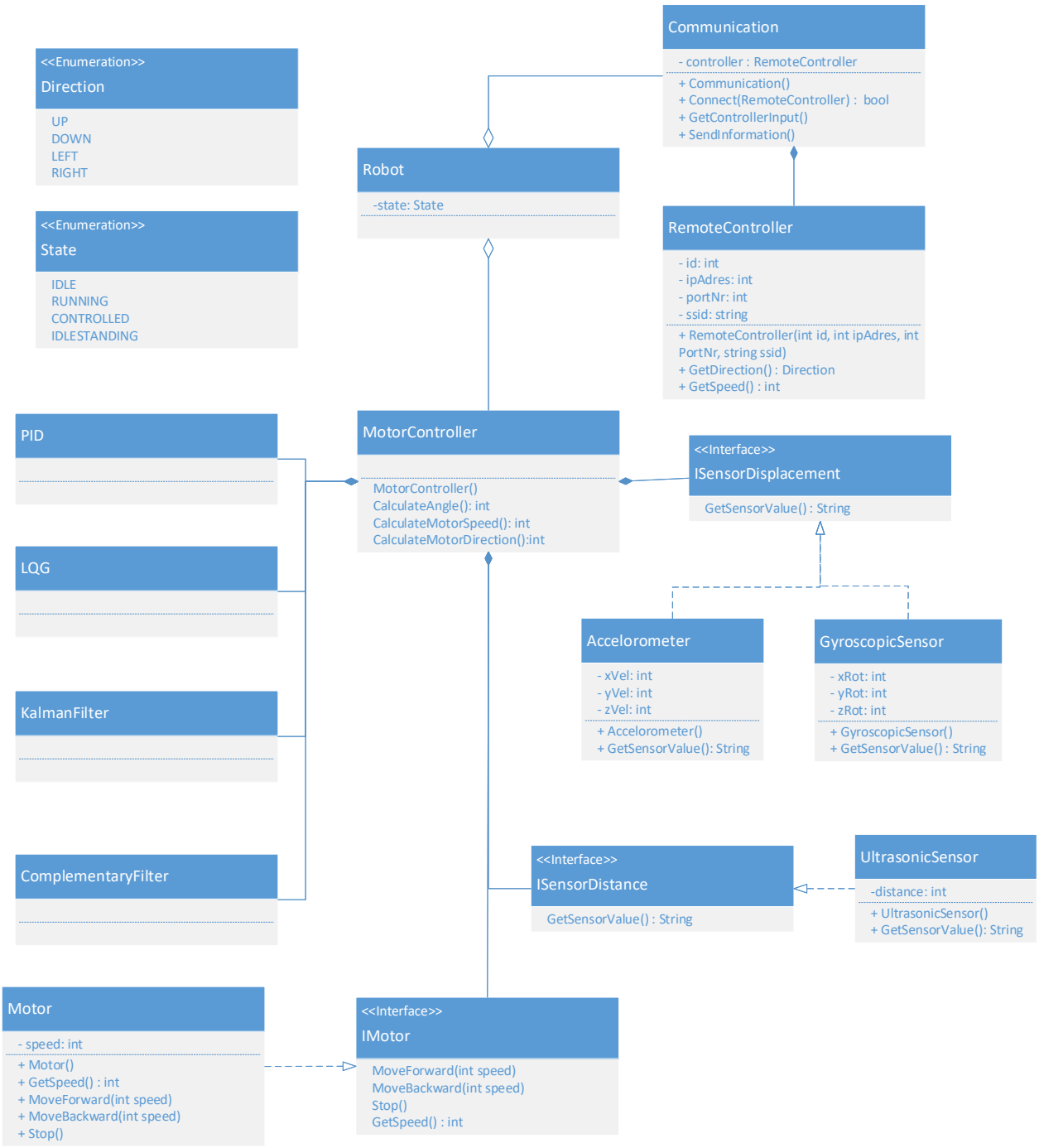
	Steps	Action
Main flow:	1	App searches for the robot hotspot.
	2	App finds robot hotspot.
	3	App and robot use handshake technique to connect.
	4	App shows on UI it is connected.
Exception flow:	1.1	App is unable to find robot.
	1.2	App displays error message.
	3.1	Handshake is unsuccessful.
	3.2	Retry handshake (this step could happen multiple times under water).
	3.3	Handshake is still unsuccessful.
	3.4	App displays error message.

UC3	Show states
Goal in context	The app is able to display status information coming from the robot.
Preconditions	The robot is on and connected.
Successful end condition	The app displays states to the user.
Failed end condition	The app displays old states or states are unknown.
Actors	User, robot and app
Trigger	Robot sends information.

