Final documentation

Android MISL Control

Android Practical Summer Term Course (SS15)

Fent, Philipp

Sautermeister, Benjamin

07/14/2015

Table of contents

1. Introduction 4
2. Project team 4
3. Project plan 5
   3.1 Project description 5
   3.2 Project flow 5
   3.3 Prototyping 6
4. Architecture 7
   4.1 Communication 7
   4.2 Virtual controls 8
5. Implemented functionality 9
   5.1 Overview 9
   5.2 Planned features 9
   5.3 Additional features 11
6. Test plan 12
7. End-user tutorial 15
8. Project conclusion 16
9. Acknowledgements 16
10. Release application and video 16

Abstract

In an international, collaborative project between Texas A&M University and Technical University of Munich, an Android application has been developed to command and control a robot vehicle that has been realized in a capstone project in Texas in the year of 2014. The ASEP robot is powered by a MISL controller introduced by NASA and allows students to have first contact with space qualified hardware. The resulting Android application that is developed by German students allows to control the robot, illustrate its position on a map, as well as to visualize its situation using a 3D model. The project's realization is oriented on the existing Windows GUI application that has been developed by the ASEP team.

1. Introduction

Due to the need for a versatile system, NASA has set standards for a stackable layer architecture that utilizes uniform power and data bus connectors between the boards. This system is called the Modular Integrated Stackable Layers (MISL) system and was developed in cooperation with Texas A&M University (TAMU) in College Station, Texas[1]. In 2014, a group of TAMU students integrated this system into an Articulated Suspension Exploratory Platform[2] (ASEP) robotic vehicle that demonstrates the capabilities of the MISL stack. To control the robot, a Windows GUI application was created by using LabView that is able to connect to ASEP using a wireless connection.

For increased visibility, the support of mobile platforms and a simplified demonstration of MISLs and ASEPs capabilities, students from the Android practical course of Technical University in Munich (TUM) and Texas A&M University (TAMU) developed an Android application for wireless control and display of sensor data in this joint international project. As a result, new ways of interaction with ASEP are going to be possible. To realize the project, the development and testing of the application is separated because only one prototype of ASEP exists. To be more precise, the German students are responsible for planning, designing and implementing the application on the one side. On the other side, the American student is responsible for the software testing. Both teams keep close contact to ensure that the final project result meets all the identified requirements.

This final documentation is for the Android part of the project only. It builds up on the previous idea report and should be used alongside with the final project report of ASEP.

2. Project team

The project team consists of three students, who design, implement and test the software. The project is supervised by Dr. Morgan of TAMU and Mr. Kannengießer of TUM.

Benjamin Sautermeister is studying in his 2nd semester of the Master in Computer Science at Technical University of Munich. He is responsible for the major parts of the application's user interface, such as the virtual control elements, the rendering of the 3D model and the display of location information in Open Street Maps.

[1] More information: http://engineering.tamu.edu/news/2013/08/23/texas-am-engineering-students-to-collaborate-with-nasa

[2] URL: http://asee-gsw.tulane.edu/pdf/the-modular-integrated-stackable-layers-system-a-nasa-development-partnership.pdf

Philipp Fent is currently studying in his 4th semester of Computer Science BSc. at Technical University of Munich and implemented the major parts of the application's networking interface, as well as the communication mechanism to ASEP.

Hector M. Paz is an Electronic Systems Engineering Technology student at Texas A&M University in College Station, Texas. He is in his senior year and tested the application's interaction with ASEP and validated the project's requirements.

3. Project plan

The following sections describe the project and give an overview about its realization plan. Moreover, the project flow is mentioned where problems that occurred during the development of the Android application are described.

3.1 Project description

This collaborative project between Technical University of Munich and Texas A&M University was developed during the Android practical course for the summer term 2015. *Android MISL Control* is an Android application designed to control the MISL-powered ASEP robotic vehicle. The goal of this project was the support of a mobile platform and to allow an easier demonstration of the ASEP robot. To achieve this, a simple and intuitive Android user interface has been developed that is similar to the existing Windows GUI application. It allows the user to directly connect to and control an ASEP robot with just a few simple steps. In addition, several kinds of sensor information have to be received and visualized within the Android application, as defined in the project requirements:

- Gyroscope data
- Accelerometer data
- Location data

As there is just a single ASEP prototype yet, there also needed to be extensive transcontinental communication, to allow testing and development to be on different sides of the Atlantic.

