RemoteMD

- **Project description**

  RemoteMD is a web application interface that helps a remote doctor to communicate with his patient via a robot.
  The doctor can keep tracking his patient by moving the robot around the patient's home or by receiving updates from the robot.

- **Installation - NodeJS:**

  1. Windows:
     a. Download NodeJS installer: https://nodejs.org/en/download/
     b. Run the installer and follow instructions.
  2. Linux (Ubuntu):
     a. `'sudo apt-get install nodejs'`
     b. `'sudo apt-get install npm'`
  3. Ensure NodeJS has been installed: run `'node -v'` in your cmd.
  4. Update version of npm: `'npm install npm --global'`
  5. Create package.json file: `'npm init'` and follow instructions.
  6. Install packages : `'npm install <package name>'`
     a. in Robot: express, ws, spawn-handler, readline.
     b. in Cloud: express, http, ws, fs.

- **Running the project**

  1. The Robot: in the folder contains robot_app.js file run
     `'node robot_app.js'`
  2. The cloud (website): in the folder contains cloud_app.js file run
     `'node cloud_app.js'`

- ## Files description

1. views folder:
    a. index.html & index.css – contains the homepage content: login page.
    b. doctor.html & doctor.css – the doctor's page: choosing a Robot.
    c. navigation.html & navigation.css – navigation page: contains navigation control buttons and the Robot's state window.
    d. script.js – JavaScript code for navigation page.
    e. Icons folder – icons images.
2. routes folder:
    a. index.js – manages routing.
3. cloud_app.js – manages a web server: handles user's requests and manages log files.
4. robot_app.js – manages the Robot's server.

- ## How is it works?

The connection between the browser and the cloud, and between the cloud and the robot established by Websockets.

The cloud and the Robot are **Websocket servers –** simply a TCP application listening on any port.

In out case, the cloud listening on port 80 and the Robot on port 8080.

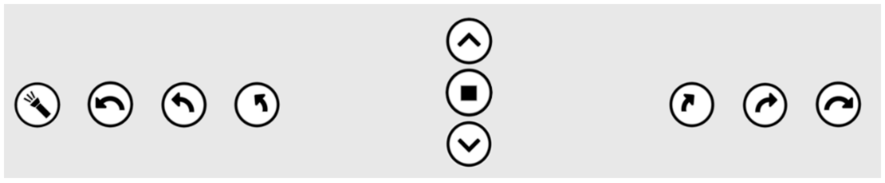
The browser and the cloud are **Websocket clients –** application that use the websocket API to communicate with websocket server.

Further reading: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

- **The Robot**

The Robot consists of a Arduino board and a LIDAR.
Arduino and LIDAR code runs by a C++ program named MAPING.exe, and our Robot's server runs MAPING.exe by a NodeJS module (named spawn).
through this module we are passing commands to the program and passing back indications (confirmation/success/failure).

- **Commands and Responses**

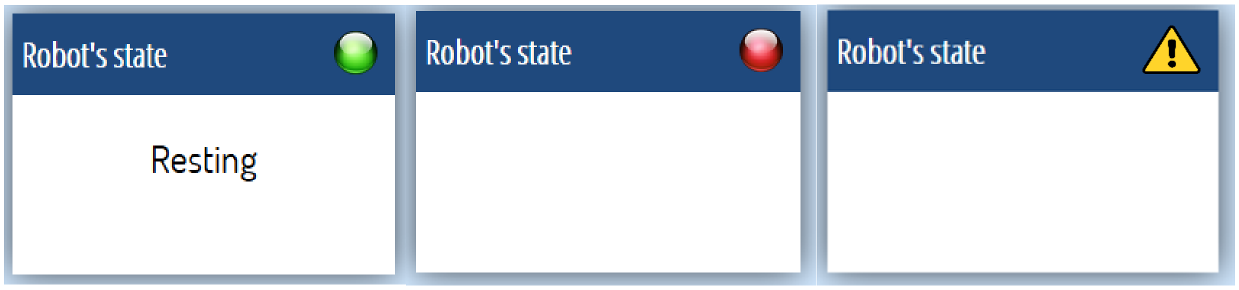


Commands		**Responses**		
description	**Representation**	**Confirmation**	**Success**	**failure**
Flashlight ON/OFF	FL_1 / FL_0	C_FL_1 / C_FL_0	S_FL_1 / S_FL_0	F_FL_1 / F_FL_0
Turning <num> degrees left. Num: {20, 45, 90}	TU_- <num>	C_TU_- <num>	S_TU_- <num>	F_TU_- <num>
Moving forward (30 cm)	FO_30	C_FO_30	S_FO_30	F_FO_30
Stop	ST	C_ST	S_ST	F_ST
Moving backwards (30 cm)	FO_-30	C_FO_-30	S_FO_-30	F_FO_-30
Turning <num> degrees right. Num: {20, 45, 90}	TU_<num>	C_TU_<num>	S_TU_<num>	F_TU_<num>

❖ The Robot sending response in JSON form: { conf: <msg> }

❖ Responses appears in the Robot's state window.

- **Robot's state**



- The Robot has 3 states: connected, closed and error.
- The cloud informs the browser by sending JSON: { robot_conn: <state> }