# Anforderungsanalyse

## Software-Ist-Zustand

Gegeben ist ein System zur Kommunikation zwischen zwei Komponenten über das TCP/IP-Protokoll mittels Wireless-Lan. Es handelt sich um eine Client-Server-Architektur. Das bestehende System definiert zwei Subsysteme, eines ist stationär und das andere mobil. Hierbei übernimmt das mobile Subsystem die Aufgabe als Server.

Das gegebene System wurde ursprünglich entwickelt um Messdaten eines Sportlers mobil aufzunehmen und anschließend an einen stationären Client weiterzusenden. Um diese Aufgabe zu erfüllen wurden Funktionen zum Ansprechen und Auslesen von Analog-Digital-Wandlern über serielle Schnittstellen definiert. Die mobile Seite des Systems besitzt diese Funktionen und kann über spezielle Kommandos von der stationären Seite angesprochen werden. Den Aufbau der Software sieht man in Grafik 1(!). Es werden zwei Prozesse gestartet, wobei der Netzwerk-Prozess den Socket öffnet und eingehende Nachrichten von dort abruft und der ProzessPi-Prozess die empfangenen Kommandos verarbeitet und die Wandler ausliest.

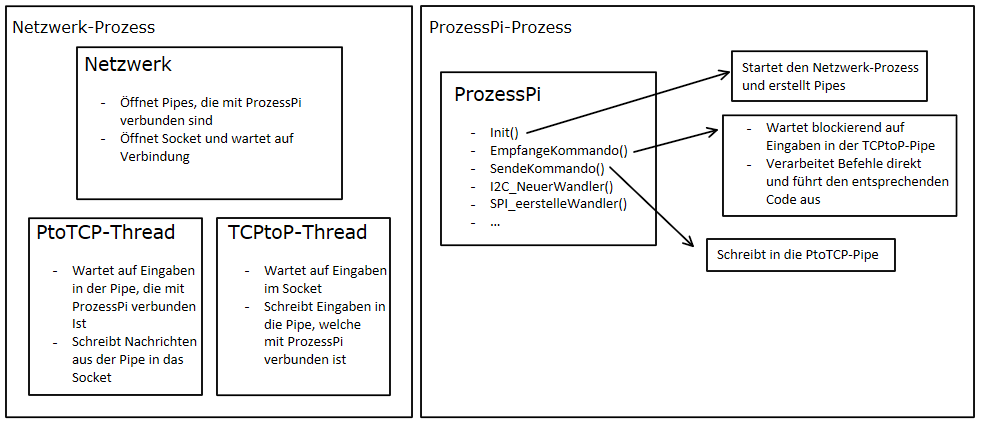


Abbildung 1: Aufbau der Ist-Software

Der Netzwerk-Prozess wird in der init()-Funktion von ProzessPi gestartet. Er läuft komplett unabhängig vom restlichen Programm ab und wird mittels Betriebssystem-Signalen (siehe Grundlagen) von ProzessPi beendet. Der Prozess startet zwei eigene Threads, den Eingangs- und Ausgangsthread. Der Eingangsthread wartet, bis eine Nachricht im Socket angekommen ist und schreibt diese dann ungefiltert in eine Pipe, die von ProzessPi gelesen wird. Der Ausgangsthread wartet bis in eine zweite Pipe, die ebenfalls mit ProzessPi verbunden ist, eine Nachricht geschrieben wird und leitet diese an das Socket weiter. So können zu jeder Zeit Nachrichten über das Netzwerk gesendet werden. Zum genaueren Verständnis der Netzwerk-Programmierung, siehe den entsprechenden Punkt in den Grundlagen.

