

Pioneer Roboter App



Mobile Anwendungen, SS17

Gruppe 1

Reupold Michael, Turzer Stefan, Simon Uwe, Targowicki Aleksandra



Gliederung

- Motivation und Projektablauf
- Bedienung
 - Steuerung
 - Anleitung + Hilfe
- Testing
- Kamera
- Features
 - NFC
 - Sound
 - Design
 - Telemetrie
- Server



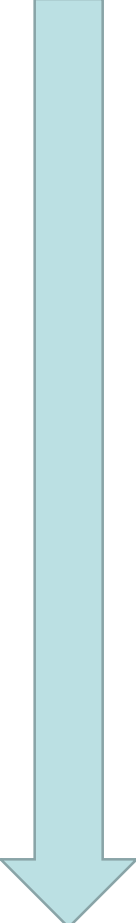
Motivation

- Mehrfach angefragtes, nie umgesetztes Projekt
- Roboter vorhanden, aber mit aufwändiger, unpraktischer Steuerung und unausgereifter Software
- Interesse an Robotik





Projektablauf

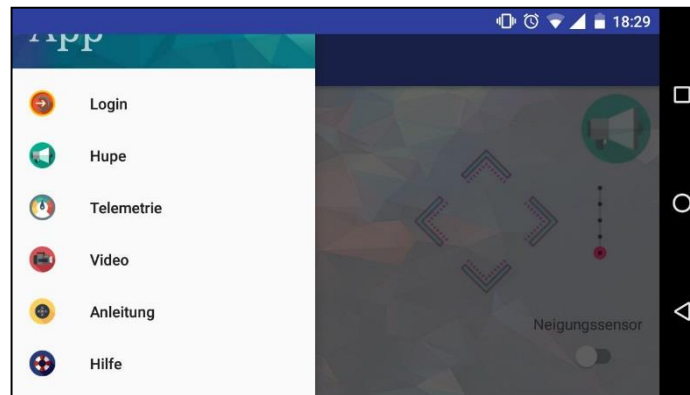
- 
- MS 1: Roboter steuern (Python)
 - MS 2: Roboter per Remote steuern über WLAN
 - MS 3: Kameraübertragung
 - MS 4: Oberfläche (App)
 - MS 5.1: NFC
 - MS 5.2: Kamera in App testen
 - MS 5.3: Sound (Hupe)
 - MS 5.4: Telemetrie
 - MS 6: Merge in App und Design
 - MS 7: Abschlusspräsentation

