
Visual Based Landing System und Leonardo Mixer

VBLS: Eingebettete Systeme im SS 2016 & WS 2016/-17

Leonardo Mixer: Systemtechnik im WS 2016-/17

Projektvorstellung - Lukas Friedrichsen, Philipp Stenkamp

Gliederung

1. Motivation Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Motivation

- Wir mögen Quadrocopter!
- Autonomie ist sexy!

Wie lässt sich für ein mobiles System Autonomie in einem unbekannten Umfeld realisieren?

- Bilderkennung
- Tastsinn
- Ultraschall
- Laser /LiDAR
- ...

Motivation

Quadrokopter + Autonomie = ...?



Bild: <https://grabcad.com/library/terminator-hk-drone-1>

Motivation

Was kann mittels Bilderkennung an Autonomie realisiert werden?

Autonome Landeplatzlokalisierung und automatische Zentrierung über diesem im Schwebeflug

Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

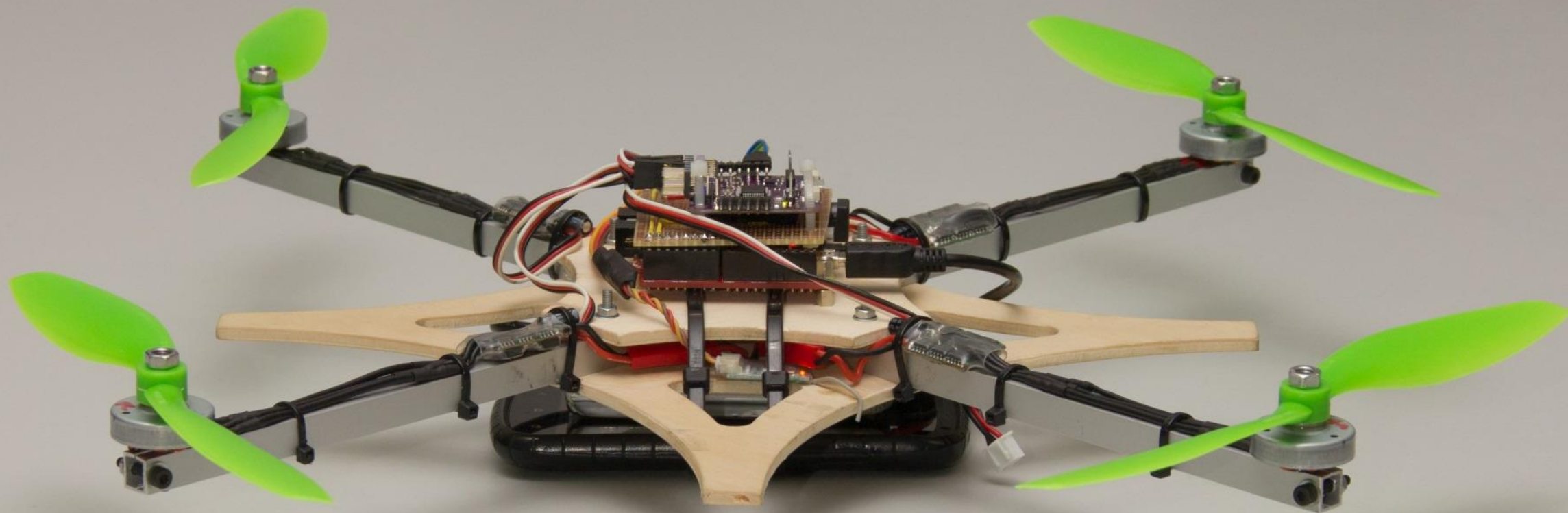
Voraussetzungen & Anforderungen

- Einfache Verwendbarkeit und Erweiterbarkeit
 - Modularer Aufbau
- Leichte Zugänglichkeit
 - Verwendung gut dokumentierter, freier Software
 - Sämtliche für die Umsetzung der Projekte verwendete Software ist Open Source!
- Funktionalität
 - Stabilität der Applikation
 - Leistungsfähige, schnelle Bildverarbeitung
 - Kommunikationsschnittstelle

Voraussetzungen & Anforderungen

Verwendete Komponenten:

