

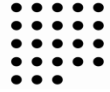
# Eine Testautomatisierung für Roboter-Steuerprogramme auf Android unter Verwendung von TTCN-3

Yassine El Amrani  
Hans W. Nissen

FH Köln  
Fakultät für Informations-, Medien- und Elektrotechnik  
Institut für Nachrichtentechnik  
Labor für Informatik

# Übersicht

- Motivation
- Unsere Inspiration
- Zielsetzungen der Testumgebung
- TTCN-3 kurz vorgestellt
- Die Beispiel-App
- Umsetzung der Testumgebung
  - Semi-automatische Prüfung
  - Automatische Prüfung
  - Ad-hoc Testfälle
- Fazit und Ausblick



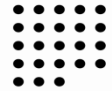
# Motivation

- Viele Geräte lassen sich durch Smartphone-Apps steuern
  - im Freizeit- / Spielbereich
    - » Helikopter
    - » Roboter
    - » Wir verwenden in der Ausbildung:
      - Lego Mindstorms Roboter
      - RP6 Roboter
  - aber auch ernsthafte Anwendungen:
    - » Gebäudetechnik
    - » Haus-Automatisierung
    - » Steuerung von Laborgeräten
    - » kommend: Systeme im Automobil



# Motivation

- Allgemein: Steuerung durch App auf Handy/Tablet sinnvoll für (eingebettete) Systeme,
  - die unerreichbar verbaut sind
  - keine eigene graphische Oberfläche besitzen
  - nicht ausreichend Speicher- und Rechenkapazität besitzen
- identische Situation:
  - App liest Sensorwerte
  - App steuert Aktuatoren
- Frage: Wie kann man diese Steuer-Apps testen?



# Motivation

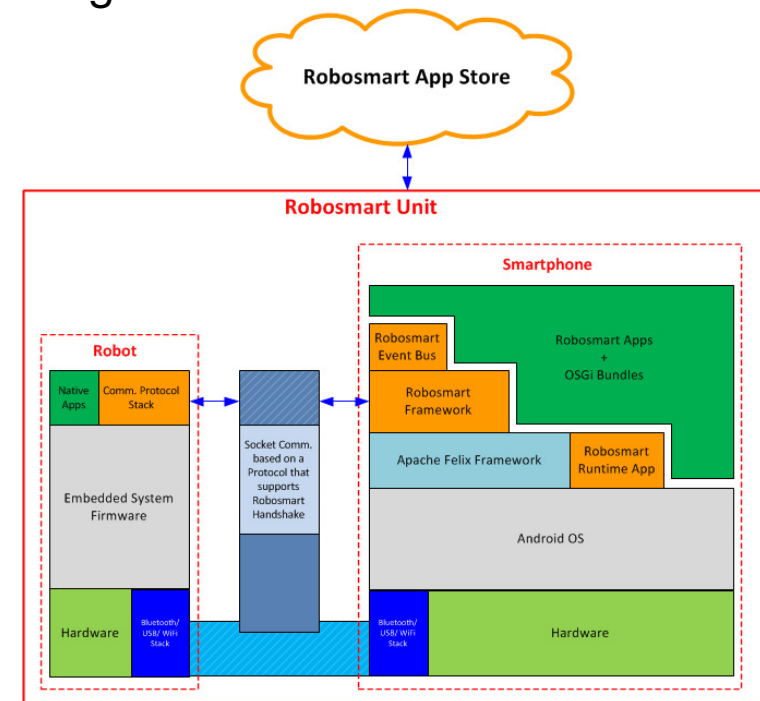
- Werkzeuge zum Test einer App existieren
  - z.B. Android Testing Framework
  - für Übersicht siehe
    - » Gao et al. "Mobile Applications Testing: A Tutorial", IEEE Computer, Februar 2014
  - Jedoch keine Betrachtung von Apps zur Steuerung anderer Geräte
- Systemtest einer Steuer-App
  - erfordert Interaktion der App mit dem gesteuerten Gerät
  - erfordert oftmals komplizierte Testaufbauten
  - Extremsituationen sind nur bedingt prüfbar
    - » Hält Roboter an der Tischkante an?
    - » Reagiert der Rettungs-Roboter in einem brennenden Haus richtig?
    - » Stoppt das Steuersystem in einem Labor das Aufheizen einer gefährlichen Mischung, wenn diese bereits kocht?

