iDogstra

itestra Coding Camp 2017

- Projekt 1 -

Betreuer	Dr. Christian Rehn, Ingo Schnabel
Gruppenteilnehmer	Dennis Ciba, Jan Oliver Rollmann, Lukas Karnowski, An Ngo Tien, Johannes Vedder, Paul Dieterich

01.09.2017

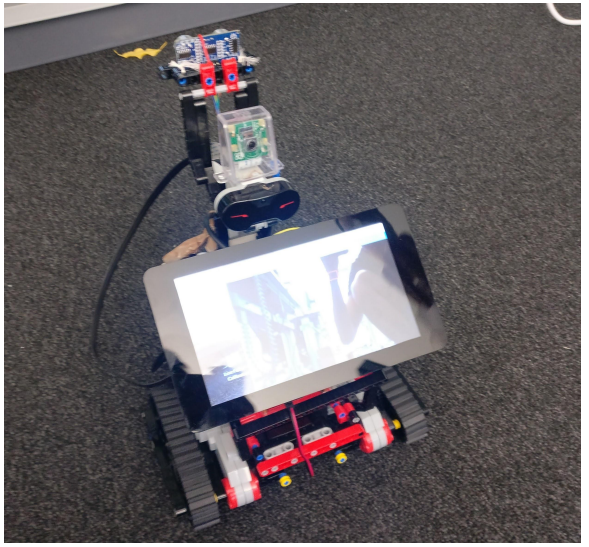
Gliederung

itestra
Software Productivity

1. Features
2. Hardware und Technologie
3. Roboter Demo
4. Aufbau Roboter
5. Arbeitsprozess
6. Experimente
7. State Machine
   - 7.1. iDogstra Statemachine
   - 7.2. Pipelines
   - 7.3. Bildverarbeitung
   - 7.4. Gesten
8. Fazit
9. Ausblick und Möglichkeiten
10. Q & A

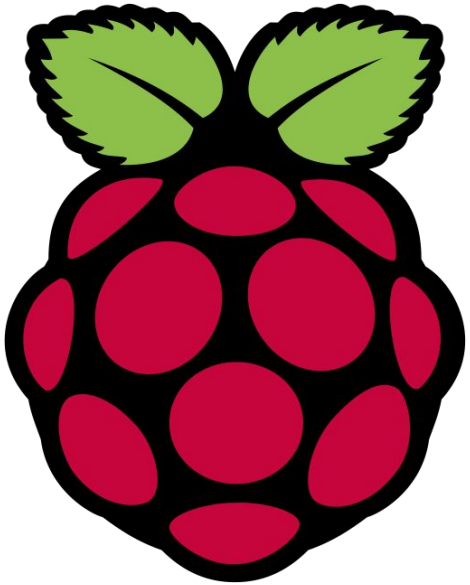
Features

- Mobiler Roboterhund zum Verfolgen einer Person
- Selbstständige Erkennung von Kollisionen
- Dynamische Geschwindigkeit beim Verfolgen
- Display zum Anzeigen des aktuellen Zustands



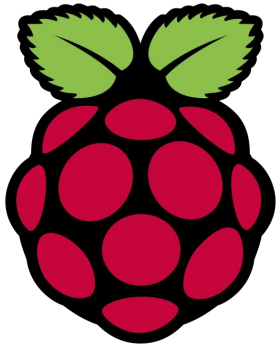
Verwendete Hardware

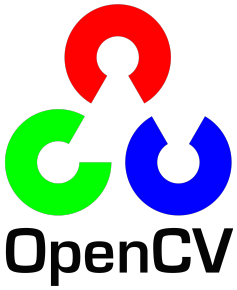
- RaspberryPi V3 & BrickPi
- Diverse USB Geräte
  - Kamera
  - Ultraschallsensor
  - Infrarotsensor
  - Bluetooth Dongles
- Display
- Lautsprecher