3.2 Project flow

The initial development of the application started good. Nevertheless, we soon hit a first obstacle since the project documentation of ASEP is very hardware focused and mentions only few details about the software running in LabView. In particular, detailed information about the used communication protocol

and the used coordinate system of the 3D model would have been helpful. To master the difficulties with the lack of documentation, we used among other things several reverse engineering approaches.

Firstly, we proceeded to look at network dumps using Wireshark and the existing Windows GUI application to figure out, which control value commands the robot to stand still or to move. After the wireless communication worked completely, we implemented various dependent features, such as parsing and displaying sensor information.

Secondly, displaying the rotation of ASEP in a 3D model was also more problematic than expected, because there was no specification on how the Euler angles represent the roll, pitch or yaw value, which coordinate system is used and how the sensor is mounted on the robot. However, we could finally implement the proper rotation of the 3D model by extracting real sensor values from a video recording of Hector Paz into our mocking connector that simulated the ASEP robot for local testing purposes.

Thirdly, due to the fact that our phone has to connect to the wireless hotspot of ASEP, we were not able to use GoogleMaps to show the robot's current position. To solve this problem, we replaced our first implementation with another that that was based on OpenStreetMaps. The latter supports the use of offline maps that fit perfectly together with our requirements.

Last but not least, we invested a lot of time in finding a solution to implement the optional video streaming feature. Because the existing camera that is mounted on top of ASEP uses a dedicated wireless connection, it was not possible to use it out of the box. As a work around, we tried to create a work around by using a second Android phone that provides the video stream. Therefore, we planned to extend the Android application with additional functionality to send and receive a video stream that is broadcasted using ASEP's hotspot. Indeed, we have been able to access, record and send the video signal using an UDP stream. But unfortunately, we ran out of time in implementing a stable working video receiver component. At the end, we came to the decision to keep this feature out of the final release in order to be able to deliver a full functional, well tested and stable application.

3.3 Prototyping

A total of seven prototypes have been released, all of them can be found in the releases section on GitHub:

https://github.com/pfent/Android-MISL-Control/releases

The first four prototypes have been released in a rapid test feedback circle with fixing potential bugs and adjustments according to the test results provided by Hector Paz. In the subsequent prototypes, the remaining functionalities have been integrated and tested to improve the stability and quality of the final application.

To make an interim conclusion, the short testing circle to validate functionality and the corresponding high feedback rate have been proven to be a good strategy and allowed an optimal use of available resources in Munich as well as in Texas.

4. Architecture

The following section gives an overview of the architecture of the two main components of the application. The presented components are used for the core functionality of the overall project, in detail for connecting, communicating and steering the ASEP robot.

4.1 Communication

Figure 1 illustrates the communication module of the Android application. The nested data package implements the used communication protocol of ASEP where command packets are sent to the robot for command and control and telemetry packets are received to obtain the required sensor information for visualization. Furthermore, and *ASEPConnector* has been implemented to encapsulate the whole communication mechanism. For testing purposes, an additional *MockConnector* has been realized which simulates the ASEP robot locally. This mock implementation has proved beneficial especially for testing the UI elements, such as the proper rotation of the 3D model.