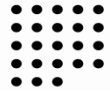
# Motivation

- Unser Ziel:  
Entwicklung einer Testumgebung zur Simulation der gesteuerten Hardware durch ein Testprogramm
  - Hierdurch sollen komplizierte und gefährliche Testaufbauten entfallen
  - Insbesondere sollen automatisierte Regressions-Tests möglich werden

# Motivation

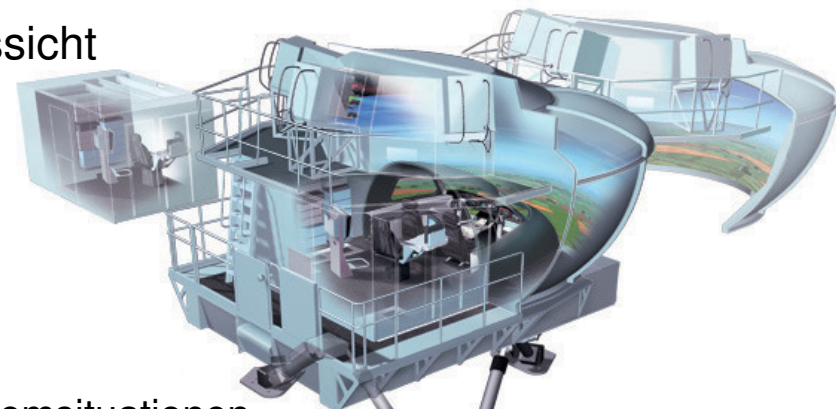
- Unser Hintergrund:
  - SW-Plattform zur Unterstützung des „Cloud Robotics“
    - » Kombination Lego Roboter und Smartphone
    - » problem-orientierte Erweiterung der Fähigkeiten eines Roboters aus der Cloud zur Laufzeit
  - Middleware für Android: *Robosmart*
    - » Umgebung für Steuer-Apps
    - » Basierend auf OSGi (Apache Felix)
    - » Service-orientierte Architektur
    - » Hot Deployment von Apps
  - Beispielanwendung: Pacman



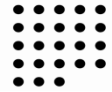


# Unsere Inspiration

- Flugsimulator zur Ausbildung von Piloten
  - nachgebautes Cockpit mit
    - » Anzeige-Instrumente
    - » Bedieneinrichtungen
    - » Visuelle Präsentation der Aussicht
  - 2 Simulationsarten
    - » Vordefiniertes Szenario
      - Einübung bekannter Abläufe
    - » Ad-hoc Szenario
      - Training für Sonderfälle / Extremsituationen
- *Steuerprogramm entspricht dem Piloten*
- *Testumgebung entspricht dem Flugsimulator*

