

Dokumentation AirSwimmer

1	Anforderungen	2
2	Vorbereitung	2
2.1	Versionsverwaltung	2
2.2	Android-Tutorium	7
2.3	Zeiterfassung.....	7
3	Alternativen.....	8
3.1	Alternativen ohne Infrarot.....	8
3.2	Simulation/Emulation	10
3.2.1	Raspberry Pi.....	10
4	Projektleitung	12
5	Empfangen	12
5.1	Analyse der Übertragungsstrategie	12
5.2	Aufnehmen des eigenen Infrarotsignals.....	13
5.2.1	Versuch 1: Untersuchen eines Infrarot Signals mittels Oszilloskop.....	13
5.2.2	Versuch 2: Aufnahme des Signals mit dem Raspberry Pi	14
5.2.3	Versuch 3: Aufzeichnung von Infrarotsignalen über die Soundkarte	15
5.2.4	Versuch 4: Aufzeichnung von Infrarotsignalen über die serielle Schnittstelle des Computers (RS232).....	15
5.3	Lirc-File	15
5.3.1	Analyse durch WinLirc	15
5.3.2	Analyse und Verbesserung der Lirc Files	16
6	Senden	22
6.1	Irdroid-App.....	22
6.2	Senden	22
7	Oberfläche	22
7.1	Konzept	22
7.2	Startseite	23
7.3	Steuerung mittels Buttons	25
7.3.1	Variante A (erste Testversion)	26
7.3.2	Variante B (ImageButtons)	27
7.3.3	Reaktion auf Druck und visueller Effekt	29
7.4	Steuerung durch Wischen.....	31
7.5	Steuerung durch Kippen	34
7.6	Lautstärkenkalibrierung	34
7.6.1	Oberfläche	34
7.6.2	Logik.....	37
7.6.3	Einbau und permanentes Speichern	37
7.7	Menü	37
8	Verbindung von Oberfläche und Senden	42
8.1	Permanente Bewegung.....	42
9	Graphische Gestaltung der Oberfläche	44
9.1	Konzept	44
9.2	Launcher-Icon	48
9.3	Animierter Hintergrund	49
9.4	Hintergrundbilder	49
9.5	Graphiken.....	49
9.5.1	Startseite (Buttons)	49
9.5.2	Steuerungsseiten (Buttons und Hai).....	50
10	Gehäuse	53

10.1	Recherche Gehäusemöglichkeiten	53
10.2	Gehäuse bemalen und verschönern.....	53
11	Sonstiges	53
11.1	Linux Library.....	53

1 Anforderungen

2 Vorbereitung

2.1 Versionsverwaltung

Von Andreas Gerken

Bei der Softwareentwicklung (vor allem bei größeren Projekten im Team) lohnt es sich auf jeden Fall ein Versionsverwaltungsprogramm einzurichten. Durch die Versionsverwaltung können Änderungen nachvollzogen werden (wann hat wer welche Änderung vorgenommen) und alte Versionen geladen werden. So lässt sich, wenn ein Fehler vorkommt, schnell nachvollziehen seit welcher Version der Fehler im Programm ist und wer ihn eingebracht hat. Im Normalfall kann der „Verursacher“ den Fehler am schnellsten beheben. Ein weiterer großer Vorteil von Versionsverwaltung ist, dass alle Teammitglieder gleichzeitig entwickeln können und ihre Änderungen immer wieder zusammenführen. Das Programm zur Verwaltung achtet dabei darauf, dass keine Änderung verloren geht und weist die Benutzer auf eventuelle Konflikte (wenn mehrere Nutzer eine Datei geändert haben) hin.

Es gibt viele verschiedene Versionsverwaltungsprogramme die sich in Bedienung und Funktionsweise zum Teil stark unterscheiden. Die meisten Systeme speichern alte Versionen auf einem zentralen Server ab und die Benutzer können auf diese über spezielle Programme direkt abrufen. Dabei unterscheiden sie sich insofern, dass der Nutzer bei manchen Systemen bei jeder Verwendung auf den Server zugreifen muss und bei Anderen immer alle alten Versionen in Form eines Repositorys herunterladen kann und auch ohne Internet Verbindung Zugriff auf alle Daten hat. Der Vorteil der „Online-Systeme“ ist, dass immer sichergestellt ist, dass der Benutzer immer Zugriff auf die neusten Versionen hat. Der Vorteil der Systeme die Repositorys herunterlädt ist, dass der Server keinen Dienst anbieten muss sondern nur einen Webspace auf den man von Überall zugreifen kann. Außerdem ist man unabhängig vom Server und durch die lokalen Kopien auf allen Rechnern können sogar alle Daten auf dem Server verloren gehen, ohne dass der komplette Projektfortschritt verloren ist.

Wir haben uns für ein System namens „GIT“ entschieden, das keine dauerhafte Internetverbindung benötigt, sondern immer die komplette Historie von einem Webspace lädt und lokal auf jedem Entwicklerrechner speichert. Dadurch benötigen wir keinen teuren Server auf dem ein Programm laufen muss sondern können einen kostenlosen Dienst namens „GIT Hub“ verwenden. Um ihn nutzen zu können, muss man sich nur registrieren und ein Repository auf dem Server anlegen. Man bekommt eine URL, von der man sich das komplette Repository herunterladen kann. Auf der Homepage von Git Hub kann man sich außerdem einloggen und Änderungen über eine

Weboberfläche nachvollziehen, ohne ein Programm auf dem PC installiert haben zu müssen.

Zum verwalten des Repositorys gibt die GIT Software die man kostenlos herunterladen kann. Sie bietet zwar alle nötigen Funktionen an, ist jedoch Kommandozeilenbasiert. Somit müsste jedes Teammitglied erst die Befehle lernen und der Aufwand wäre recht groß. Deswegen gibt es mehrere GUI basierte Programme die GIT steuern. Wir haben uns für Tortoise GIT entschieden, das direkt in das Windows Filesystem eingebunden ist. So bekommt man alle Funktionen über einen Rechtsklick auf Ordner über ein Kontextmenü zur Verfügung gestellt. Die Verwendung von GIT ist dadurch sehr intuitiv und kann schnell erlernt werden.

Das GIT System beruht auf dem runterladen, verändern und wieder hochladen des Repositorys. Die beiden Operationen dazu heißen Pull (runterladen) und Push (hochladen) wobei beim herunterladen Änderungen die auf dem Server gespeichert sind mit denen im lokalen Repository verschmolzen werden (merge).

Um das Repository das erste Mal herunter zu laden, muss die Funktion clone aufgerufen werden. Hier kann man die URL des Repositorys auf dem GitHub-Server eingegeben werden. Für unser Projekt wurden zwei Repositorys eingerichtet. Ein Produktives namens „AirSwimmer“ und Eins das zum Testen der GIT Funktionen gedacht ist („TestRepository“). Somit kann Jeder alle Funktionen testen ohne Angst haben zu müssen etwas falsch zu machen.

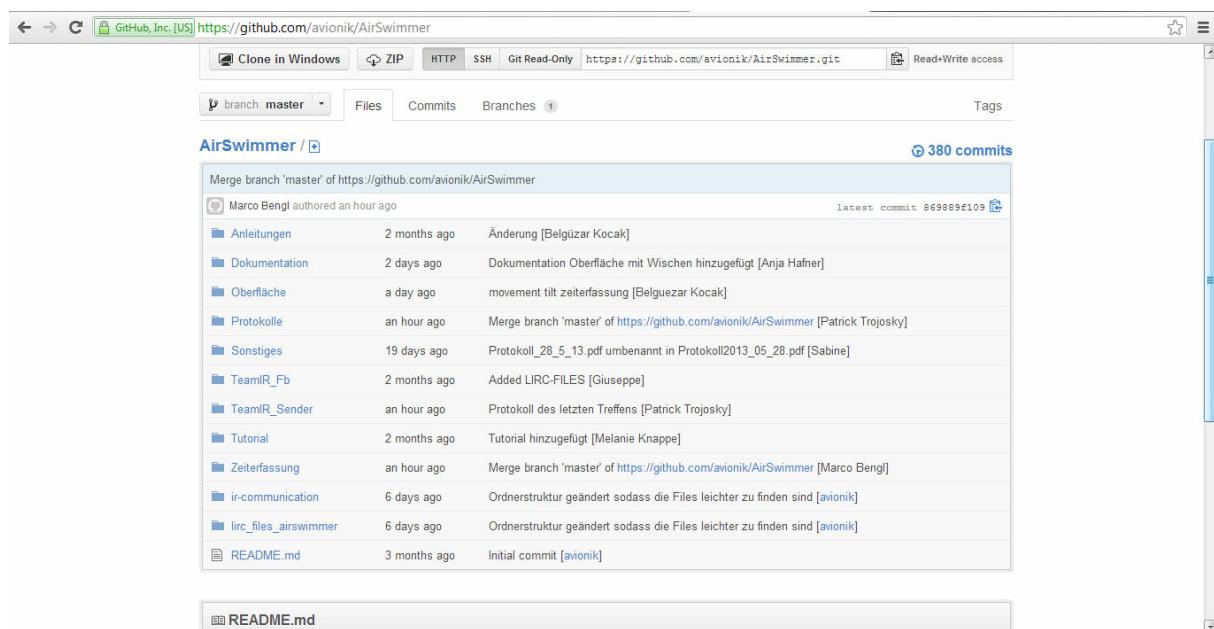


Abbildung 1
Weboberfläche von GitHub

Hier sieht man direkt auf den ersten Blick, wer in welchem Bereich aktuell entwickelt. Die Messages der neusten Änderung in einem Ordner werden mit dem Namen des Benutzers der die Änderung getätigt hat angezeigt.

Bibliothek "Dokumente"

Oberfläche

Name	Änderungsdatum	Typ	Größe
✓ Oberflächen_Entwurf	05.05.2013 15:01	Dateiordner	
✓ src	29.05.2013 21:38	Dateiordner	
✓ Beschreibung_einzelneOberflaechen.doc	04.06.2013 12:07	Microsoft Office ...	25 KB
✓ Neues Textdokument.txt	17.06.2013 19:10	TXT-Datei	0 KB
✓ ZurInfo.txt	17.06.2013 19:09	TXT-Datei	1 KB

Abbildung 2

Einbindung von GIT in das Windows Dateisystem durch TortoiseGit

Dateien und Ordner mit grünem Haken sind mit der aktuellsten Version im Lokalen Repository identisch. Dateien mit rotem Ausrufezeichen wurden geändert und müssen noch in das Repository „committed“ werden. Dateien ohne Symbol wurden neu angelegt und noch nicht ins Repository hinzugefügt.

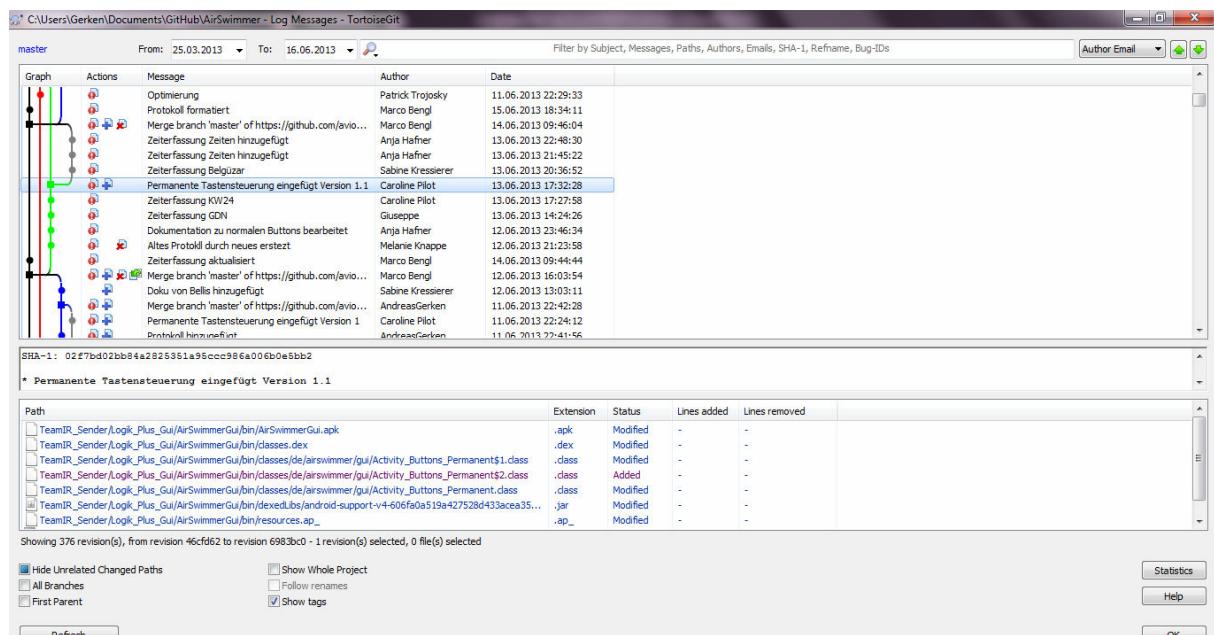


Abbildung 3

Exemplarische Änderungshistorie mit der Möglichkeit genau zu sehen, was bei einer Version geändert wurde.

Anleitung zur Benutzung des Git Systems

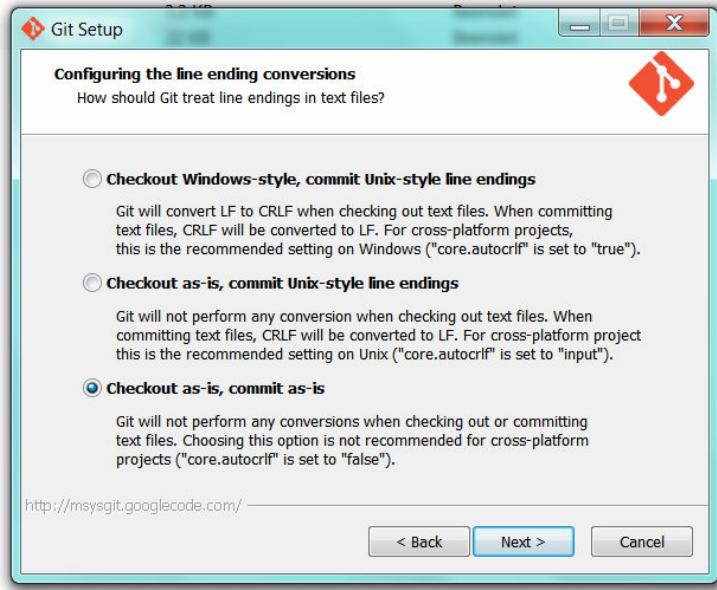
Von Andreas Gerken, Andreas Weber, Caroline Pilot

1. Schritt

Git runterladen und installieren:

<http://git-scm.com/>

Hierbei ist darauf zu achten, dass beim Installationsprozess folgendes eingestellt ist:



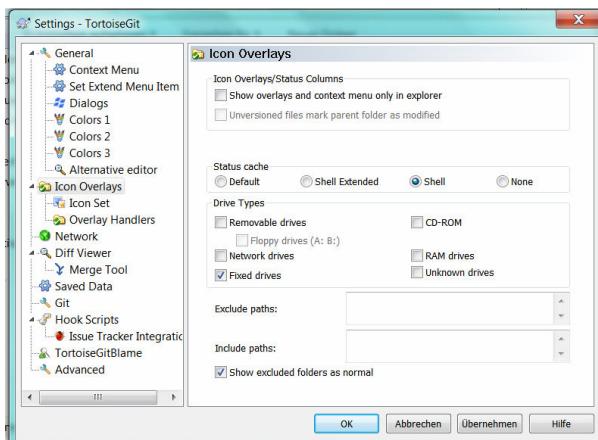
TortoiseGit runterladen und installieren:

<https://code.google.com/p/tortoisegit/wiki/Download>

2. Schritt

Git einrichten

- Ordner auf eigenem Rechner anlegen (in dem die Dateien des Projekts gespeichert werden sollen) z.B. Avionik Projekt (nicht AirSwimmer!)
- Rechter Mausklick im leeren Ordner → „TortoiseGit“ → „Settings“
 - → „Git“, Name und E-Mail einfügen
 - → „IconOverlays“, Einstellungen siehe Bild



3. Schritt

Klonen der Server-Ordner auf den Rechner

- Rechter Mausklick im Ordner → „Git Clone“
- Folgende URL ist einzugeben
 - <https://github.com/avionik/TestRepository.git>

- Dieser Ordner ist zum Ausprobieren gedacht‘
- Letzte zwei Punkte sind zu wiederholen, diesmal mit folgender URL
 - <https://github.com/avionik/AirSwimmer.git>
 - Dieser Ordner ist für die richtigen Dateien des AirSwimmers gedacht

4. Schritt

Arbeiten mit dem GitHub Server

Grundbegriffe und Funktionen:

- Pull
 - Lädt die neuste Version vom Server herunter und verschmilzt (Merge) diese mit der lokalen Version
 - Merge kann evtl. zu Problemen führen, da dieser nicht trivial ist, man muss die zu übernehmende Version manuell über den Menüpunkt „Conflict“ auswählen
- Push
 - Lädt die lokal gespeicherte Version auf den Server
- Commit
 - Speichert die Änderung als neue Version auf dem lokalen Repository

Vor Änderungen ist es sinnvoll erst die neuste Version herunterzuladen um Konflikte zu vermeiden.

Änderungen ins lokale Repository speichern

- Änderungen auf Datei ausführen
- Rechter Mausklick auf geänderte Datei → „Git commit -> master“
- Message hinzufügen (ausdrucksstarke Info!)
- Bestätigen

Lokales Repository auf den Server Laden

- „Git commit -> master“ falls noch nicht geschehen
- Im Order, in dem geänderte Datei liegt rechter Mausklick
- Git Sync
- Pull
- Push (Eingabe von Nutzernamen und Passwort erforderlich)

Wenn das Pull dabei nicht durchgeführt wird, kann beim Push die Meldung kommen, dass das lokale Repository gegenüber dem aktuellen Repository (auf dem Server) zurück liegt. Da die Push Operation dann nicht durchgeführt werden kann, ist es also sehr wichtig den Pull Schritt nicht zu vergessen.