Der ProzessPi-Prozess ist dafür zuständig die Nachrichten aus dem Netzwerk zu analysieren und entsprechend zu antworten. Dafür gibt es einige Funktionen zur Ansteuerung von seriellen Schnittstellen, wie zum Beispiel I2C oder SPI. Die Befehle aus der Nachricht werden in einer Funktion mit bekannten Befehlen verglichen und dementsprechend gehandelt und eine Antwort gesendet. Zum Starten von ProzessPi muss die init()-Funktion aufgerufen werden. Diese initialisiert die Pipes und führt den Netzwerk-Prozess aus. Für die Kommunikation wichtig sind die Funktionen SendeKommando() und EmpfangeKommando(), welche die Werte entsprechend dem Protokoll verpacken oder entpacken und in die Pipe schreiben, die mit dem Netzwerk-Prozess verbunden ist. Zum Beenden des Netzwerk-Prozesses wird beim Empfangen des „Beenden“-Kommandos das Signal „SIGUSR“, das vom Linux-Betriebssystem bereitgestellt wird, gesendet. Es gibt ebenfalls die Möglichkeit die empfangenen Werte aus den Wandlern zwischenzuspeichern, sollte die Verbindung zur stationären Einheit abbrechen. Diese Werte können dann von der Speicherkarte der mobilen Einheit genommen werden.

Für die Kommunikation zwischen der mobilen und dem stationären Subsystem zu ermöglichen wurde ein einfaches proprietäres Protokoll entwickelt. Dabei wird der auszuführende Befehl und der dazugehörige Wert als String übertragen und mit Komma getrennt. Ein Semikolon kennzeichnet das Ende der Nachricht.

Der Aufbau des bestehenden Protokolls:

Kommando,Wert;

ProzessPi und Netzwerk-Prozess sind auf beiden Subsystemen ähnlich. Auf der stationären Seite fehlt allerdings die Möglichkeit auf den Zugriff von seriellen Schnittstellen, da dies bei der Entwicklung nicht die Anforderung war.

Eine Plattformunabhängigkeit ist nicht vorhanden. Beide Seiten, die mobile sowie die stationäre, sind für ein Linux-Betriebssystem programmiert worden. Um eine Unabhängigkeit zu erreichen müsste die Netzwerk-Programmierung geändert bzw. ergänzt werden. Außerdem muss eine andere Interprozesskommunikation gewählt werden, da nicht alle Betriebssysteme Signale in dieser Form unterstützt.

Nähere Informationen können der Masterarbeit „Entwicklung eines mobilen Systems zur Erfassung und Verarbeitung online anfallender Prozessdaten“ von Theo Gabloffsky entnommen werden.

## Hardware-Ist-Zustand

Die Hardwareauswahl des bestehenden Systems ist sehr eingeschränkt. Nach genauer Analyse der Anforderungen des Projekts wurde entschieden, dass die mobile Einheit ein Raspberry Pi sein soll. Dieser Einplatinenrechner ist klein genug und besitzt ausreichend Rechenkraft um Messwerte aufzunehmen und über Wireless Lan zu kommunizieren. Mit dem Raspberry Pi können sehr einfach Sensoren über Pins angeschlossen werden und auch die USB-Anschlüsse können hilfreich sein. Eine besondere Eigenschaft des Raspberry Pis ist, dass man ein komplettes Betriebssystem aufspielen kann und dort wie auf einem „normalen“ PC Programme starten kann. Dies macht einige Dinge einfacher, hat aber auch Nachteile (siehe Grundlagen Raspberry Pi). Ein weiterer Vorteil bei der neuesten Version dieses mobilen Systems ist der integrierte WLan-Chip. Da die Software direkt für den Raspberry Pi geschrieben wurde, ist es schwierige diese auf ein neues System zu portieren. Der Raspberry Pi ist also die gegebene Hardware für den mobilen Teil des Systems. Es ist möglich über die Software AD-Wandler anzusprechen. Mit Hilfe dieser AD-Wandler sollten analoge Messdaten kontinuierlich aufgenommen werden und digital an das mobile Subsystem weitergeleitet werden.