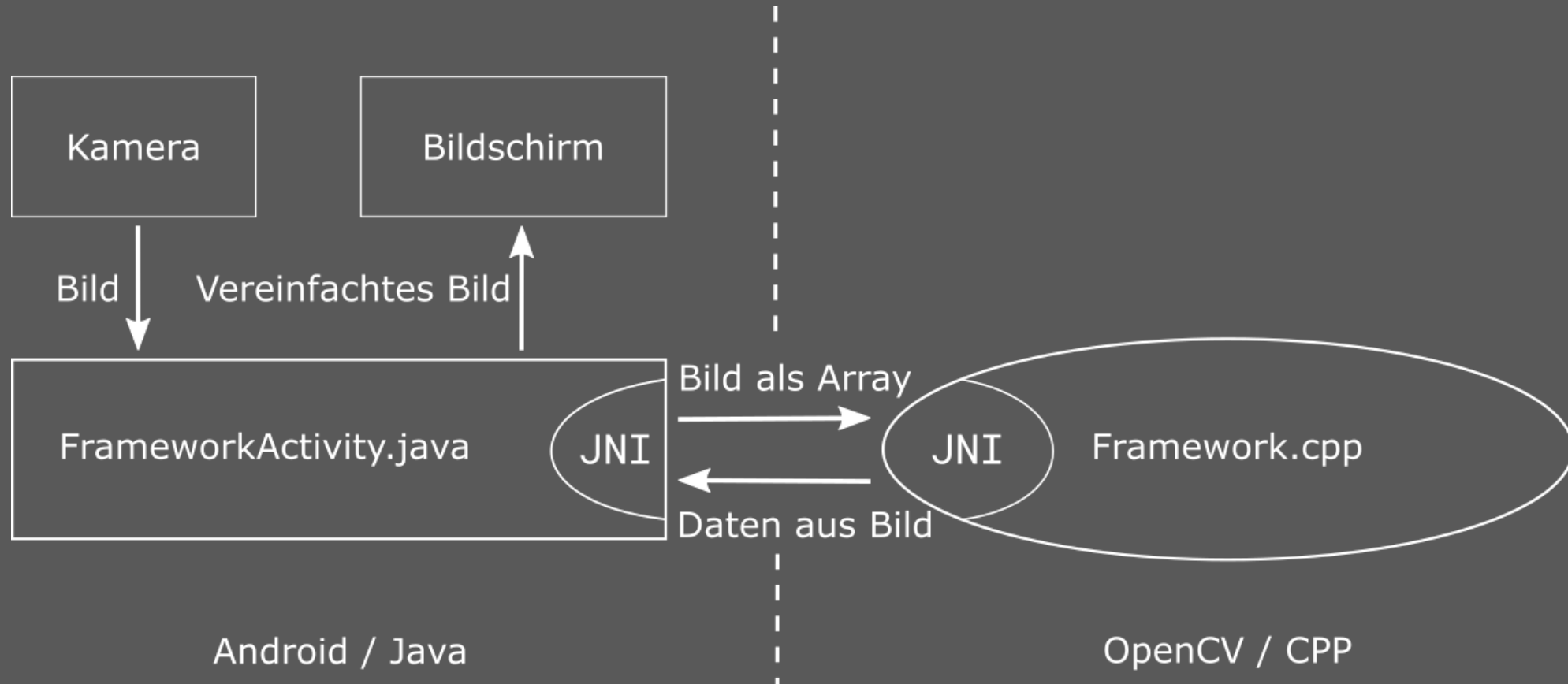
- Trägerplattform: Android (Java)
 - Das Android-Open-Source-Project (AOSP) steht unter der Apache Software License, Version 2.0
- Bildverarbeitung: OpenCV (C++)
 - Die Open Source Computer Vision Library (OpenCV) steht unter der Modified BSD License
- Kommunikationsschnittstelle:
 - Intern: JNI (Java Native Interface – GNU GPL)
 - Extern: USB (MIT License)



Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Generalisierung in Form eines Frameworks



Generalisierung in Form eines Frameworks

```
public Mat onCameraFrame(CvCameraViewFrame inputFrame) {  
    // Gray - images  
    mGray = inputFrame.gray();  
    [...]  
    byte[] jImageData = null;  
    try {  
        jImageData = transformationTools.jMatToArray(mGray);  
    }  
    [...]  
    double[] jImageDataProcessed = null;  
    try {  
        jImageDataProcessed = transformationTools.process(jImageData);  
    }  
    [...]  
    /* Possibly do stuff with your processed image here  
    *  
    */  
    return mGray;  
}
```