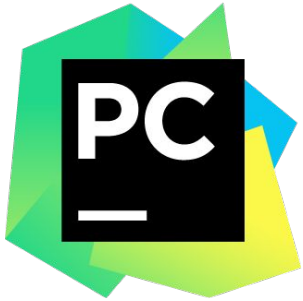


Verwendete Technologien

itestra
Software Productivity

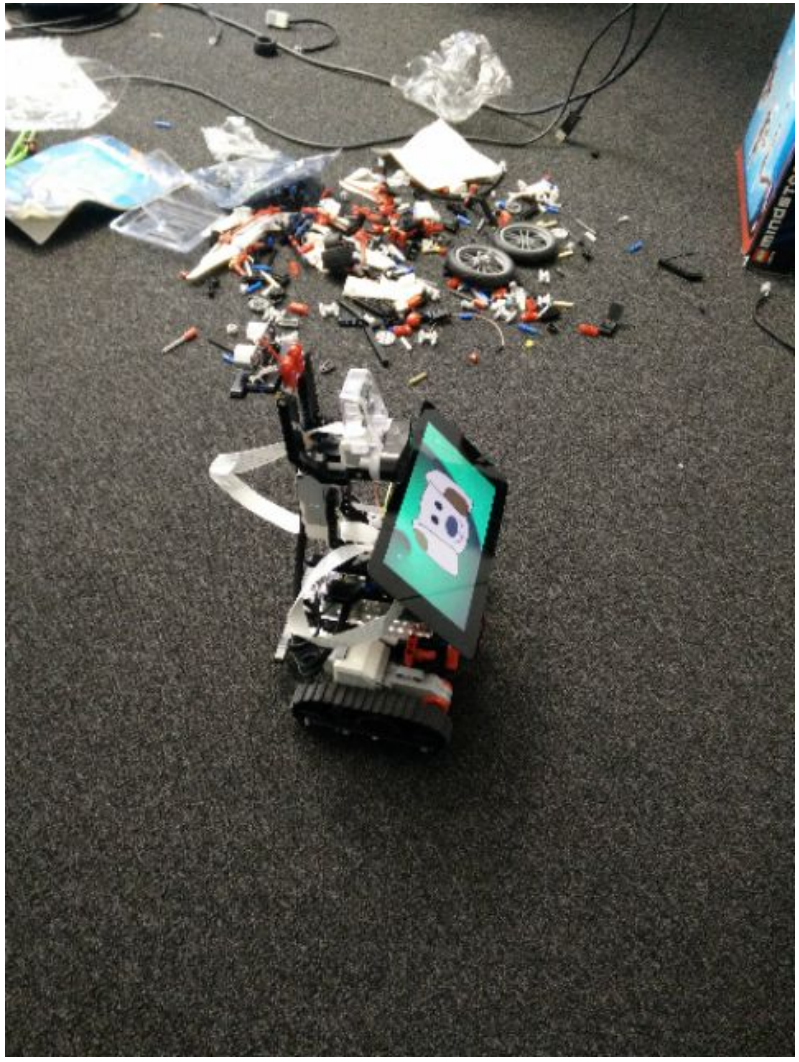




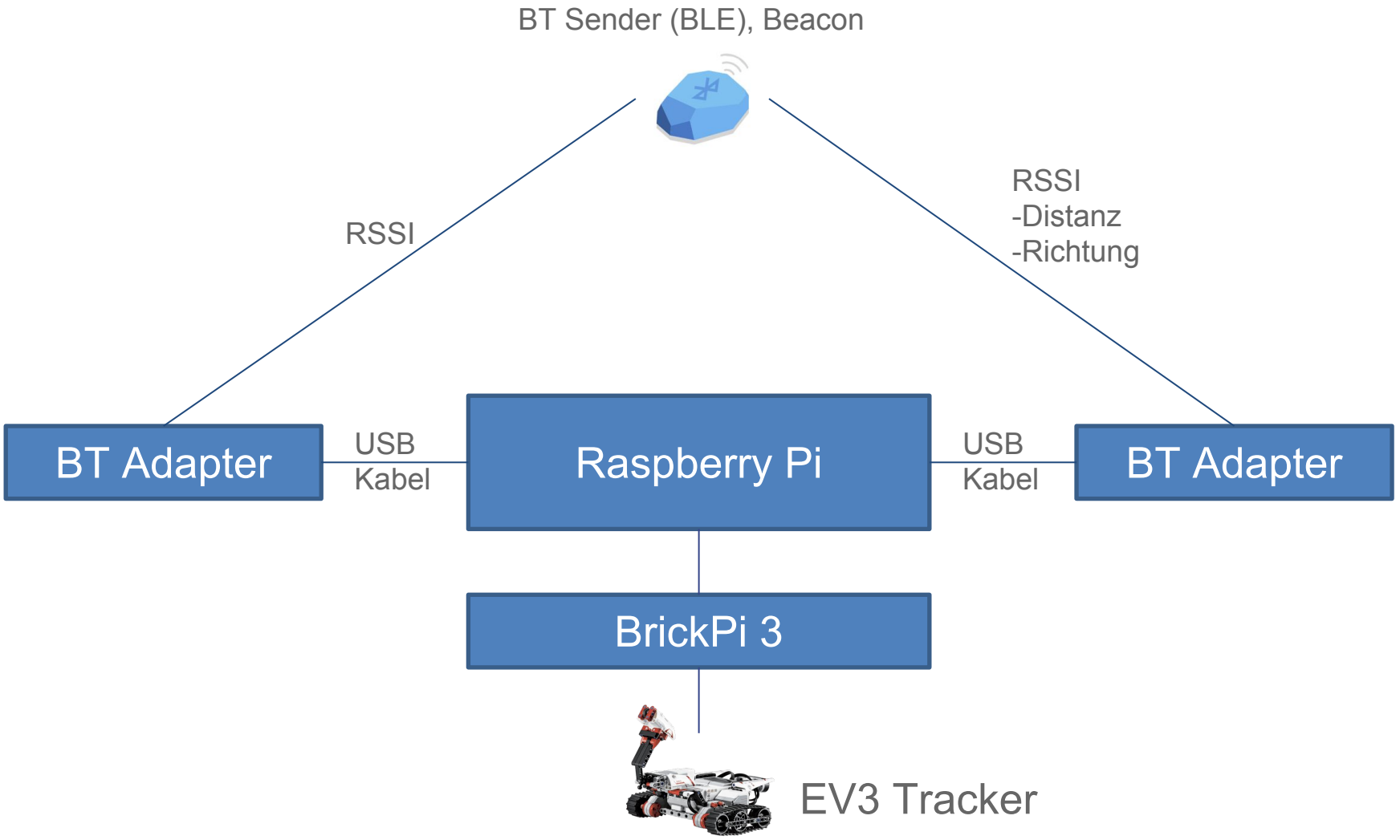


DEMO

itestra
