
Documentation

for

Android Practical Course

Robotic Simulator Controller

Prepared by
Rui Choo
Pongwanit Jeaperapong
Tobias Bauer

Technische Universität München

08.08.2017

Application Programming Interface (API)

Server Adapter Translation

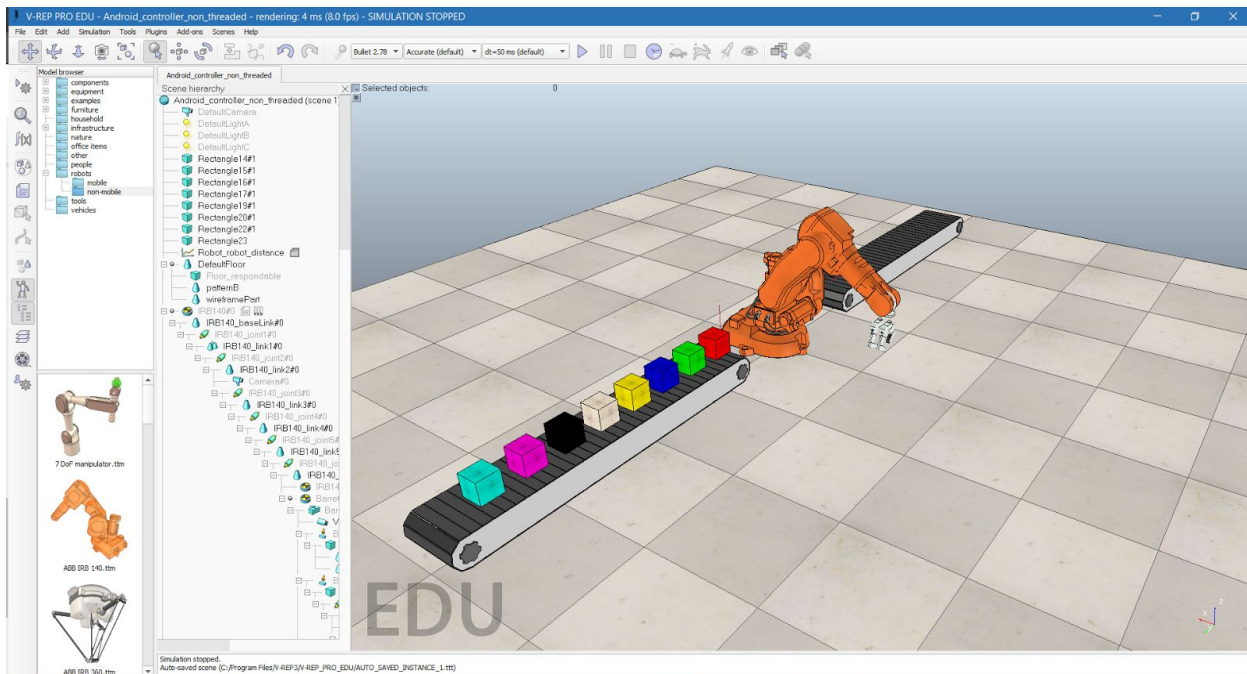
Server Adapter receive command from Android app by using Java method "BufferedReader(new InputStreamReader(Socket.getInputStream()))". It send feedback data to Android app by using Java Method "DataOutputStream.writeBytes()".

