APPLIED COMPUTER SCIENCES

FH JOANNEUM
University of Applied Sciences

# Project „RaspBot“

Overall Information on Raspberry Pi Robot Kits

```
   ___                ___       __
  / _ \___ ____ ___  / _ )___  / /_
 / , _/ _ `(_-</ _ \/ _  / _ \/ __/
/_/|_|\_,_/___/ .__/____/\___/\__/
             /_/
---RaspBian Wheezy------------------
```

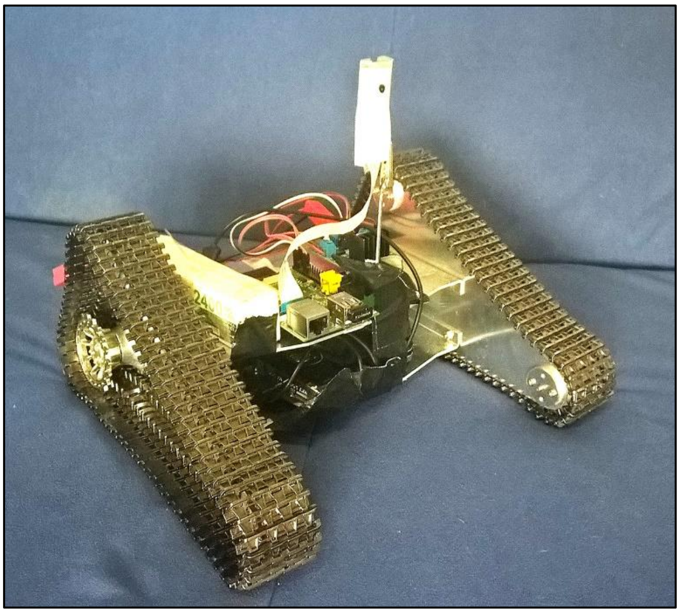


*Image 1 Assembled Robot*

**Note:**

This is a work in progress – several subprojects already have finished and are still ongoing- so if you choose to work with the robots, you should ask for existing project documentations!