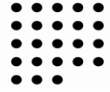






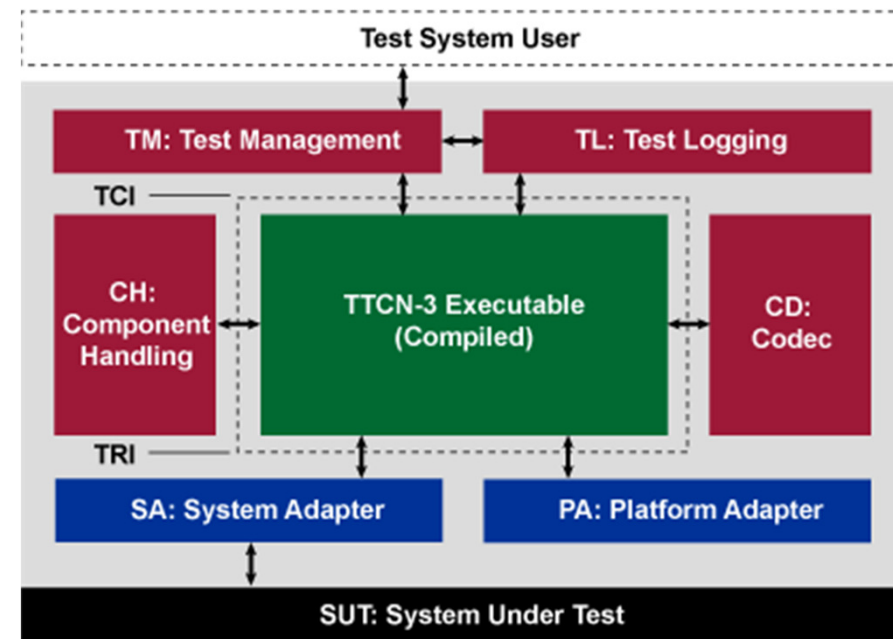
# Zielsetzungen der Testumgebung

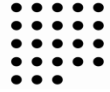
- Umsetzung in TTCN-3
  - *Testing and Test Control Notation*
- Vordefiniertes Szenario
  - teil-automatisierte Durchführung
    - » Tester muss Oberfläche der App bedienen
  - automatisierte Durchführung
    - » Oberfläche wird aus Testfall heraus bedient
- Ad-hoc Szenario:
  - Kontrollfenster ermöglicht Interaktion durch Tester
    - » Setzen von Sensor- und Aktuatorwerten
- Herausforderungen:
  - GUI einer App bedienen aus TTCN-3 Testfall heraus
  - Interaktion durch Eingabe von Testdaten an den TTCN-3 Testfall



## TTCN-3 kurz vorgestellt

- Testing and Test Control Notation Version 3 ([www.ttcn-3.org](http://www.ttcn-3.org))
- Standardisierte Testtechnologie und Testsprache zur Automatisierung von Tests für kommunikationsbasierte Systeme
- Anwendung in Telekommunikation, Automobilsektor, Medizin, ...
- Wir verwenden die TTworkbench von Testing Technologies

