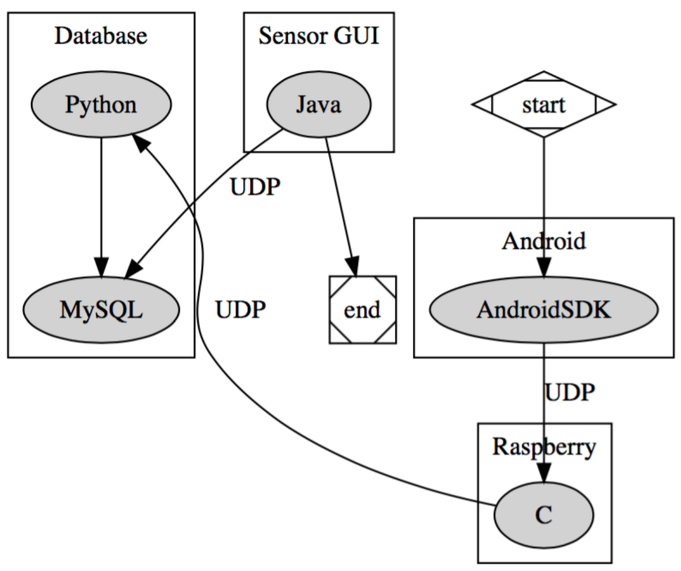
# Programmierprojekt SS16: PiSense mit Quadrocopter


Teilnehmer: Philipp Gackstatter, Marcel Früh
Dominik Heinrich, Jascha Petter, Christoph Weik