1. If Server Adapter receive "REMOTEAPI_CONNECTACCEPT", Server Adapter will reply to Android app with "REMOTEAPI_CONNECTREQ" in order to make connection handshake.
2. If Server Adapter receive "SIMULATION", it will expect to receive an integer
 - a. If receive "1\n", Server Adapter will call V-REP API vrep.simxStartSimulation() to start V-REP Simulation.
 - b. If receive "2\n", Server Adapter will call V-REP API vrep.simxPauseSimulation() to pause V-REP Simulation.
 - c. If receive "3\n", Server Adapter will call V-REP API vrep.simxStopSimulation() to stop V-REP Simulation which reset scenefile to its initial state.
3. If Server Adapter receive "MOVEMENTDATA", it will expect to receive a tiltLeftRight integer, a new line, a tiltUpDown integer, and a new line. After that it will call both vrep.simxSetFloatSignal(..., "rotate", *number*, ...) and vrep.simxSetFloatSignal(..., "moveUpDown", *number* ...). The number is speed of arm movement which is determined from value of tiltLeftRight and tiltUpDown.
4. If Server Adapter receive "GRIPPERDATA", it will expect to receive a gripperStatus integer, and a new line. After that it will call vrep.simxSetIntegerSignal(..., "closeGripper", gripperStatus,...) where gripperStatus is either 0 or 1.
5. If Server Adapter receive "MOVEMENTDATAVIABUTTON", it will expect to receive a command string, and a new line. After that it will call either vrep.simxSetFloatSignal(..., "rotate", *number*, ...) and vrep.simxSetFloatSignal(..., "moveUpDown", *number* ...) where number is -0.02 or 0.02 depending on direction. If command string is "STOP", it will call both vrep.simxSetFloatSignal(..., "rotate", 0, ...) and vrep.simxSetFloatSignal(..., "moveUpDown", 0, ...)
6. If Server Adapter receive "REQCOLORDATA", it will send integer value of RGB colors back to Android app in format of R+"\n"+G+"\n"+B+"\n" where + is string concatenation and R, G, and B represent integer value.
7. If Server Adapter receive "REQSHUTDOWN", it will close Server thread which close listening socket port. ThreadSupervisor thread remain running to listen for new incoming connection.