2.2 Android-Tutorium

2.3 Zeiterfassung

Von Sabine Kressierer

Ziel

Ziel was es, ein Tool zu finden, dass es ermöglicht die gearbeiteten Zeiten aller Teammitglieder zu erfassen und übersichtlich darzustellen. Dazu sollte jeder seine Zeiten, und die dabei erledigten Arbeiten, selbst eintragen können.

Ergebnis

Ein Großteil der im Internet vorhandenen Zeiterfassungstools ist für mehr als einen Nutzer kostenpflichtig. Zudem handelt es sich dort meist um komplettete Projektmanagement-Tools, bei denen der Aufwand zur Einrichtung und Konfiguration des Tools sowie die Einarbeitungszeit zu hoch wäre, um eine einfache Zeiterfassung für 12 Personen zu erhalten. Aus diesem Grund wird stattdessen eine Excel-Tabelle, die Andreas Weber zur Verfügung stellte, verwendet. Dort ist es möglich für die einzelnen Tage, oder ganze Wochen, gearbeitete Stunden oder Uhrzeiten einzutragen, aus denen die Arbeitszeit einer Woche sowie über die gesamte Projektphase berechnet wird. Diese Datei wurde auf den Projektzeitraum angepasst und für jedes Teammitglied dupliziert. Zudem wurde eine Übersicht erstellt, die automatisch die Gesamtstunden der einzelnen Teammitglieder, sowie die einzelnen Wochenstunden anzeigt. Die Gesamtstunden werden zudem in einem Balken-Diagramm graphisch dargestellt(Abbildung 4).

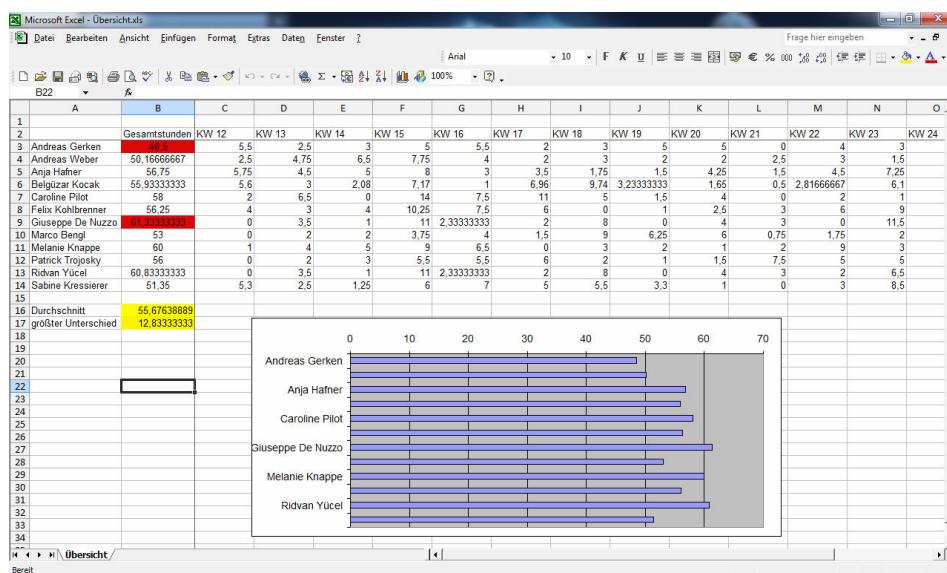


Abbildung 4

Übersicht Zeiterfassung

So ist ein schneller Vergleich der verschiedenen Zeiten möglich, die später bei der Verteilung der neuen Arbeitspakete berücksichtigt werden. Um die Zeiten nachvollziehen zu können, wird zu der entsprechenden Zeitangabe eingetragen, was bearbeitet wurde. Zu Beginn befand sich die gesamte Zeiterfassung in einer einzelnen

Excel-Datei. Da das Versionsverwaltungstool GIT jedoch keine Excel-Dateien mergen kann, führte dies immer wieder zu Konflikten. Aus diesem Grund sind die Zeiterfassungstabellen der einzelnen Teammitglieder nun in extra Dateien, die sich zusammen mit der Übersicht in einem Ordner befinden.

Zeiterfassung			
Name	Änderungsdatum	Typ	Größe
Andreas Gerken.xls	20.06.2013 20:33	Microsoft Excel-Ar...	42 KB
Andreas Weber.xls	20.06.2013 23:00	Microsoft Excel-Ar...	47 KB
Anja Hafner.xls	20.06.2013 20:33	Microsoft Excel-Ar...	40 KB
Belguezar Kocak.xls	20.06.2013 20:33	Microsoft Excel-Ar...	46 KB
Caroline Pilot.xls	20.06.2013 20:33	Microsoft Excel-Ar...	54 KB
Felix Kohlbrenner.xls	20.06.2013 20:33	Microsoft Excel-Ar...	45 KB
Giuseppe De Nuzzo.xls	20.06.2013 20:33	Microsoft Excel-Ar...	51 KB
Marco Bengli.xls	20.06.2013 20:33	Microsoft Excel-Ar...	49 KB
Melanie Knappe.xls	20.06.2013 23:00	Microsoft Excel-Ar...	48 KB
Patrick Trojosky.xls	20.06.2013 20:33	Microsoft Excel-Ar...	47 KB
Ridvan Yuecel.xls	20.06.2013 20:33	Microsoft Excel-Ar...	44 KB
Sabine Kressierer.xls	20.06.2013 20:33	Microsoft Excel-Ar...	45 KB
Übersicht.xls	20.06.2013 20:33	Microsoft Excel-Ar...	40 KB

Abbildung 5
Ordnerstruktur Zeiterfassung

Da so jeder eine eigene Datei zum Ändern hat, können keine Versionskonflikte mehr auftreten. Die Übersicht ist weiterhin mit den einzelnen Dateien verlinkt. Da in Excel alle Pfade relativ gespeichert werden, können die verlinkten Felder auch aktualisiert werden, wenn der gesamte Ordner verschoben wurde. Da sich jedoch alle Dateien in einem Ordner befinden müssen ist es so nicht mehr möglich den Stand direkt auf github.com zu aktualisieren. Dort wird nur der zuletzt gepuschte Stand angezeigt.

3 Alternativen

3.1 Alternativen ohne Infrarot

von Anja Hafner

Für den Fall, dass die Ansteuerung des AirSwimmers mit Infrarot nicht realisiert werden kann, wird nach alternativen Geräten gesucht.

Diese Geräte sollten mittels Bluetooth ansteuerbar sein, nicht mehr als 100 € kosten und fliegen können.

Warum Bluetooth?

Die Steuerung per Bluetooth ist gewünscht, da generell alle aktuellen Tablets und Smartphones Bluetooth besitzen und keine zusätzliche Hardware (wie bei der Steuerung mit Infrarot) nötig ist. Außerdem ist die Umsetzung dieser Kommunikation mit dem Fisch deutlich einfacher als mit Infrarot, da die Kommunikation mit Bluetooth besser dokumentiert ist.

Warum soll es fliegen?

Da es sich um ein Avionik-Projekt handelt, sollte der Gedanke, etwas flugfähiges zu steuern, nicht verloren gehen.

Suche nach Alternativen

Zu Beginn wird nach einem AirSwimmer gesucht, der mit Bluetooth gesteuert werden kann. Jedoch gibt es keinen alternativen AirSwimmer, der mit Bluetooth gesteuert wird, lediglich ferngesteuerte AirSwimmer werden angeboten.

Aus diesem Grund wird die Suche auf alle flugfähigen, mit Bluetooth und Android gesteuerten Geräte, erweitert.



Abbildung 6

Auch hier ist die Auswahl nicht groß, lediglich eine flugfähige Alternative der Firma BeeWi wird angeboten. Dieser Hubschrauber ist per Bluetooth 3.0 ansteuerbar und bietet bis zu 8 Minuten Flugzeit. Die Ladezeit hingegen beläuft sich auf ca. 40 Minuten und erfolgt per USB. Ein weiterer Nachteil ist die schwere Steuerbarkeit des Hubschraubers. Die Kosten des BeeWi BB/301-A0 Bluetooth Controlled Hubschraubers in schwarz belaufen sich auf 50,37 €.

Fahrfähige Alternativen

Da nur eine flugfähige Alternative mit Bluetooth gefunden wurde, wird ebenfalls nach Bluetooth-gesteuerten Fahrzeugen gesucht, auch wenn dies nicht ganz den Anforderungen entspricht. Der Vorteil von Fahrzeugen ist die einfachere Steuerbarkeit. Das erste Modell stammt ebenfalls von der Firma BeeWi und ist ein Mini Cooper. Dieses Modellauto wird mit Bluetooth 2.0 gesteuert und hat eine Reichweite von bis zu 10 Metern. Für den Betrieb sind 3 AA Batterien erforderlich und die Kosten betragen 30,54 € bei Amazon.



Abbildung 7

Eine weitere Alternative sind die so genannten Tankbots. Die kleinen Fahrzeuge werden per USB aufgeladen und besitzen 3 bereits vorhandene Fahrmodi: sie können Hindernisse selbstständig ausweichen, in freien Bewegungen "tanzen" oder per Smartphone gesteuert werden. Die Kosten belaufen sich auf 24,99 € bei Amazon.



Abbildung 8

Für den Fall, dass die Ansteuerung per Infrarot nicht realisiert werden kann, wurde der Hubschrauber als mögliche Alternative ausgewählt.

3.2 Simulation/Emulation

3.2.1 Raspberry Pi

Von Andreas Gerken

Zur Emulation der Funktionalität des Air Swimmers ist eine Möglichkeit einen Raspberry Pi (RPi) zu verwenden. Das ist ein kleiner, reduzierter aber auch vollwertiger Computer, der dazu gedacht ist, dass man auf ihm Sachen ausprobieren kann ohne Angst haben zu müssen, dass man etwas kaputt macht. Auf dem Rechner läuft eine

spezielle Linux Variante (Raspbian; abgewandelt von Debian) die auf die Leistungsmerkmale des Rechners zugeschnitten ist. Auf der Platine ist ein 700MHz ARM-Prozessor und 512MB Arbeitsspeicher verbaut. Die Platine verfügt über zwei USB Anschlüsse, einen HDMI ausgang, einen Ethernetsteckplatz und weitere Verbindungen. Vorallem aber einen „GPIO“ = General Purpose Input Output Steckplatz mit insgesamt 26 Pins über die Peripherie wie Sensoren, Servos, Motoren oder Ähnliches angeschlossen werden können. Somit könnte man über diesen Steckplatz auch einen Infrarot Empfänger anschließen und LEDs oder Motoren. Somit könnte man die Aktivitäten des Fisches Emulieren. Der RPi könnte auf die gleichen Signale reagieren, die im LIRC File definiert sind und auf die Signale mit leuchten der LED/ drehen der Motoren reagieren. Damit könnte auch bei Ausfallen des Fisches der Erfolg des Projekts demonstrativ gezeigt werden.



Abbildung 9

Die Platine des Raspberry Pi's die 26 Pins auf der linken Seite der Abbildung sind die GPIO Pins

Da das Betriebssystem Raspbian nur eine reduzierte Linux Variante ist, unterstützt es leider nicht wie die meisten Linux Distributionen. Das Programm kann auch nicht aus einem Repository nachgeladen werden sondern muss in den Linux Kernel eingebaut werden und das komplette Betriebssystem muss neu kompiliert werden. Dazu gibt es eine umfangreiche Anleitung im Internet, die Schritt für Schritt durchgeführt wurde. Die Lirc Komponente konnte zwar eingebaut werden, jedoch dauerte der Kompilierungsvorgang sehr lange (da das gesamte Betriebssystem gebaut werden muss). Dadurch ist die Fehlersuche sehr langwierig und es ist sinnvoller den Arduino als Emulationsplatine zu verwenden.

4 Projektleitung

5 Empfangen

5.1 Analyse der Übertragungsstrategie

Von Andreas Weber

Signalaufbau

IR-Fernbedienungen senden ein Signal im unsichtbaren Infrarotbereich aus. Als Strahlungsquelle dienen häufig Infrarotleuchtdioden. Das Signal wird mit einer Frequenz um 40 kHz aus- und eingeschaltet. Dadurch erhöht sich die Störsicherheit des Empfängers: Ein Bandpassfilter lässt nur diese Frequenzen passieren und sperrt zufällige Störsignale aus. Durch Modulation dieses Sendesignals werden Informationen zum Empfänger übertragen.

Abbildung kommt noch

Codierverfahren der IR Übertragung

Es gibt wohl nahezu so viele Codierverfahren, wie Hersteller von IR Geräten, leider kein einheitlicher Standard.

x-Sat

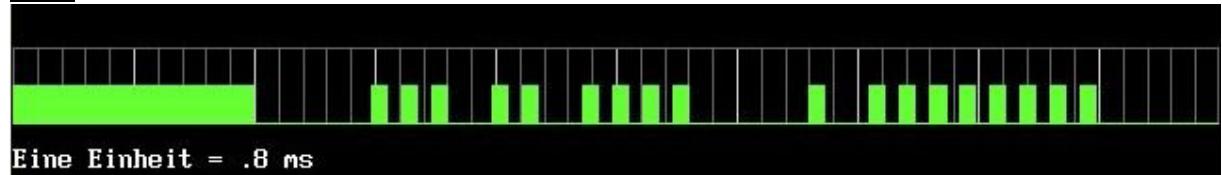
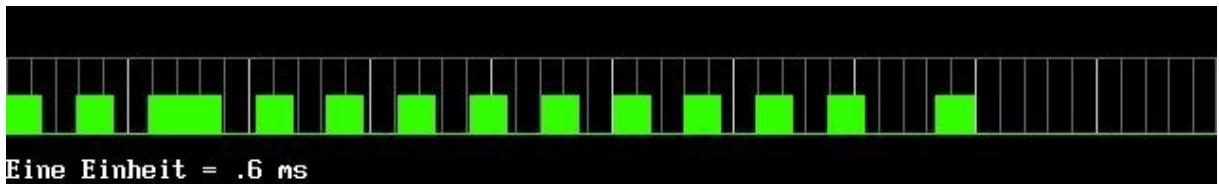


Abbildung 10

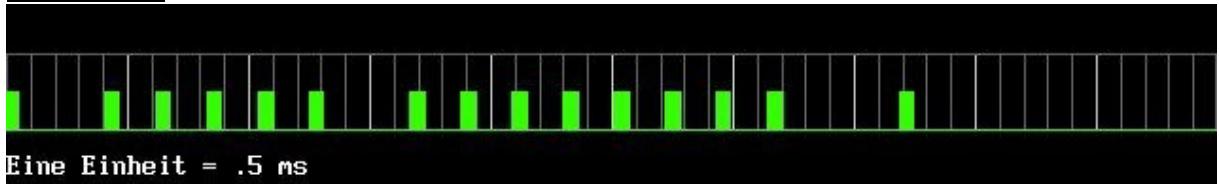
Gedrückt wurde die Taste "1" fürs erste Programm. Man kann sehr schön ablesen, dass zu Beginn ein 8ms langer Impuls gesendet wird, gefolgt von einer 4ms-Lücke. Das ist eindeutig das auf der oben angesprochenen Seite als "X-Sat" bezeichnete Protokoll. Dem Startsignal folgen zwei Gruppen kürzerer Impulse, wieder getrennt durch eine 4ms-Pause. Bei diesem Protokoll sind jeweils die ersten acht dieser Kurzen Impulse das Byte, das den Gerätecode enthält, der Letzte ist sozusagen ein Stopimpuls. Ein Impuls mit kurzer Pause danach ist eine Null, einer mit langer Pause eine Eins, gelesen wird von links nach rechts. Der Gerätecode ist also 20. Das zweite Byte ist 1, was der Code für Taste 1 ist. Auch hier ist der 9. Impuls der Stopimpuls.

Viele Protokolle arbeiten mit dieser Aufteilung in Gerät- und Funktionscode. So stören sich die Fernbedienungen z.B. von Fernseher und Videorecorder des gleichen Herstellers nicht, auch wenn sie ihre Daten nach dem gleichen Verfahren übertragen. Hier noch zwei weitere Beispiele, bei beiden wurde jeweils die Taste "1" gedrückt:

RC-5 Code:



Sharp Code:



Quelle: Skilltronics.de http://www.skilltronics.de/versuch/elektronik_pc/ir.html

Bei dem von unserem AirSwimmer Fernbedienung verwendeten Codierung handelt es sich jedoch um ein Unbekanntes Codier verfahren, über das keine weiteren Informationen gefunden wurden. Bei diesem Codier verfahren werden die Informationen rein über die Länge des „0“ Impulses Codiert, es gab keine Messbare Startsequenz.

5.2 Aufnehmen des eigenen Infrarotsignals

Von Andreas Weber

Im Internet wurde nach einer Möglichkeit gesucht, das eigene Infrarotsignal aufzunehmen. Dazu wurden mehrere Möglichkeiten gefunden. Für Apple Geräte gibt es einen fertigen Aufsatz mit dem unter anderem auch die Hochschule Rosenheim bereits den AirSwimmer gesteuert hat und dazu eine passende app geschrieben hat. Diese Version ist sehr einfach, da das Gerät und die beiliegende Software alles selbst übernimmt, es ist für die Kommunikation kein weiterer Programmieraufwand nötig.

Im Internet gibt es weitere ausgereifte Geräte, um infrarot Signale aufzunehmen. Leider sind diese recht teuer und deshalb wurde eine günstigere Variante gesucht.