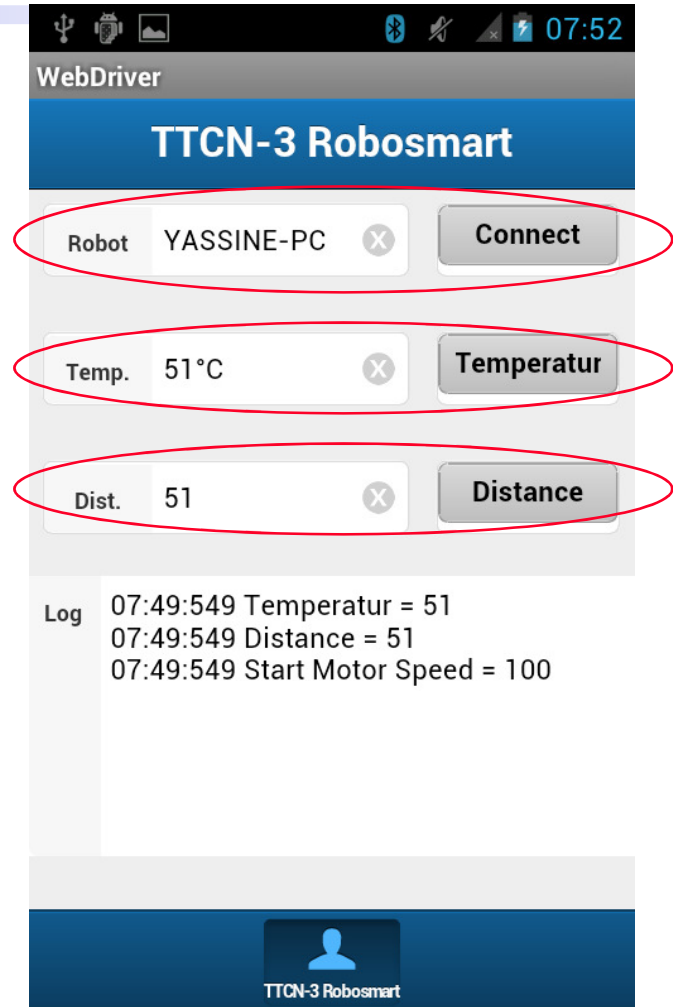


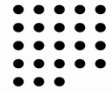


# Das Beispiel

- (sehr) einfaches Steuerprogramm als App
  - 3 Ein-/Ausgabefelder mit Buttons
    - » Connect:
      - zur Verbindung mit angegebenem Roboter
    - » Temperatur:
      - Zur Anfrage des Temperatur-Sensors des Roboters
    - » Distance:
      - Zur Anfrage des Distanz-Sensors des Roboters
  - Kommuniziert mit LCP (Lego Communication Protocol) über Bluetooth
  - Steuerlogik:

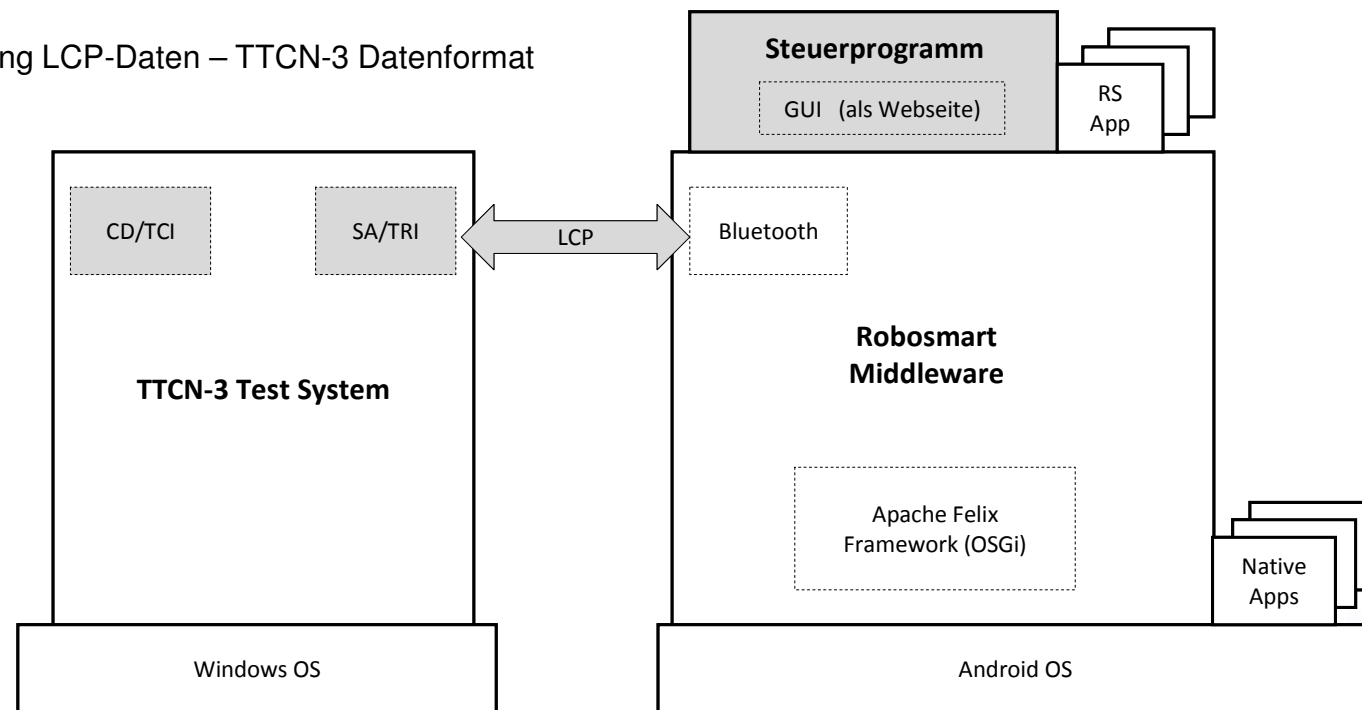
Falls	Dann
$0 < \text{Temperatur} \leq 50$ und $0 < \text{Distanz} \leq 50$	Motor = 50
$50 < \text{Temperatur} \leq 100$ und $50 < \text{Distanz} \leq 100$	Motor = 100
Sonst	Motor = 0