Bei der stationären Einheit gibt es weniger Einschränkungen. Es ist nur, wie oben in dem Software-Ist-Zustand beschrieben, nötig, dass Linux als Betriebssystem installiert ist, denn sonst funktioniert die Kommunikation der Prozesse untereinander und über die Sockets nicht.

Auch hier der Verweis auf die Masterarbeit von Theo Gabloffsky, die genauere Informationen enthält.

## Funktionale Anforderungen

### Hardware-Schnittstellenverfügbarkeit

Die mobile Einheit des Systems, die später als Basis für neue Roboter dienen soll, sollte möglichst viele verschiedene Schnittstellen zur Anbindung von Roboterkomponenten bieten. Diese Anforderung ist eine der wichtigsten, da bei dem Projekt die Modularität einen hohen Stellenwert besitzt. Zukünftige Roboter sollen einfach und ohne großen Softwareaufwand in das System integriert werden können, weswegen eine Schnittstellenvielfalt unabdingbar ist. Der Raspberry Pi bietet dafür eine perfekte Basis. Sensoren und Aktoren können über die GPIO-Pins direkt am Pi betrieben werden. Über serielle Schnittstellen, wie zum Beispiel I2C oder UART können von einem Chip gesteuerte Aktoren und Sensoren angesprochen werden. Dies ist unter anderem bei sogenannten Pi-Hats notwendig, welche auf den Raspberry Pi aufgesetzt werden können und anschließend die Steuerung von Motoren übernehmen. Diese eigenständigen Chips können mit Befehlen über die I2C-Schnittstelle gesteuert werden. Auch zur Anbindung von fertigen Robotern, wie zum Beispiel dem Epuck, kommen serielle Schnittstellen zum Einsatz. Die vorhandenen USB-Anschlüsse stellen eine einfache Möglichkeit dar eine Webcam oder Ähnliches anzuschließen.

Durch die Verwendung des Raspberry Pis stehen dem späteren Nutzer viele Schnittstellen zur Verfügung, die auch gleichzeitig zum Einsatz kommen können. Um das zu erstellende System so modular wie möglich zu gestalten, sollten diese Schnittstellen einfach zu nutzen sein.

### Kommunikationsprotokoll

Für die Kommunikation zwischen stationärer und mobiler Einheit wird ein Protokoll benötigt, welches die Nachrichten zwischen Fernsteuerung und ausführender Einheit verpackt. Das Protokoll wird eine Schicht unter dem proprietären Protokoll des ProzessPi-Prozesses (Kapitel 1.1) liegen. Der Aufbau sollte geteilte Pakete zulassen um große Mengen an Daten zu übertragen. Alle Nachrichten werden nach dem Request-Response-Prinzip versendet. Das bedeutet das Protokoll sollte Anfragen und Antworten zulassen und zuordenbar machen. Beim Transport entstehende Fehler werden von den oberen Schichten abgedeckt. Es sollten lediglich Protokoll-Fehler erkannt werden.

### Software-Schnittstellenverfügbarkeit

Auf der stationären Seite sollte eine Schnittstelle definiert werden, die einfach benutzbar eine Fernsteuerung der mobilen Einheit ermöglicht. Diese Schnittstelle wird später von Studenten in der Vorlesung C++/GUI genutzt um eine Grafische Oberfläche zu erstellen, mit der der an der mobilen Einheit angeschlossenen Roboter ferngesteuert werden kann. Dabei ist auf eine einfache Benutzbarkeit zu achten, da die Studenten kein tieferes Wissen über das zugrundeliegende System benötigen sollen. Über die Schnittstelle sollten sämtliche Funktionen zur Steuerung des Roboters verfügbar sein. Dazu zählen unter anderem: fahren, Sensoren auslesen, Kamera auslesen, andere Aktoren ansteuern, LEDs setzen.