Generalisierung in Form eines Frameworks

```
if (cchannels == 1 && cdepth == 0){
    src = Mat(crows,ccolumns, CV_8UC1);
    for (int i = 0; i < crows; i++) {
        for (int j = 0; j < ccolumns; j++) {
            pos = i*ccolumns+j;
            src.at<signed char>(i, j) = inCArray[pos];
        }
    }
}
else {
    return NULL;
}
[...]
```

/* Do stuff with the image here
*
*/

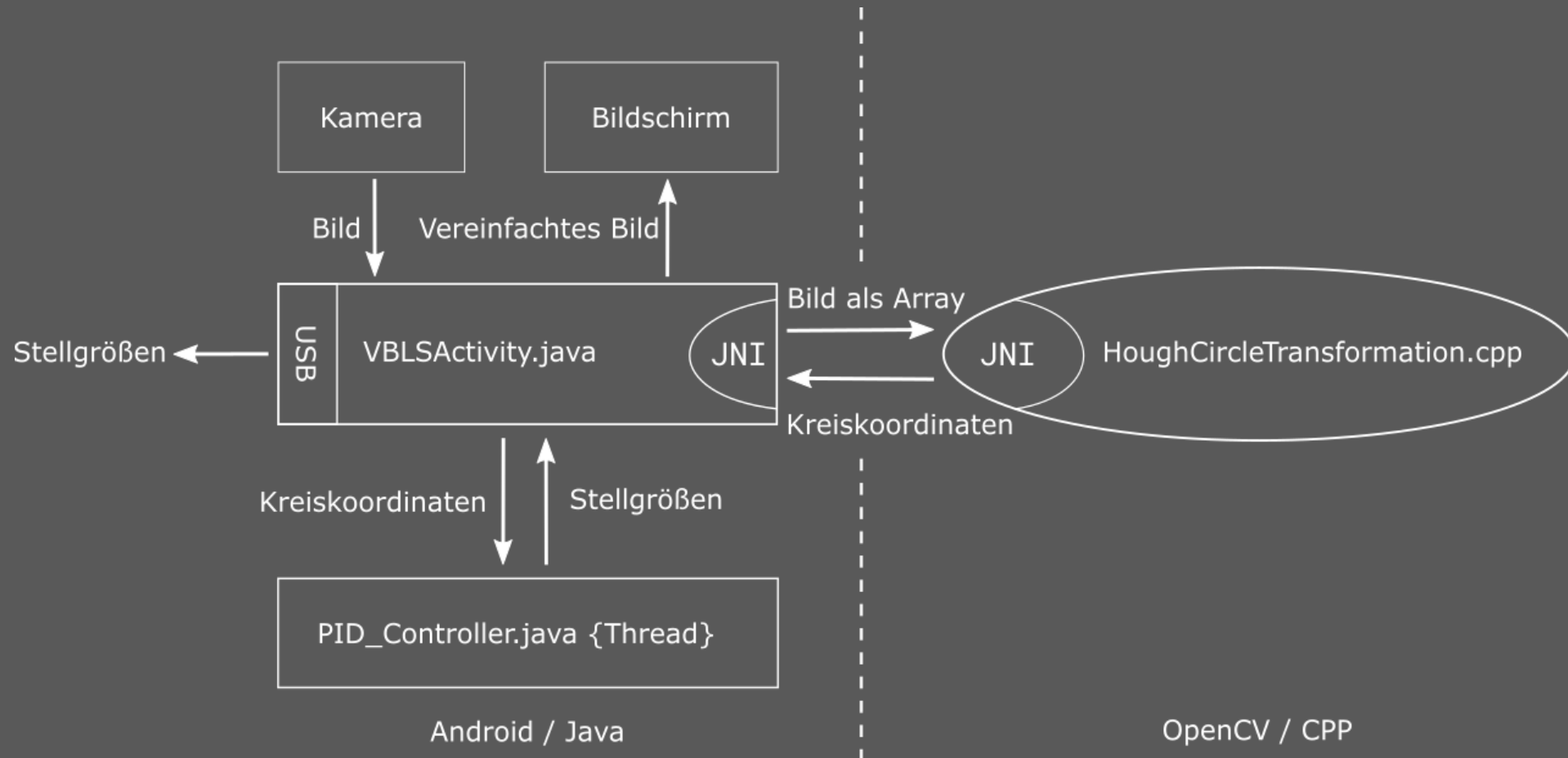
// Return your stuff if you want to
double *outCArray;
outCArray = (double *) malloc((XXXXXXXXXX)); // Malloc to prevent the application from using new memory space on every call of this method