Betreuer: Vikas Agrawal
Arbeitsbereich Eingebettete Systeme

EBERHARD KARLS UNIVERSITÄT TÜBINGEN


## DOT (GraphViz)



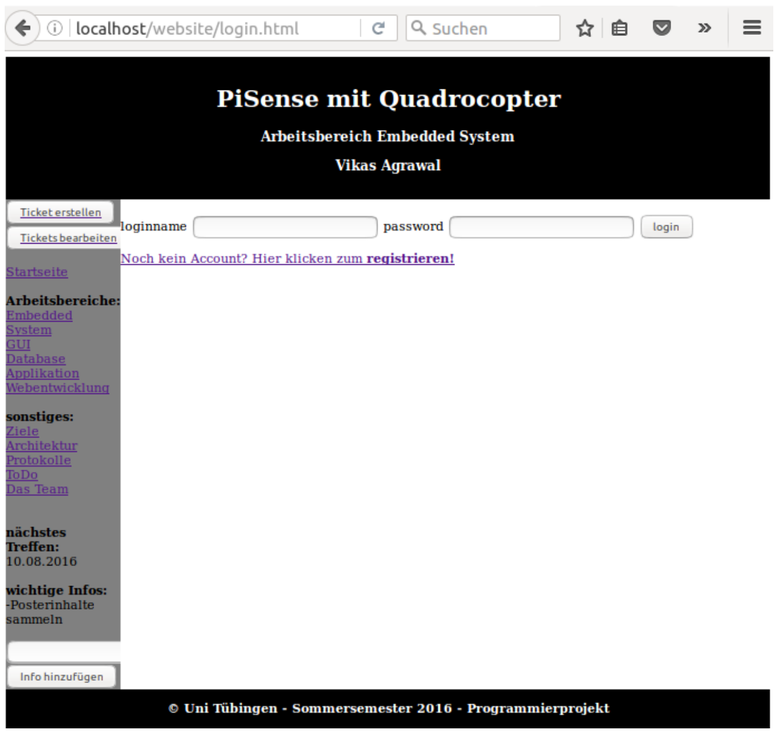

Programmierprojekt Quadrocopter

Architektur.dot

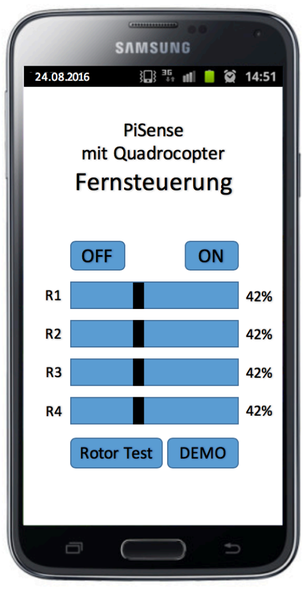
```
digraph MyGraph2 {
graph [ label="Programmierprojekt Quadrocopter" ]

subgraph cluster_0 {
node [style=filled];
C;
label = "Raspberry";
}

subgraph cluster_1 {
node [style=filled];
Python -> MySQL;
label = "Database";
}

subgraph cluster_2{
node [style=filled];
Java;
label = "Sensor GUI";
}

subgraph cluster_4 {
node [style=filled];
AndroidSDK;
label = "Android";
}


AndroidSDK->C[label="UDP"];
start-> AndroidSDK;
C -> Python [constraint=false][label="UDP"];
Java->end;
Java -> MySQL[label="UDP"];
start [shape=Mdiamond];
end [shape=Msquare];
}
```

## *WebProjectManagment:*

HTML
CSS
PHP
MySQL
LAMP




## *Applikation:*




Android SDK

UDP

## *GUI:*

**Funktionsweise**
Die GUI liest aus der Datenbank die gespeicherten Sensordaten aus und stellt diese in Graphen dar. Dabei werden aus Konfigurationsdateien die Details für Serverkommunikation, Bezeichnung der Daten und Dimensionen der GUI gelesen. Diese Dateien sind importierbar und können für verschiedene Nutzfälle angepasst werden. Der Nutzer hat die Möglichkeit Daten auszublenden, die dann automatisch auf gute Sichtbarkeit skaliert werden. Außerdem gibt es eine Vielzahl von Optionen durch welche die Anzeige vom Benutzer verbessert werden kann, wie etwa das verschieben, zoomen und pausieren.


Java

Für die GUI Programmierung möchten wir uns bei den Vorarbeitern Alexander Deitche und Juan-Carlos Barradas-Palmeros bedanken.

UDP

## C

## *Embedded Systems:*

**Funktionsweise**
Das Rasperry Pi bekommt seine Befehl von der App via UDP und führt je nach Protokoll unterschiedliche Funktionen aus. Jeder der vier Rotoren lässt sich einzeln ansteuern und die Umdrehung festlegen.

```
// Motor Data
sendto(clientSocket, Motor1, m1, 0,
        (struct sockaddr *) &serverAdress, addressSize);
sendto(clientSocket, Motor2, m2, 0,
        (struct sockaddr *) &serverAdress, addressSize);
sendto(clientSocket, Motor3, m3, 0,
        (struct sockaddr *) &serverAdress, addressSize);
sendto(clientSocket, Motor4, m4, 0,
        (struct sockaddr *) &serverAdress, addressSize);

//Sending Values over I2c

sendBuffer[0] = pwmValue;

g_lldI2c_WriteI2c_bl(BLCtrlADRExecuteOrder[0],
        &sendBuffer[0], 1);
g_lldI2c_WriteI2c_bl(BLCtrlADRExecuteOrder[1],
        &sendBuffer[0], 1);
g_lldI2c_WriteI2c_bl(BLCtrlADRExecuteOrder[2],
        &sendBuffer[0], 1);
g_lldI2c_WriteI2c_bl(BLCtrlADRExecuteOrder[3],
        &sendBuffer[0], 1);
usleep(100000);

pwmValue = pwmValue + STEPSIZE;

}

while (pwmValue > 5) {
    g_halImu_triggerImuReading_bl();
    g_halImu_triggerBaroReading_bl();
    g_halImu_triggerGyroReading_bl();
    g_halImu_triggerAccReading_bl();
```

```
pi@helikopter: ~

pi@helikopter ~ $ ./HELIKOPTER
pleasefly
Received string is pleasefly
Beschleunigungstest
Eingabe maximaler PWM Wert, 2 Ziffern < 70
100
Eingabe Stepsize, 1 Ziffer, < 10
1
AFAF
TimS1434498113
TimM993
AccX-0.026346
AccX-0.026346
AccZ-10.641397
MagX0.000004
MagY-0.000003
MagZ0.000021
GyrY-0.579852
GyrP0.885037
GyrR-0.671407
Temp28.512501
Pres969.748291
```


Zudem möchten wir uns bei Chris Mönch, Oliver Breuning, Jürgen Schmidt für die Rasperry Programmierung bedanken.


UDP

MySQL

## *DB:*

UDP

Python


**Zukünftige Ziele:** Real Time Code Generation mit MatLab Simulation