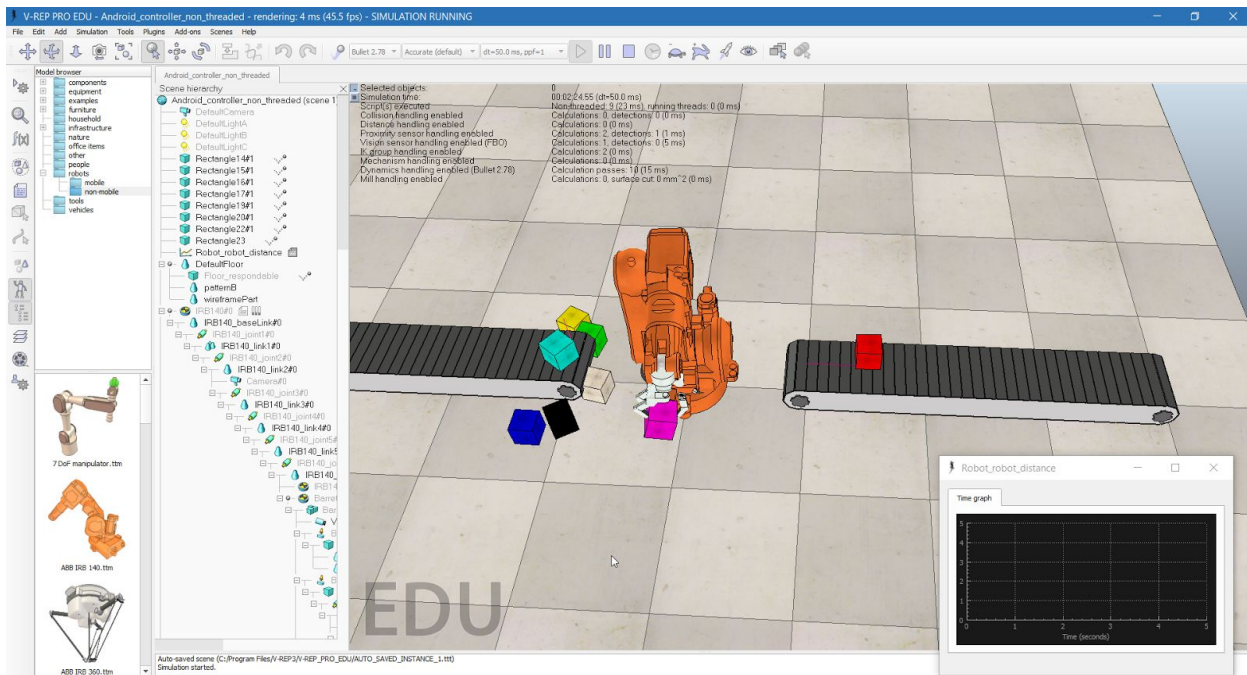
Readme/Install

1. Download and Install V-REP from <http://www.coppeliarobotics.com/downloads.html>
The link can also be obtained by googling with “V-REP Download” which is easier if this document is being read from physical paper. Optionally, you can change V-REP Remote API port from default of 19997 to something else by editing “remoteApiConnections.txt” found in root of V-REP installation directory.
2. Start V-REP and open the scene by clicking File > Open scene..., then navigate to “scenefiles/Android_controller_non_threaded.ttt”. Run Server Adapter via “Start ServerAdapter.bat”. You can edit created “serveradapterconfig.txt” if you want to connect to other computers V-REP or change its server adapter port.
3. Install the Android app with “V-REP Controller.apk” provided. Run the app and connect to Server Adapter.
4. Press play button to start simulation, and then control robot arm with either accelerometer sensor or on screen button. Press pause button to pause simulation. Press stop button to restart scenefile to its initial state.

