



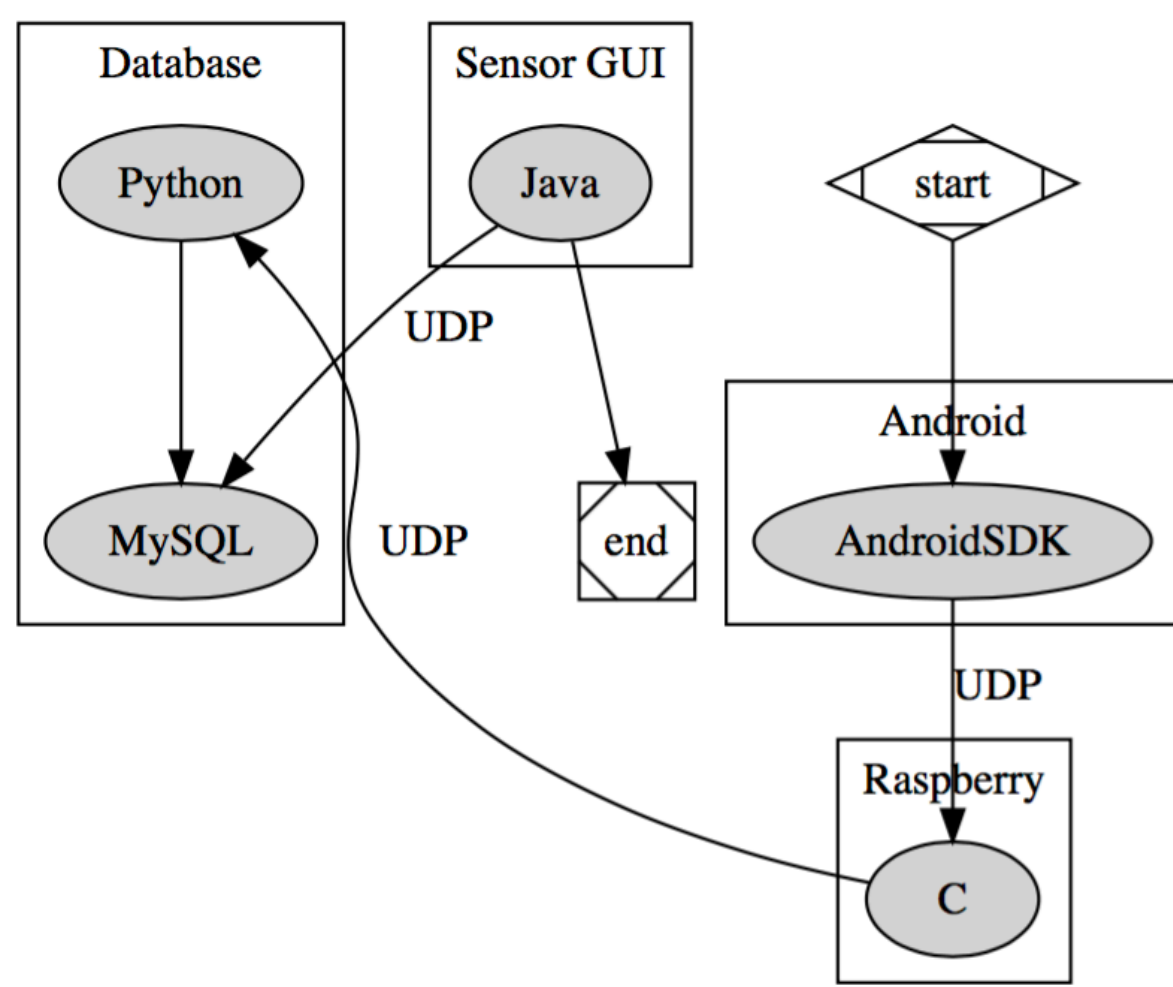
Programmierprojekt SS16: PiSense mit Quadrocopter

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

Betreuer: Vikas Agrawal
Arbeitsbereich Eingebettete Systeme



DOT (GraphViz)