Bei der Suche sind wir auf den so genannten TSOP Empfänger gestoßen. Dieser wird an 5V Gleichspannung angeschlossen und wandelt das empfangene Lichtsignal in eine digitale Spannung um. Außerdem filtert er bereits die Grundfrequenz von 40HZ heraus.

5.2.1 Versuch 1: Untersuchen eines Infrarot Signals mittels Oszilloskop

Von Andreas Weber

Zum Untersuchen eines Signals, in diesem Fall ein Spannungssignal dass eine Infrarote Leuchtdiode (LED) ansteuert, kann ein handelsübliches Oszilloskop hergenommen werden. Dies wird einfach zu den beiden Anschlüsse der Infrarot LED parallel angeschlossen. Dazu wurde eine Universalfernbedienung, die auf nahezu jedes Fernsehgerät angelernt werden kann, verwendet. Diese wurde dazu zerlegt, und in den Versuchsaufbau eingefügt:

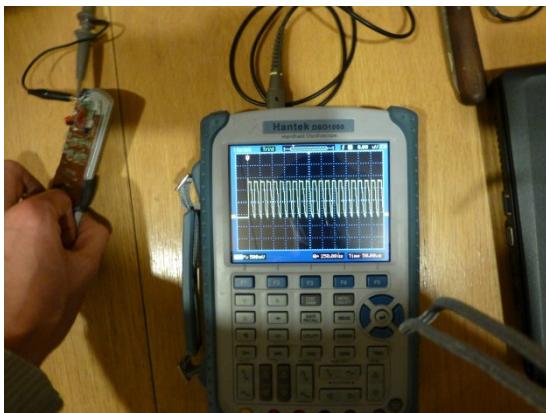


Abbildung 11

Anschließend konnte bereits eine Voranalyse des Signals vorgenommen werden. Leider war die Auflösung des Oszilloskops nicht ausreichend genug, bzw. das Fenster in dem Das Signal angezeigt wurde nicht breit genug, deswegen wurde nach einer Möglichkeit gesucht, das Signal über einen längeren Zeitraum (ca. 0.5s) aufzuzeichnen.

Im Internet werden solche Signale über den Mikrofoneingang des Notebooks aufgezeichnet. Mit geeigneter frei verfügbarer Software, kann das Signal dann als Spannungsamplitude über die Zeit angesehen werden.

Das Problem dabei ist, dass der Mikrofoneingang hardwaremäßig auf der Soundkarte bereits über entsprechende Filterschaltungen verfügt, und somit alles was nicht im hörbaren Bereich liegt (also von der Frequenz kleiner als ca 20kHz) herausfiltert. Da das Infrarotsignal über eine Grundfrequenz von 40kHz verfügt, kann es zu falschen Messpunkten kommen.

5.2.2 Versuch 2: Aufnahme des Signals mit dem Raspberry Pi

Von Andreas Gerken

Wie in 2.3 (Emulation) beschrieben, kann über die GPIO Pins das Signal des Infrarotempfängers über LIRC analysiert werden. Es kann jedoch nicht nur zur Erkennung der Befehle verwendet werden, sondern genauso um ein Signal aufzunehmen. Parallel zur Bearbeitung der anderen Lösungsmöglichkeiten (Analyse per Oszilloskop, WinLirc) wurde auch für die Aufnahme, LIRC in das Raspbian Betriebssystem eingebaut. Dabei gab es zunächst zwei relativ „einfache“ Probleme:

- Bildschirme werden normalerweise über den HDMI-Port an den Raspberry PI angeschlossen. Die Bildschirme in der Hochschule haben aber keinen solchen Eingang. Die Lösung stellte ein HDMI-DVI Adapter von Herr Schäffer dar.
- Um die benötigten Programme zu installieren benötigt man einen Internetzugang. In der Hochschule gibt es aber keine frei zugänglichen Ethernet Stecker, an die man den Raspberry anschließen könnte. Es gelang über einen WLAN Stick, dessen Einrichtungsumgebung aber auch nicht auf die Authentifizierung mit Username und Passwort ausgelegt war.

Diese Verzögerungen führten dazu, dass kaum noch Zeit war um LIRC einzurichten und die Teammitglieder die es mit WinLirc Probieren haben schneller zu einem Ergebnis gekommen sind.

5.2.3 Versuch 3: Aufzeichnung von Infrarotsignalen über die Soundkarte

5.2.4 Versuch 4: Aufzeichnung von Infrarotsignalen über die serielle Schnittstelle des Computers (RS232)

5.3 Lirc-File

5.3.1 Analyse durch WinLirc

Von Andreas Gerken und Andreas Weber

```
begin remote

    name  \\rz-home\ff9315\Desktop\FbLeftSend
    flags RAW_CODES|CONST_LENGTH
    eps      30
    aeps     100

    gap      210903

    begin raw_codes

        name DIVE
        309    712    370    657    352    130
        360    162    304    718    325    215
        285    205    316    707    303    728
        298    216    302    715    358    672
        332    205    276    215    337    199
        297    703    330    695    293    220
        298    210    360    159    355    159
        324    701    306    712    308    719
        317    96223   296    724    308    719
        306    205    302    218    302    719
        304    211    325    190    289    735
        305    777    267    210    285    711
        312    719    329    207    315    202
        302    182    298    732    304    715
        337    203    333    153    312    206
        311    202    364    652    339    680
        316    717    306

    end raw_codes

end remote
```

Abbildung 12

Raw File mit exemplarischem Befehl

Das obere Bild zeigt ein Lirc File, wie es direkt nach dem einlesen aussieht. Die Infrarotsequenz wird dabei nur als „raw code“ aufgezeichnet. Die in der Datei abgespeicherten Werte zeigen immer abwechselnd die Dauer eines High Pegels und eines Low Pegels. Bei der Abbildung wurde nur die Sequenz einer Taste dargestellt. Es handelt sich um die Taste um das Gewicht nach vorne zu bewegen und den Fisch somit nach vorne zu kippen.

Die dargestellten Werte zeigen immer abwechselnd die Dauer eines High Pegels und eines Low Pegels. Man sieht jedoch, dass die Werte stark schwanken und kaum geeignet sind, um erneut gesendet zu werden. Deshalb bietet Lirc die Möglichkeit,

Signale in Hexadezimalen Werten abzuspeichern und einheitlich das Signal für ein 1 Bit und ein 0 Bit abzuspeichern.

Um ein Signal vom „Raw Code“ in Hexadezimale Werte umzuwandeln muss die irrecord.exe (in Winlirc inbegriffen) mit dem Attribut –a für Analyse und dem Dateinamen der Raw Datei aufgerufen werden. Das Ergebnis wird auf die Konsole ausgegeben und muss somit noch in ein File umgeleitet werden. Der gesamte Konsolenauftruf sieht folgendermaßen aus:

Irrecord.exe –a rawfile.txt > hexfile.txt

Das Programm sucht intern nach einem Muster in den Pegelzeiten und findet zum Beispiel dass die ersten beiden Signale (309ms, 712ms und 370ms, 657ms) relativ ähnlich sind. Die darauf folgenden Signal (352ms, 130ms und 360ms, 162ms) unterscheidet sich von den ersten Beiden, sind aber untereinander wiederum Ähnlich. Wenn man die beiden ersten beiden Bits als Eins definiert sind die anderen automatisch Nullen. So erhält man $1100 = 0xC$. So werden alle Pegelzeiten analysiert und zusammengesetzt. Das Signal aus der Abbildung besteht aus 3 Bytes und wird nach der Umwandlung durch $0xC9B187$ beschrieben. Zusätzlich werden alle Pegeldauern einer Art (z.B. der Low Pegel einer 0) gemittelt und abgespeichert. Das neue File ist dadurch viel kompakter und einfacher zu lesen. Außerdem werden die Signale viel genauer beschrieben, weil es keine zeitlichen Schwankungen zwischen den einzelnen Pegeldauern gibt.

5.3.2 Analyse und Verbesserung der Lirc Files

Von Andreas Gerken

Die „Hex Files“ sehen folgendermaßen aus:

```

#
# this config file was automatically generated
# using lirc-0.9.0(emulation) on Thu Apr 18 15:38:59 2013
#
# contributed by
#
# brand:          \\rz-home\ff9315\Desktop\FbLeftSend
# model no. of remote control:
# devices being controlled by this remote:
#

begin remote

    name  \\rz-home\ff9315\Desktop\FbLeftSend
    bits      24
    flags  SPACE_ENC|CONST_LENGTH
    eps       30
    aeps      100

    one        315    709
    zero       315    198
    ptrail     319
    gap        114972
    toggle_bit_mask 0x0

    begin codes
        DIVE           0xC9B187
        CLIMB          0xC9B14B
        TAILRIGHT      0xC9B11E
    end codes

end remote

```

In der Abbildung erkennt man einige Daten, die im Folgenden erklärt werden:

- Name: Name unter dem die „raw Datei erstellt wurde“
- Bits: Bits die pro Signal gesendet werden. In unserem Fall 3 Bytes.
- Flags:
 - o SPACE_ENC: Das Flag heißt, dass Bits durch die Abstände (SPACE) zwischen den Flanken kodiert (ENCryption) werden. Es gibt zwar noch andere Kodierungen (RC5,...) jedoch wurde diese Kodierung in allen analysierten LIRC Files automatisch gesetzt.
 - o CONST_LENGTH: Alle Befehle haben die gleiche Länge. (Auswirkung wird bei gap beschrieben)
- Eps, Aeps: Kennzahlen zur Toleranz beim Empfangen von Infrarotsignalen. Eps ist dabei ein relativer Wert (30% Toleranz) und Aeps ein absoluter Wert in microsekunden (100µs).
- One, Zero: gibt die Dauer der verschiedenen Pegel an. Zuerst die Dauer des „High Pegels“ und dann die Dauer des „Low Pegels“ im obigen Beispiel wird also eine Eins durch 315ms an und 709ms aus kodiert. Äquivalent dazu die Null mit 315ms an und 198ms aus.
- Ptrail: Nach den letzten Daten wird ein „High Pegel“ von 319ms gesetzt
- Gap: Der Gap ist die Lücke zwischen zwei Befehlen. Da bei Flags

CONST_LENGTH gesetzt ist, gibt der Wert die Pause in Millisekunden an, die zwischen Ende eines Befehls bis Anfang des nächsten Befehls gewartet werden soll. Ohne CONST_LENGTH würde der Wert die Pause zwischen jeweils den Anfängen zweier Befehle festlegen.

- toggle_bit_mask 0x0Laut (<http://winlirc.sourceforge.net/technicaldetails.html>) können über die Maske Bits, bei jedem erneuten drücken einer Taste gekippt werden.

Zusätzliche Möglichkeiten

1. pre_data, pre_data_bits: Über pre_data können Bits angegeben werden, die vor jedem Befehl gesendet werden sollen. Die Länge wird über pre_data_bits angegeben
2. post_data, post_data_bits: Äquivalent zu pre_data nur nach dem Befehl.

Codes:

Zwischen begin codes und end codes werden nun die einzelnen Befehle Angegeben. Im vorherigen Beispiel sind nur drei Befehle aufgezeichnet worden. Der fehlende (TAILLEFT) wurde fehlerhaft aufgezeichnet und konnte nicht analysiert werden.

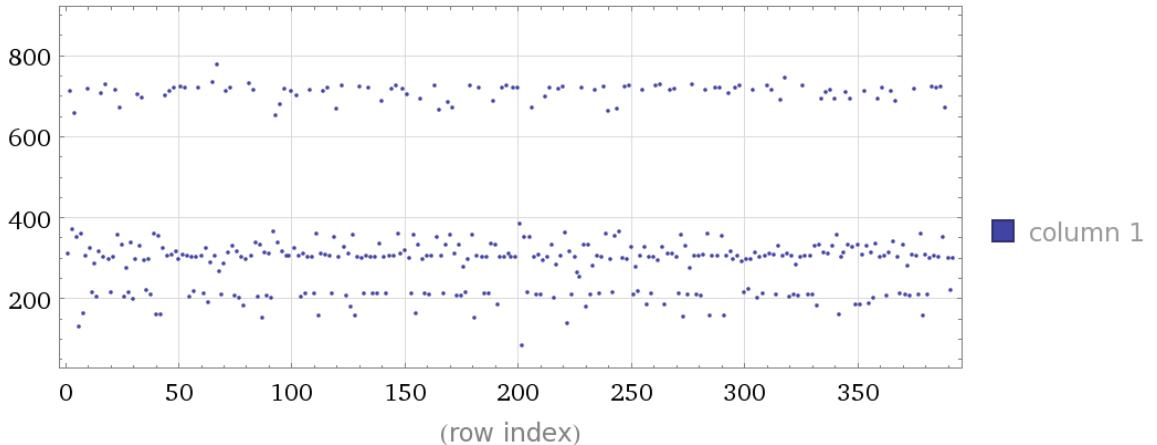
Optimierung

Die aufgenommenen LIRC Files funktionierten leider nur sehr schlecht, da sie Fehlerhaft waren. Durch das „zusammenbauen“ mehrerer Files konnten die Fehler korrigiert werden. Die Fehler kommen dadurch, dass erstens die Fernbedienung keine präzisen Signale sendet, die Abtastrate des Seriellen Eingangs zu langsam ist oder der Prozess auf dem Computer zu spät dran kommt.

Dadurch treten folgende Fehler auf:

- Die Dauer der Pegel wird nur ungenau aufgenommen
- Dadurch können manche Signale nicht analysiert werden oder es werden die falschen Hex Werte erkannt
- Die Pegeldauer (One, Zero) werden falsch übernommen. Da der falsche Wert auch zum Senden benutzt wird werden die Signale dadurch vom Empfänger zum Teil nicht erkannt.

Raw data plot :



In der Abbildung ist die Verteilung der Dauer einzelner Pegel (Raw values vor der Analyse) zu sehen hier fällt schon auf, dass die Werte stark variieren. 4 Werte wurden bereits aus der Graphik entfernt die ca. 100.000ms lang waren da sie sonst die Graphik zu sehr verschoben hätten. Rein logisch waren sie nicht zu erklären und im Internet ließ sich nichts über solch große Werte finden. Sie werden vom Analysetool anscheinend nicht beachtet, da sie anscheinend keinen Einfluss auf die Hex Werte haben.

Da auch beim Senden mit der Irdroid weitere Ungenauigkeiten auftreten, summieren sich alle Fehler und die Befehle werden nur schlecht oder gar nicht vom Fisch erkannt. Deshalb wurde das LIRC File optimiert was im Endeffekt auch Wirkung zeigte:

Zunächst wurden die Kommandos aus allen Files verglichen und die ausgewählt, die in den meisten Files vorgekommen sind. Dass manchmal Befehle nicht erkannt wurden, war dadurch kein Problem, da jeder Befehl in mindestens einer Aufnahme vorhanden war.

Um die Richtigkeit der Hexwerte zu verifizieren, wurde das Bitmuster analysiert:

Da die ersten beiden Bytes für alle Befehle gleich waren (0xC9B1) wurde nur das letzte Betrachtet:

DIVE	0x87	1	0	0	0	0	1	1	1
CLIMB	0x4B	0	1	0	0	1	0	1	1
TAILLEFT	0x2D	0	0	1	0	1	1	0	1
TAILRIGHT	0x1E	0	0	0	1	1	1	1	0

Wie man in der Tabelle sieht, sind die Bits immer entweder bei drei Befehlen 1 und bei den anderen 0 oder genau andersrum. Somit hat jeder Befehl zwei einzigartige „Bit-Merkmale“ und beim Empfangen können Bit-Fehler ausgeglichen werden. Dass die

Anordnung zufällig entstanden ist, ist sehr unwahrscheinlich und so, konnte bestätigt werden, dass die Hex Werte stimmen.

Des Weiteren wurde die Pegeldauer von Zero und One von allen Aufnahmen gemittelt um den Fehler auch hier zu reduzieren. Die gemittelten Werte waren:

One	296	720
Zero	296	221
Ptrail	296	

Die gemessenen Zeiten weichen bis zu 12% von diesen Durchschnittswerten ab. Zum Testen wurden drei Versionen bereitgestellt: die Original-Zeiten, die Durchschnittswerte und gerundete Werte (auf 300, 720 und 220). Durch Testen mit dem Fisch stellte sich heraus, dass der Fisch auf die gerundeten Werte am besten reagierte.

Die gleiche Analyse wurde genauso für den neuen Fisch durchgeführt, bei dem sich die Analyse als schwerer herausstellte, da die aufgenommenen LIRC-Files stark voneinander abwichen und viele Fehler dabei waren. Das kann entweder daran liegen, dass die Fernbedienung ungenauer schickt oder das Aufnahmesetup ungenauer war (Anderer PC, weitere Entfernung vom Sender zum Empfänger, etc).

Änderungen bei der neuen Fernbedienung:

Es gibt einen Befehl mehr, bei dem der Fisch mit der Schwanzflosse wedelt und somit nach vorne schwimmt. Die Daten sind 11 Bits lang wobei die ersten 2 immer Nuller sind, die restlichen 9 Bits sind ähnlich wie bei der alten Fernbedienung aufgebaut:

DIVE	0x0FF	0	1	1	1	1	1	1	1	1
CLIMB	0x17C	1	0	1	1	1	1	1	0	0
TAILLEFT	0x1B5	1	1	0	1	1	0	1	0	1
TAILRIGHT	0x1D9	1	1	1	0	1	1	0	0	1
SWIM	0x1FD	1	1	1	1	0	1	1	1	1

Auch hier gibt es für 8 Bits jeweils eine einmalige Belegung. Die Ausnahme beim vorletzten Bit, bei dem Dive und Forward beide eine Eins belegen muss daher kommen, dass die Befehle in 9 Bits kodiert werden. Mit 10 Bits wäre es genau aufgegangen, weswegen nicht ganz verständlich ist, warum das eine Bit nicht noch spendiert wurde und solche krummen Zahlen verwendet werden.