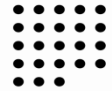




# Umsetzung: teil-automatisierte Durchführung der Prüfung

- TTCN-3 System läuft auf Windows-Rechner
- TTCN-3 Anpassungen:
  - System Adapter zur Anpassung an SUT
    - » Bluetooth-Verbindung mit Bibliothek BlueCove
    - » Lego Communication Protocol (LCP)
  - Codec
    - » Übersetzung LCP-Daten – TTCN-3 Datenformat
- Architektur:

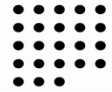




# Umsetzung: teil-automatisierte Durchführung der Prüfung

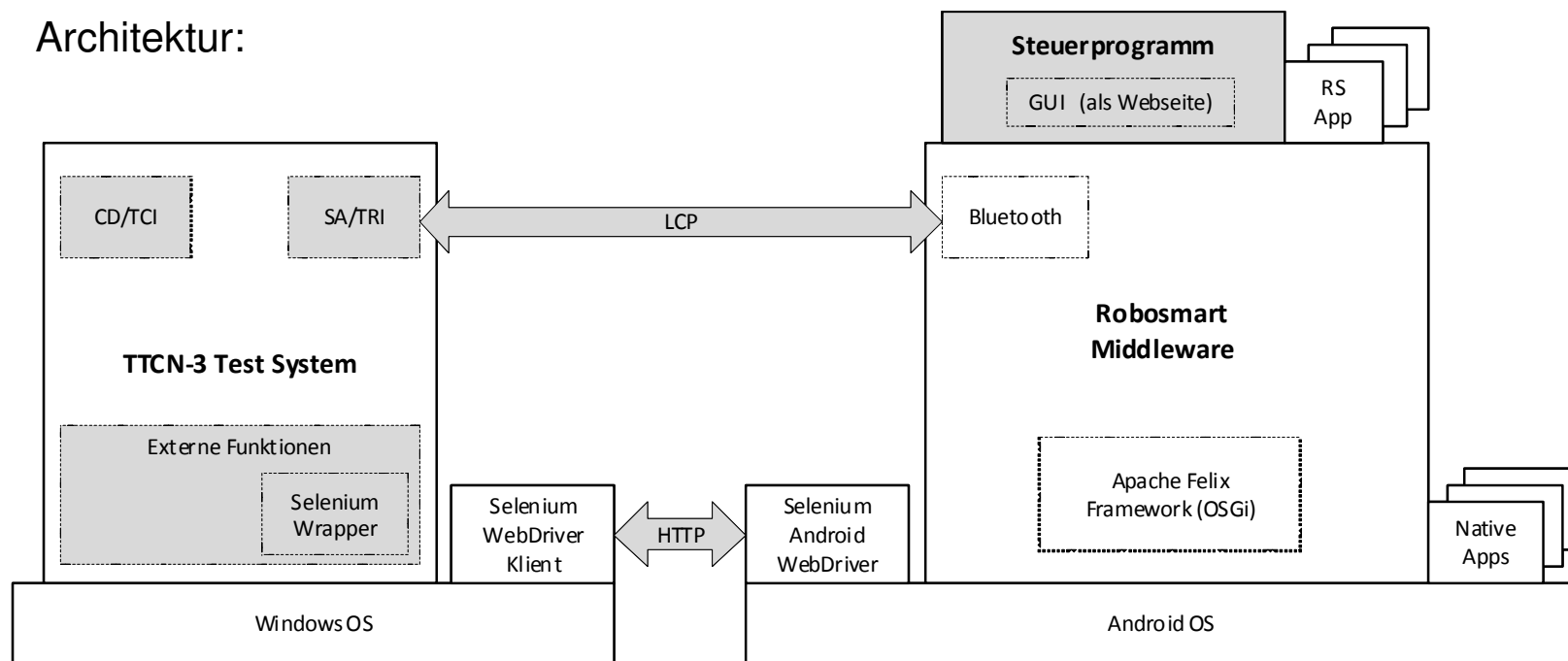
- Testfall prüft, ob Steuer-App
  - den Motor mit Geschwindigkeit 50 ansteuert, wenn
  - Temperatur-Sensor den Wert 30 und
  - Distanz-Sensor den Wert 20 liefert