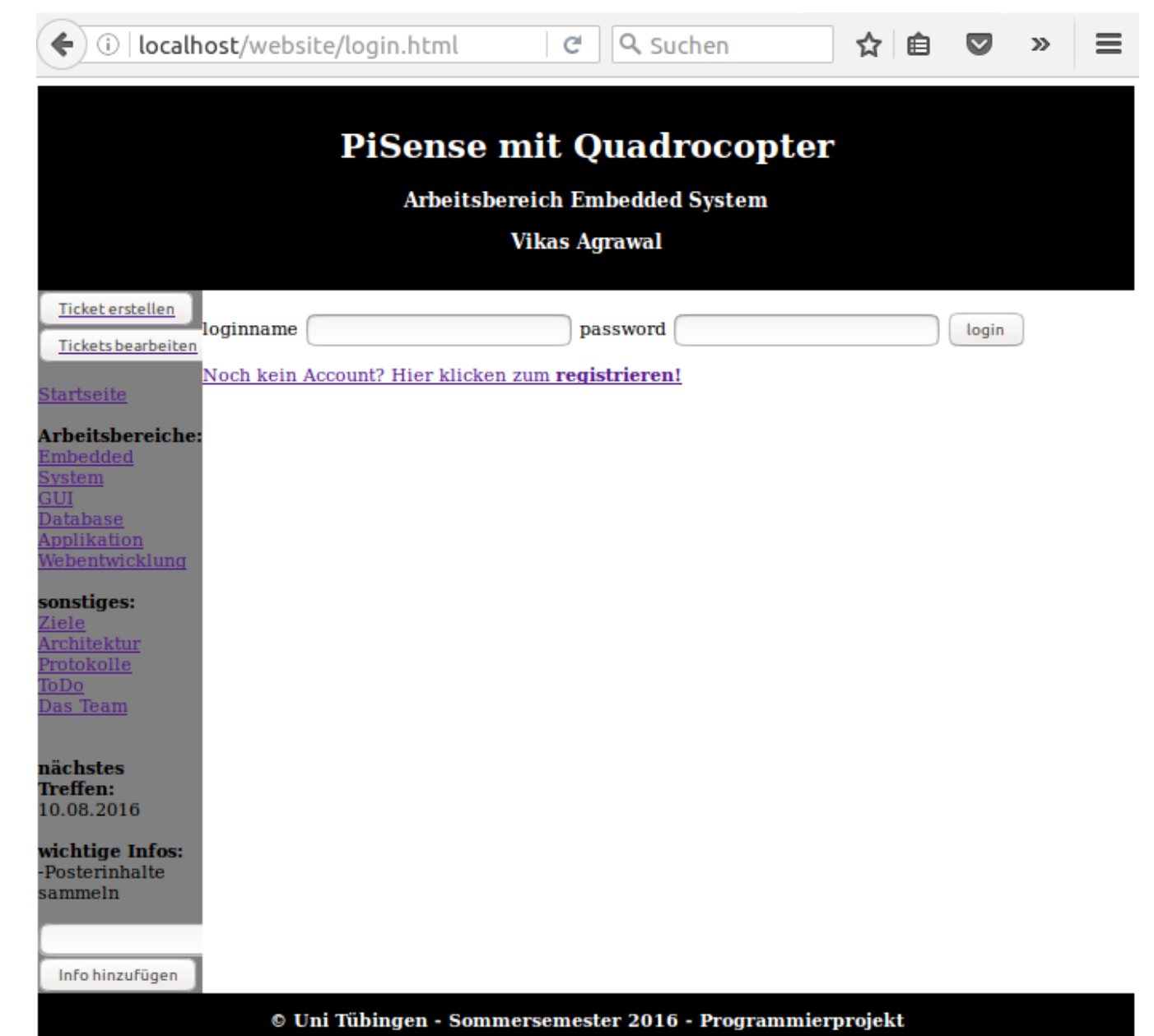
Programmierprojekt Quadrocopter

```
graph TD
    subgraph cluster_0 [ ]
        direction TB
        start((start))
        end((end))
    end
    subgraph cluster_1 [ ]
        direction TB
        Python((Python))
        MySQL((MySQL))
    end
    subgraph cluster_2 [ ]
        direction TB
        Java((Java))
    end
    subgraph cluster_3 [ ]
        direction TB
        Android((Android))
        AndroidSDK((AndroidSDK))
    end
    subgraph cluster_4 [ ]
        direction TB
        Raspberry((Raspberry))
        C((C))
    end
    start --> Android
    Android -- UDP --> Raspberry
    Raspberry -- UDP --> C
    C -- UDP --> MySQL
    MySQL -- UDP --> Python
    Python -- UDP --> Java
    Java -- UDP --> end
```

HTML
CSS
PHP

LAMP

WebProjectManagement:



TODO: Anderes Bild

Applikation:

Bild APP

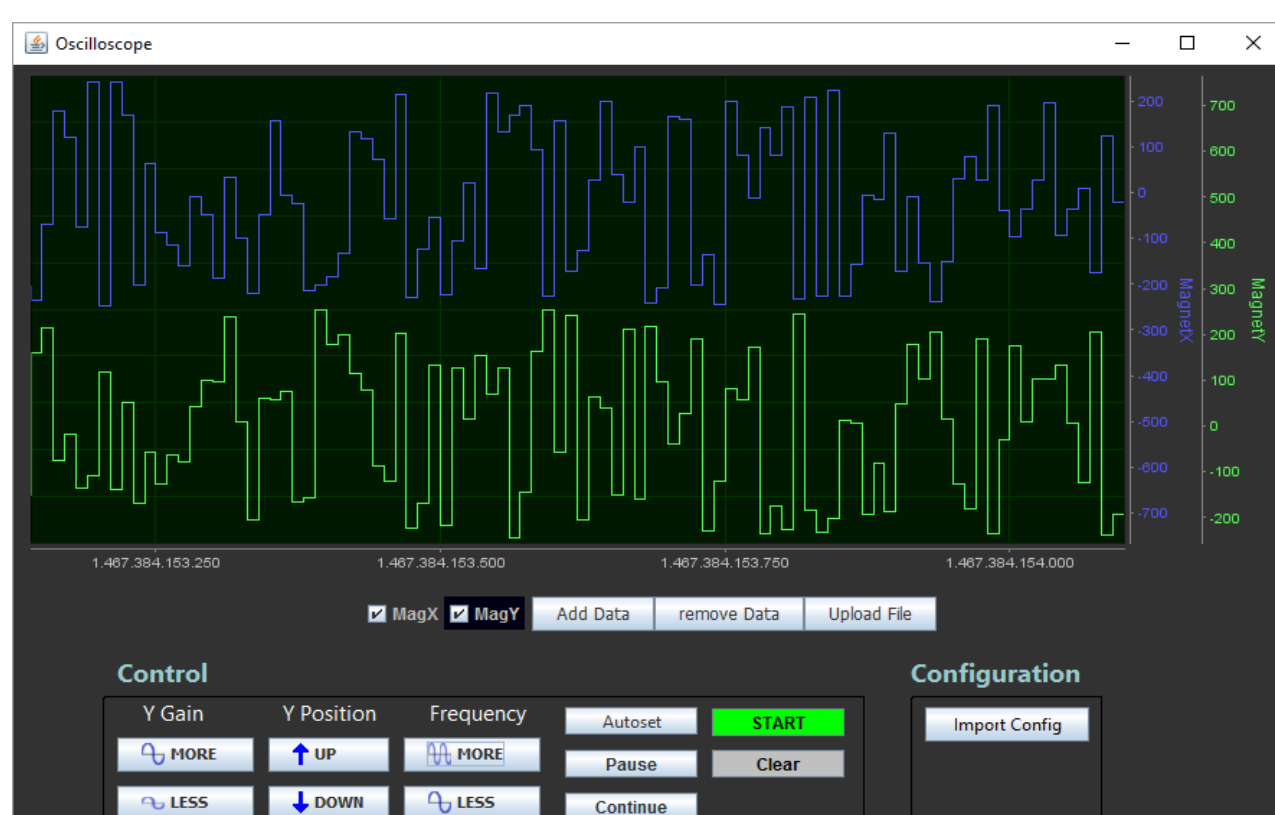
Android SDK

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.



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

C

Funktionsweise

Das Raspberry Pi bekommt seine Befehle 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 *) &serverAddress, addressSize);
sendto(clientSocket, Motor2, m2, 0,
(struct sockaddr *) &serverAddress, addressSize);
sendto(clientSocket, Motor3, m3, 0,
(struct sockaddr *) &serverAddress, addressSize);
sendto(clientSocket, Motor4, m4, 0,
(struct sockaddr *) &serverAddress, addressSize);

//Sending Values over I2C
sendBuffer[0] = pwmValue;
g_i2cWriteI2C_b1(BLCTRL_EXECUTE_ORDER[0],
&sendBuffer[0], 1);
g_i2cWriteI2C_b1(BLCTRL_EXECUTE_ORDER[1],
&sendBuffer[0], 1);
g_i2cWriteI2C_b1(BLCTRL_EXECUTE_ORDER[2],
&sendBuffer[0], 1);
g_i2cWriteI2C_b1(BLCTRL_EXECUTE_ORDER[3],
&sendBuffer[0], 1);
usleep(100000);
pwmValue = pwmValue + STEPSIZE;

while (pwmValue > 5) {
g_hlimu_triggerimuReading_b1();
g_hlimu_triggerBaroReading_b1();
g_hlimu_triggerGyroReading_b1();
g_hlimu_triggerAccReading_b1();
}
```

UDP

MySQL

DB:

Python

Java

Zukünftige Ziele: Real Time Code Generation mit MatLab Simulation