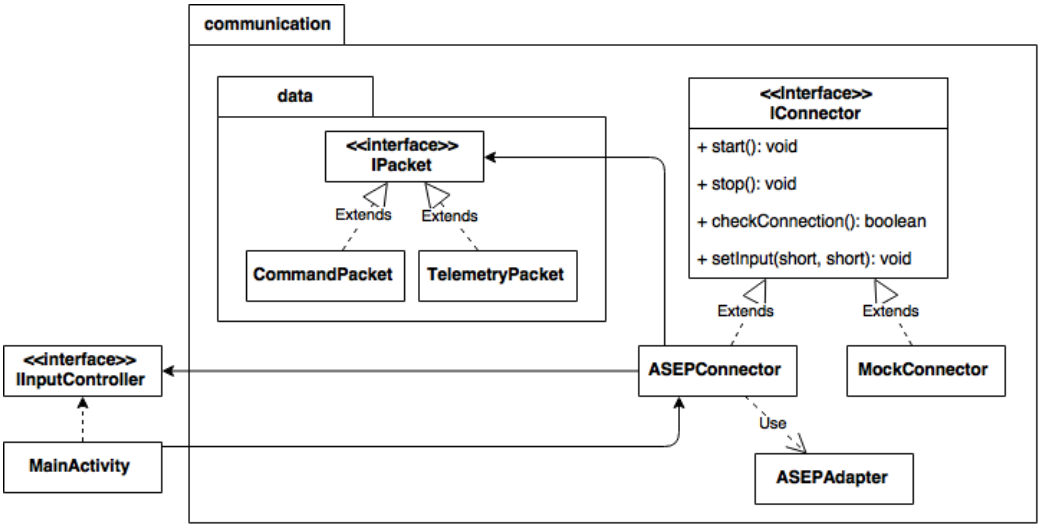

Figure 1 - Structure of the communication mechanism

4.2 Virtual controls

To control the ASEP robot, two different control mechanisms have been realized. While the first implementation is a fully touch-based virtual joystick, we also created an alternative sensor-based control mechanism. The latter allows steering using the accelerometer sensor of the Android device and acceleration using ordinary touch control.

In Figure 2, the simplified class diagram of the control mechanism is shown. Each controller is implemented as a *SurfaceView* and is based on the Model-View-Controller pattern, where the model is reused in each implementation. To encapsulate the rendering of the view, a separate render thread was implemented that updates the virtual control elements in a fixed frame rate.

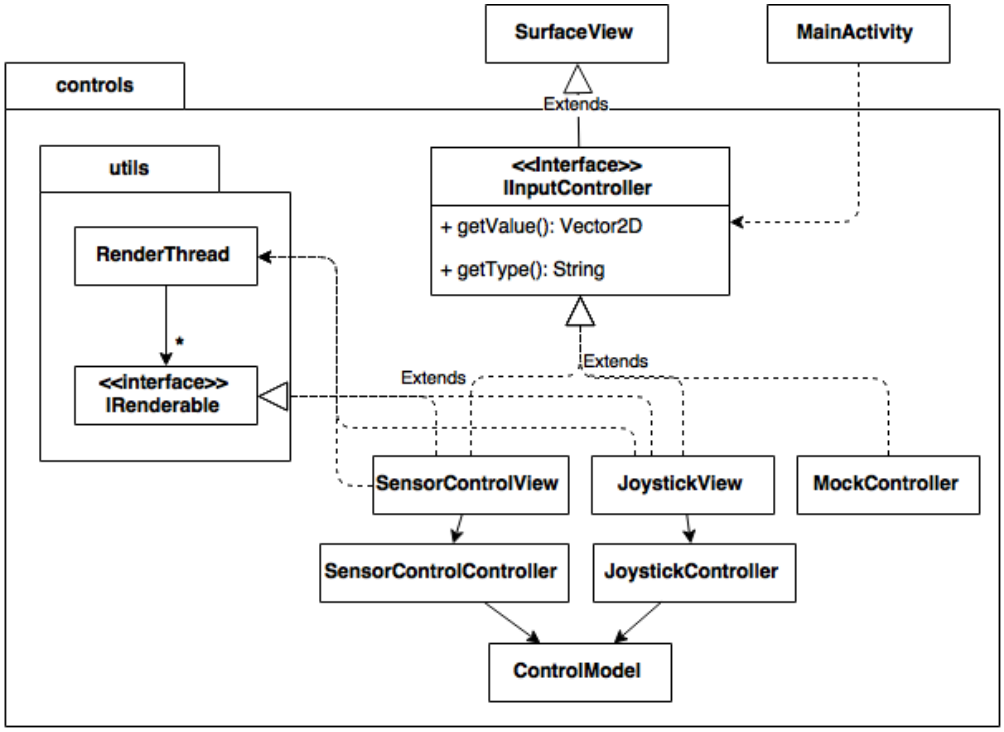

Figure 2 - Structure of the control mechanism

5. Implemented functionality

The following chapter describes the implemented functionality of the Android application. After a short overview, we will distinguish between planned and additional features and present how they have been realized.