Software Productivity

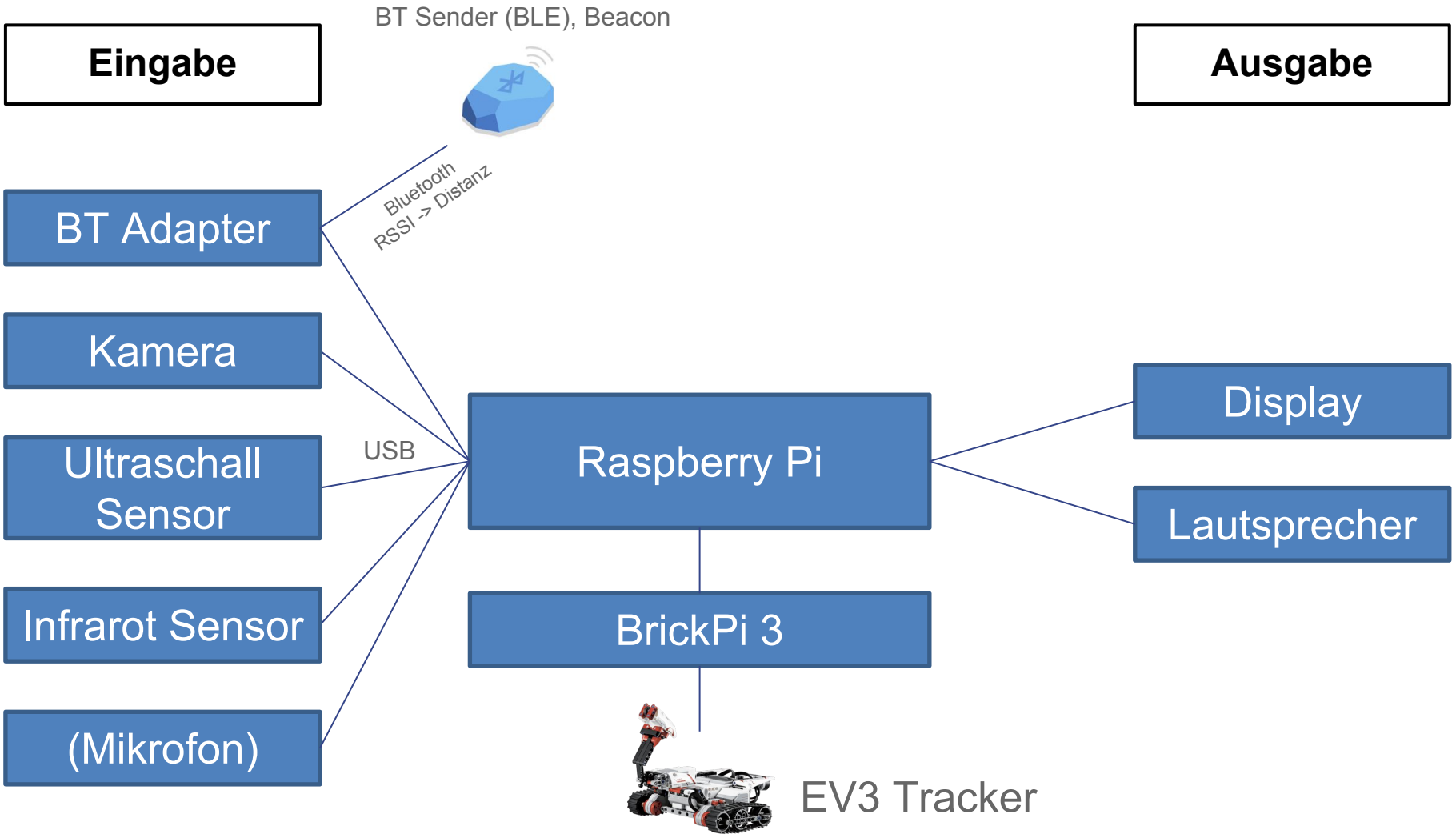


Aufbau (Anforderungen)



Aufbau (Implementierung)

itestra
Software Productivity



Arbeitsprozess

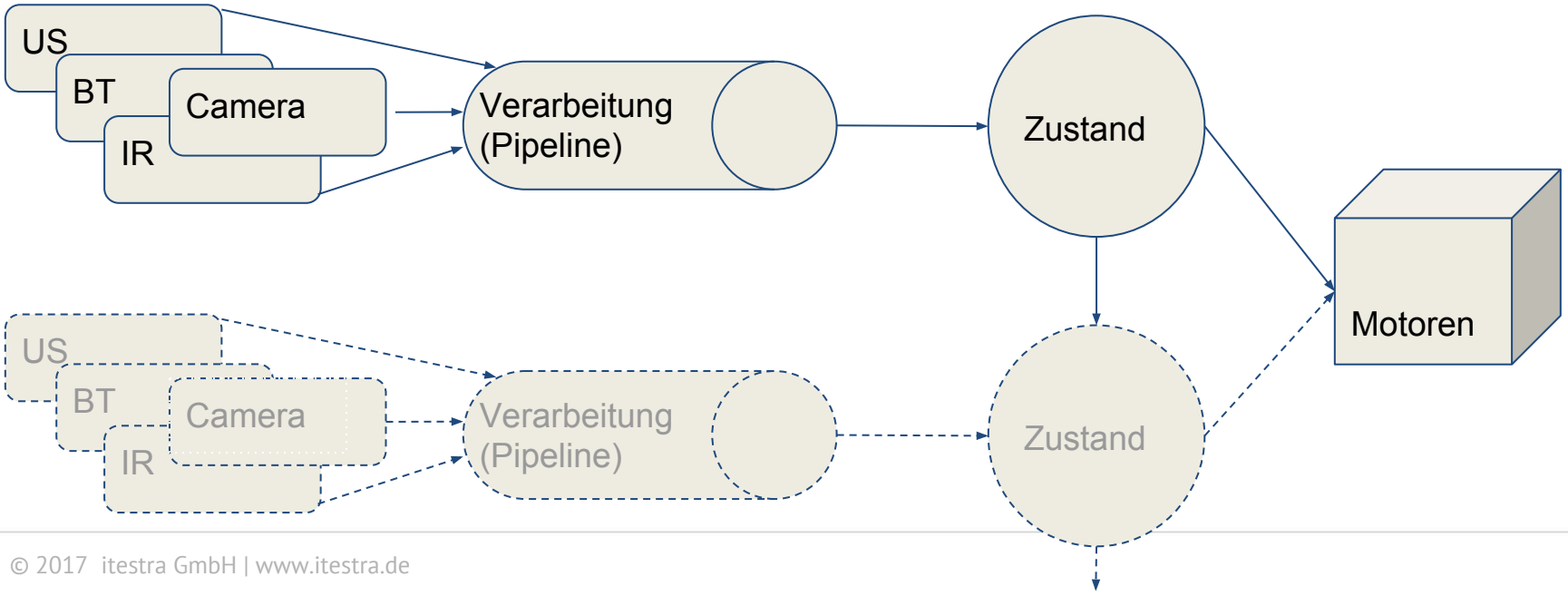
- Stand-Up Meetings (morgens, mittags, abends)
  - Fortschritts Besprechungen
- Aufteilung in kleinere Gruppen
- Nutzung des Whiteboards
  - Zusammenfassungen
  - Weitere Arbeitsplanung und -einteilung
- Styleguide