// Return the results
[...]
}

Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Konkreter Lösungsansatz in Form des VBLS



Konkreter Lösungsansatz in Form des VBLS

```
// Calculates the nearest circle and writes the coordinates in the USB-write-buffer
if (jImageDataCircles.length > 0){
    jImageDataCircles = transformationTools.calculateNearestCircle(jImageDataCircles, IMG_WIDTH,
                                                                    IMG_HEIGHT, VEC_SUB_SIZE);

    // PID - controller
    pid.set_values(jImageDataCircles[0], jImageDataCircles[1]);

    // Set the USB-output-data to the current actuating variables
    double[] actuating_variables_copy = pid.get_actuating_variables();
    double x_temp = actuating_variables_copy[0]*(255.0/IMG_WIDTH)+127.0;
    double y_temp = actuating_variables_copy[1]*(255.0/IMG_HEIGHT)+127.0;
    [...]
    usb_data[0] = (byte) x_temp;
    usb_data[1] = (byte) y_temp; // Scaling to img_width to ensure identical x- and y-resolution
    usbService.write(usb_data);
}
```

Konkreter Lösungsansatz in Form des VBLS

```
src_gray = src; // Gray - image

// Reduce the noise so false circle detection is avoided (or rather reduced)
GaussianBlur(src_gray, src_gray, Size(9, 9), 2, 2);

// Apply the Hough Transformation to find the circles
HoughCircles(src_gray, circles, CV_HOUGH_GRADIENT, 1.0, (double) src_gray.rows/8, 150.0, 75.0, 0, 0);

// Convert the vector containing the coordinates and the size of the detected circles to the output-array
vec_size = circles.size();
sub_vec_size = 3;
double *outCArray;
outCArray = (double *) malloc((vec_size*sub_vec_size)*sizeof(double));
for (int i = 0; i < vec_size; i++) {
    for (int j = 0; j < sub_vec_size; j++) {
        outCArray[i*sub_vec_size+j] = cvRound((double)circles[i][j]);
    }
}
```

Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

5. Leonardo Mixer – Das Projekt im Projekt

1. Motivation für den Leonardo Mixer
2. Voraussetzungen & Anforderungen
3. Scheduler
4. Umsetzung als LeonardoMixerIO



Bild: https://commons.wikimedia.org/wiki/File:Matryoshka_transparent.png, User:Fanghong

Motivation für den Leonardo Mixer

- Trotz aller Autonomie muss manuelle Kontrolle gewährleistet bleiben!
 - Vgl. Folie 5...
 - Sicherheitsaspekte!
- Notwendigkeit einer Signalkonvertierung und –verarbeitung
 - Schnittstellenwandlung
 - Wertekonvertierung

*Entwicklung eines echtzeitfähigen RC-Mischers
mittels Arduino*

Voraussetzungen & Anforderungen

- Echtzeitfähigkeit
 - Einhaltung der Sendefrequenz der Eingangssignale
 - Reaktionsgeschwindigkeit auf Änderungen
- Stabilität
 - Deterministisches Verhalten
- Hardwareschnittstellen zur Vernetzung der einzelnen Komponenten
- Einfache Bedienbarkeit und freie Zugänglichkeit
 - Verwendung von freier Software

Voraussetzungen & Anforderungen

Verwendete Komponenten:

- Trägerplattform: Olimexino 32u4 (Arduino-Leonardo-kompatibel)
 - Open Source Hardware
 - Programmierung in C++ mittels freier Software
- Scheduler (C++)
- Kommunikationsschnittstellen:
 - USB -> Mobilgerät
 - Serielle Schnittstellen -> Flugsteuerung, Fernbedienung