Die gemittelten Werte für One und Zero sind bei dem neuen Fisch nicht so einheitlich und intuitiv „rundbar“

One	777	314
Zero	382	314

Ptrail entfällt bei der Fernbedienung

Da bei der alten Fernbedienung Änderungen der Zeiten um wenige Millisekunden reichten um das Signal zu verbessern versuchte ich auch bei den neuen Werten ein System zu entdecken.

$$\frac{t_{One, on}}{t_{Zero, on}} = \frac{777}{382} = 2,034$$

$$\frac{t_{One, ges}}{t_{Zero, ges}} = \frac{777 + 314}{382 + 314} = \frac{1091}{696} = 1,5675$$

Es kann natürlich Zufall sein, dass das Verhältnis zwischen den „An Zeiten“ ungefähr Zwei ist und das Verhältnis zwischen den Gesamtzeiten ca. 1,5. Ein Versuch war es auf jeden Fall wert, Werte zu finden, die die genauen Verhältnisse bringen und ähnlich zu den Gemittelten sind.

Durch lösen der Gleichungen $\frac{x}{y} = 2$ und $\frac{x+z}{y+z} = 1,5$ resultierte die Lösung $y = z = \frac{x}{2}$.

Somit müsste man die „An Dauer“ von Zero und die „Aus Dauer“ um jeweils mindestens 20% verändern um das Verhältnis zu erlangen was zu unrealistisch erschien.

Die Lösung des zweiten Versuchs (mit $\frac{x+z}{y+z} = 1,6$) ist $y = \frac{x}{2}$ und $z = \frac{x}{3}$

Eine Lösung dafür, bei der jeder Wert um 10% geändert werden würde (weniger war nicht möglich) ist

One	840 (+63)	280 (-34)
Zero	420 (+38)	280 (-34)

Die Veränderung wäre immer noch sehr drastisch also müssen die Anforderungen reduziert werden.

Wenn man die zweite Bedingung ($\frac{x+z}{y+z} = 1,5$ oder $1,6$) vernachlässigt und nur versucht die Erste zu erreichen, ist die Lösung mit sehr kleinen Korrekturen zu erreichen. Wenn danach noch der „Low“ Wert auf 315 aufgerundet wird, stellt sich genau der Faktor zwischen den Gesamtzeiten 1.55. Die beiden geraden Faktoren (ohne viele Nachkommastellen) und die einzelnen Werte die auf 0 und 5 enden scheinen gefühlsmäßig plausibel.

One	770 (-7)	315 (+1)
Zero	385 (+3)	315 (+1)

Die geänderten Zeiten werden auf den Server geladen und sie müssen getestet werden. Das gefundene „System“ ist eventuell ein wenig aufgezwungen doch es wird sich zeigen, ob die Werte besser funktionieren.

Schönheitsänderungen:

Da bei der ersten Fernbedienung die ersten 2 Bytes immer gleich waren (0xC9B1) wurden sie in pre_data geschrieben, sodass nur noch der veränderliche Teil pro Befehl angegeben werden muss.

Bei manchen LIRC Files war keine CONST_LENGTH angegeben. Das Flag wurde hinzugefügt und der Gap entsprechend geändert.

6 Senden

6.1 Irdroid-App

6.2 Senden

7 Oberfläche

7.1 Konzept

Von Anja Hafner, Melanie Knappe, Belgüzar Kocak

Zielsetzung

Ziel war es mit der Oberflächen Gruppe Anja Hafner, Melanie Knappe & Belgüzar Kocak Konzepte zur Steuerung des Air Swimmers zu entwickeln.

Vorüberlegung und Analyse

Die Ansteuerung des Air Swimmers erfolgt durch eine Infrarot Verbindung. Diese enthält folgende Funktionen:

1. Ansteuerung der hinteren Flosse (nach links oder nach rechts)
2. Ansteuerung des Gewichtes am AirSwimmer (Flug nach oben oder nach unten)

Auf diese Funktionen soll die Oberflächensteuerung aufbauen. Somit wurden drei verschiedene Ansteuerungsvarianten zur Realisierung der Oberfläche erstellt, die wiederum die Steuervarianten Single und Permanent enthalten.

Bei der Ansteuerungsart Button stellt die Bewegungsart Single die Funktion der Fernbedienung nach. Die Ansteuerungsart Permanent enthält einen zusätzlichen Button Start. Durch betätigen des Start Buttons bewegt sich der Air Swimmer aufgrund der permanenten Ansteuerung der hinteren Flusse vorwärts. Die Buttons Links & Rechts ermöglichen eine Links-/ Rechtskurve.

Beim Wischen stellt die Single Bewegungsart folgendes dar: durch eine Wischbewegung nach Links bewegt sich die Flosse einmal nach links. Durch die Wischbewegung nach rechts bewegt sich die Flosse einmal nach rechts. Bei der Permanentfunktion fliegt der AirSwimmer durch eine Berührung der Oberfläche nach vorne. Durch das Wischen nach rechts/links fliegt der AirSwimmer die entsprechende Kurve.

Bei der Nutzung der Ansteuerungsart Kippen erscheint beim Start ein Start Button. Durch Drücken auf diesen startet der Fisch. Bei der Bewegungsart Single bewegt sich die Hinterflosse einmal nach links oder rechts, je nach Kipprichtung. Durch das Kippen nach oben/unten wird das Gewicht am AirSwimmer angesteuert. Wird der Bildschirm berührt, reagiert der AirSwimmer nicht mehr und der Startbildschirm erscheint. Bei

der Bewegungsart Permanent erscheint auch zunächst der Start Button, nach Betätigung beginnt der Air Swimmer geradeaus zu schwimmen. Durch das Kippen nach links/rechts wird eine Links-/Rechtskurve geflogen. Auch hier erzeugt die Berührung an den Bildschirm ein Stoppen und der Startbildschirm erscheint.
Die Realisierung der verschiedenen Konzepte wird in den nächsten Kapiteln eindeutig erklärt.

7.2 Startseite

Von Caroline Pilot

Zielsetzung

Jede App benötigt eine Startseite, die es dem Benutzer ermöglicht, eine Auswahl zu treffen, zwischen verschiedenen Seiten und Funktionen zu wechseln und somit den Benutzer empfängt und leitet.

Da die Startseite einen Knotenpunkt für alle Teilanwendungen (Tasten, Kippen, Wischen) darstellt, müssen diese hiermit verknüpft werden.

Vorüberlegung

Aussehen

Da die Startseite in erster Linie die Navigation zu den verschiedenen Benutzermodi übernehmen soll, werden drei Buttons gebraucht.

Außerdem soll die erste Seite der App natürlich optisch ansprechend wirken, deshalb wäre eine Eröffnungsanimation oder eine einladende Graphik des AirSwimmers von Vorteil.

Programmcode

Mit Hilfe der Java-Klasse „Activity“ können die Buttons mit Aktionen belegt werden, die zu den einzelnen Teilfunktionen führen.

Der Programmcode soll in Englischer Sprache erstellt werden.

Außerdem soll für ein Tablet der Größe 10,1 Zoll mit einer Auflösung von 1280 x 800 Pixeln programmiert werden.

Vorgehensweise

Erstellen der Startseite

a) Konfiguration

Um die Einbindung der Teilfunktionen einfacher zu gestalten, wird ein neues Android Application Project aufgesetzt („AirSwimmer“) und mit folgenden Einstellungen konfiguriert:

Minimum Required SDK → API 7: Android 2.1 (Eclair)

Target SDK → API 7: Android 2.1 (Eclair)

Compile with → API 7: Android 2.1 (Eclair)

Um Probleme bei der Abwärtskompatibilität zu vermeiden, werden niedrige SDK-Versionen verwendet, damit die App auch auf Endgeräten mit älteren Android-Versionen lauffähig ist.

Die Launcher-Konfiguration wird über ein „Image“ der Form „Circle“ mit der erstellten Graphik gespeist.

Der „Activity-Name“ ist passend dazu „FrontPage“ sowie der zugehörige „Layout Name“ „front_page“ lautet.

b) Layout Erstellung

Im Unterordner „Layout“ kann nun per Drag&Drop das Erscheinungsbild der Startseite, wie in der Vorüberlegung angesprochen, erstellt werden. Diese setzt sich aus folgenden Komponenten zusammen:

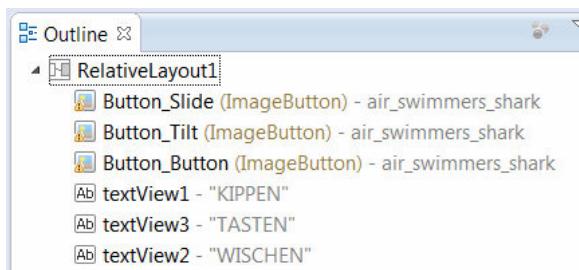


Abbildung 13

Komponenten des Layouts der Startseite

Die Buttons werden je nach Funktion mit einem String beschriftet.

Nach Erstellen der Buttons befindet sich für jeden Button ein Codeblock in der xml-Datei. Dieser wird jeweils um folgende Codezeile erweitert, damit der Benutzer diesen Button tatsächlich anklicken kann.

`android:onClick="onButtonClick"`

Außerdem wird ein passender Hintergrund gewählt, der das „AirSwimmer-Thema“ aufgreift

und die Buttons werden in ihrer Größe und Ausrichtung auf die Maßen eines Tablets ausgerichtet.

Einbinden der Teilfunktionen

a) Layout und Source-Code einbinden

Durch Erzeugen neuer xml-Dateien („activity_layout_buttons“, „activity_layout_wipe“ und „activity_layout_tilt“) im Unterordner „Layout“ und Einfügen des bereits vorhandenen Codes für die Modi, sind die Layouts der Unterfunktionen nun Bestandteil des Hauptprogramms.

Dasselbe Vorgehen findet auch für den Source-Code im Unterordner „src“ statt, mit neuen Java-Dateien („Activity_Buttons“, „Activity_Wipe“ und „Activity_Tilt“).

Diese Funktionen müssen nun im AndroidManifest unter „Application“ als Knoten hinzugefügt werden:

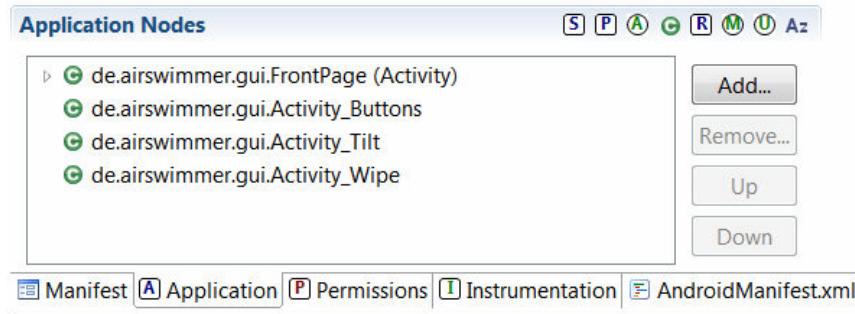


Abbildung 14

b) Funktion programmieren

Da durch den Buttonklick zu der jeweiligen Funktion gewechselt werden soll, wird der Source-Code der Startseite um switch-case Anweisungen erweitert:

```
// What will happen when you click on a button
public void onButtonClick(View view){
    switch(view.getId()){
        case R.id.Button.Buttons:
            startActivity(new Intent(this, ActivityButtons.class));
            break;
        case R.id.Button_Tilt:
            startActivity(new Intent(this, ActivityTilt.class));
            break;
        case R.id.Button_Wipe:
            startActivity(new Intent(this, ActivityWipe.class));
        default:
    }
}
```

Insgesamt muss beachtet werden, dass alle Namenskonflikte aufgelöst werden, alle zusätzlich genutzten Pakete ebenfalls implementiert werden und alle eingefügten Graphiken in das neue Projekt eingepflegt werden.

Ergebnis

Der Benutzer gelangt mit Hilfe dieser App nun über die Startseite hin zu den Teillfunktionen (Tasten, Wischen, Kippen) und kann somit den AirSwimmer steuern.

7.3 Steuerung mittels Buttons

Ziel:

Ziel ist es, eine Oberfläche mit Buttons zu erstellen. Diese soll vier Buttons zur Steuerung enthalten. Somit wird die Steuerungsarten rechts, links, oben und unten realisiert. Auch soll diese Oberfläche die Fernbedienung des AirSwimmers ersetzen.

Zu Beginn werden zwei Ausführungen erstellt: Einerseits mit Buttons mit Beschriftung(Variante A) und andererseits mit Buttons mit Bildern (sogenannten ImageButtons) (Variante B). Als endgültige Version der Buttonssteuerung wird Variante B gewählt.

7.3.1 Variante A (erste Testversion)

von Anja Hafner

Vorgehensweise:

Die Oberfläche des Projektes „Android Fernsteuerung des AirSwimmers“ wird in der Entwicklungsumgebung „eclipse“ mit Add-Ons für die Android-Programmierung entwickelt.

Zu Beginn wird ein neues „Android Application Project“ in eclipse angelegt (zu finden unter File → New → Other... → Android → Android Application Project).

Die „main_activity“, die nach der Erstellung des Projekts erscheint, zeigt eine Darstellung eines Handybildschirms mit dem Projektnamen als Überschrift und „Hello World“ als Text im Bildschirm an.

Um nun den ersten Entwurf mit vier einfachen Knöpfen mit Beschriftung zu erzeugen, wird aus der Palette, die sich neben der Darstellung der zukünftigen App befindet, im Abschnitt „Form Widgets“ der Button ausgewählt.

Die Beschriftungen der Buttons werden zunächst in „strings.xml“ definiert (res → values → strings.xml). Für die Initialisierung wird der Button „Add...“ gedrückt. Im erscheinenden Fenster wählt man den Typ „String“ aus und gibt als Namen den Namen des Strings ein, mit dem er angesprochen werden soll. In Value wird die zukünftige Beschriftung (zum Beispiel: „up“) definiert.

Anschließend wird der Button viermal mit der Maus in den Handybildschirm gezogen. Mit dem Befehl „`android:text="@string/*"`“ wird der String dem Button zugeordnet, wobei der Stern für den Namen des definierten Strings steht.

Für eine einfachere Benutzung der Knöpfe während dem Programmieren, werden die Namen der vier Buttons in der „main_activity.xml“ in „button_up“, „button_down“, „button_left“ und „button_right“ geändert.

Um die Abstände der Knöpfe zueinander anzupassen, markiert man den Button, den man verschieben möchte. Danach wird in der Symbolleiste des „activity_main.xml“ Graphical Layout-Fensters der Button, der einen gestichelten Rahmen zeigt („Change Margins...“), angeklickt. In dem erscheinenden Fenster trägt man die gewünschte Position ein. Dabei ist zu beachten, dass die Abstände mit der Endung „dp“ (Density Independend Pixel) angegeben werden müssen, damit die Oberfläche richtig angezeigt wird.

Nun kann die App getestet werden. Dazu wird sie zunächst gespeichert. Anschließend folgt ein rechts-Klick auf den Projektordner und unter „Run As“ wird die Option „Android Application“ ausgewählt.

Besitzt man ein Android-fähiges Handy oder Tablett, wird dieses angeschlossen und von eclipse erkannt. Sollte es nicht erkannt werden, müssen zur Erkennung zusätzliche Treiber für das Handy / Tablett installiert werden. Die .apk-Datei der App wird danach auf das Gerät installiert und es kann direkt auf dem Handy / Tablett ausgeführt werden.

Eine .apk-Datei ist eine Art Ordner, die alle erforderliche Daten für die Ausführung der App enthält.

Soll die App auf dem Computer ausgeführt werden, ist ein „Virtual Device“ (ein virtuelles Handy, auf das die App gespielt wird) nötig.

Dieses wird im „Android Virtual Device Manager“ erzeugt. Dorthin gelangt man über „Window → Android Virtual Device Manager“ oder in der Symbolleiste über das Handysymbol. Per Mausklick öffnet sich ein Fenster. In diesem Fenster drückt man den Button „New...“ und ein neues Virtual Device kann erstellt werden.

Da die App für Tabletts gedacht ist, wird als darzustellendes Gerät „10.1“ WXGA (Tablet) (1280 x 800: mpdi) ausgewählt. Als Target (die minimale Android-Version, auf der die App laufen wird) wird die API „Android 2.2 – API Level 8“ festgelegt.

Nachdem das „Android Virtual Device“ gestartet ist, kann die App, die „AirSwimmer“ genannt wird, gestartet und die Steuerung mit Buttons getestet werden.

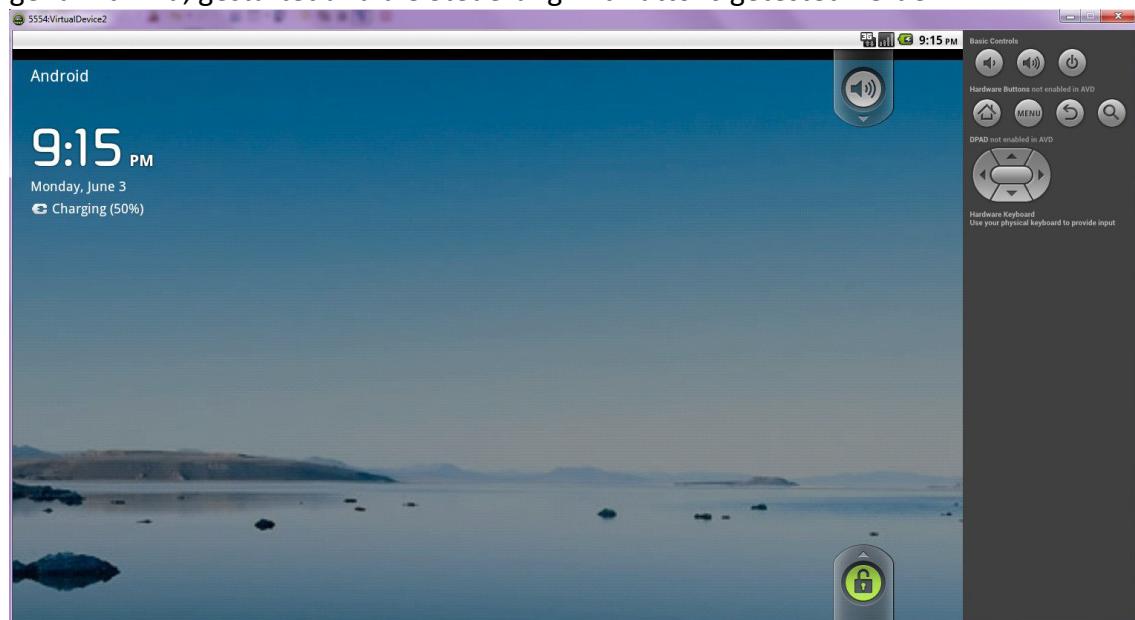


Abbildung 15

Android Virtual Device

7.3.2 Variante B (*ImageButtons*)

Von Belgüzar Kocak

Vorgehen

Zuerst stellt uns eclipse beim Erstellen von Projekten ein Standard Layout zur Verfügung.