Bedienung

- Login



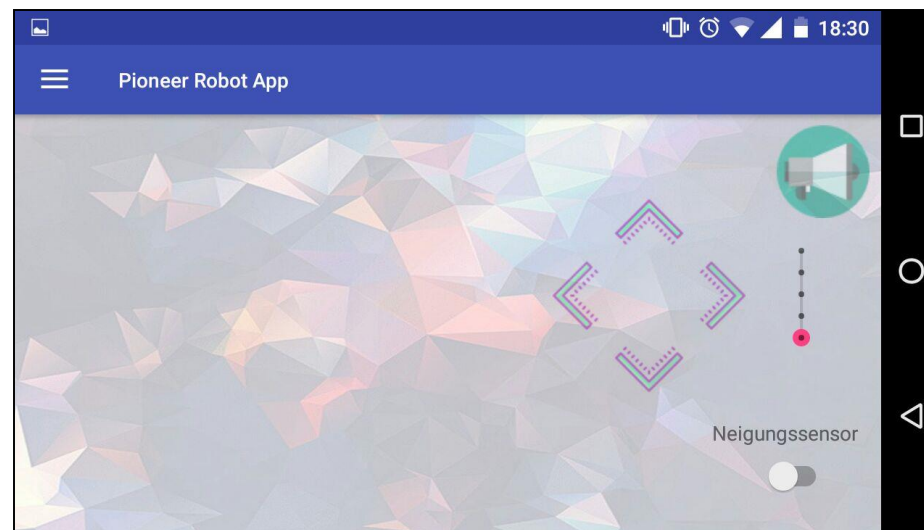
- Features



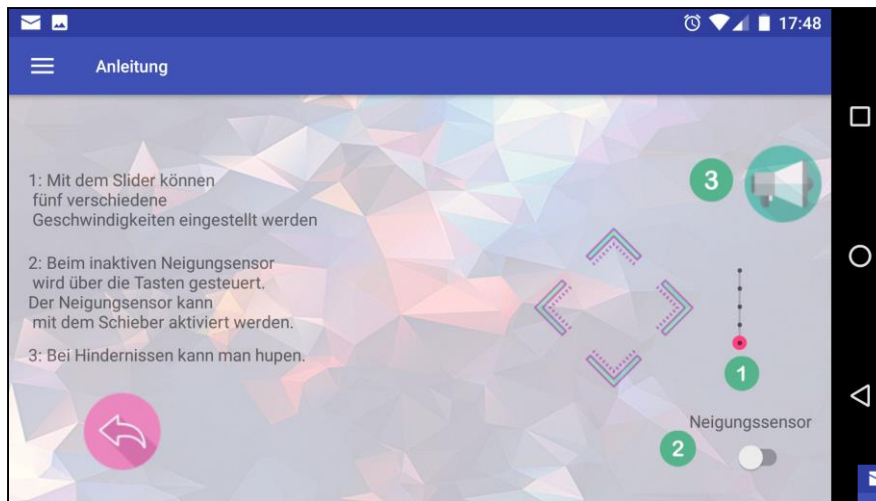
Steuerung



- Tastensteuerung
- Intensität mit Stufenverstellung
- Neigungssensor

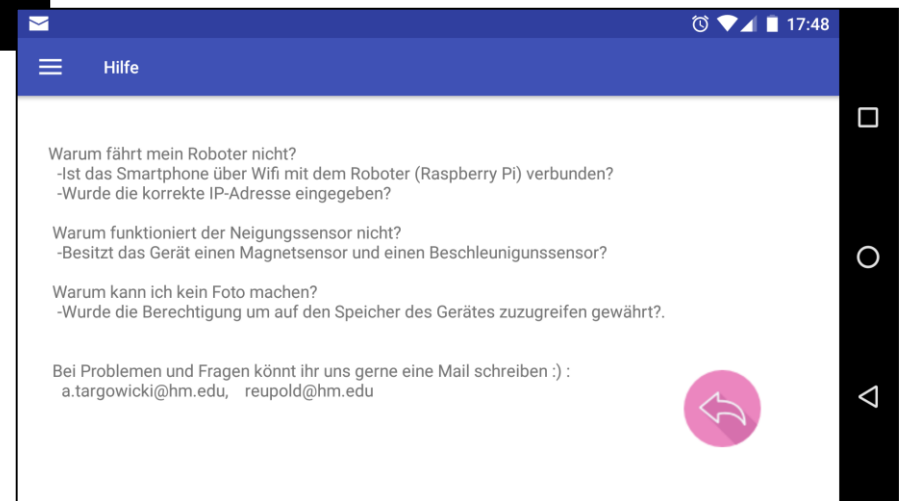


Anleitung und Hilfe



Anleitung: Übersicht der Hauptoberfläche

Hilfe: Tipps und Kontaktdaten





Testing

- Login mit Espresso getestet
- Steuerung, Dataframe, UDP-Sender mit Espresso getestet
- Dataframe Unit-Test mit Tastensteuerung
- Dataframe Unit-Test mit Neigungssensor
- Ausführliche Funktionalitätstest für Video, Telemetrie, Empfangsserver, Backstates, Hupe, Anleitung Hilfe, Navigation View



Kamera *Live-Stream*



Stream erzeugen

```
#first install motion
sudo apt-get install motion

# configure motion.conf
sudo nano /etc/motion/motion.conf

# change the following entries:
daemon on (für daemon, bei uns off)
output_pictures off (default on )
stream_localhost off
stream_maxrate 100
framerate 30
width 320
height 480
#sdl_threadnr 0

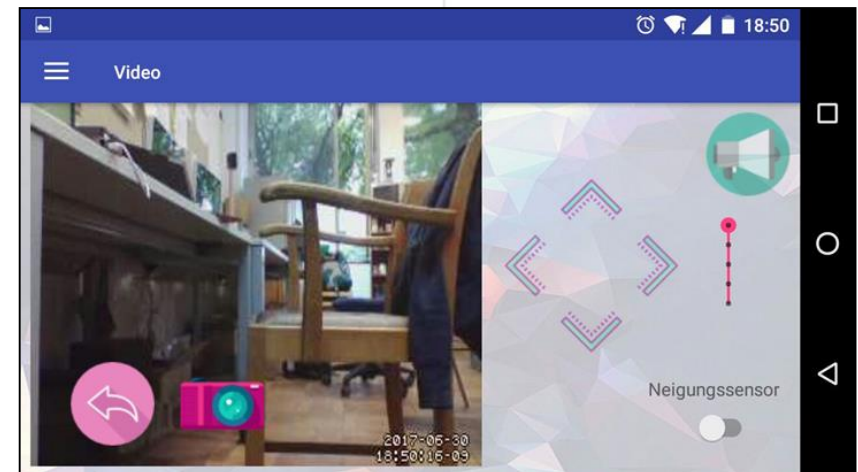
#save it

#start
sudo motion start
```