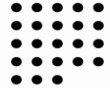
```
testcase tc_semiautomated() runs on mtcType system systemType {
  const hexstring speed := str2hex(intToHex(50, 2));
  map(mtc:mtcPort, system:systemPort);
  localtimer.start;
  alt {
    [] mtcPort.receive(GETINPUTVALUES(SENSOR_PORT_1)) {
      localtimer.stop;
      mtcPort.send(GETINPUTVALUESREPLY(30, SENSOR_PORT_1));
      alt {
        [] mtcPort.receive(GETINPUTVALUES(SENSOR_PORT_2)) {
          mtcPort.send(GETINPUTVALUESREPLY(20, SENSOR_PORT_2)); }
        [] mtcPort.receive {setverdict(fail;)}
      }
      alt {
        [] mtcPort.receive(SETOUTPUTSTATE(MOTOR_PORT_A, MOTORON, speed)) {
          mtcPort.send(SETOUTPUTSTATEREPLY);
          setverdict(pass, "TEST PASSED"); }
        [] mtcPort.receive {setverdict(fail;)}
      }
    [] mtcPort.receive {localtimer.stop; setverdict(fail, "TEST FAILED");}
    [] localtimer.timeout {setverdict(inconc, "Nothing received");}
  }
  unmap(mtc:mtcPort, system:systemPort);}
}
```



# Umsetzung: automatisierte Durchführung der Prüfung

- Oberfläche der Steuer-App wird durch Testfall bedient
- Verwendung Selenium WebDriver Framework:
  - Server auf Android-Gerät: *Selenium Android WebDriver*
    - » Führt Aktionen auf GUI entsprechend Anfragen eines Klienten durch
  - Klient auf TTCN-3 Gerät: *Selenium WebDriver Klient*
    - » API als externe Funktionen in TTCN-3 eingebunden: Selenium Wrapper
- Architektur:





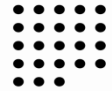
# Umsetzung: automatisierte Durchführung der Prüfung