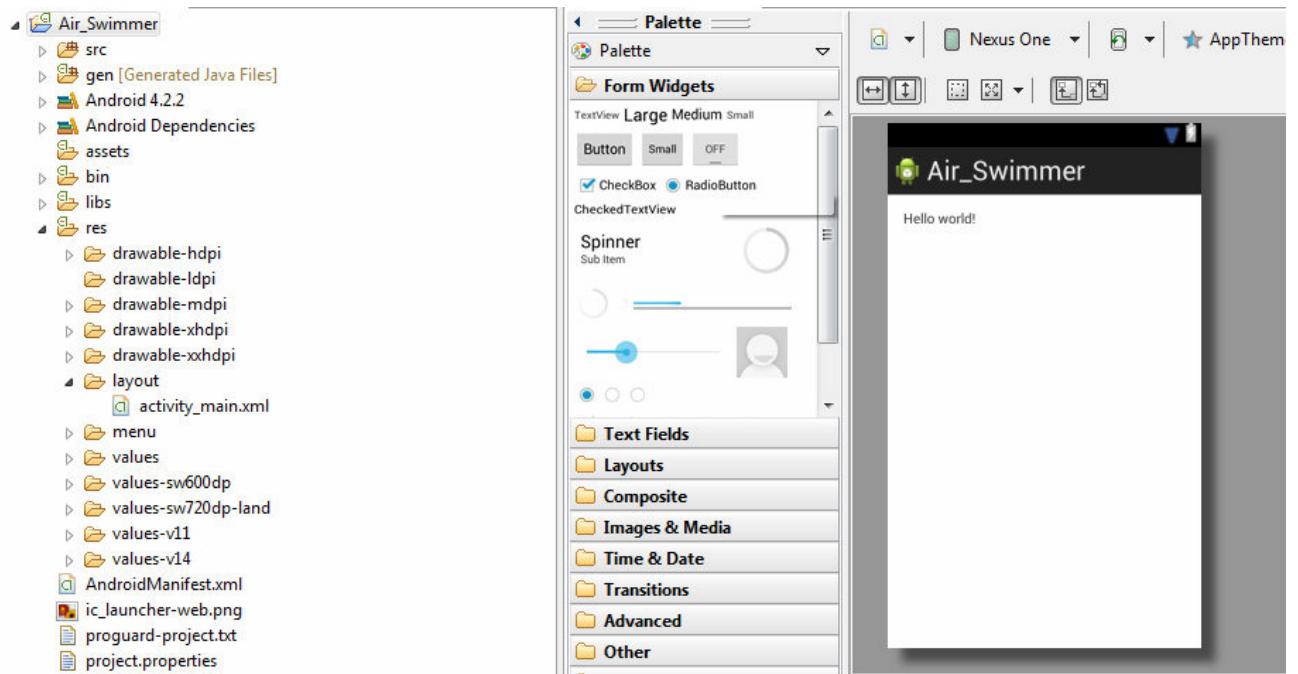


Abbildung 16

Dieses enthält folgende XML-Struktur:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Zu allererst wird dabei die TextView entfernt. Unter dem Punkt Image & Media werden 4 ImageButtons in das Layout gezogen und durch einen Doppelklick gelangt man direkt in die XML-Datei. Anhand dieser Zeile in der XML-Datei wird die entsprechende Grafik in den Button gesetzt.

`android:src="@drawable/up"`

Auch wird der Hintergrund des Layouts geändert indem beim RelativeLayout die folgende Zeile hinzugefügt wurde:

`android:background="@drawable/ic_sky"`

Diese Grafik stellt ein Wolkenhintergrund dar und wird vom drawable-hdpi aufgerufen. Um die Buttons transparent zu gestalten, wird folgender Code editiert.

```
android:background="#00000000"
```

Um auch später die Bewegungssteuerung für die Buttons schnell und einfach zu implementieren wurde für jeden Button ein ButtonListener eingebaut. Ein Listener überprüft wann und ob ein Button gedrückt wurde. Falls ein Button gedrückt wurde kann man eine Funktion zur Steuerung implementieren.

```
button_up = (ImageButton) findViewById(R.id.imageButtonUp);  
button_up.setOnTouchListener(this);
```

Ergebnis

Durch diese Vorgehensweise wurde ein Layout mit Buttons-Steuerung erstellt. Somit wurde der Start zur Oberflächenprogrammierung gegeben. Auf diese Oberflächen wurden die folgenden Ansteuerungsarten gebaut.

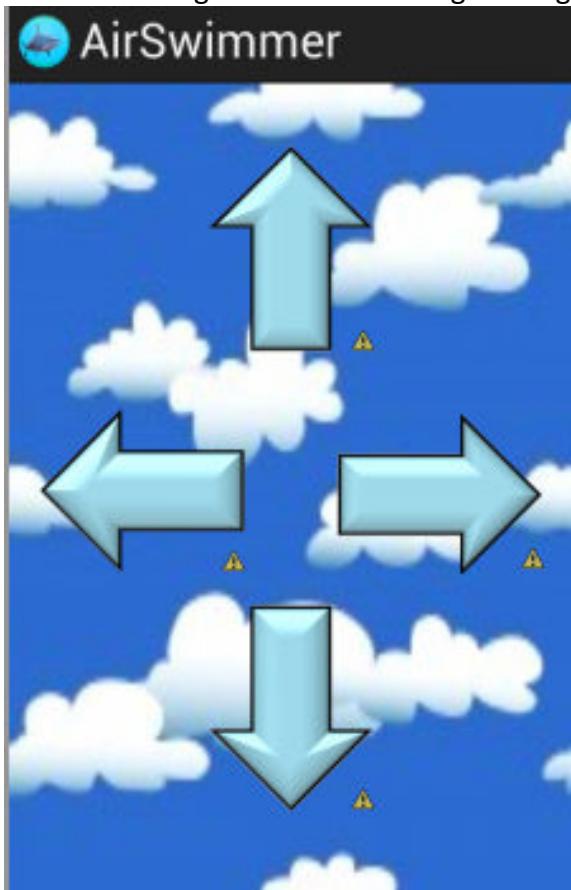


Abbildung 17

7.3.3 Reaktion auf Druck und visueller Effekt

Von Caroline Pilot

Zielsetzung

Da die Tasten bei einer einfachen „if-Abfrage“ in den „Activitiy_Button“ Klassen nur auf ein Loslassen der Tasten vom Benutzer reagieren, muss hier, vor allem aus Logikgründen, eine Reaktion beim Drücken der Tasten implementiert werden.

Zudem soll ein visueller Effekt dem Benutzer deutlich machen, wann ein Signal gesendet wird und wann nicht.

Vorüberlegung

Die Reaktion beim Drücken bedarf keiner weiteren Überlegung.

Der visuelle Effekt jedoch, kann verschieden implementiert werden:

- Ein Text mit der Aufschrift der jeweiligen Richtung besteht bereits, dieser wird allerdings nur für eine kurze Zeitdauer angezeigt
- Ein farblicher Rahmen um die Tasten beim Druck
- Die gedrückte Taste an sich, könnte die Farbe verändern

Ergebnis: Der Text soll weiterhin kurz angezeigt werden, dies soll der reinen Information für den Benutzer dienen und außerdem wird so sichergestellt, dass auch tatsächlich die Signale für beispielsweise die Aufwärtsbewegung ausgegeben werden.

Auch die Taste an sich soll keine Farbänderung erfahren, denn dies könnte möglicherweise nicht direkt gesehen werden, wenn der Benutzer seinen Finger darauf legt.

So bleibt die gebräuchliche Möglichkeit, einen dickeren Rahmen um jede Taste zu legen und diesen beim Druck einzufärben und beim Loslassen verblassen zu lassen.

Vorgehensweise

Reaktion beim Druck der Tasten

Durch eine einfache Erweiterung der „If-Anweisung“ in der Base-Class der Buttons kann die Reaktion nur beim Druck erreicht werden.

```
if (event.getAction() == MotionEvent.ACTION_DOWN){...}  
bzw.  
if(event.getAction() == MotionEvent.ACTION_UP){...}
```

Visueller Effekt

Mit dem Befehl lässt sich der Hintergrund der Buttons einfärben:

```
button_up.setBackgroundColor(Color.BLUE);
```

Dies setzt den Hintergrund des „button_up“ Bildes auf Blau. Diese Farbe wurde gewählt, weil man sie auf jedem Hintergrundbild gut sehen kann und auch zu den Farben des AirSwimmers passt.

Da dies erst geschehen soll, sobald der Nutzer die Taste drückt (`ACTION_DOWN`), wird dieser Befehl in die erste „if-Anweisung“ eingefügt und ein passender Befehl (`Color.TRANSPARENT`), welcher den Hintergrund der Taste wieder farblich zurücksetzt, in die zweite „if- Anweisung“ eingefügt.

Ergebnis

Der Benutzer erhält nun visuelle Rückmeldung, dass der Button gedrückt wird, diese Farbänderung hält die ganze Zeit an, bis der Benutzer den Finger wieder hebt. In genau dieser Zeit sollen auch die Signale an den Fisch gesendet werden.

7.4 Steuerung durch Wischen

von Anja Hafner

Ziel:

Ziel ist es, eine Ansteuerung des AirSwimmers per Drag and Drop Prinzip zu realisieren. Dabei soll ein Bild, das in dem „drawable“-Ordner im „res“-Ordner des Projektes gespeichert ist, per Drag and Drop Prinzip über den Handybildschirm bewegt werden. Unter Drag and Drop versteht man, dass mit dem Finger das Bild des Fisches in der App berührt wird und dieses in die gewünschte Position gezogen wird. Der AirSwimmer ahmt anschließend die Bewegung des Fisches in der App nach.

Vorgehensweise:

Für die Steuerung der App durch Wischen wird ein neues Projekt mit dem Namen „SlideApp“ in eclipse erstellt.

Zunächst wird im „Graphical Layout“ der „activity_layout_slide.xml“ die Oberfläche erstellt. Dazu wird die Palette eingesetzt. In dem Ordner „Images & Media“ der Palette wird die ImageView in den Handybildschirm gezogen. In dem erscheinenden Dialogfenster wählt man in den Projektressourcen das Bild des Fisches aus, das eingefügt werden soll.

Alternativ kann das Bild auch als „ImageView“ direkt in der „activity_layout_slide.xml“ Datei mit dem Befehl „`android:src="@drawable/fish"`“ eingefügt werden (hier ist der Name des Bildes: fish).

Wird das Bild direkt in der xml-Datei eingefügt, befindet sich das Bild im oberen linken Rand des Bildschirms. Wird es hingegen per Hand in den Bildschirm gezogen, ist die Position frei wählbar. Das zu bewegende Bild soll zentriert angezeigt werden. Dazu wird im ImageView der Befehl „`android:layout_centerInParent="true"`“ hinzugefügt. In diesem Fall ist der Parent die View, in dem das Bild nun zentriert plaziert ist.



Abbildung 18

Bild des Graphical Layouts mit der Palette zur Erstellung der Oberfläche

Der Befehl „centerInParent“ existiert nur im „RelativeLayout“, darum wird dieses Layout gewählt.

Um auch den Hintergrund dem Thema entsprechend anzupassen, wird hier ein Bild mit dem Befehl: „android:background="@drawable/sea"“ eingefügt. Dieses Bild befindet sich ebenfalls im „drawable“-Ordner des Projektes.

```
1① <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:background="@drawable/sea"
6     android:id="@+id/layout"
7     tools:context=".MainActivity" >
8
9      <ImageView
10        android:id="@+id/img"
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:layout_centerInParent="true"
14        android:src="@drawable/fish" />
15
16 </RelativeLayout|
```

Abbildung 19

Screenshot der „activity_layout_slide.xml“ Datei

Die Bewegung des Fisches wird in der „BaseSlide.java“-Datei implementiert. Sie wird mit Hilfe eines sogenannten „onTouch“-Listeners realisiert. Dieser ist in zwei Ausführungen im Programm vorhanden: Einmal ein „onTouch“-Listener für die ViewGroup (eine View, die andere Views beinhalten kann, in diesem Fall das Bild des Fisches) und für das Bild, welches bewegt werden soll.

Im „onTouch“-Listener des Bildes wird mit „MotionEvent.ACTION_DOWN“ überprüft, ob der Benutzer das Touch-Display berührt. Ist dies der Fall, wird ein neues Offset ermittelt. Mit diesem Offset wird im „onTouch“-Listener der ViewGroup die aktuelle Postion des Bildes berechnet.

Findet keine Berührung statt, passiert nichts.

```
ImageView img = (ImageView) findViewById(R.id.img); // fish image is initialized
img.setOnTouchListener(new View.OnTouchListener() { // onTouchListener for fish

    @Override
    public boolean onTouch(View view, MotionEvent event) {
        switch (event.getActionMasked()) {
            case MotionEvent.ACTION_DOWN: // if someone touches the fish
                offset_x = (int) event.getX(); // calculate new offset
                offset_y = (int) event.getY();
                selected_item = view; // view is selected
                break;

            default: // fish is not touched
                break;
        }
        return false;
    }
});
```

Abbildung 20

Der „onTouch“-Listener der ViewGroup wird in der Methode „onCreate“ gezetzt. Der Listener wird in dieser Methode aufgerufen, damit der Fisch nicht bei jeder beliebigen Berührung an die berührte Stelle springt.

In diesem Listener wird ebenfalls überprüft, ob eine Bewegung innerhalb des Bildes während einer Berührung stattfindet. Hierbei werden die Koordinaten des Bildes mit der Methode „getLocationOnScreen“ ermittelt. Diese Koordinaten werden von den Koordinaten des gedrückten Punktes abgezogen. Anschließend wird überprüft, ob die Bewegung innerhalb eines bestimmten Toleranzbereiches liegt.

```
(if (MotionEvent.ACTION_MOVE >= imageX - 70 &&
    MotionEvent.ACTION_MOVE < imageX + 70 || 
    MotionEvent.ACTION_MOVE >= imageY - 30 &&
    MotionEvent.ACTION_MOVE< imageY + 30).
```

Wird das Bild berührt, wird die Methode zum Bewegen des Fisches („moveImage“) aufgerufen. Die Methode „moveImage“ ermöglicht die Bewegung des Fischbildes. Wird das Bild bewegt, wird die neue Position des Fisches in der x- und der y-Achse jeweils mit dem im „onTouch“-Listener ermittelten Offsets berechnet. Um zu verhindern, dass der Fisch an allen vier Seiten aus dem Bildschirm herausgezogen werden kann, wird mit „get WindowManager().getDefaultDisplay().getWidth“ und „get WindowManager().getDefaultDisplay().getHeight“ die Breite und Höhe des Displays ermittelt. Mit diesen Werten ist es möglich, mit einer einfachen if-Abfrage zu ermitteln, ob der Fisch zu weit nach links/rechts oder oben/unten gezogen wurde.

Wird das Bild nicht bewegt, passiert nichts.

Es fällt auf, dass der Fisch bei Annäherung an den rechten und unteren Rand der App kleiner wird.

Durch Ersetzen von „android:layout_width="fill_parent““ und „android:layout_height="fill_parent““ durch „android:layout_width="match_parent““ und „android:layout_height="match_parent““ wird das Verkleinern des Fisches verhindert.