Kamera



```
private void setWebView() {  
    view.setBackgroundColor(Color.TRANSPARENT);  
    String htmlPart1 = "<html><body><img src=\"http://\"";  
    String htmlPart3 = "/\" alt=\"\\n\\n\\n\\n\"Not able to connect.\\nMaybe Robot is offline?\" width=\"\"";  
    String htmlPart5 = "\" height=\"\"";  
    String htmlPart7 = "\"/> </body></html>\"";  
    String mime = "text/html";  
    String encoding = "UTF-8";  
    String port = ":8081";  
  
    view.setWebViewClient(new WebViewClient());  
    String width = "95%";  
    String height = "95%"; // no scrolling in view  
  
    String html = htmlPart1+message+port+htmlPart3+width+htmlPart5+height+htmlPart7;  
    view.loadDataWithBaseURL(null,html,mime,encoding,null);  
}
```





Berechtigung für ein Foto

```
/**
 * checks necessity of asking for permission on Runtime
 * @return boolean of necessity of asking for permission
 */
private boolean checkPermissionNecessity(){

    return(Build.VERSION.SDK_INT>Build.VERSION_CODES.LOLLIPOP_MR1);

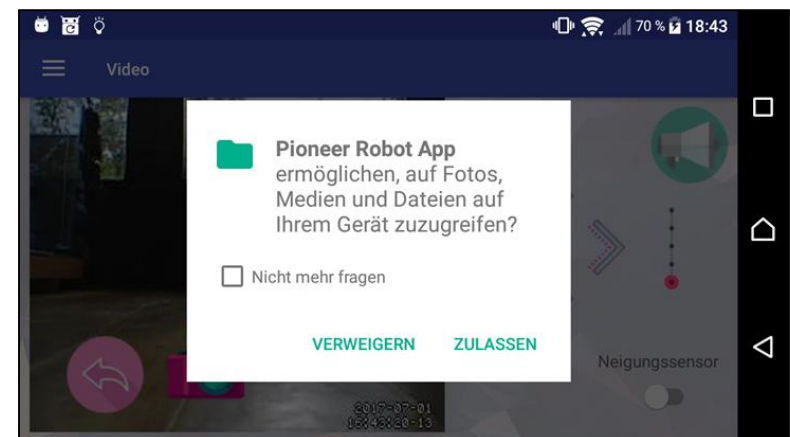
}

/**
 * Get answer of permission to read and write storage
 * @param permsRequestCode Code to ask for permission
 * @param permissions Array of permissions to ask
 * @param grantResults Result of the questions for permission
 */
@Override
public void onRequestPermissionsResult(int permsRequestCode, String[] permissions, int[] grantResults){

    if (permsRequestCode == 200){
        readAccepted = grantResults[0]==PackageManager.PERMISSION_GRANTED;

        writeAccepted = grantResults[1]== PackageManager.PERMISSION_GRANTED;

        if (!readAccepted || !writeAccepted) {
            Toast.makeText(getContext(),"Zugriff verweigert. Ich darf leider kein Foto machen!", Toast.LENGTH_LONG).show();
        }
    }
}
```



Foto

```
@Override
public void run() {
    readAccepted = true;
    writeAccepted = true;

    Picture picture = view.capturePicture();
    Bitmap b = Bitmap.createBitmap(picture.getWidth(),
        picture.getHeight(), Bitmap.Config.ARGB_8888);
    Canvas c = new Canvas(b);

    picture.draw(c);
    Calendar cal = Calendar.getInstance();
    int year = cal.get(Calendar.YEAR);
    int month = cal.get(Calendar.MONTH) + 1; // starts at 0
    int day = cal.get(Calendar.DAY_OF_MONTH);
    int hour = cal.get(Calendar.HOUR_OF_DAY);
    int minute = cal.get(Calendar.MINUTE);
    int second = cal.get(Calendar.SECOND);

    boolean necessity = checkPermissionNecessity();

    if (necessity) {
        // ask for permission
        String[] perms = {"android.permission.READ_EXTERNAL_STORAGE", "android.permission.WRITE_EXTERNAL_STORAGE"};

        int permsrequestCode = 200;

        requestPermissions(perms, permsrequestCode);
    }
    if (readAccepted && writeAccepted) {
        FileOutputStream fos = null;
        try {
            String file = "robot-" + year + "-" + month + "-" + day
                + "-" + hour + "-" + minute + "-" + second + ".jpg";

            fos = new FileOutputStream("mnt/sdcard/Pictures/" + file);
            if (fos != null) {
                b.compress(Bitmap.CompressFormat.JPEG, 100, fos);
                fos.close();
                Toast.makeText(getContext(), file + " saved", Toast.LENGTH_LONG).show();
            }
        } catch (Exception e) {
            //Real message ("Permission denied").
            //Toast.makeText(getContext(), e.getLocalizedMessage(), Toast.LENGTH_LONG).show();
        }
    }
}
```