Scheduler

```
struct task
{
    void (*activity) (void);
    unsigned long timestamp;
};
```

```
struct rtTask: task
{
    unsigned long cycleTime;
};
```

```
struct nRtTask: task
{
    unsigned long priority;
};
```



```

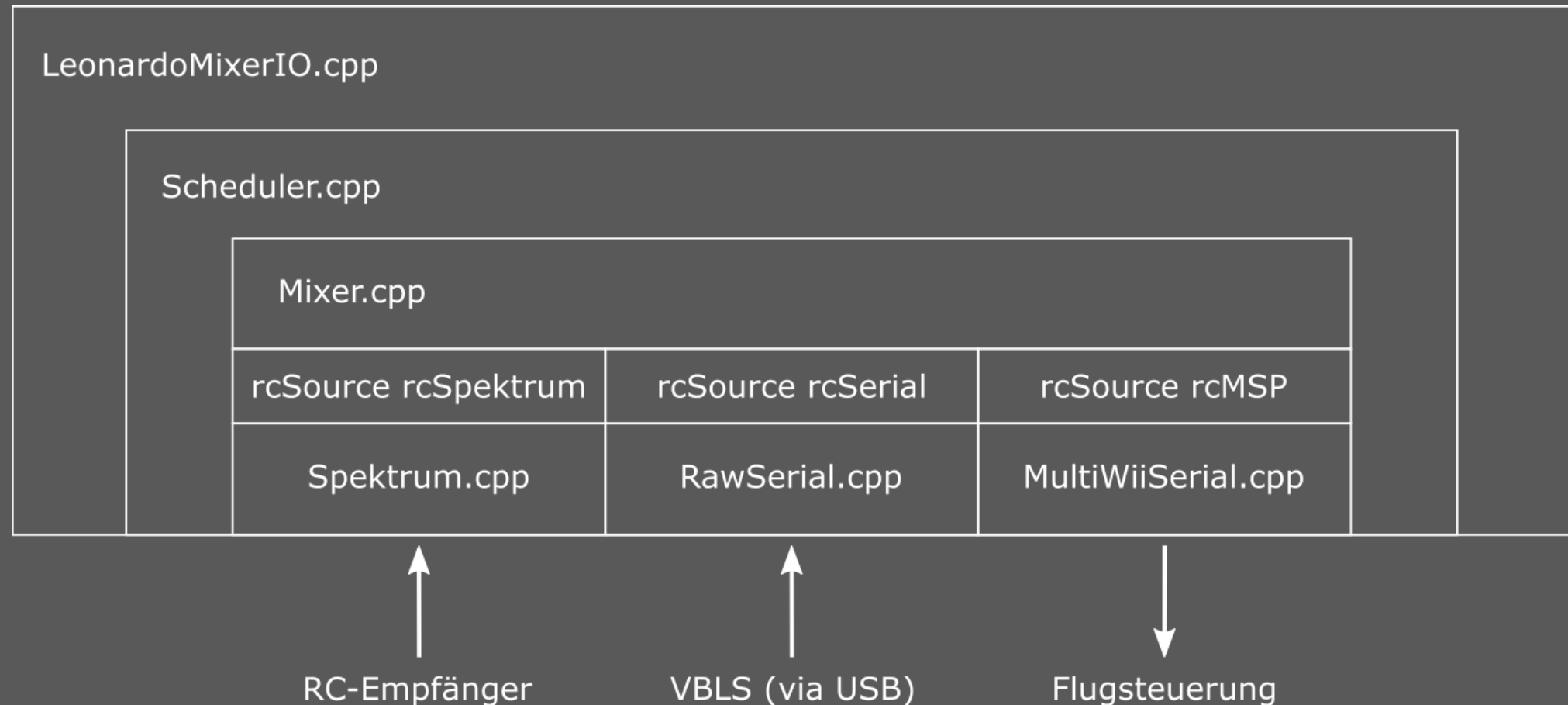
void Scheduler::schedule (void) {
    if (rtTasks) {
        unsigned long timerDiff = getTaskTimerDiff(rtTasks->listElement);
        unsigned long cycleTime = getTaskCycleTime((rtTask *) rtTasks->listElement);
        if (timerDiff >= cycleTime) {
            if (((timerDiff-cycleTime)*100/cycleTime) > OVERLOAD_THRESHOLD_PERCENT) {
                overloadCounter++;
                if (overloadCounter > OVERLOAD_THRESHOLD_TIMES) {
                    exterminate();
                }
            }
            rtTasks->listElement->activity();
            updateTaskTimer(rtTasks->listElement);
            rtTasks = sortRtTasks(rtTasks);
            if (nRtTasks){
                setPriorities();
                taskCounter = nRtTasks;
            }
        }
        else if (nRtTasks) {
            perform();
        }
    }
    else if (nRtTasks) {
        perform();
    }
    else {
        // No tasks... Nothing to do!
    }
}

```

Scheduler

```
void Scheduler::perform (void) {  
    if (taskCounter && *taskCounter) {  
        (*taskCounter)->activity();  
        updateTaskTimer((task *) *taskCounter);  
        taskCounter++;  
    }  
    else if (nRtTasks) {  
        taskCounter = nRtTasks;  
    }  
}
```

Umsetzung als LeonardoMixer IO



Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Kommen wir zum spannendsten Teil...