Experimente

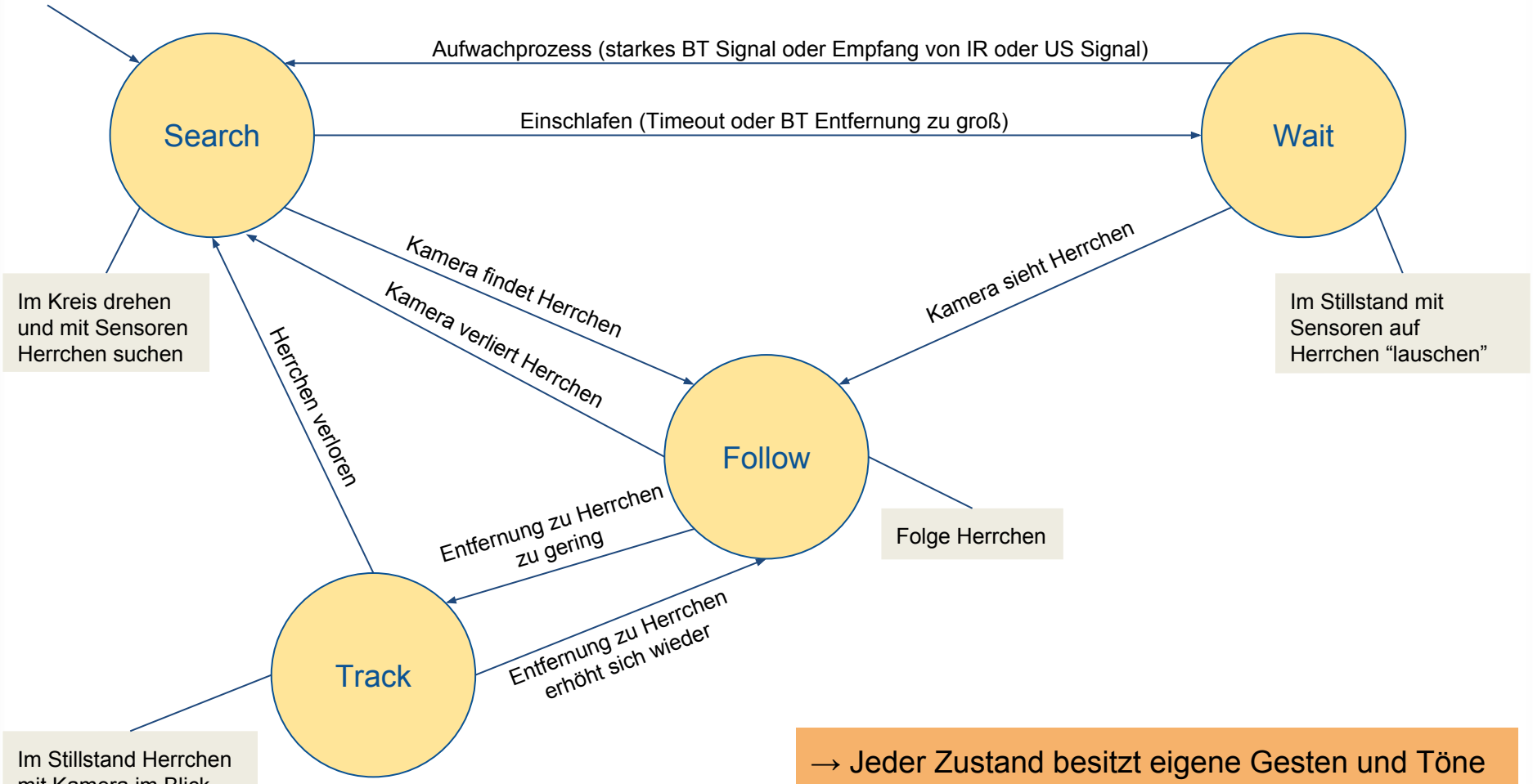
- Höchster Zeitaufwand: Testing von Schnittstellen und Verlässlichkeit
- Position der Hardware auf Roboter (Stabilität, Erreichbarkeit)
- Werte auslesen durch Python
- Testscripts schreiben
- Daten filtern
- Relevanz der Daten (Anwendungsfälle, Bsp: Gesten)
- Zeitaufwand des Einbaus
- Positivbeispiel: Bluetooth
- Negativbeispiel: GPS