Features

Sensorik und Design

NFC

- Login
- Inhalt: IP-Adresse
- Tag am Roboter



NFC Tag



Ohne NFC



Mit NFC

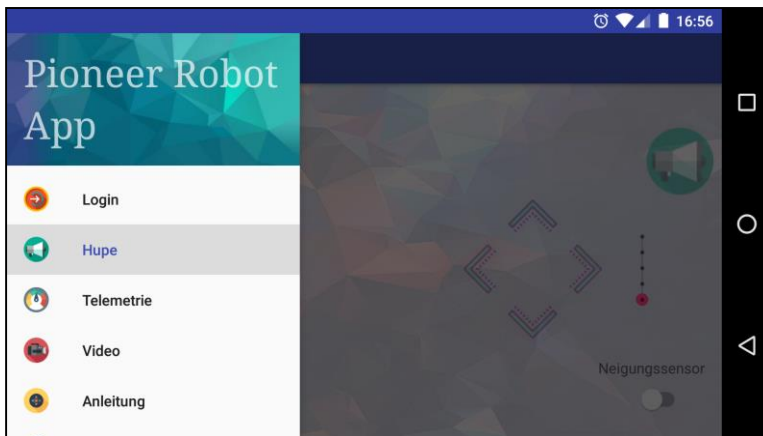
Sound



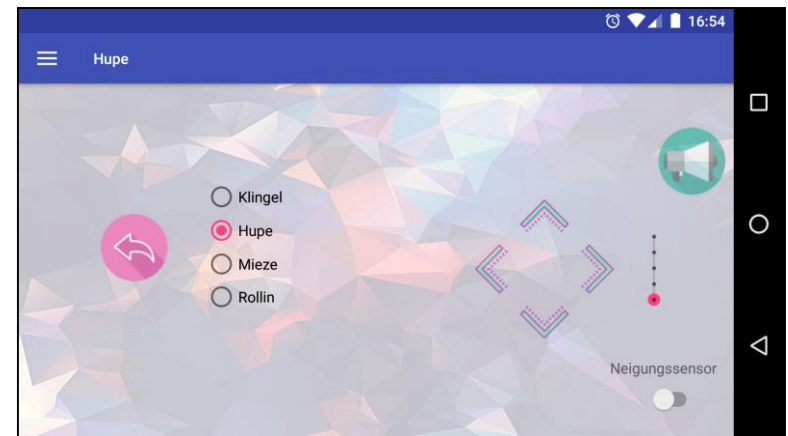
- Einsatz als Hupe
- Lautsprecher beim Roboter + UDP Server (Python)
- Verschiedene Töne wählbar