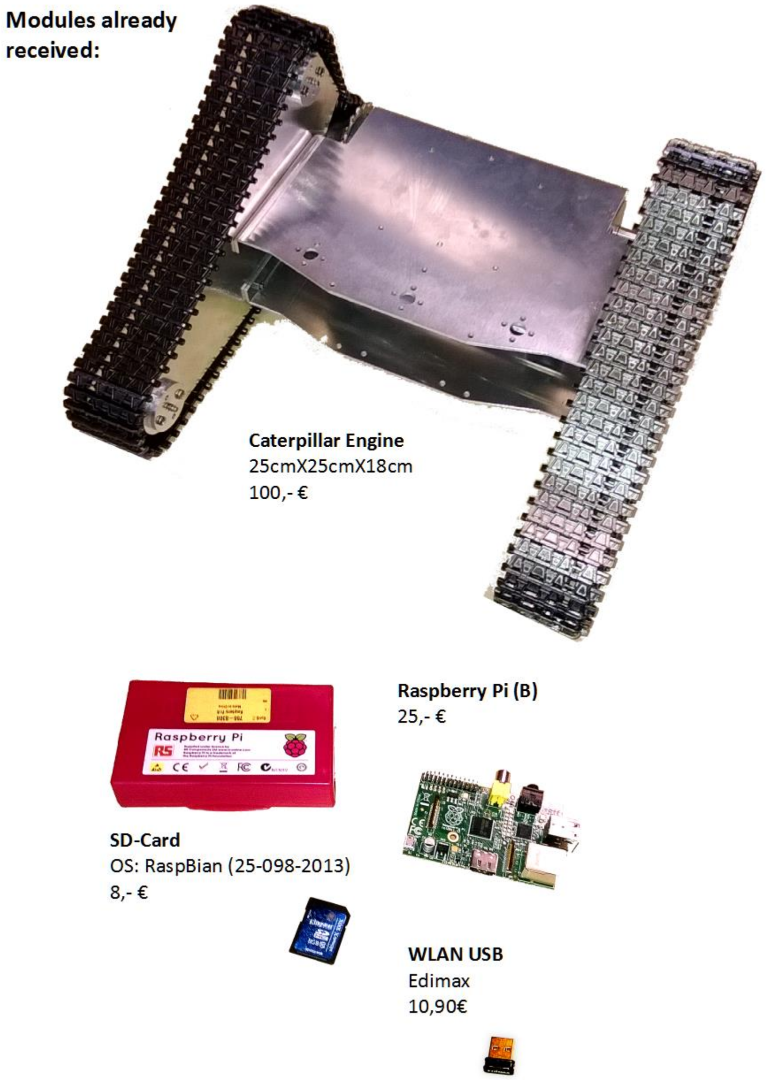
# Table of Contents

## Overview

Since we have a few robot kits at our disposal, it seems to be a good idea to build an Internet controlled (WLAN) robot with a camera, microphone and loudspeakers. The following tasks might arise in order to achieve this goal:

- Server – Client Architecture for controlling the robot
- Web Site with a HTML5/CSS3/JS client to access the robot
- Audio/Video Streaming
- Sensors (Distance Measures, Gyroscope, etc.)

Additionally to the tasks ahead, another possibility to customize the robot might come in handy: Crafting pieces with the 3D Plotter".



**Modules already received:**

**Caterpillar Engine**
25cmX25cmX18cm
100,- €

**Raspberry Pi (B)**
25,- €

**SD-Card**
OS: RaspBian (25-098-2013)
8,- €

**WLAN USB**
Edimax
10,90€

## Robot Kit

The basic material list for the robot is beneath. A detailed list will follow later with the assembly instructions:

- Robot Kit
- Raspberry PI (B) 512MB
- 4GB SD Card
- Edimax Wifi USB
- Lithium Polymer (LiPo) Battery Pack
- LiPo Checker
- Motor Controller
- Raspberry Camera
- Micro/Speaker

*Image 2 All Parts*

Aside the chassis, the RaspBot consists of

Raspberry PI (B) 512MB & SD Card

Edimax Wifi USB

Lithium Polymer (LiPo) Battery Pack 240mAh 30C 11.1V

L298N Motor Controller

HD44780 1602 LCD Module

Raspberry Camera

## Table of Parts

| Device / Part | Description | Amount / Pcs. | Check |
|---|---|---|---|
| Motor Controller | L298 | 1 | |
| Raspberry Pi | B | 1 | |
| Powerpack | Emergency Powerpack EPB80 | 1 | |
| USB Normal2Micro | Cable | 1 | |
| Battery Pack<br>+ Bag (fireproof) | LiPo 3900mAH 3S1P 11.1V 43.3Wh POLICE greenline | 1 | |
| LiPo Checker | Hitec LiPo Checker | 1 | |
| Robiot Chassis | Tri-Track Chassis Kit TTRK-KT | 1 | |
| Female to Male Cables | | | |
| Male to Male Cables | | | |
| Female to Female Cables | | | |
| Ethernet Cable | | 1 | |
| HDMI (male2male) Cable | | 1 | |
| Charging Set | IMAX B6AC | 1 | |
| WLAN USB adapter | Edimax | 1 | |
| SD Card | 4GB | 1 | |
| Blue Box | | 1 | |

Parts acquired from:

- robotshop.com
- Amazon
- etc.

# Common Setup

## Message of the Day

```
vi /etc/motd
   ___                ___       __
  / _ \___ ____ ___  / _ )___  / /_
 / , _/ _ `(_-</ _ \/ _  / _ \/ __/
/_/|_|\_,_/___/ .__/____/\___/\__/
             /_/
---RaspBian Wheezy------------------
```

## Hostname

```
vi /etc/hostname
raspbot
```

## Syntax Highlighting and Beep in VIM

```
vi ~/.vimrc
syntax on
set vb
```

## Save SD Card as an Image

```
dd if=/dev/sdx of=/path/to/image bs=1M
```

## Write Image to SD Card

```
dd if=/path/to/image of=/dev/sdx bs=1M
```

## WLAN Configuration

Added to /etc/wpa_supplicant/wpa_supplicant.conf:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
        ssid="wpa4fh"
        scan_ssid=1
        key_mgmt=WPA-EAP
        pairwise=TKIP
        group=TKIP
        eap=PEAP
        identity="knolm"
        password="*************"
        ca_cert="/etc/ssl/certs/UTN_USERFirst_Hardware_Root_CA.pem"
        phase1="peaplable=0"
        phase2="auth=MSCHAPV2"
}
```

## WLAN Interface Settings

This is /etc/network/interfaces:

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet static
        address 192.168.2.2
        netmask 255.255.255.0

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
        wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf

iface default inet dhcp
```