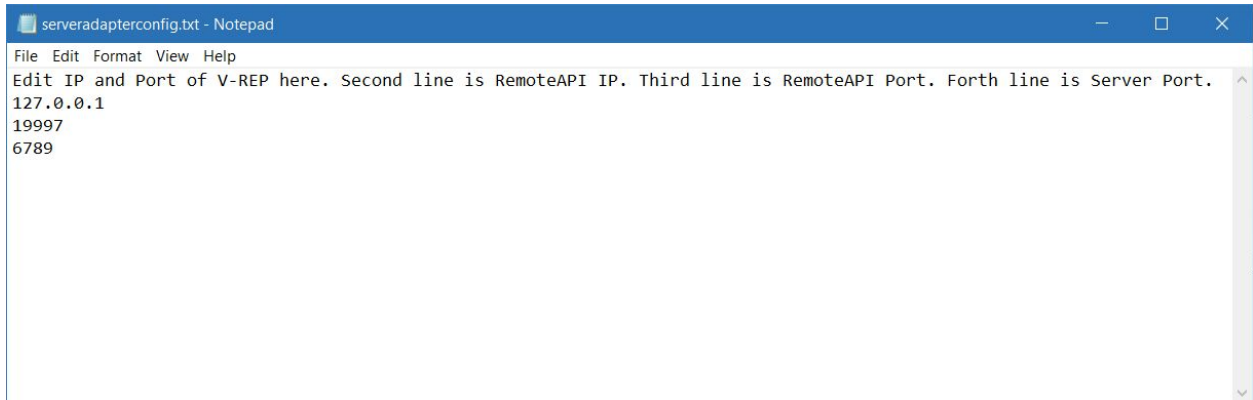
Pictures



Scene file initially

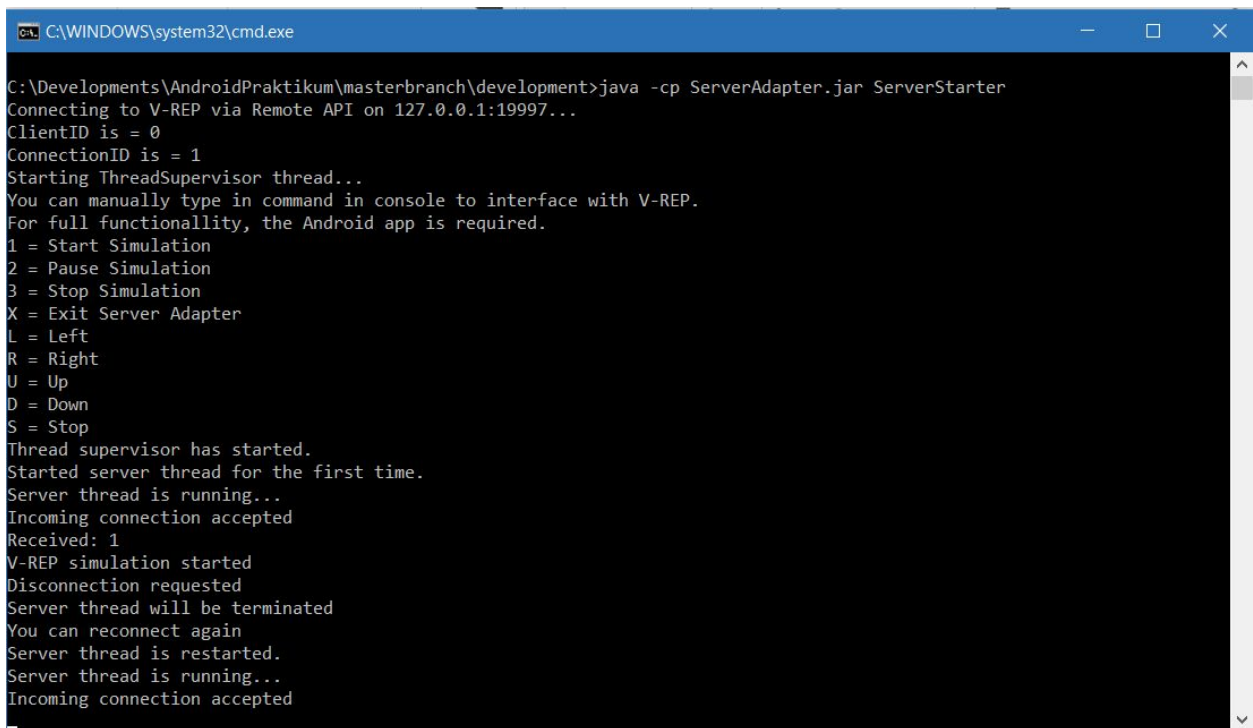


V-REP in action



```
serveradapterconfig.txt - Notepad
File Edit Format View Help
Edit IP and Port of V-REP here. Second line is RemoteAPI IP. Third line is RemoteAPI Port. Forth line is Server Port.
127.0.0.1
19997
6789
```

Configuration of IP and Port



```
C:\WINDOWS\system32\cmd.exe
C:\Developments\AndroidPraktikum\masterbranch\development>java -cp ServerAdapter.jar ServerStarter
Connecting to V-REP via Remote API on 127.0.0.1:19997...
ClientID is = 0
ConnectionID is = 1
Starting ThreadSupervisor thread...
You can manually type in command in console to interface with V-REP.
For full functionallity, the Android app is required.
1 = Start Simulation
2 = Pause Simulation
3 = Stop Simulation
X = Exit Server Adapter
L = Left
R = Right
U = Up
D = Down
S = Stop
Thread supervisor has started.
Started server thread for the first time.
Server thread is running...
Incoming connection accepted
Received: 1
V-REP simulation started
Disconnection requested
Server thread will be terminated
You can reconnect again
Server thread is restarted.
Server thread is running...
Incoming connection accepted
```

Server Adapter in action

Connect to Server Adapter



V-REP
CONTROLLER



 192.168.0.156

 6789

CONNECT

First connect screen

Connect to Server Adapter




V-REP

CONTROLLER

Ensure that the Server Adapter and V-REP with the loaded scene file are open.