5.1 Overview

Feature	Req. ID	Priority	Status
Wireless communication	1	MUST	✔
Setup wizard	2	OPTIONAL	✔
Controlling ASEP	3	MUST	✔
Gyroscope information	4	MUST	✔
3D rendering of ASEP's orientation	5	OPTIONAL	✔
GPS location	6	MUST	✔
Target platform constraint	7	MUST	✔
Reception of camera feed	8	SHOULD	✘
Alternative sensor-based controls	9a	ADDITIONAL	✔
Position rendering on a map with offline support	10a	ADDITIONAL	✔
Possible configuration of networking interface	11a	ADDITIONAL	✔
Support for phones (portrait layout)	12a	ADDITIONAL	✔

5.2 Planned features

The most fundamental requirement to setup wireless communication with the robot was also one of the most challenging ones, since the whole communication mechanism needed to be implemented from scratch. The documentation about the details of the protocol has also been rare and mostly had to be reverse engineered from previously recorded network dumps, which also contained some illogical quirks. As an example, the previously used GUI sent all its data to the global IPv4 broadcast address and didn't increment the used sequence numbers that is defined in the protocol. After some analysis of the code and the implementation of the first working prototype, the use of ASEP's configured IP address worked without any problems, whereas the incrementation of the sequence number resulted in a noticeable delay in response time.

The realization of the setup wizard was rather straightforward, since Hector was familiar with getting ASEP to work and provided a list of steps needed to set it up. This list has been transformed into an interactive step-by-step guide to provide a good user experience.

Controlling ASEP heavily depends on the wireless communication requirement, and couldn't be developed entirely on its own. But because the original source code of ASEP was provided, we were able to identify the relevant methods and transform the C code into Java. In case this source would not have been available, the implementation of the control mechanism would have been quite problematic, since the documentation does not contain any information about a mapping between values and commands. For instance, it is not documented that the left and right wheels can be controlled separately or that a value of 0 represents full reverse, 6 means stop and 12 relates to full throttle.

Receiving and displaying the transmitted gyroscope information worked right away, since the implementation transmitted standard *IEEE 754* 32-Bit floating point values. But as before, the range of these values was not defined and also lacked a specification of used units, respectively normal values. After comparing received and actual values, numbers in radian are used in a range between -π and +π, wrapping around at these extremes. The observed normal value for the X-Euler angle (roll) is ±π, for Y-Euler (pitch) 0 and no obvious value for Z-Euler (yaw).

To display the gyroscope information, a view of a 3D rendered and rotated model of ASEP has been implemented. The basic rendering of the model is done using the lightweight rendering framework min3D[3], an open source OpenGL ES wrapper that allows loading and parsing OBJ model files. To realize a smooth rotation of the 3D model and to counteract variations in received sensor values, an exponential low-pass filtering has been implemented.

The transmitted GPS coordinates use a decimal degree latitude and longitude format transmitted as standard single precision *IEEE 754* 32-Bit floating point values. Since this is the de facto industry standard for geographic reference systems, the unprocessed data could be used. To display this information, the Android application utilizes the *osmDroid*[4] library, which provides a *MapView* using OpenStreetMaps card data. On this map the current position of ASEP is displayed, as well as all the previously received positions are recorded and visualized as a path on the map.

An additional feature of osmDroid is offline map support that allows displaying map data, even when the device is not connected to the internet. As an additional feature, the position of the user could also be displayed on the map and a warning message could be displayed when the Android device exceeds the WiFi range of the robot's hotspot.

[3] URL: https://code.google.com/p/min3d/
[4] URL: https://github.com/osmdroid/osmdroid

Fulfillment of the target platform constraint was not an issue, since a Galaxy Tab 10.1 has been provided by Dr. Morgan and has been our main platform for development and testing of our prototypes. Nevertheless, final testing with a real WiFi connection between the tablet and ASEP can only be done after the development has finishes and the tablet is shipped back to Texas.

5.3 Additional features

The implementation of a sensor-based control of ASEP was a natural idea in contrast to using a virtual joystick only, since several racing games utilize such a control mechanisms. Because a control abstraction layer using MVC[5] has already been introduced for the virtual joystick, an additional mechanism could be added with less effort.

Adding a configurable networking interface was the practical result of testing the functionality of the application in a simulated environment, which did not necessarily have the same network address or WiFi SSID. To provide a testable Android application, a settings page has been implemented which allows adjusting several network related parameters and to test it within a local hotspot without having an actual ASEP prototype.

To also support other devices beside our main target platform, we extended the app to support portrait as well as landscape mode. Therefore, we realized separate XML layouts which are optimized for the corresponding form factors. The additional support of phones also had practical reasons, since Hector used his Android smartphone to test the functionality of the application's prototypes. As a result, it is now possible to flip the device to change the orientation and the level of information detail. In theory, any device type should now be supported.

[5] Model-View-Controller, an software architectural pattern.