## **Webcam** with MJPG Streamer

Install necessary libraries:

```
apt-get install subversion-tools libjpeg8-dev imagemagick fswebcam
```

Prepare and download mjpg-streamer service:

```
mkdir -p /root/software/streaming
cd /root/software/streaming
svn checkout svn://svn.code.sf.net/p/mjpg-streamer/code/ mjpg-streamer-code
```

Build the service

```
cd mjpg-streamer-code/mjpg-streamer
make
make install
```

Run service

```
./mjpg_streamer -i "./input_uvc.so -n -y -f 15 -r 320x240" -o
"./output_http.so -n -w ./www -p 8040"
```

Works with Logitech C600! (Low-cost chinese cams won't work.)

### Raspberry Pi Camera

```
apt-get upgrade

raspi-config
```

Edit configuration the following way:

*camera enable/disable camera support*

Reboot afterwards

```
apt-get install mplayer netcat
```

On client Linux computer:

```
nc -l -p 5001 | mplayer -fps 18 -cache 1024 -
```

On Raspberry Pi:

```
raspivid -t 999999 -o - | nc [insert the IP address of the client] 5001
```

Nearly Lag-free:

```
raspivid -w 320 -h 240 -t 999999 -o - -fps 18 -b 5000000| nc 10.52.200.82
5001
```

# Hardware

## GPIOs in Raspberry Pi

| | | A | B | |
|---|---|---|---|---|
| 01 | 3.3V | o | o | 5V |
| 02 | GPIO 2 | o | o | 5V |
| 03 | GPIO 3 | o | o | GND |
| 04 | GPIO 4 | o | o | GPIO 14 |
| 05 | GND | o | o | GPIO 15 |
| 06 | GPIO 17 | o | o | GPIO 18 |
| 07 | GPIO 27 | o | o | GND |
| 08 | GPIO 22 | o | o | GPIO 23 |
| 09 | 3.3V | o | o | GPIO 24 |
| 10 | GPIO 10 | o | o | GND |
| 11 | GPIO 09 | o | o | GPIO 25 |
| 12 | GPIO 11 | o | o | GPIO 8 |
| 13 | GND | o | o | GPIO 7 |

## GPIO to Motor Connections

(See chapter about Motor Controller as well!)