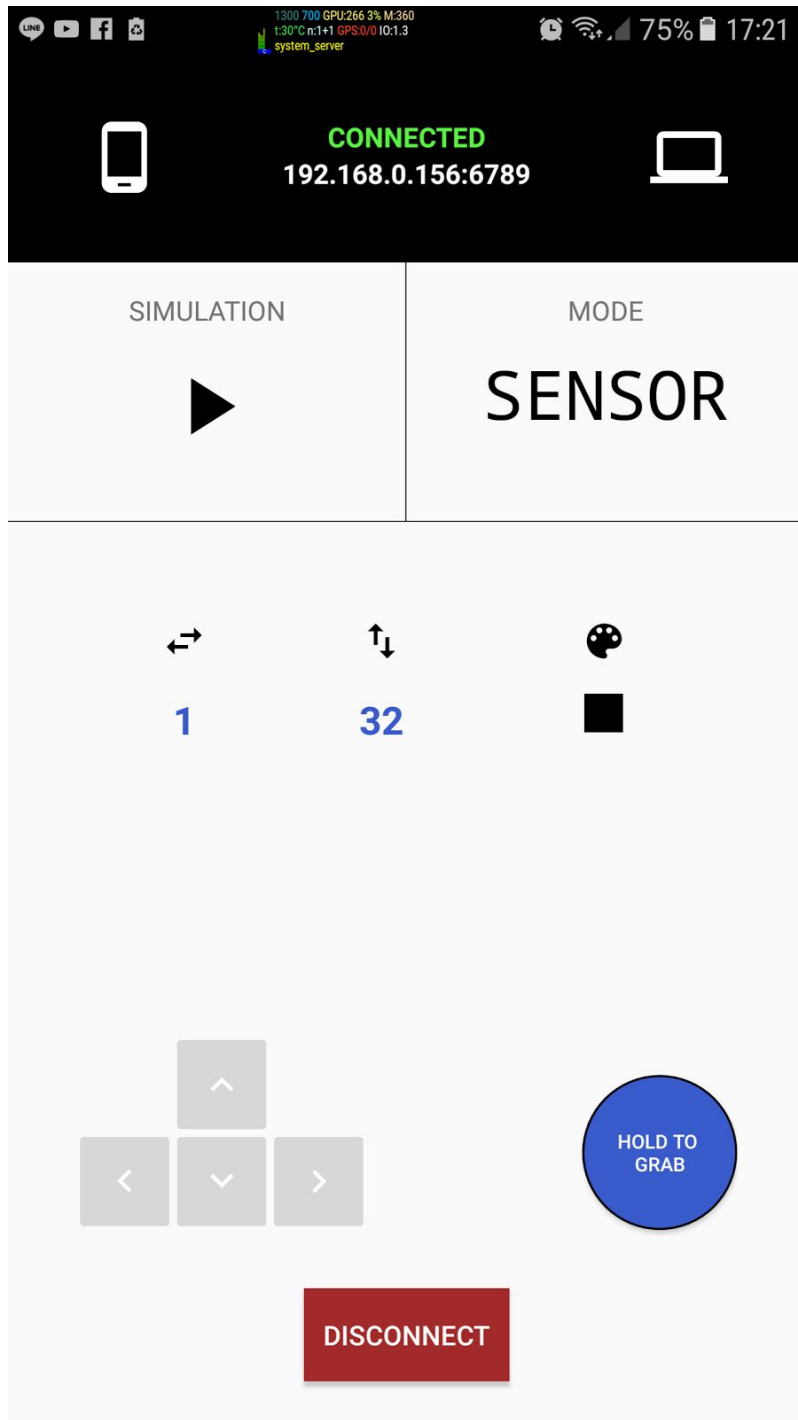


 192.168.0.156

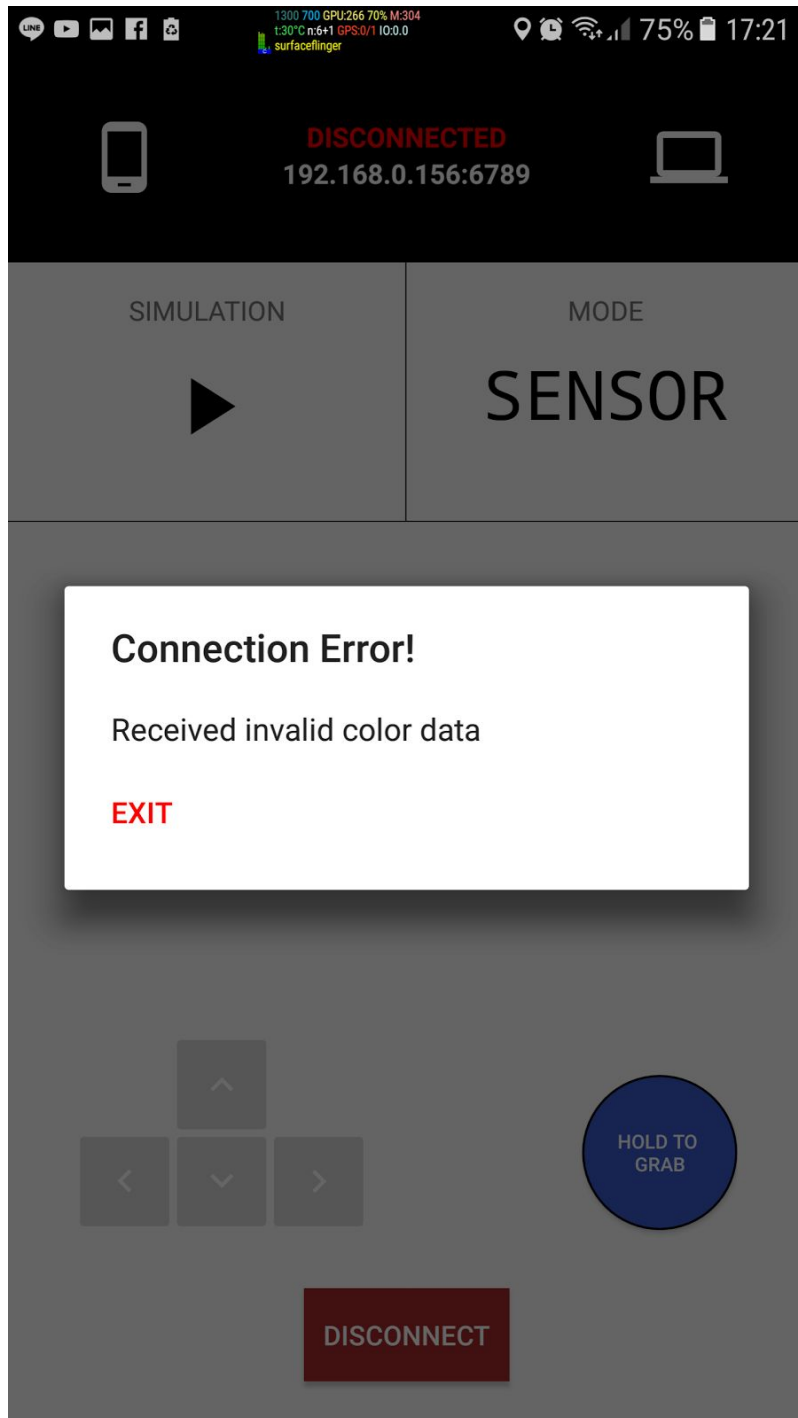
 6789

CONNECT

[Additional information on how to connect](#)