6. Test plan

This chapter describes the overall testing procedure including the test planning by presenting step-by-step lists to reproduce them, as well as the test results of the proceeded tests.

Test ID	T1	**Req. ID**	2
Title	Setup wizard		

Procedure	• Start Android-MISL-Control for the first time with your WiFi turned off
	• Follow the instructions displayed on the screen:
	  • Batteries charged and plugged in
	  • Click on next or swipe to the left
	  • Power on the T-rex board
	  • Click on next or swipe to the left
	  • Initialize MISL
	  • Click on next of swipe to the left
	  • Connect your WiFi to the ASEP WiFi: This should work automatically, if you had been connected to ASEP before
	  • Click on finish		

Expected	A MISL initializing screen appears and the app starts.		
Result	In v0.2.1, stepping through the setup wizard worked until the very last step, and then the App crashed. This issue has been resolved since v0.3.0. Following releases passed the test.		

Test ID	T2	**Req. ID**	1
Title	Wireless communication		

Procedure	• Follow instructions as in T1
	• The App automatically communicates with ASEP, as long as it is connected
	• Now disconnect your device and ASEP by either:
	  • switching of your WiFi
	  • exceeding the WiFi range (ca. 100 meter / yards)		

Expected	After a timeout of approximately 2 seconds the app informs the user that ASEP is disconnected.		
Result	In v0.1.1 - v0.1.3 the communication was unstable and timeouts have been reported. This issue has been resolved since v0.1.4. Following releases passed the test.		

Test ID	T3	Req. ID	3
Title	Controlling ASEP		
Procedure	• Follow instructions as in T1 • In default configuration, a round virtual joystick is displayed on the bottom right of your device • Tap and hold your finger on the black middle section of the joystick and drag it out of position		
Expected	ASEP should react to your instructions according to your displacement of the virtual joystick.		
Result	Implemented in v0.1.4 and passed test all the time.		

Test ID	T4	Req. ID	4 / 5
Title	Gyroscope information and 3D rendering		
Procedure	• Follow instructions as in T1 • Change the situation of the robot		
Expected	The 3D model rotates according to ASEP's orientation		
Result	Tests with v0.2.1 reported inconsistencies between actual and displayed ASEP displacement. This issue has been resolved since v0.3.0. Following releases passed the test.		

Test ID	T5	Req. ID	6 / 10a
Title	GPS location and position rendering on a map		
Procedure	• Follow instructions as in T1 • In the App's main screen a map is drawn • Control ASEP to receive several GPS fixes		
Expected	If there is no GPS position available (e.g. indoors), the map shows a generic location. In case there is a GPS position available, the map surrounding the current area and a red line of the path is drawn. Remarks: Depending on your device it is possible that a generic gray background is visible instead of the map.		
Result	Implemented in v0.2.0 and passed test all the time.		

Test ID	T6	Req. ID	7
Title	Target platform constraint		
Procedure	• Use a Galaxy Tab 10.1 running 4.0.4		
Expected	The app should start without any problems.		
Result	Passes tests in all versions, since this is the development platform.		

Test ID	T7	Req. ID	9a
Title	Alternative sensor-based controls		
Procedure	• Follow instructions as in T1 • Tap on the Settings Button in the upper right corner • Tap on Controls • Tap on Control Type • Select Sensor-based Steering • Tap on the back button 2 Times • Instead of the round virtual joystick, there is a virtual lever that allows you to control the forward motion of the robot		
Expected	By tilting your device left or right you should now be able to steer ASEP.		
Result	Implemented in v0.2.0 and passed test all the time.		

Test ID	T8	Req. ID	11a
Title	Possible configurations of networking interface		
Procedure	• Startup the App, ignoring the wizards instructions • Tap on the settings button in the upper right corner • Tap on Network • On a separate computer, setup a WiFi hotspot, disable any firewalls and note your local IP address • Now start the “Mock ASEP” by running “make run” in Android-MISL-Control/vendor/ASEPMock/. • After a few seconds a message “no packets for 5 seconds!” are given out • On the Android device, tap on ASEP IP address and enter the previously noted IP address • Connect your devices WiFi to the laptops hotspot • Now restart the App by closing it via the task switcher and tapping on the App icon again		
Expected	After skipping the wizard, no error messages should appear on the App’s main screen and sample test data is shown. On the computer, the program should print the data received from the Android application.		
Result	Implemented in v0.2.1 and passed test all the time.		