Symbol



Menü



Design



- Icon
- Geometrie:
Orientierung an
Desktop Umgebung KDE Plasma

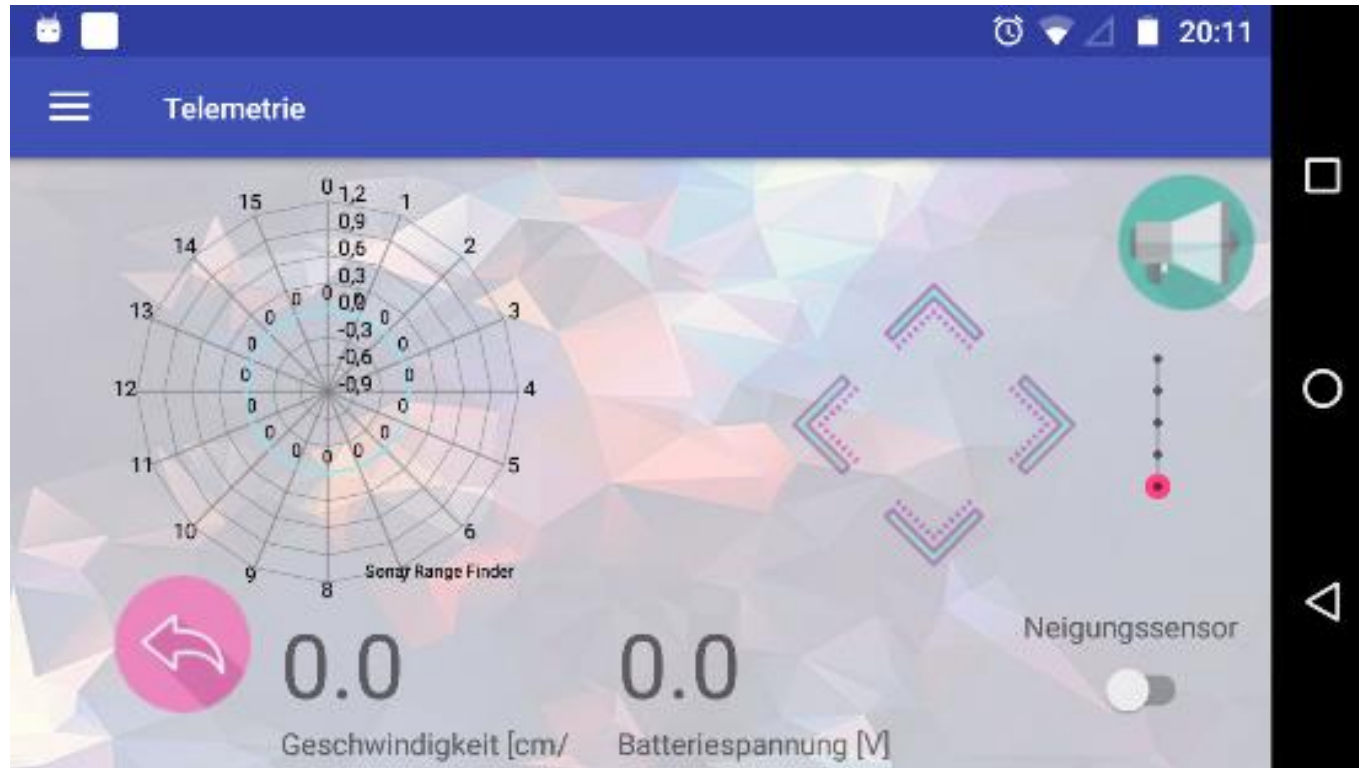


KDE Plasma 5.3

- Material Design von Google
 - Minimalismus
 - Schichten, sowie Licht und Schatten
 - Color Tool für Farben, z.B. Texte, verwendet



Telemetrie Anzeige



Telemetrie



Serverseitig:

- Daten werden auf dem UDP Server via ARIA gesammelt
- JSON-Encoded String wird zum Smartphone geschickt
 - Battery Spannung
 - Roboter Geschwindigkeit
 - Sonar Rangefinder Readings
- Als UDP Paket verschickt, an Port 8845

Telemetrie



Clientseitig:

- HandlerThread im Client wartet auf Paket an Port 8845
- JSON wird ausgewertet und in eine TelemetryMessage verpackt
- EventBus um TelemetryMessage an Fragment zu schicken
- Dort ist eine subscriber Methode angelegt, die Werte darstellt
- Sonar Ranges werden mit hilfe einer externer Bibliothek dargestellt



Server *Roboter*

UDP Control Server



- Umgesetzt:
 - UDP Listener auf Port 8844 für Control Messages
 - Kollisionskontrolle mit SafetyLimits basierend auf Sonar Ranges
 - Telemetry Sender und Datensammler
 - ARIA Bibliothek um Roboter anzusprechen (über USB)



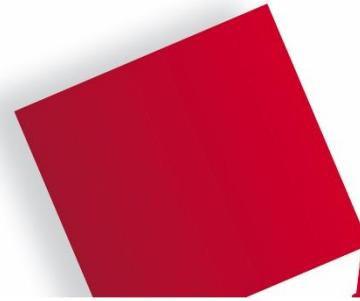
Zusammenfassung

Pioneer Roboter App



Zusammenfassung

- Visuell schön gestaltete, einsatzfähige App
- Steuerung des Pioneer Roboters mit Tasten, oder mit Hilfe des Neigungssensors
- Erleichterte Anmeldung dank NFC
- Akustische Warnung bzw. Sound
- Live-Bildübertragung aus Sicht des Roboters
- Telemetrie Daten
- Anleitung und Hilfe mit Kontaktdaten



HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN · FH
MÜNCHEN

Vielen Dank für eure Aufmerksamkeit!
Live-Demo im Anschluss