01B Motor - 5V

03B Motor - GND

02A Motor Control Right Pin 2 - GPIO 2

03A Motor Control Right Pin 1 - GPIO 3

06A Motor Control Left Pin 2 - GPIO 17

07A Motor Control Left Pin 1 - GPIO 27

## Motor Control Program

This is the python script to run the motors (raspbot.py)

```
#!/usr/bin/python
#
# Motor Left:
#  Pin 01: Port 13 = GPIO 27
#  Pin 02: Port 11 = GPIO 17
# Motor Right:
#  Pin 01: Port 5 = GPIO 03
#  Pin 02: Port 3 = GPIO 02
#
# Imports
import RPi.GPIO as gpio
import time
import sys
import usb.core
import usb.util

#
# Initial setup and basic methods
#
# If Joystick has control over movement
hasControl=False
# GPIO Pins
motorLeftPin01=13
motorLeftPin02=11
motorRightPin01=5
motorRightPin02=3
# Address for Joystick
joyVendor=0x16c0
joyProduct=0x27dc
dev = usb.core.find(idVendor=joyVendor, idProduct=joyProduct)
interface = 0
endpoint = dev[0][(0,0)][0]
#
# Main function
#
def main():
      try:
            raspbot()
      except KeyboardInterrupt:
            pass
      finally:
            gpio.cleanup()
#
# RaspBot Logic
#
def raspbot():
      initAll()
      moveIt()
      time.sleep(1)
      stopAll()
#
```

```
# Initialize all devices
#
def initAll():
      print "Init GPIOs ..."
      gpio.setmode(gpio.BOARD)
      gpio.setup(motorLeftPin01, gpio.OUT)
      gpio.setup(motorLeftPin02, gpio.OUT)
      gpio.setup(motorRightPin01, gpio.OUT)
      gpio.setup(motorRightPin02, gpio.OUT)
      print "Init USB Joystick ..."
      if endpoint is None:
            print "Joystick endpoint not found!"
            raise ValueError('Device endpoint not found')
      if dev.is_kernel_driver_active(interface) is True:
            # tell the kernel to detach
            dev.detach_kernel_driver(interface)
            # claim the device
            usb.util.claim_interface(dev, interface)
#
# Movement commands
#
def moveLeftForward():
      gpio.output(motorLeftPin01, True)
      gpio.output(motorLeftPin02, False)
def moveRightForward():
        gpio.output(motorRightPin01, False)
        gpio.output(motorRightPin02, True)
def moveLeftBackward():
        gpio.output(motorLeftPin01, False)
        gpio.output(motorLeftPin02, True)
def moveRightBackward():
        gpio.output(motorRightPin01, True)
        gpio.output(motorRightPin02, False)
def stopMovement():
      gpio.output(motorLeftPin01, False)
        gpio.output(motorLeftPin02, False)
        gpio.output(motorRightPin01, False)
        gpio.output(motorRightPin02, False)
#
# Moving loop
#
def moveIt():
      print "Starting moving loop ..."
      runIt=True
      while(runIt):
            try:
                  data =
dev.read(endpoint.bEndpointAddress,endpoint.wMaxPacketSize)
                  print data
                  runIt = moveBot(data)
            except usb.core.USBError as e:
                  data = None
                  if e.args == ('Operation timed out',):
                        continue
#
```

```
# One move for the bot
#
def moveBot(data):
      global hasControl
      vVertical = data[2]
      vHorizontal = data[1]
      vButton = data[3]

      if vVertical == 128 or vHorizontal == 128:
            stopMovement()

      if hasControl:

            if vVertical == 0:
                  moveLeftForward()
                  moveRightForward()
            elif vVertical == 255:
                  moveLeftBackward()
                  moveRightBackward()

            if vHorizontal == 0:
                  moveRightForward()
                  moveLeftBackward()
            elif vHorizontal == 255:
                  moveLeftForward()
                  moveRightBackward()

      if vButton == 1:
             hasControl = not hasControl

      return True

#
# Stop all GPIOs
#
def stopAll():
      print "Full stop ..."
      gpio.output(motorLeftPin01, False)
      gpio.output(motorLeftPin02, False)
      gpio.output(motorRightPin01, False)
      gpio.output(motorRightPin02, False)

#
# Call main
#
if __name__ == "__main__":
    sys.exit(main())
```

## pyUSB

### Settings for simple Joystick (not part of package)

```
array('B', [1, 128, 128, 0])          → IDLE
array('B', [1, 128, 0, 0])              → UP
array('B', [1, 128, 255, 0])            → DOWN
array('B', [1, 255, 128, 0])            → RIGHT
array('B', [1, 0, 128, 0])              → LEFT

array('B', [1, 128, 128, 1])            → BOTTON

array('B', [1, 0, 0, 0])                → UP LEFT
```

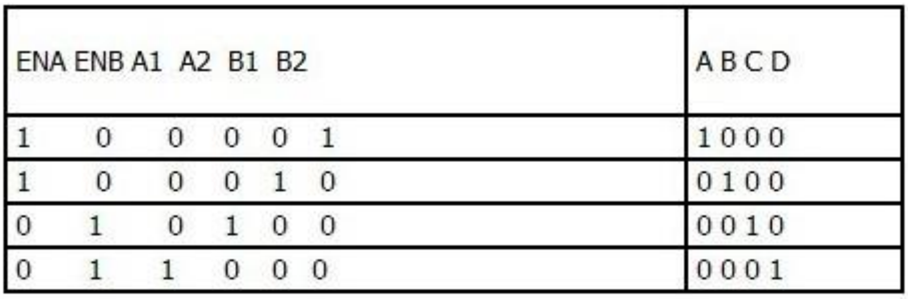
## **Motor Controller** (cheap Chinese part for 3-4€)

### Instructions for use:

ENA and ENB board is high effective and here refers to the level TTL level.