State Machine

- Das Verhalten des Roboters wird durch eine State Machine definiert
- Der Roboter befindet sich zu jeder Zeit in einem Zustand
- Bei Aktualisierung der State Machine
  - Der Roboter bekommt neue Eingaben von Sensoren
  - Die Motoren des Roboters werden angesteuert
  - Ein neuer Folgezustand wird festgesetzt

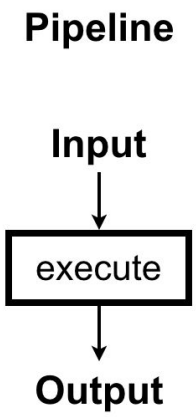


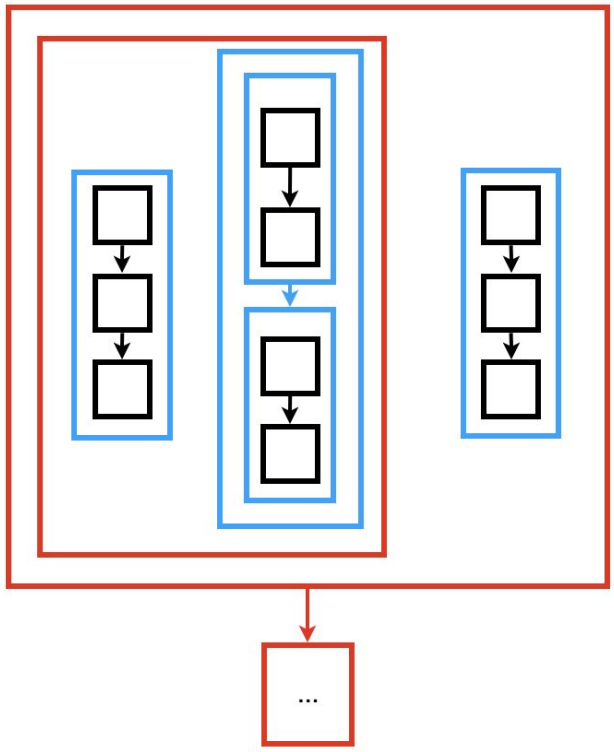
iDogstra State Machine

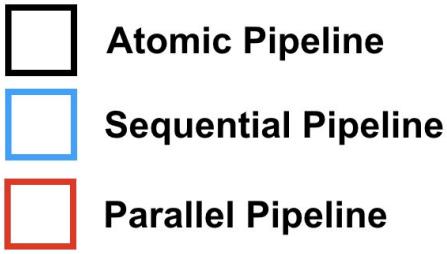


→ Jeder Zustand besitzt eigene Gesten und Töne

Pipelines







Arten von Pipelines

itestra
Software Productivity

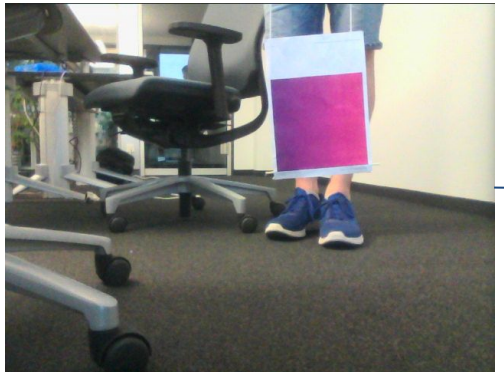
- Hardware-Pipelines:
  - Lesen der Sensorwerte
  - Bsp: Bluetooth, Infrarot, Ultraschall, Camera