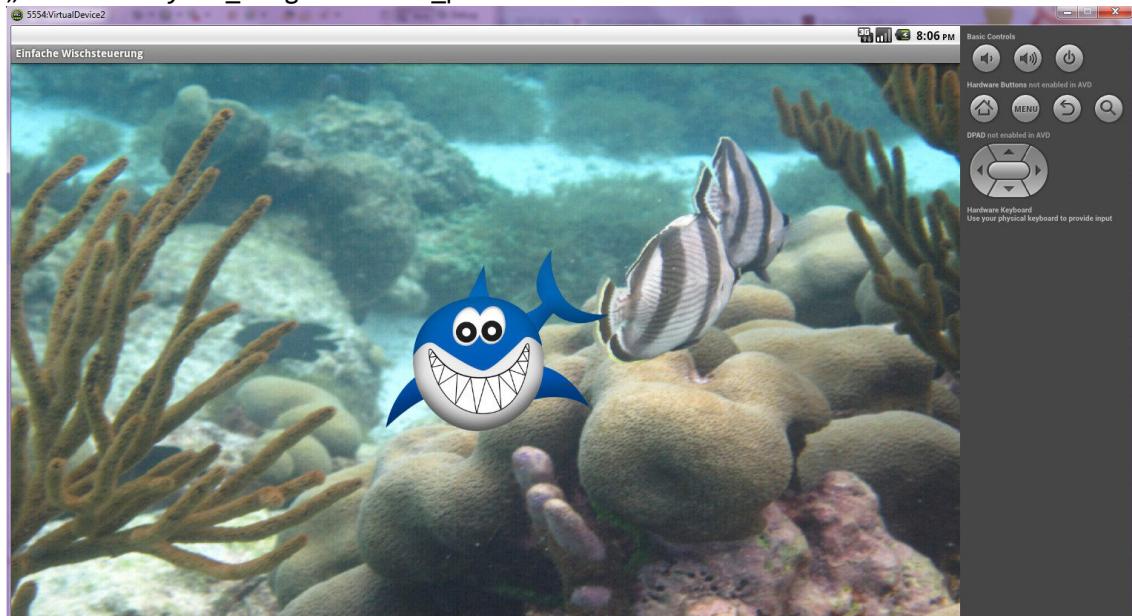


Abbildung 21
Bild der SlideApp

7.5 Steuerung durch Kippen

7.6 Lautstärkenkalibrierung

7.6.1 Oberfläche

von Anja Hafner

Ziel:

Ziel ist es, eine Oberfläche zur Lautstärkeneinstellung zu entwickeln. Diese ist nötig, um die korrekte Lautstärkeneinstellung für verschiedene Geräte zu ermitteln, da der Irdroid-Adapter je nach Gerät verschiedene Lautstärkeeinstellungen benötigt um ein Signal zu versenden. Das Fenster soll beim erstmaligen Öffnen der App erscheinen und die ausgewählte Lautstärke im Gerät einstellen und diese speichern. Nach dem erstmaligen Einstellen besteht die Möglichkeit, sie über das Menü bei Bedarf erneut zu ändern.

Vorgehensweise:

Die Anzeige des momentanen Lautstärkenlevels erfolgt über eine sogenannte „SeekBar“. Diese ist in der Palette der „activity_set_sound.xml“ Datei unter „Form Widgets“ zu finden. Die SeekBar wird mittig in den Handybildschirm platziert. Darüber steht als kurze Erklärung in einer TextView, dass bei Bewegung des Fisches auf OK gedrückt werden soll und unter der SeekBar ist der OK-Button platziert.



Abbildung 22

App zur Ermittlung der richtigen Lautstärke

Mit dem Befehl „`android:max="15"`“ in der „activity_set_sound.xml“ Datei wird der SeekBar ein maximaler Einstellungswert von 15 zugewiesen (da die Lautstärkenangabe bei Android-Geräten in einem Bereich von 0 bis 15 angegeben und eingestellt wird).

Um den ermittelten Wert der SeekBar zu erhalten, wird der SeekBar ein sogenannter „OnSeekBarChangeListener“ zugewiesen. Dieser zeigt an, ob der Fortschritt der

SeekBar verändert wurde. Diese Änderung wird mit der Methode „onProgressChanged (SeekBar seekBar, int progress, boolean fromUser)“ erkannt.

Mit einem „OnTouchListener“ und mit Hilfe der Methode „onTouch“, wird der Button zum Bestätigen der Bewegung implementiert.

Wird der Button gedrückt (if (event.getAction() == MotionEvent.ACTION_DOWN)) ändert er seine Farbe (als Feedback dass gedruckt wurde) und ruft den Fortschritt der SeekBar mit „bar.getProgress()“ auf.

Mit Hilfe des AudioManager kann die Lautstärke am Audiojack-Ausgang geregelt werden. Der Befehl

„audioManager.setStreamVolume(AudioManager.STREAM_SYSTEM, value, 0);“ setzt die Lautstärke auf den ermittelten Wert (value).

Wird der Button losgelassen (if (event.getAction() == MotionEvent.ACTION_UP)), ändert er lediglich seine Farbe.

```
61@    @Override
62    public void onClick(View v) {
63
64        if (v == okbutton) {
65            okbutton.setOnTouchListener(new OnTouchListener() { //create OnClickListener for okbutton
66
67                @Override
68                public boolean onTouch(View v, MotionEvent event) { //called when screen is touched
69
70                    if (event.getAction() == MotionEvent.ACTION_DOWN){ //if button is pressed
71                        okbutton.setBackgroundColor(Color.GRAY); //change button colour (so you can see that button has been pushed)
72                        int value = bar.getProgress();
73                        AudioManager audioManager = (AudioManager) getSystemService(AUDIO_SERVICE);
74                        audioManager.setStreamVolume(AudioManager.STREAM_SYSTEM, value, 0);
75
76                        return true;
77                    }
78                    else if (event.getAction() == MotionEvent.ACTION_UP){ //if button is released
79                        okbutton.setBackgroundColor(Color.LTGRAY); //change button colour again (so you can see that button has been released)
80                        return true;
81                    }
82
83                    return false;
84                }
85            });
86        }
87    }
88}
89}
```

Abbildung 23

Bild des „onTouch“-Listeners mit Setzen der Lautstärke

Korrekturen:

Es folgen noch ein paar kleinere Korrekturen.

So soll die SeekBar nicht per Hand veränderbar sein. Mit Hilfe des Befehls „setEnabled(false)“ ist eine manuelle Steuerung nicht mehr möglich.

Auch ist die momentane Beschreibung über der SeekBar nicht sehr eindeutig und der Benutzer weiß nicht genau, wie er sich verhalten soll. Zur Vermeidung unnötiger Verwirrung wird eine zusätzliche TextView mit einer näheren Beschreibung („Automatische Lautstärkeneinstellung, bitte warten.“) hinzugefügt.

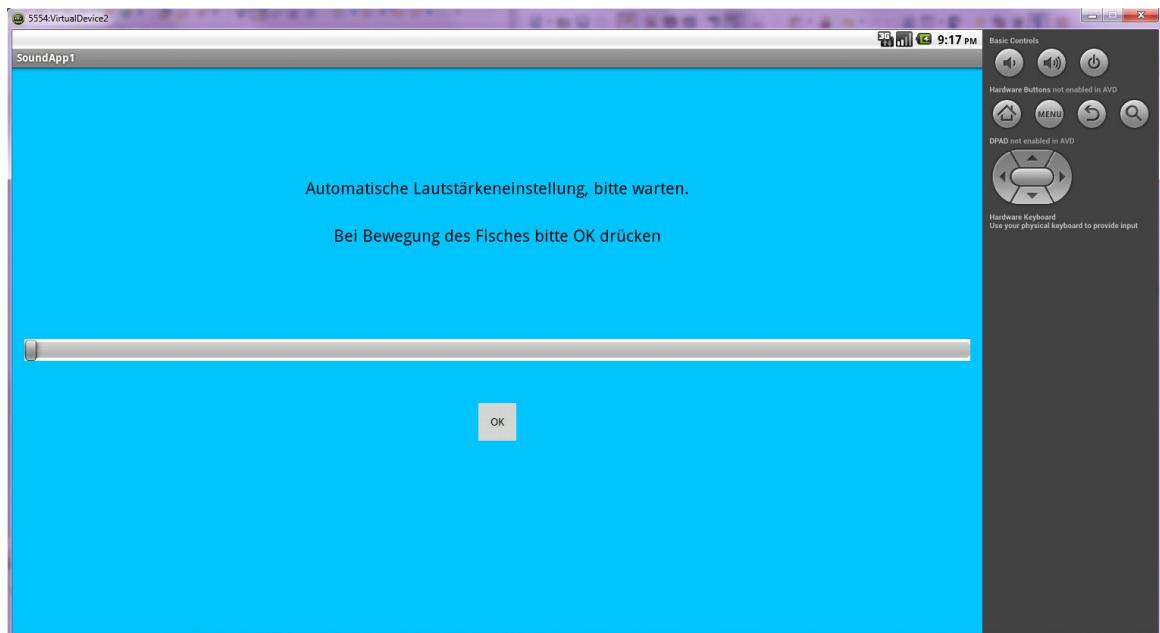


Abbildung 24

Bild der SoundApp mit erweiterter Erklärung

Für eine ansprechendere Optik wird der Hintergrund der Lautstärkeneinstellung an den Hintergrund der Startseite angepasst.

Mit dem Befehl „`android:background="@drawable/frontpage_background"`“ wird das Bild eingefügt.

Da die schwarze Schrift nicht sehr gut auf dem dunklen Hintergrund zu sehen ist, wird sie mit „`android:textColor="#f6f60e"`“ in gelb umformatiert.

Der Balken der SeekBar muss eine helle Farbe besitzen, da er sonst nicht erkennbar ist.

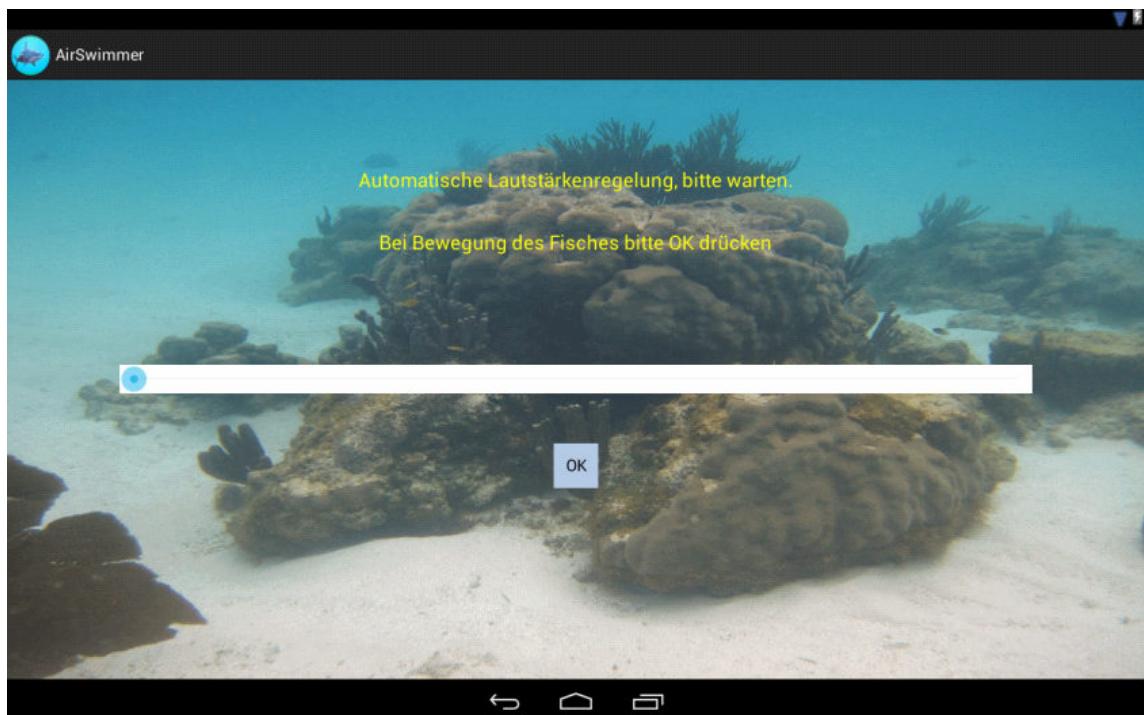


Abbildung 25

Lautstärkeneinstellung mit neuem Hintergrund

7.6.2 Logik

7.6.3 Einbau und permanentes Speichern

7.7 Menü

Steuerungsoberflächen

von Sabine Kressierer

Zielsetzung

In den verschiedenen Oberflächen zum Steuern des AirSwimmers soll ein Menü eingefügt werden, dass das Umschalten zwischen den einzelnen Steuerungsmodi sowie eine Auswahl des Hintergrund ermöglicht.

Vorgehensweise

Ich Ordner „res“ wurde ein Ordner „menu“ angelegt, der die XML-Dateien zum darstellen der Menüs enthält. Da die Menüpunkte der verschiedenen Oberflächen zum Steuern des AirSwimmers gleich sind, wurde für sie nur eine XML-Datei „control.xml“ erstellt. Diese enthält die Menüpunkte mit allen Unterpunkten wie hier für den Menüpunkt „Hintergrund ändern“:

```
<item android:title="@string/change_background"
      android:id="@+id/change_background">
    <!-- Submenu with different background pictures -->
    <menu>
        <group android:id="@+id/submenu_changeBackground" >
            <item
                android:id="@+id/water"
                android:title="@string/water_picture"/>

            <item
                android:id="@+id/sky"
                android:title="@string/sky_picture"/>
            <item android:id="@+id/th_picture"
                  android:title="@string/th_picture"/>
        </group>
    </menu>
</item>
```

In einer Basisklasse „ BaseActivity.java“, von der alle Activities zum Steuern des Airswimmers ableiten, wird die Logik für das Menü implementiert. Das Erzeugen des richtigen Menüs erfolgt durch überschreiben der onCreateOptionsMenu-Methode, in der den Activities mit Hilfe des Befehls
getMenuInflater().inflate(R.menu.control, menu) das Menü der Datei „control.xml“ zugewiesen wird. Die Reaktion auf die Auswahl der Menüpunkte erfolgt in der Funktion onOptionsItemSelected, die ebenfalls in der Basisklasse überschrieben wird. Auf diese Weise werden die nachfolgenden Menüpunkte realisiert.

a) Hintergrund ändern

von Sabine Kressierer

Zweck

Dem Benutzer soll die Möglichkeit zur Verfügung stehen zwischen verschiedenen Hintergründen für die Steuerungsoberflächen zu wählen.

Umsetzung

Der Menüpunkt „change_background“ enthält ein Untermenü mit den verschiedenen Hintergrundbildern. Wird einer dieser Punkte ausgewählt wird mittels `background.setBackgroundDrawable(source.getDrawable(R.drawable.ic_sky));` das entsprechende Bild (in diesem Fall ein Himmel) als Hintergrund verwendet. „background“ ist in diesem Fall das Layout der Activity. Zur einheitlichen Verwendung hat das Layout bei allen Activities die Id „layout“.

Besonderheit der Oberfläche Kippen: Da beim Kippen der Hintergrund anstatt einem einfachen Drawable eine BitMap ist wird hier das ändern des Hintergrunds überschrieben. Hier wird der Hintergrund geändert, indem das gewünschte Bild in einer Variablen „background“ des Threads zum Zeichnen der Oberfläche gespeichert wird. Das dort gespeicherte Bild wird in der run()-Methode des Threads gezeichnet.

b) Modus auswählen

von Sabine Kressierer

Zweck

Um die Bedienung zu vereinfachen soll man nicht nur über die Startseite sondern auch über das Menü zwischen den einzelnen Steuerungsmodi (Kippen, Wischen, Tasten) wechseln können

Umsetzung

Der Menüpunkt „change_mode“ enthält ein Untermenü mit den Namen der einzelnen Steuerungsarten. Wird ein Punkt ausgewählt, wird mittels `startActivity(new Intent(this, Activity_Buttons.class));` die entsprechende Activity gestartet. Um ein erneutes Auswählen der bereits laufenden Activity zu verhindern, wird bei Erzeugung des Menüs, der entsprechende Menüpunkt ermittelt und unsichtbar gemacht.

Besonderheit der Oberfläche Kippen: Da in dieser Oberfläche ein Thread verwendet wird, muss dieser bei Verlassen der Oberfläche gestoppt werden, da sonst auf einigen Geräten eine Fehlermeldung auftritt. Dies geschieht durch überschreiben der Methode `onOptionsItemSelected` in der entsprechenden Activities. Damit bei Rückkehr zur Oberfläche durch Betätigen des Zurück-Knopfes die im Thread realisierte Animation der Oberfläche wieder stattfindet, wird in der Methode `onResume()` dieser Thread wieder gestartet.

c) Steuerungsart ändern

von Caroline Pilot

Zielsetzung

Der Benutzer soll die Möglichkeit haben das Versenden von Signalen an den AirSwimmer über verschiedene Steuerungsarten zu realisieren.

Zum einen soll genau ein Signal (ein einziger kompletter Bewegungsbefehl) je Tastendruck ausgeführt werden (→ Einfach), zum anderen soll eine permanente Bewegung mit nur einem Klick ebenfalls möglich sein (→ Permanent). Diese Steuerungsart soll sowohl im Tastenmodus als auch im Wisch- und Kippmodus möglich sein.

Im Folgenden wird das Einfügen des Menüpunkts für die Tastensteuerung beschrieben, da die anderen Steuerungsmodi analog ablaufen.

Vorüberlegung

Da es sich zwar bei der Optik jeweils um die gleichen Klassen handelt (Einfache Tastensteuerung und Permanente Tastensteuerung), jedoch die Logik dahinter anders ist, muss die vorhandene Klasse geklont werden, damit ein Menüpunkt später dorthin verweisen kann.

Dieser Menüpunkt („Steuerungsart ändern“) muss als ein weiterer in die Liste eingetragen werden, wobei ein Untermenü zum Wechseln zwischen „Einfach“ und „Permanent“ ebenfalls implementiert werden muss.

Damit der Benutzer auch weiß, in welchem Modus er sich befindet, muss der Android Action Bar Text (Wird am oberen Bildschirmrand während jeder Aktivität angezeigt) angepasst werden.