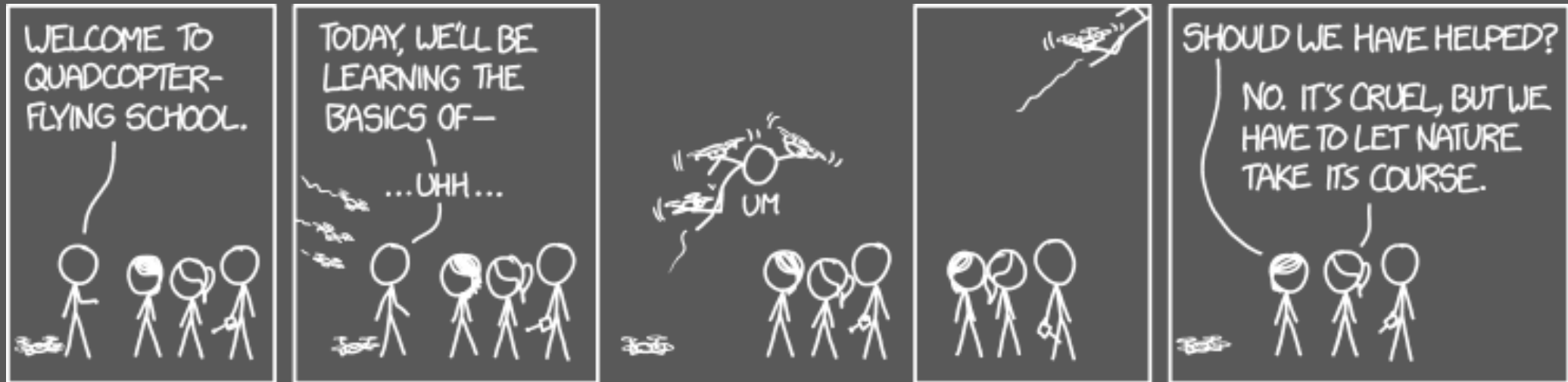


BILD: [HTTPS://XKCD.COM/1630/](https://xkcd.com/1630/)

Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Fazit

- Einzelne Systemkomponenten sind vollständig einsatzfähig
 - Einzelziele der jeweiligen Projekte erreicht
- Gesamtprojekt stellt funktionales „Proof of Concept“ dar
- Erkannte Probleme:
 - Optimierungspotential bzgl. Parametrierung des Reglers
 - Starre Befestigung des Mobilgeräts an Versuchsaufbau problematisch

Fazit

Auch wenn ein autonomes Landen des Quadropters noch nicht möglich ist, können beide Projekte als Erfolg betrachtet werden!

Gliederung

1. Motivation für das Visual Based Landing System
2. Voraussetzungen & Anforderungen
3. Generalisierung in Form eines Frameworks
4. Konkreter Lösungsansatz in Form des VBLS
5. Leonardo Mixer – Das Projekt im Projekt
6. Demonstration des Gesamtprojektes
7. Fazit
8. Ausblick

Ausblick

Das Projekt stellt in seiner aktuellen Form eine gute, einfach erweiterbare Grundlage für zukünftige eigenständige oder Folgeprojekte dar:

- Optimierung des Reglers
- Verbesserung der mechanischen Befestigung des Mobilgeräts
- Optimierung und Stabilisierung des Erfassungsbereiches der Kamera
- Miniaturisierung bzw. Konsolidierung der Hardware
 - Verwendung kleinerer und leichter Hardware
 - Zusammenfassen der einzelnen Komponenten
- ... weitere Ideen?

Vielen Dank für Ihre und Eure Aufmerksamkeit!

Bildquellen

BO-Logo & CVH Logo
Hochschule Bochum
<http://www.hs-bochum.de/campus-velbert-heiligenhaus.html>

Sonstige Bildquellen sind unter den jeweiligen Abbildungen angegeben.

Alle Bildrechte verbleiben bei den Eigentümern.

Eine Weitergabe ist nur hochschulintern gestattet!

Link zum Projekt:
gitlab.cvh-server.de/lf.ps/vbls





BILD: [HTTPS://XKCD.COM/1207/](https://xkcd.com/1207/)