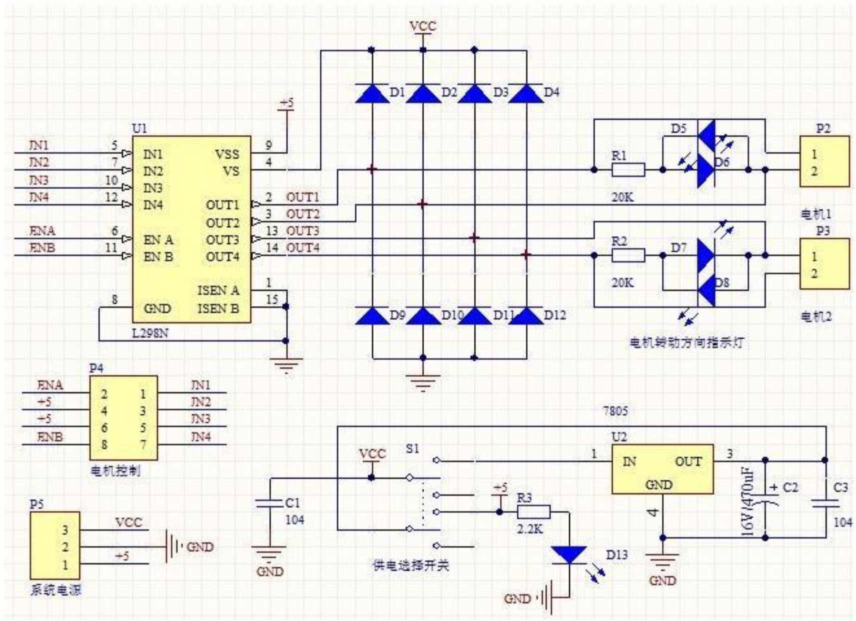
ENA A1 and A2, Enable, ENB for the B1 and IB2 enable end BJ then stepping motor common.
Stepper motor control logic as follows, where A, B, C, D four coils of the stepper motor, said there is a current of 1, 0 means no current flow. Coil connection is shown below

Example of a four-phase stepper motor

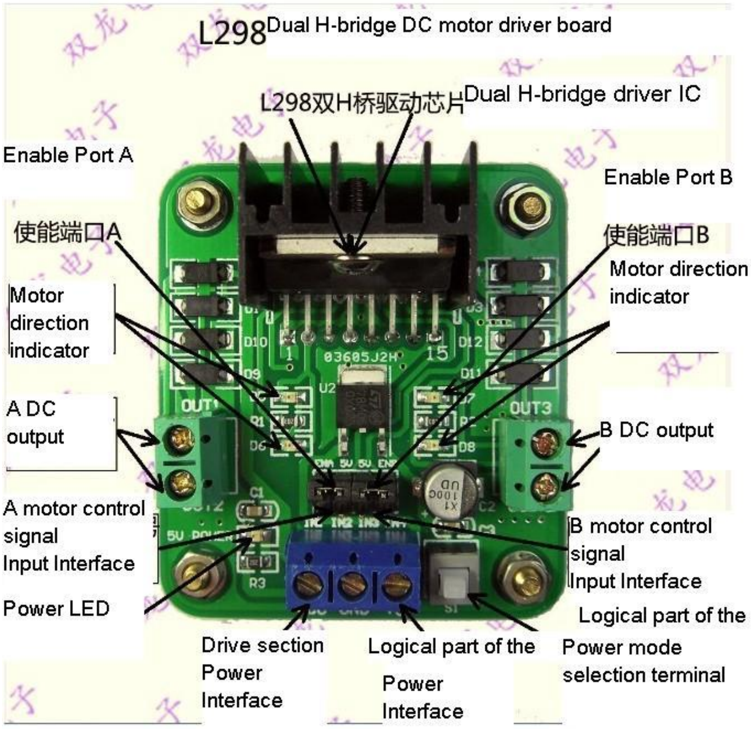


| ENA ENB A1 A2 B1 B2 | A B C D |
|---|---|
| 1 0 0 0 0 1 | 1 0 0 0 |
| 1 0 0 0 1 0 | 0 1 0 0 |
| 0 1 0 1 0 0 | 0 0 1 0 |
| 0 1 1 0 0 0 | 0 0 0 1 |