- Einschrittige (atomare) Pipelines
  - Führen einen Berechnungsschritt aus
  - Beispiel: Konvertierung RGB -> HSV

- Zusammengesetzte Pipelines
  - Verknüpfen mehrere Pipelines zu einer großen Pipeline
  - Bsp. sequentielle Pipeline, parallele Pipeline

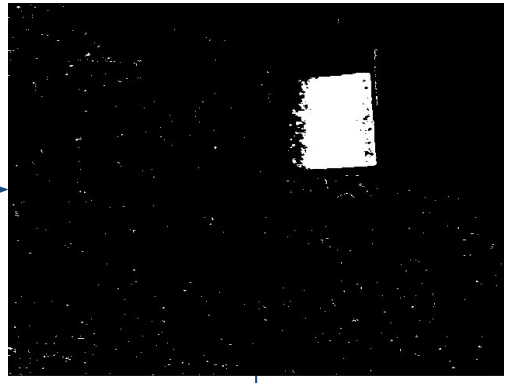
Beispiel: Bildverarbeitung

Kamerabild aufnehmen



Konvertierung nach HSV

Nach Farbe filtern



Noise rausfiltern

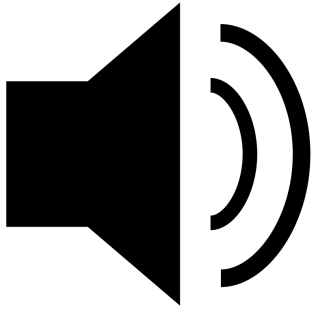
Größten zusammenhängenden Bereich finden

0.334719 x-Abweichung

0.209138 Kalman-Filterung

Gesten

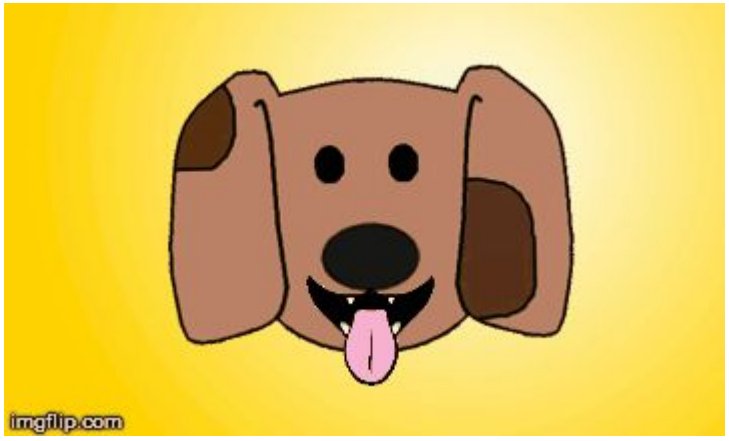
itestra
Software Productivity



Gesten

- Gesture-Thread läuft im Hintergrund
- Wechselt je nach State die Animation auf dem Display
- Töne werden je nach State unterschiedlich abgespielt
- Animation/Bilder mit GIMP erstellt

Gesten



Fazit

- Python zu langsam?
  - Problem: Echtzeit-Bildverarbeitung
- Leistungsbegrenzung durch CPU & GPU des RaspberryPi
- Bluetooth ungeeignet für Triangulationsmessung der aktuellen Position
- Eismaschine fördert Produktivität
- GPS Sensor indoors unbrauchbar

Ausblick/Möglichkeiten...

itestra
Software Productivity

- Einsatz stärkerer Hardware
- C/C++ statt Python -> Performance
- Genauere Lokalisierung von Roboter und Herrchen
  - Einsatz von “Hundeflöte”?
- Automatische Kollisionserkennung
- Robusteres Gehäuse -> stabilere Fahrweise
- Wechsel von Statemachines während der Ausführung

Noch Fragen?

itestra
Software Productivity

Q & A