- Testfall prüft, ob Steuer-App
  - den Motor mit Geschwindigkeit 50 ansteuert, wenn
  - Temperatur-Sensor den Wert 30 und
  - Distanz-Sensor den Wert 20 liefert
- Eingabe und Buttons werden durch Testfall bedient

```
testcase tc_automated() runs on mtcType system systemType {
```

```
    const integer temperatur := 30;
    const integer distance := 20;
    const hexstring mode_port_a := MOTORON;
    const integer power_port_a := 50;
    const hexstring speed := str2hex(intToHex(power_port_a, 2));
    map(mtc:mtcPort, system:systemPort); localtimer.start;

    initRemoteWebDriver (hub);
    openPage (url);
    sendKeys (robotTextField, „YASSINE-PC“);
    pushButton (CONNECT_BUTTON);
    pushButton (TEMPERATUR_BUTTON);
    alt {
        [] mtcPort.receive (GETINPUTVALUES (SENSOR_PORT_1)) { localtimer.stop;
            mtcPort.send (GETINPUTVALUESREPLY (temperatur, SENSOR_PORT_1));
            pushButton (DISTANCE_BUTTON);
            alt {
                [] mtcPort.receive (GETINPUTVALUES (SENSOR_PORT_2)) {
                    mtcPort.send (GETINPUTVALUESREPLY (distance, SENSOR_PORT_2)); }
                [] mtcPort.receive {setverdict (fail;)}
            }
            alt {
                [] mtcPort.receive (SETOUTPUTSTATE (MOTOR_PORT_A, mode_port_a, speed)) {
                    mtcPort.send (SETOUTPUTSTATEREPLY);
                    setverdict (pass, "TEST PASSED"); }
                [] mtcPort.receive {setverdict (fail;)}...}
    }
```



# Umsetzung: ad-hoc Szenario

- Prüfung von Sonderfällen und Extremsituationen
- Testdaten werden vom Tester über Kontrollfenster definiert
  - Und sind nicht im Testfall fest programmiert
- Kontrollfenster:

**TTCN-3 Test Simulator**

**Input**  
Hier können Sie Roboter Sensoren Werte simulieren.

Temperatur 51

Distanz 51

Sensor 3

Sensor 4

Send

**Output**  
Hier können Sie die TTCN-3 Tests Rückgaben sehen, z. B. Motoren Status.

Servomotor Mode MOTORON Power 100

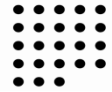
Motor B Mode MOTORON Power

Motor C Mode MOTORON Power

TTCN-3 Output TEST PASSED

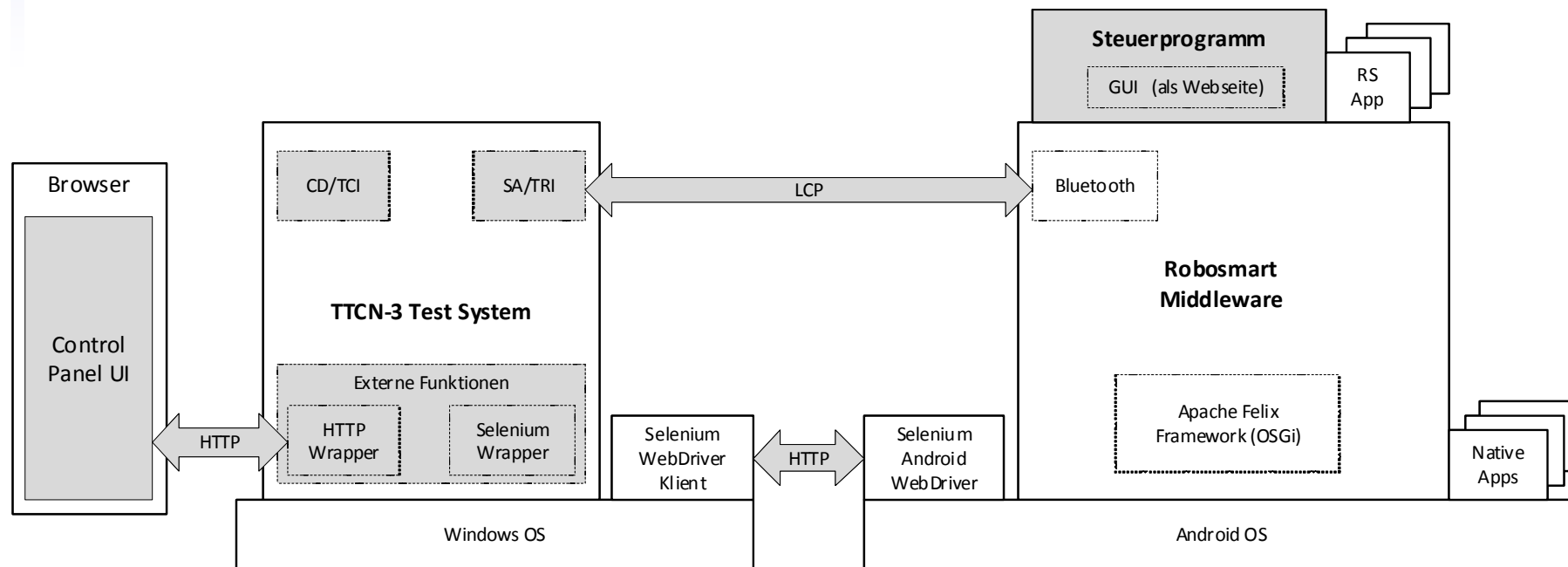
Simulator Einstellungen





# Umsetzung: ad-hoc Szenario

- Kontrollfenster als Webseite realisiert
- HTTPWrapper bietet Funktionen für Zugriff auf eingestellte Werte



## Umsetzung: ad-hoc Szenario

- Testwerte für Sensoren und Aktuatoren werden nicht als Konstanten gesetzt, sondern vom Kontrollfenster abgefragt
- Testlogik bleibt unverändert

```
testcase tc_controlpanel_on() runs on mtcType system systemType {

  simulator();
  var integer temperatur := str2int(getKeyValue(SENSOR_PORT_1));
  var integer distance := str2int(getKeyValue(SENSOR_PORT_2));
  var hexstring mode_port_a := str2hex(intToHex(str2int(getKeyValue(MOTOR_PORT_A_MODE)), 2));
  var hexstring speed := str2hex(intToHex(str2int(getKeyValue(MOTOR_PORT_A_POWER)), 2));

  map(mtc:mtcPort, system:systemPort);
  initRemoteWebDriver(hub);
  sendKeys(robotTextField, „YASSINE-PC“);
  pushButton(CONNECT_BUTTON);
  alt {
    [] mtcPort.receive(GETINPUTVALUES(SENSOR_PORT_1)) {
      localtimer.start;
      openPage(url);
      localtimer.stop;
      pushButton(TEMPERATUR_BUTTON);
      mtcPort.send(GETINPUTVALUESREPLY(temperatur, SENSOR_PORT_1));
      alt {
        [] mtcPort.receive(GETINPUTVALUES(SENSOR_PORT_2)) {
          mtcPort.send(GETINPUTVALUESREPLY(distance, SENSOR_PORT_2));
        }
        [] mtcPort.receive {setverdict(fail;)}
      }
      alt {
        [] mtcPort.receive(SETOUTPUTSTATE(MOTOR_PORT_A, mode_port_a, speed)) {
          mtcPort.send(SETOUTPUTSTATEREPLY);
          sendSocketAnswer("TEST PASSED"); setverdict(pass, "TEST PASSED");
        }
        [] mtcPort.receive {setverdict(fail;)}...}
    }
  }
}
```

# Vorführung

- automatisierte Durchführung der Prüfung
- ad-hoc Szenario

# Fazit und Ausblick

- Ergebnisse:
  - Ziel war Testumgebung für Steuerprogramme auf Android
  - Integration der GUI-Bedienung in den TTCN-3 Testfall
    - » automatisierte Testfälle möglich
    - » geeignet für automatisierte Regressions-Tests
  - Ad-hoc Testfälle
    - » Übernahme von Testwerten aus externem Kontrollfenster
    - » Prüfung von Sonderfällen
  - Intensive Verwendung von externen Funktionen in TTCN-3
  - Inspiration des Flugsimulators konnte umgesetzt werden
- Ausblick:
  - Kontinuierliche Testausführung mit Kontrollfenster
  - Einfaches Capture&Replay
    - » Capture: manuelles Ausführen des Steuerprogramms und „Fahrtschreiber“ auf Roboter protokolliert Sensor- und Aktuatorwerte
    - » Replay: „Abspielen“ der protokollierten Werte