Test ID	T9	Req. ID	12a
Title	Support for phones (portrait layout)		
Procedure	• Use an Android Phone and hold it horizontally • Follow instructions as in T1 • Now rotate your phone to portrait • Repeat with holding your phone vertically and switching to horizontal		
Expected	The app does not crash after changing the device orientation.		
Result	Passed all tests on the target platform and an OnePlus One device.		

7. End-user tutorial

After starting the Android MISL Control application, you will see an interactive guide that guides you to set up ASEP and getting it configured as well as ready to go. This guide can be navigated through with the next and previous buttons or using swipe gestures. After you are finished in setting up ASEP, a splash screen will appear while the Android application is loading. After a few seconds, the main screen appears and is ready for interaction.

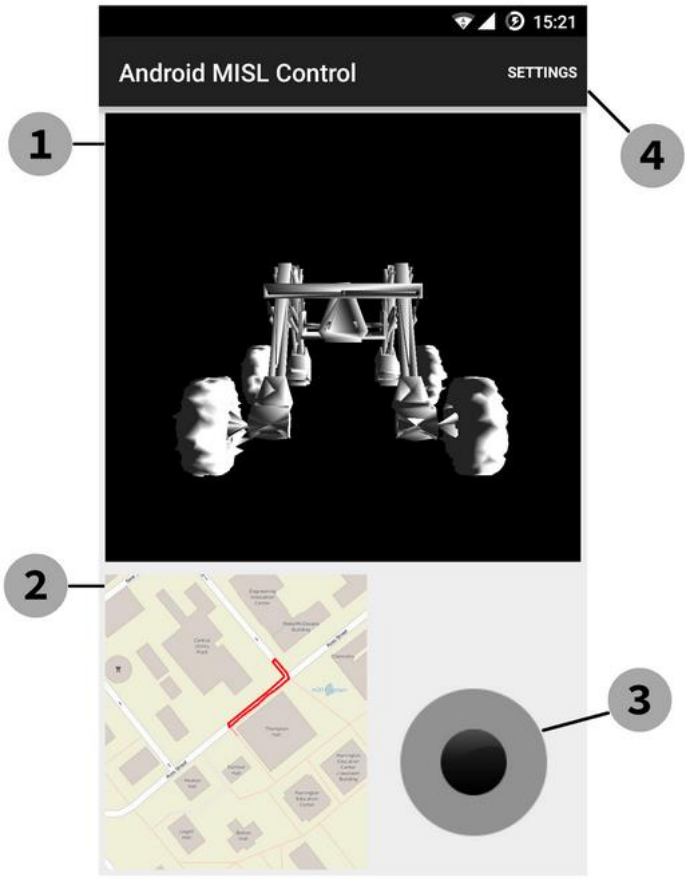

Figure 3 - Application screenshot

In Figure 3 you can see the user interface of the application's main screen. Annotation 1 shows the 3D display of ASEP, with its current orientation (normal in this case). Next, annotation 2 shows the map which displays the recorded track that ASEP has traveled so far as a bright red line. Annotation 3 shows the virtual joystick, which allows controlling ASEP by tapping and dragging the black circle in the center of the grey region in the direction you want the robot to move. Lastly, annotation 4 shows the settings button, which you can tap to get to the settings page. There you can access all the information on the app itself and its contributors. Changing any settings other than the used control type is not recommended.

8. Project conclusion

To sum up the overall project, we can say that we were able to gain a lot of experiences in working with Android. Considering the resulting application, we were able to deliver a fully functional and well tested application that is able to command and control a robot using space qualified hardware. Furthermore, working in such an international project was an amazing opportunity as well as a completely new experience for all of us. Although there were several difficulties and hidden obstacles, the development of Android MISL Control advanced significantly and has been released on 14 July 2015. Since the application is open source under an Apache License and available on GitHub, the project can easily be extended.

9. Acknowledgements

We would like to thank Mr. Kannengießer to give us the opportunity to work in such an international project including an actual stay at Texas A&M University in Texas to learn more about the culture of our American project partners. Additional thanks to Dr. Morgan for offering this interesting project as well as to Hector Paz for the successful collaboration.

10. Release application and video

The signed release application (APK) and a video that demonstrates its functionalities by controlling the actual ASEP prototype at Texas A&M University can be found on GitHub:

https://github.com/pfent/Android-MISL-Control/releases/tag/v1.0.0