Vorgehensweise

Erstellen des Menüpunkts „Steuerungsart ändern“

In der bereits vorhandenen *control.xml* Datei werden im Layout folgende Punkte hinzugefügt:

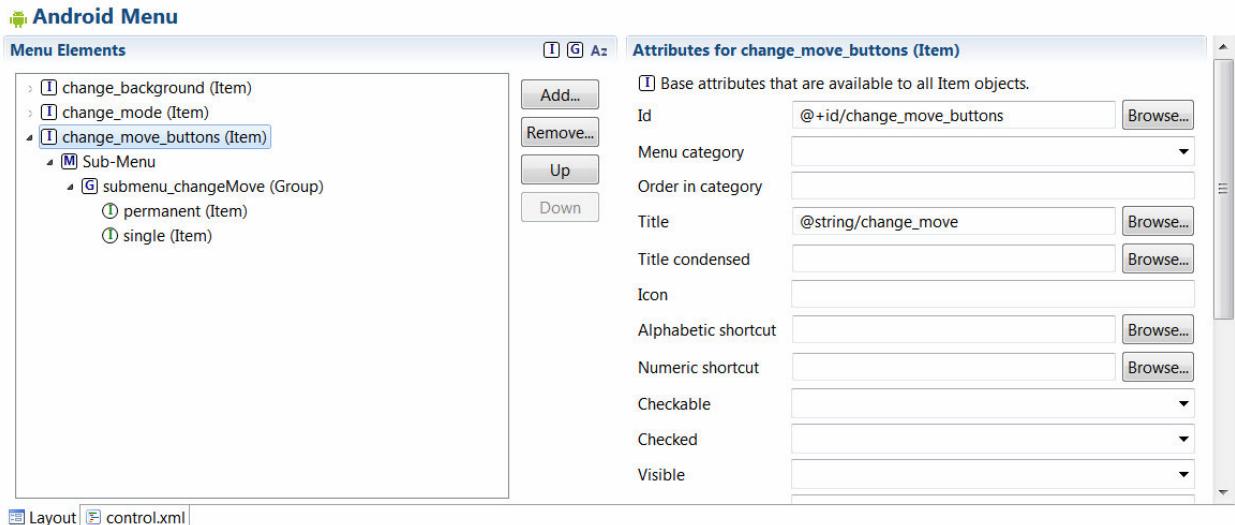


Abbildung 26

Menüpunkt für Steuerungsmodus hinzugefügt

Passende „id’s“ und „strings“ werden eingetragen. So findet sich unter *@string/change_move* der Text „Steuerungsart ändern“.

Hiermit hat man nun im Menü diesen Punkt, sollte dieser durch Klicken aufgerufen werden, so öffnet sich ein Untermenü mit den Unterpunkten „Einfach“ und „Permanent“.

Um auch die Logik dahinter zu realisieren wird in der BaseActivity Klasse strukturiert abgefragt, um welche Klasse es sich aktuell handelt:

```
if (currentActivityName.equals(getResources().getString(R.string.title_activity_activity_buttons))){...}
```

und ob der „Steuerungsart ändern“ Menüpunkt aufgerufen wurde:

```
if (item.getGroupId() == R.id.submenu_changeMove){...}
```

Sollte dies der Fall sein, so wird die Nutzereingabe über `switch (item.getItemId())` eingelesen und zur jeweils zugehörigen Klasse gesprungen:

```
case R.id.permanent:
    startActivity(new Intent(this, Activity_Buttons_Permanent.class));
    return true;
case R.id.single:
    startActivity(new Intent(this, Activity_Buttons.class));
    return true;
default:
    return false;
```

Ändern des Android Action Bar Texts

Der Android Action Bar Text kann im *AndroidManifest.xml* verändert werden.

Da hier ohnehin schon zu jeder Activity ein Codeblock mit einem Label (default: Namen der Klasse) generiert wurde kann dieser mit gebräuchlicheren und verständlicheren Namen ersetzt werden.

```
<activity
    android:name="de.airswimmer.gui.Activity_Buttons"
    android:label="Einfache Tastensteuerung"
    android:screenOrientation="Landscape" >
</activity>
```

Hier wird unabhängig von definierten Strings gearbeitet und macht ein Modifizieren und Erweitern einfach.

Ergebnis

Der Benutzer gelangt nun mit Hilfe eines „Steuerungsart ändern“ Menüpunkts zu der Auswahl, ob der aktuelle Modus in einer Permanenten oder Einfachen Steuerungsart bedient werden soll.

Optisch unterscheiden sie sich kaum, jedoch ist die Logik dahinter jeweils eine andere. Für eine bessere Orientierung sieht der Benutzer am oberen Bildschirmrand nun in welchem Modus und welcher Steuerungsart er sich befindet.

Startseite

Abfrage Aux-Anschluss

Von Sabine Kressierer

Zielsetzung

Die Ausrichtung der Oberfläche soll so erfolgen, dass sich der Aux-Anschluss immer an der oberen Seite des Tablets befindet. Somit kann der IrDroid Sender direkt auf den AirSwimmer gerichtet werden.

Umsetzung

Bei erstmaligem Öffnen der App wird zuerst ein Dialog geöffnet, in dem angegeben werden muss wo sich der Aux-Anschluss befindet. Dies geschieht mit Hilfe eines AlertDialog, der in der onCreate()-Methode der FrontPage-Activity erzeugt und angezeigt wird. Die Reaktion auf die Auswahl eines Punktes erfolgt mittels onClick-Listener, in dessen onClick-Methode der Index des gewählten Punktes bekannt ist. Entsprechend dieser Auswahl wird die ActivityInfo-Konstante für das entsprechende Layout gespeichert.

```
if (item == 0) { //get value for requested screen orientation  
    layout = ActivityInfo.SCREEN_ORIENTATION_PORTRAIT;  
} else if (item == 1) {  
    layout = ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE;  
}1
```

Da sich die Position des Aux-Anschlusses nicht verändert, wird dieser Wert dauerhaft gespeichert. Dazu werden shared-preferences verwendet, die es ermöglichen innerhalb der gesamten App auf den Wert zuzugreifen. Mit Preferences = getSharedPreferences("AirSwimmerPrefs", Context.MODE_WORLD_READABLE); werden die preferences in der Datei „AirSwimmerPrefs“ ermittelt. In diesen Preferences wird dann mit dem SharedPreferences.Editor der int-Wert zum Schlüssel „layout“ auf die oben ermittelte Konstante gesetzt. Am Ende wird in der onCreate-Methode der Startseite mittels setRequestedOrientation(layout); die Orientierung entsprechend verändert. Mit der selben Funktion wird in der onCreate-Methode der „ BaseActivity“ die Orientierung gesetzt. Um dort den in den Preferences gespeicherten Wert zu erhalten, werden wie oben die Preferences ermittelt und anschließend preferences.getInt("layout", -1) aufgerufen.

Bei erneutem Starten der App wird der Dialog nur noch gestartet, wenn in den Preferences kein Wert gespeichert ist.

Menü der Startseite

Von Sabine Kressierer

Zweck

Die Ausrichtung der App soll nachträglich veränderbar sein.

Umsetzung

In der Datei „start.xml“ wurde ein Menüpunkt „change_layout_orientation“ eingefügt, mit einem Untermenü für die beiden Orientierungsmöglichkeiten. Das Erzeugen des Menüs erfolgt in der FrontPage-Aktivity analog zum control-Menü. Bei Auswahl einer

¹ 0=Aux-Anschluss an kurzer Seite → Portrait / 1= Aux-Anschluss an langer Seite → Landscape

Orientierung im Untermenü, wird dieser Wert wie beim AlertDialog in den Preferences gespeichert und die Ausrichtung der FrontPage angepasst.

8 Verbindung von Oberfläche und Senden

8.1 Permanente Bewegung

Von Caroline Pilot

Zielsetzung

Der Permanent-Modus soll dazu dienen, dass der AirSwimmer in einer kontinuierlichen Bewegung geradeaus schwimmt.

Somit soll sich der Benutzer, nach dem Betätigen des Start/Stopp Buttons (siehe Abbildung 27

Oberfläche des Permanent-Modus, in der Mitte ist der Start/Stopp Button zu sehen), nicht mehr um die Fortbewegung im Allgemeinen kümmern müssen, sondern lediglich um die speziellen Richtungswechsel.

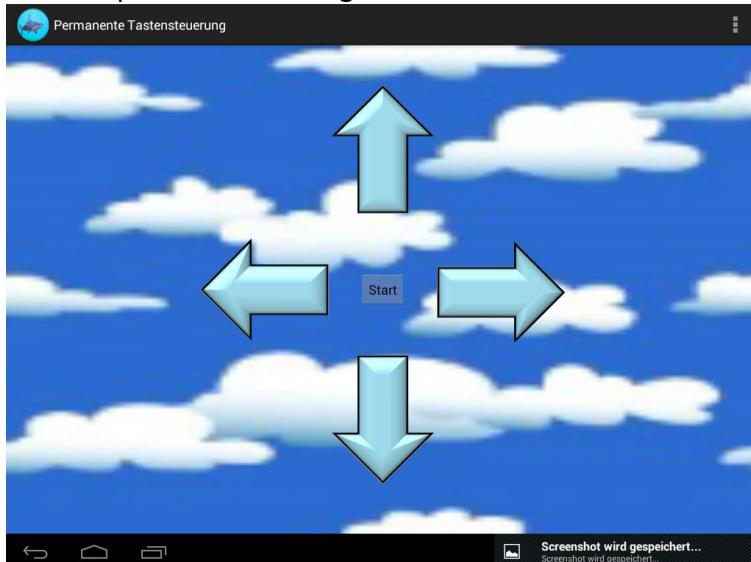


Abbildung 27

Oberfläche des Permanent-Modus, in der Mitte ist der Start/Stopp Button zu sehen

Vorüberlegung

Im Allgemeinen muss zur permanenten Bewegung folgende Punkte in dieser Reihenfolge bearbeitet werden:

1. Eine Linksbewegung der Schwanzflosse
2. Eine kurze Wartezeit von etwa 5sec , damit die Schwanzflosse wieder zur Mitte zurückkehren kann
3. Eine Rechtsbewegung der Schwanzflosse

Wiederholen der Punkte 1 – 3, solange bis der Button erneut gedrückt wurde.

Um dies zu realisieren gibt es im Grunde zwei Möglichkeiten:

1. Eine einfache do-while Abfrage

2. Ein Runnable-Aufruf

Vorgehensweise

1. Realisierung über eine einfache do-while Abfrage

Da bereits eine Methode implementiert ist, die lediglich den Schriftzug des Buttons beim Drücken ändert, bietet es sich an, die hier markierten „// TODO“ Stellen zu füllen.

```
-- 17 + public void changeMoveState(View view){  
18 +     Button button_start = (Button) view;  
19 +     if(forwardMovement){  
20 +         forwardMovement=false;  
21 +         button_start.setText("Start");  
22 +         //TODO stop forward movement  
23 +     }  
24 +     }else{  
25 +         forwardMovement=true;  
26 +         button_start.setText("Stop");  
27 +         //TODO start forward movement  
28 +     }  
29 + }  
30 + }
```

Abbildung 28

Hierfür wird eine neue Methode „swim()“ geschrieben, die genau die oben genannten Punkte abarbeitet:

```
public boolean swim() {  
  
    moveLeft();  
    action.finishMovingLeft();  
    SystemClock.sleep(waiting_time);  
    moveRight();  
    action.finishMovingRight();  
  
    return false;  
}
```

Die verwendeten Befehle (`moveLeft();action.finishMovingLeft(); ...`) sind Funktionen der Klasse, die das Senden realisiert.

Mittels einer do-while Schleife soll nun die `swim()` Methode aufgerufen werden, solange (while) der `Button_Start` Button wieder gedrückt wird.

2. Realisierung über das Runnable-Interface

Eine weitere Möglichkeit eine permanente Bewegung zu realisieren, besteht darin, ein Runnable-Interface zu starten, sobald der Button gedrückt wird.

Hierin soll die Selbe Reihenfolge der Punkte abgearbeitet werden, wie oben

beschrieben. Somit läuft das Interface parallel zur restlichen Bewegung bzw. zu den anderen Aktionen der Oberfläche und beendet sich, sobald der Button erneut gedrückt wurde.

Das Verhalten eines Runnable-Interfaces, lässt sich durch das Überschreiben der „run()“-Methode modifizieren:

So wird auch hier eine do-while Schleife über die Signale für die Links- und Rechtsbewegung gelegt. Das Warten wird mittels einer try-catch Abfrage behandelt und lässt das Runnable-Interface für 5 s schlafen.

Ergebnis

Das Ergebnis beider Methoden ist leider ganz und gar nicht zufriedenstellend.

1. Realisierung über eine einfache do-while Abfrage

Das Testen ergibt leider, dass die Idee, einer simplen do-while Abfrage, in einer Endlosschleife endet, da während dieser Ausführung der Rest der Oberfläche nicht mehr läuft und somit kann ein Drücken des Buttons nicht mehr wahrgenommen wird.

2. Realisierung über das Runnable Interface

Auch hier wird leider kein befriedigendes Ergebnis erreicht.

Nach vielen Modifikationen werden nie alle Fehler beseitigt und ein fehlerfreies Laufen möglich gemacht.

Zum einen ist auch hierbei wieder die restliche Oberfläche eingefroren, was dazu führt, dass das Interface nie beendet wird.

Zum anderen wird, sobald der Runnable-Aufruf in den Code eingefügt wurde, nicht mal mehr der Schriftzug des Buttons verändert, obwohl ein Runnable-Aufruf noch gar nicht stattfand.

Außerdem erlangt man undefiniertes bzw. nicht-deterministisches Verhalten, was das Senden der Befehle angeht. Sie werden, teils in großen, teils in sehr kurzen Abständen aufgerufen.

Eine weitere Realisierungsmöglichkeit für den permanenten Modus ist leider aus Zeitgründen nicht mehr möglich gewesen.

Ausblick

Um diese Funktion zu implementieren wäre ein weiterer Versuch gewesen, einen „Prioritäten-Handler“ zu implementieren, der die Oberfläche, parallel zum Permanenten Geradeaus schwimmen, parallel zum Senden der Signale, verwaltet.

9 Graphische Gestaltung der Oberfläche

9.1 Konzept

Von Caroline Pilot

Zielsetzung

Ziel eines Prototyps für das Design der Oberfläche² ist es, der Gruppe einen ersten visuellen Eindruck für die App zu geben. So bekommt das gemeinsame Ziel des Teams ein Gesicht, auf das motivierter hingearbeitet werden kann.

Außerdem bieten Modelle eine bessere Diskussionsgrundlage für Wünsche, Ideen und Verbesserungsvorschläge. So kann die Oberflächenentwicklung mit konkreten Designs ausgeführt werden.

Vorüberlegung

Zunächst muss ein Graphikprogramm gefunden werden, das den Anforderungen entspricht.

Hierbei ist folgendes zu beachten:

- Vielerlei Möglichkeiten Bilder zu bearbeiten (Kontraste, Layer, Transparenz, ...)
- Erstellen von Graphiken in gängigen Bildformaten
- Einfache Bedienung, damit keine lange Einarbeitungszeit nötig ist
- Kostengünstig, am besten ein Freeware Download Programm

Des Weiteren muss der Kreativität bei folgenden Punkten freien Lauf gelassen werden:

- Wie soll der Hintergrund aussehen?
- Wie soll das Objekt dargestellt werden?
- Welche Besonderheiten benötigen die einzelnen Benutzermodi?

Vorgehensweise

Unter [http://www\(chip.de/downloads/PhotoFiltre_13012070.html](http://www(chip.de/downloads/PhotoFiltre_13012070.html)) ist das Freeware Graphikprogramm als Download verfügbar. Es entspricht unseren Anforderungen und enthält zudem sehr gute Nutzerrezensionen.

Aus dem Brainstorming zum Thema Oberflächendesign ergibt sich folgendes:

- Hintergrund:
 - o Himmel
 - o Unterwasserwelt
 - o Stadt/Skyline
- Objekt
 - o Bild des AirSwimmers
 - o Bild eines Fisches
 - o Bild eines Köders, dem der echte AirSwimmer „hinterher schwimmt“
- Besonderheiten einzelner Modi
 - o Tastensteuerung
 - Tasten in gewöhnlicher Buttonform
 - Tasten in Form eines Fisches
 - Kontrastreiche Farben <-> Gedeckte Farben
 - o Kippsteuerung

² Hiermit sind im Kapitel 5.1 „Konzeptentwicklung für das Oberflächendesign“ die unterschiedlichen Benutzermodi zum Steuern des AirSwimmers gemeint, nicht die Startseite der App

- Angabe von Himmelsrichtungen (north, south, west, east)
 - Angabe von Bewegungsrichtung (up, down, left, right)
 - Keine Angabe
- Wischsteuerung
 - Angabe von Himmelsrichtungen (north, south, west, east)
 - Angabe von Bewegungsrichtung (up, down, left, right)
 - Keine Angabe

Ergebnis

Bilder zum Verwirklichen dieser Ideen und zum Erstellen von Designprototypen lassen sich im Internet finden oder mit dem Graphikprogramm erstellen.

In verschiedenen Kombinationen, umgesetzt mit dem Photofiltre7 ergeben sich folgende Entwürfe:

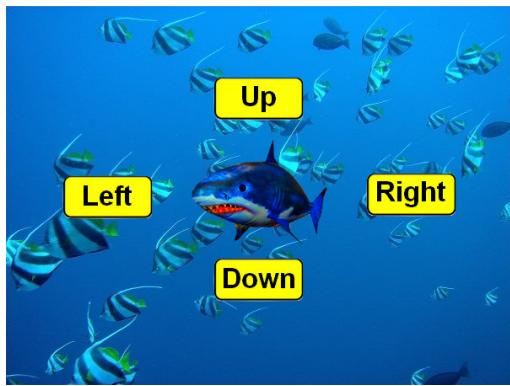


Abbildung 29

Tastensteuerung;
Unterwasserwelt, Kontrastreiche Buttons



Abbildung 30

Tastensteuerung;
Himmel, Gedeckte Buttons



Abbildung 31

Kippsteuerung;
Unterwasserwelt, Angabe von Himmelsrichtungen

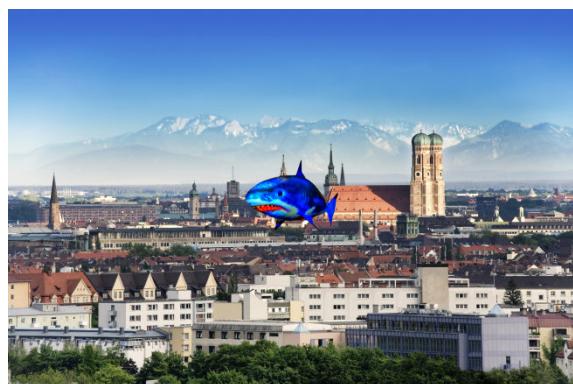


Abbildung 32

Kippsteuerung;
Skyline, ohne Angaben



Abbildung 33

Wischsteuerung
Benutzergrenzen, für definierte Bewegungen, Köder als Objekt



Abbildung 34

Wischsteuerung
Benutzergrenzen, für definierte Bewegungen

Die Gruppendiskussion bringt hervor, dass es in der App einen Punkt für einen Wechsel des Hintergrunds zwischen Himmel und Unterwasserwelt geben wird, je nach Wunsch des Benutzers.