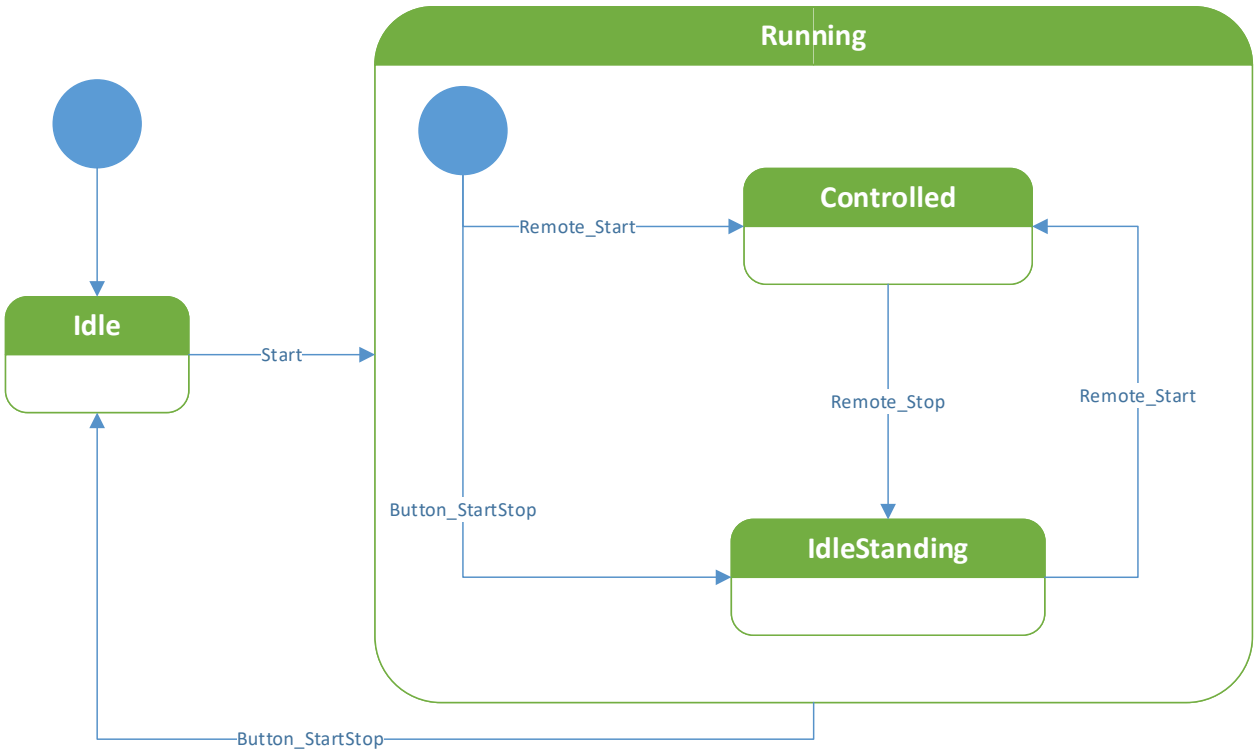
	Steps	Action
Main flow:	1	App receives message from robot.
	2	App unpacks the message.
	3	App correctly displays the information (see also app design document)
Exception flow:	1.1	App does not receive message from robot.
	1.2	Nothing changes in UI (old information stays)
	2.1	App is unable to read the message.
	2.2	App displays old information

Class diagrams

Robot



State diagram - Robot



Protocols

Identify

When a device connects with the hotspot on the robot it must identify itself in order to get a valid connection.

From device	App
To device	Robot
Description	Connect and identify this device as controller.

App	Robot
Identify:CONTROLLER:01 (<action>:<device type>:<ID>)	
	ACK
ACK	

Update information

When the robot has new information for the app it shall send it. The information includes e.g. battery level and speed.

From device	Robot
To device	App
Description	Sending new information

App	Robot
	Info:<long string message>
ACK	

Instructions

When the app is connected to the robot, the user can decide to start controlling the robot. In order to establish this, the app sends the user input (instructions) to the robot.

From device	App
To device	Robot
Description	Sending new instruction

App	Robot
Instruction:<Direction><speed> (e.g. instruction:LeftDown50)	
	ACK

Disconnect

When the user disconnects the app from the robot, the connection is broken and the robot goes into idle while still balancing itself.

From device	App
To device	Robot
Description	App disconnect from the robot.

App	**Robot**
Disconnect:CONTROLLER:01 (<action>:<device type>:<ID>)	
	ACK