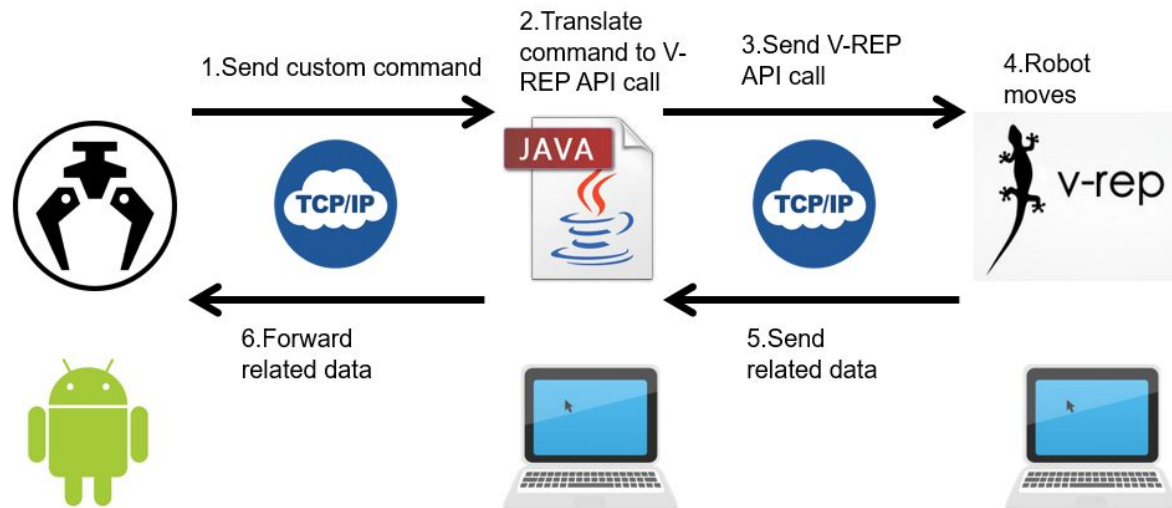
User Interface for Control Screen



What happens if there is connection error

Project Source Tree

Overview of project



The project consists of 3 parts.

1. "V-REP Controller" Android app
2. "Server Adapter" Java program that runs on Windows/Linux/MacOS
3. V-REP with scene file "Android_controller_non_threaded.ttt"

"V-REP Controller" Android app

The Android app itself consists of 3 screens - the Main activity, the About page and the Controller activity.

1. Main activity

Launching the app will display the main screen, which contains the following widgets:

App logo and header	Clicking on this will display the About page (more details below)
Tooltip	Reminds the user to launch the server adapter and the V-REP scene file first prior to attempting to connect via the app

Input text box for IP address	The user should input the IP address of the computer which he/she wishes to connect to, previously entered IP address will be loaded from SharedPreferences and pre-inputted for convenience.
Input text box for port	The user should input the port that the server adapter is listening on for incoming connections, previously entered Port will be loaded and from SharedPreferences and pre-inputted for convenience.
"Connect" button	Begins a connection attempt to the input IP address and port, and IP and Ports will be saved via SharedPreferences

2. About page

This page lists the members of our group as the developers of the app under the supervision of Sebastian Eckl as part of the IN0012 Google Android Practical Course conducted at TUM during the summer semester of 2017.

3. Controller activity

This page can be divided into 6 sections as displayed in the diagram on the right (in top-down and left-right order):

(a) Connection details header

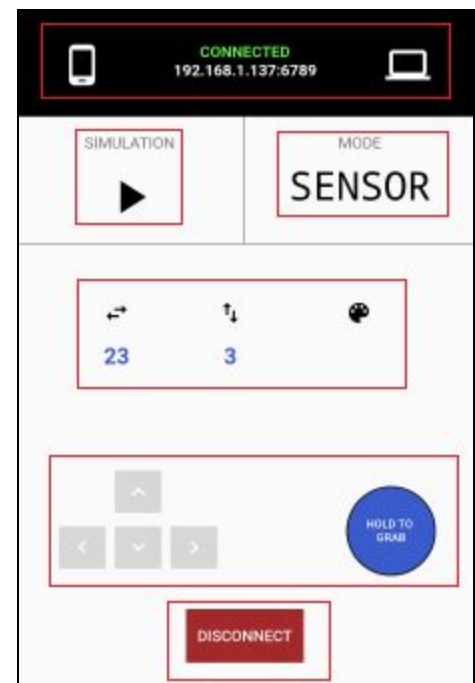
This header displays the current connection status and the IP address and port that the Android device is using to connect to the server adapter running on the PC.

(b) Simulation control section

In this section, the user can either start, pause or stop the simulation. The simulation has to be started before the robot will respond to any movement commands.