Des Weiteren werden nicht die Himmelsrichtungen im Kippmodus angegeben, denn dies könnte zu Verwirrungen führen, stattdessen werden keine Angaben platziert, ebenso im Wischmodus.

Das Objekt in der Mitte soll ein Fisch sein.

Ausblick

Da es sich bei den verwendeten Bildern um Eigentum Dritter handelt, könnte man sich durch dessen Verwendung rechtlich Probleme einhandeln. Aus diesem Grund ist es wünschenswert, eigene Graphiken, vielleicht auch dynamische, zu erstellen oder die Urheber schriftlich um eine Genehmigung zu bitten.

9.2 Launcher-Icon

von Caroline Pilot

Zielsetzung

Um einen einheitlichen Auftritt der App zu ermöglichen, wird ein Launcher-Icon benötigt. Dieser repräsentiert die App auf dem Bildschirm und ist auch bei der Ausführung am Rand zu sehen.

Vorüberlegung

Da bereits das Graphikprogramm PhotoFiltre vorhanden ist, kann dieses verwendet werden, um den Launcher-Icon zu erstellen.

Des Weiteren muss der Kreativität bei folgenden Punkten freien Lauf gelassen werden:

Welche Form soll der Icon bekommen?

Welches Objekt soll darauf zu sehen sein?

Welche Farben könnten passen?

Vorgehensweise

Aus dem Brainstorming ergeben sich folgende Punkte:

Form

Rund, da es einer Wasserblase entspricht, in dem sich der AirSwimmer fortbewegt

Objekt

Da es eine App für den AirSwimmer sein soll, liegt es nahe, dass auch auf dem Icon der Fisch abgebildet ist. Somit wäre es eindeutig auf dem Desktop zu identifizieren

Farben

Die Farben müssen herausstechen, sodass die App nicht auf dem Desktop untergeht und man sie nicht sehen kann (Beispiel schwarz), dennoch müsste die Farbe zum Thema Wasser/Himmel passen

Ergebnis

Das Bild des AirSwimmers lässt sich im Internet finden.

In verschiedenen Kombinationen mit Farben und Hintergründen, umgesetzt mit dem Photofiltre7 ergeben sich folgende Entwürfe:

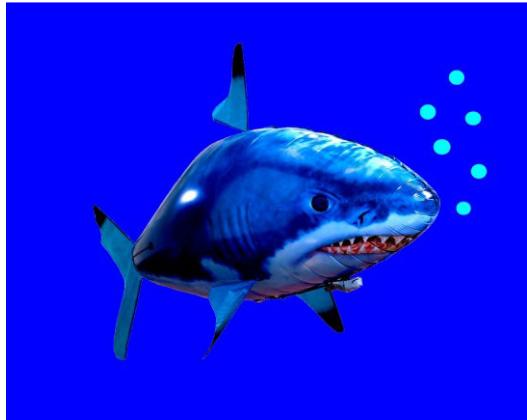


Abbildung 35

Launcher-Icon mit dunklem Hintergrund

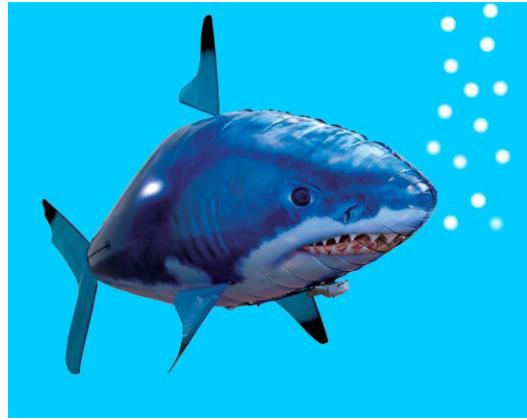


Abbildung 36

Launcher-Icon mit hellem Hintergrund

Nach dem Testen auf dem Tablet wird festgestellt, dass der dunkle Hintergrund bei einem schwarzen Desktop untergeht, daher fiel die Entscheidung auf einen runden Launcher-Icon mit hellem Hintergrund (Abbildung 35 Fehler! Verweisquelle konnte nicht gefunden werden.).

9.3 Animierter Hintergrund

9.4 Hintergrundbilder

9.5 Graphiken

9.5.1 Startseite (Buttons)

Von Belgüzar Kocak

Zielsetzung

Die Aufgabe bestand darin, die Startseite zu verschönern und die Standard Buttons auszutauschen, da diese wie Balken über dem Bildschirm lagen. Aufgrund dessen waren größere Buttons und auch für die Startseite passende Buttons notwendig. Ziel war es Hai-Buttons für die Startseite zu erstellen.

Vorgehen

Bevor die Buttons ausgetauscht werden konnten, mussten sie in der XML-Datei in ImageButtons umgewandelt werden. In diese ImageButtons wurde eine Hai Grafik eingefügt und der Hintergrund der Buttons ausgeblendet damit diese die Form eines Haies haben.

In der XML-Datei wurden die normalen Buttons rausgelöscht und neue ImageButtons in die Oberfläche gezogen und die entsprechenden Images an die Buttons zugewiesen. Vorab wurde die Grafik im res Ordner unter drawable-hdpi als air_swimmer_shark gespeichert.

Diese wurden dann mittels folgender Anweisung den einzelnen Buttons zugewiesen:
`android:src="@drawable/air_swimmers_shark"`

Der Hintergrund wurde transparent gestaltet um einen Hai-Förmigen Button zu erstellen

```

        android:background="#00000000"

<ImageButton
    android:id="@+id/Button_Slide"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:background="#00000000"
    android:onClick="onButtonClick"
    android:scaleType="center"
    android:src="@drawable/air_swimmers_shark" />

```

Die Beschriftung der einzelnen Buttons wurde anhand einer TextView dargestellt.
Dabei wurde die Schriftgröße, Schriftart und der Text der View implementiert.

```

<TextView
    android:id="@+id/textView2"
    android:layout_width="1189dp"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/textView3"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/Button_Slide"
    android:layout_toRightOf="@+id/Button_Button"
    android:text="@string/button_wischen"
    android:textSize="30sp"
    android:textStyle="bold"
    android:typeface="sans" />

```

Auch wurde der Hintergrund der Startseite verändert.

```
    android:background="@drawable/frontpage_background"
```

Am Ende hatten die Buttons der Startseite folgendes Aussehen.



Abbildung 37

9.5.2 Steuerungsseiten (Buttons und Hai)

Von Belgüzar Kocak

Zielsetzung

Ziel war es für die Oberfläche Kippen und Wischen eine Grafik von einem Hai zu erstellen und auch Buttons für die Buttons Oberfläche zu zeichnen. Genutzt wurden hier folgende Tools: Microsoft Word, Paint.Net und Photoshop.

Vorgehen und Erstellen

Verlangt wurde, dass der Hai direkt den Benutzer der Oberfläche anschaut. Aufgrund dessen wurden vorerst Zeichnungen per Hand erstellt genutzt wurden dabei ein einfaches Papier so wie Bleistift. Somit war der erste Schritt der Zeichnung getan. Jetzt war es wichtig die Zeichnung vom Papier auf den Bildschirm als Grafik zu verwirklichen. Dies gelang mit dem Photoshop Programm.

Somit entstand folgende Grafik:

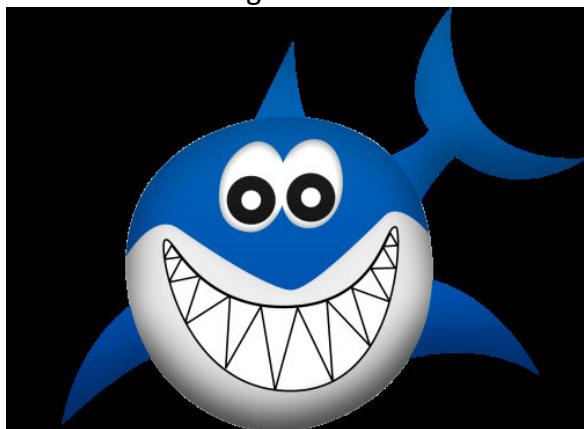


Abbildung 38

Die Pfeile wurden in Microsoft Word erstellt, die Pfeilform wurde in die passenden Größe skaliert. Nach der Skalierung wurde unter „Form formatieren“ in 3D-Format passende Werte eingegeben sowie Farben geändert. Anschließend in Paint.Net der Hintergrund mithilfe der Zauberstabfunktion entfernt.

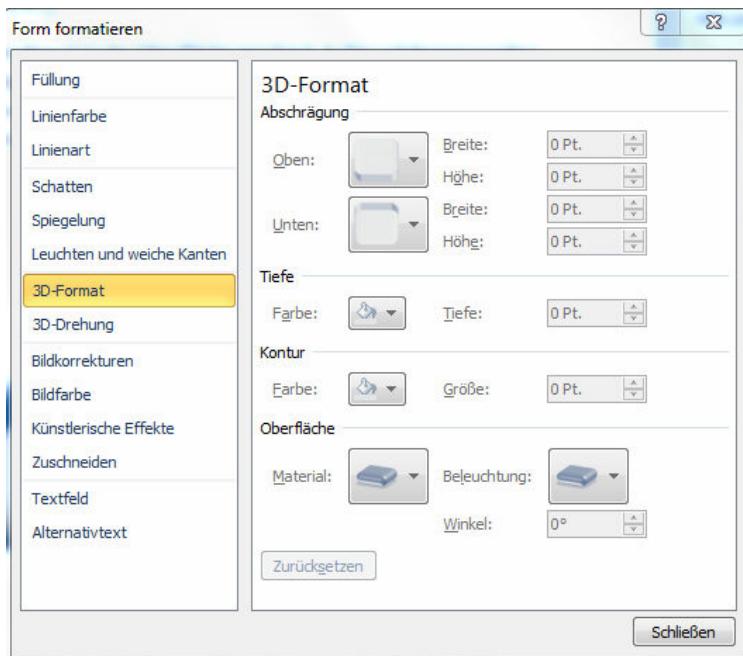


Abbildung 39

Nach dem Erstellen der Grafiken wurden diese in die Oberfläche eingebaut. Wie zum Beispiel in die Buttonoberfläche.

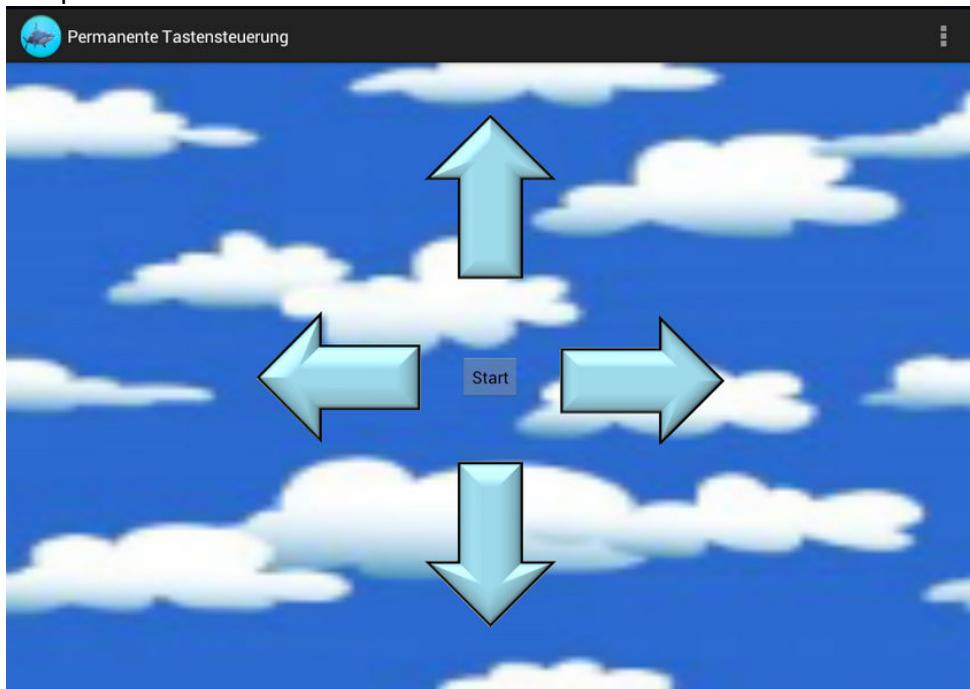


Abbildung 40

10 Gehäuse

10.1 Recherche Gehäusemöglichkeiten

10.2 Gehäuse bemalen und verschönern

Von Belgüzar Kocak

Nachdem ein passendes Gehäuse gefunden wurde und auch der Anfang zur Gestaltung von Andreas Gerken gegeben wurde, musste dieses noch farblich gestaltet werden. Als erstes wurden die Plastikflossen gefeilt anschließend mit Tesa Sekundenkleber angebracht. Die farbliche Gestaltung wurde mit Acrylfarbe und Naturschwamm erarbeitet.

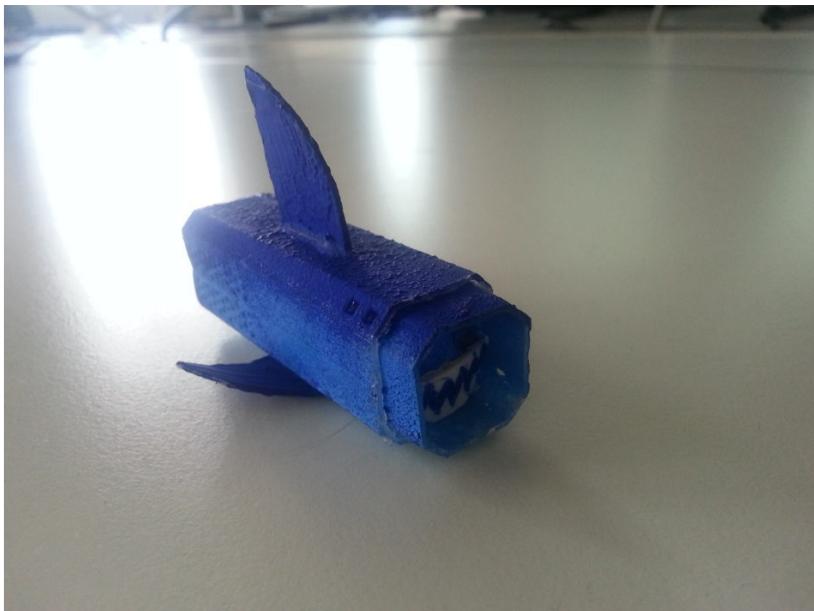


Abbildung 41
fertiges Gehäuse für Irdroid-Adapter

11 Sonstiges

11.1 Linux Library

Von Andreas Gerken

Um in einer App nativen Code verwenden zu können, müssen die C-Dateien zu einem .so File kompiliert werden. SO heißt Shared Object Library und ist das Linux Äquivalent zu DLLs auf Windows Systemen. Die Libraries speichern alle referenzierten Files mit ab und Verweise werden nur relativ gemacht, sodass die Library an einen beliebigen Ort verschoben werden kann. Da die Hardware Unterstützung für den Irdroid Adapter in C geschrieben ist, und die Irdroid App sie genau über so ein so File einbindet bietet sich das für unsere App genauso an.

Um die Shared Object Library zu verwenden, muss in Java das JNI (Java Native Interface) verwendet werden, das die Verbindung zwischen Java und nativ darstellt. Die Library kann über „System.loadLibrary(“<Name der Bibliothek“) geladen werden und die Prototypen der Funktionen müssen mit native deklariert werden. Der GNU Compiler GCC kann solche .SO Dateien erstellen. Dazu muss man bei der Erstellung der Objekt dateien die Option –fPIC aufrufen. Sie macht, dass der Code unabhängig der Referenzen ist.

```
gcc -c -fPIC datei.c -o datei.o
```

Die so erzeugte Objekt Datei kann nun über den folgenden Aufruf in eine Library eingebunden werden.

```
gcc -shared -Wl,-soname,libdatei.so.0.1 -o libdatei.so  
datei.o
```

Das Attribut –shared macht, das seine Library erstellt wird. Mit –soname kann ein Name eingestellt werden, der bei einer Referenzierung auf die Library verwendet werden muss. Er muss mit lib anfangen muss, gefolgt von einem Namen und beendet mit .so.<Versionsnummer>. Die Versionsnummer soll immer inkrementiert werden, wenn das Interface geändert wird. Libdatei.so ist der Name den die neu Kompilierte Datei bekommen soll.

So ist es möglich einzelne Dateien zu kompilieren. Wenn mehrere Dateien kompiliert werden sollen, müssen sie entweder im Aufruf angegeben werden, oder der Prozess wird über ein Makefile gesteuert. Die Irdroid App liefert einige Makefiles mit, die jedoch alle sehr kryptisch sind. Sie sind nicht dokumentiert und es gibt Mehrere die untereinander verlinkt sind. Es wird eine fertige Library mitgeliefert, die zwar auch Funktionen umfasst, die wir nicht benötigen und unserer Namenskonvention nicht entspricht, jedoch ist sie nur 180KB groß und die zusätzlichen Funktionen stören insofern nicht. Um die Namen zu ändern muss neu kompiliert werden und die Namen in die App auch noch ein gepflegt werden. Diese Erkenntnisse wurden in der Projektbesprechung vorgestellt und es wurde beschlossen, dass der Aufwand nur um die Namen zu ändern zu groß wäre. Somit wurde die orginal Library verwendet.