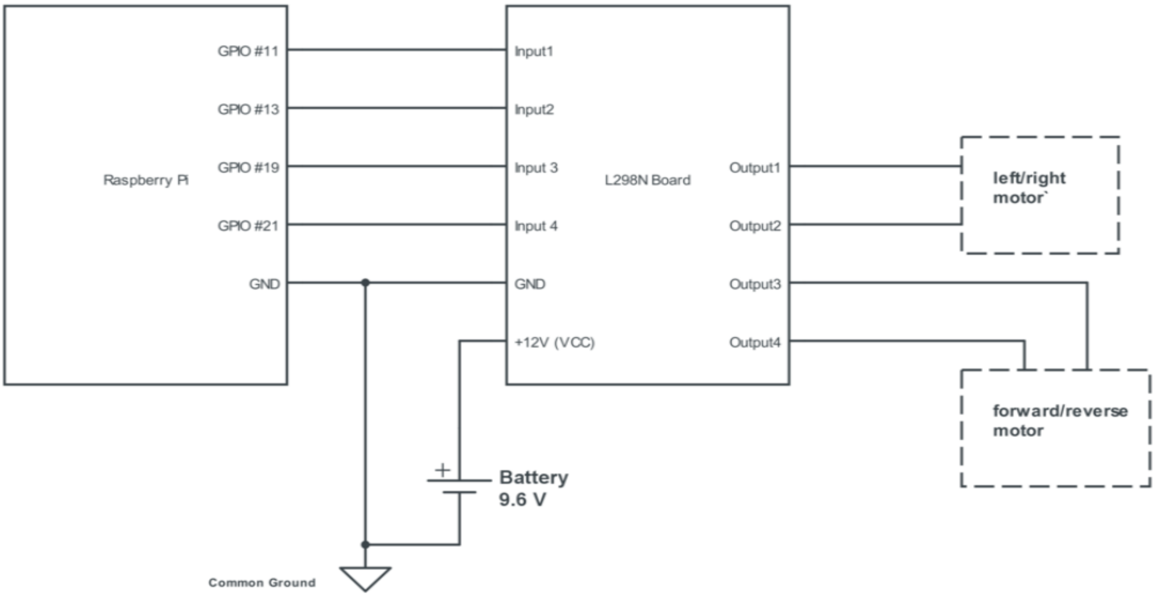
系统原理图如下： System works as follows



板子端口介绍：


L298Dual H-bridge DC motor driver board
L298双H桥驱动芯片Dual H-bridge driver IC
Enable Port A
使能端口A
Motor direction indicator
A DC output
A motor control signal Input Interface
Power LED
Drive section Power Interface
Logical part of the Power Interface
Enable Port B
使能端口B
Motor direction indicator
B DC output
B motor control signal Input Interface
Logical part of the
Power mode selection terminal
OUT1
OUT3
03605J2H
5V POWER

How it is done:



## Projektaufgaben (ger)

- **Server – Client**
  Dieser Teil soll eine Server - Client sein. Ich stelle mir ein Nachrichtensystem vor. Am Server meldet sich der Client(Roboter) an. Ein "Operator" meldet sich auch am Server an und sieht alle verfügbaren Roboter- und kann die Steuerung übernehmen. Ich dachte an eine Art STOMP System, bin aber für Vorschläge offen!
- **Webauftritt & Client**
  DerRoboter wird über einen Server gesteuert. Auf diesem Server soll auch eine Website dafür sein. Die Steuerung soll auch über einen HTML5/CSS3/JS Client geschehen!
- **A/V Streaming**
  Die ersten beiden Aufgaben beinhalten einen Server, einen Client, die Steuerung und den Webauftritt. jetzt braucht der Roboter noch Audio/Video Streaming- damit man ihn von überall steuern kann und sieht/hört, was drumherum los ist....
- **Sensorik**
  Der Roboter sollte mit den ersten paar Projekten der Liste voll steuerbar sein. Wenn man allerdings eine autonome Steuerung will (Der Robit fährt alleine wohin...), dann braucht man zusätzliche Sensoren, damit er niemanden überrollt... (Wenn der Robot größer wär- ein Problem).

Jede Teilaufgabe ist für ein Team von 1-2 Studierenden gedacht- insegsamt bilden alle Teams ein großes Projektteam.