(c) Control mode section

Here, the user toggles between "Sensor" or "Button" as the control mode. Sensor mode utilises the Android device sensors to send movement data to the robotic arm, while Button mode utilises the 4-directional arrow keys mentioned in (e) below to move the arm. Switching between each mode will toggle the transparency setting of the relevant data for each mode accordingly.



(d) Data section

This section displays 2 types of data:

I. Left/right and up/down tilt angle of the device

This is based on the data obtained via the sensors on the device; the data is then processed to calculate the current tilt angle of the phone.

II. Colour of object detected by the image sensor on the robotic arm

The robotic arm has an image sensor attached to its end which reads in the colour of what it is detecting at the moment, i.e. if the arm is holding onto an object, then the image sensor will read in the colour of that object. This colour data is refreshed every 0.5 seconds.

(e) Arm control buttons

This section contains the 4-directional keypad that is used when "Button" mode is activated. Pressing a direction will send the corresponding movement data to the arm.

In addition to the keypad, there is a button to control the gripper. Holding the button down will close the gripper and releasing it will open the gripper.

(f) Disconnect button

The user can choose to stop the connection between the Android device and the server adapter by clicking on the "Disconnect" button.

Activity lifespan

When this activity is started, the relevant sensors in the device are started. Only the accelerometer is utilised here.

The SensorEventListener implementation will be initialised; any time there is a change in the sensor values, the listener will be called and the updated sensor values will be read by the app.

The raw movement data is processed and the left/right and up/down tilt angles of the phone are calculated. There is a parameter included such that the data will only be updated every 100ms.

The new tilt angle values are displayed in the app and sent to the server adapter. Similarly, when the gripper button is pressed down or released, a command will be sent to the server adapter for handling of the scene file.

If there is any kind of connection error occurs, this activity stop trying to communicate with Server Adapter, then show error dialog with error message and exit button. If user press exit button, the activity will terminate and return to connect screen (MainActivity).

Server Adapter

It consists of 3 java class files.

1. "ServerStarter.java" is the main class. After it is started, it will try to read config file "serveradapterconfig.txt" for V-REP Remote API address, port, and server port. If the config file is not found, it will create a new one with default config. V-REP Remote API address is 127.0.0.1, port is 19997, server adapter port (for android app to connect to) is 6789. Then, server adapter will try to connect V-REP Remote API. If it fails, it will exit. If succeeded, it will instantiate class ThreadSupervisor. Finally, it will stay in loop to receive user input on PC for robot arm control.
2. "ThreadSupervisor.java" will open server port. Once there is an incoming connection from Android app, ThreadSupervisor will start a new thread of Server.java class. ThreadSupervisor will continuously monitor for event of Server.java class thread is killed in event of Android app disconnection. If that happens, ThreadSupervisor will restart a new Server.java class thread when there is new incoming connection.
3. "Server.java" is the class responsible for receiving custom command from the Android app, and translate to corresponding V-REP Remote API command. The class will also receive feedback data from V-REP such as detected color, and forward the data to Android app. If there is a connection error or Android app request disconnection. The Server thread will terminate itself.

V-REP with scene file

Startup and Handling

After starting V-REP and loading the scene file you can pan, rotate and shift the camera position, by clicking the corresponding symbol on the toolbar. You can also manually start, pause and stop the scene with the toolbar.

The scene

The V-REP scene file contains two opposing conveyor belts, each having a sensor that detects objects and controls the corresponding conveyors according to the sensor data. On the first belt lie 8 Blocks in a row, each having a different color. In between there is an IRB140 Robot with an attached gripper, that includes a color detecting vision sensor.

Motion and Communication

The Robot and the gripper have each an independent script that tells them how to behave, these scripts communicate over a tube. At runtime of the Main scene, global variables - called signals - can be created. We used these signals to trigger functions in the IRB140 script, that effectively move the Robot. These signals can also be set by the Remote API connection, this is basically how the app Controls the robot.