### Netzwerkkommunikation

Diese Anforderung wird weitestgehend vom bestehenden System erfüllt. Hier ist es nur wichtig, dass notwendige Anpassungen gemacht werden, um dauerhaft neue Clients auf Serverseite annehmen zu können. Bisher wird hier nach beenden einer Verbindung auch das Netzwerkprogramm beendet. Dies ist für eine Roboteranwendung sehr unpraktisch, da nicht ständig das Programm auf der mobilen Einheit neugestartet werden soll. Außerdem können eventuell weitere Anpassungen notwendig sein um die Kommunikation über WLAN zu ermöglichen. Da das System später eigenständig laufen soll, ist es wichtig, dass auf dem Raspberry Pi ein mobiler Hotspot eingerichtet wird. So können sich stationäre Clients mit der mobilen Einheit ohne ein externes Netzwerk verbinden.

### Kamerainput

Um einen zukünftigen Roboter vernünftig zu steuern, ist es wichtig, dass Bilder einer Kamera eingelesen und übertragen werden können. Der bisher benutze Roboter besitzt eine Kamera, die jedoch nur eine sehr geringe Auflösung von 640x480 Pixel unterstützt. Durch den schwachen Prozessor und den geringen Arbeitsspeicher des Epucks, muss diese Auflösung noch reduziert werden auf 40x40 Pixel. Mit dem Raspberry Pi als Recheneinheit, soll das jetzt verbessert und eine normale Webcam getestet werden. Die Schwierigkeit besteht darin, dass eine große Menge an Daten übertragen werden müssen um ein flüssiges Anzeigeergebnis zu bekommen. Deshalb soll die benötigte Bandbreite, mit der die Übertragung der Bilder das Netzwerk belastet, getestet werden.

### Beispielimplementation

Zu Vorführungszwecken soll eine Grafische Oberfläche entwickelt werden, die die geforderten Funktionen präsentiert. Damit dieses Beispiel in der Vorlesung verwendet werden kann, soll die Oberfläche mit Qt programmiert werden.

## Nichtfunktionale Anforderungen

### Benutzbarkeit

Grundlegend müssen Softwareschnittstellen immer einfach zu benutzen sein, denn sonst würden sie ihren Sinn verfehlen. Deshalb ist beim Entwickeln solcher Schnittstellen die Benutzbarkeit ein wichtiger Aspekt.

Bei dem gesamten System handelt es sich hauptsächlich um Softwareschnittstellen: Eine Schnittstelle befindet sich auf der stationären Seite und eine auf der mobilen.

Die Schnittstelle zur Fernsteuerung der mobilen Einheit soll einfach benutzbar sein, damit jeder diese Schnittstelle nutzen kann, ohne tiefe Programmierkenntnisse in C++. Da diese Schnittstelle von Studenten genutzt wird, die vermutlich in der Vorlesung ihre ersten Erfahrungen mit der Programmiersprache C++ machen, ist eine einfache Einbindung der Fernsteuerung notwendig. Sie sollte allerdings auch von C++-Features Gebrauch machen und so die Studenten zum Verwenden der neuen Techniken bringen. Da in der Vorlesung nicht die Zeit vorhanden ist, das komplette System vorzustellen, sollte die Schnittstelle ohne jegliches tiefere Wissen zum System verwendbar sein.

Die Schnittstelle zur Hardware des Roboters auf der mobilen Einheit kann im Gegensatz zu eben beschriebener Schnittstelle etwas mehr Fachwissen voraussetzen, da sie für einen professionellen Nutzer geschrieben wird. Dieser möchte Komponenten des Roboters mit dem System verbinden und bekommt dafür eine Hilfe, um nicht alles selbst programmieren zu müssen. Allerdings sollte auch hier auf Benutzbarkeit geachtet werden, da eine Anforderung an das System das schnelle Verständnis der zu nutzenden Funktionen ist.

### Zuverlässigkeit

Da die mobile Einheit nicht ständig an einen Monitor angeschlossen wird, ist eine Garantie, dass das Programm nach dem Start ständig läuft notwendig. Bei fehlerhaftem Verhalten von der stationären Seite soll das System weiterlaufen und Rückmeldung über den Fehler geben. Auch beim Abbrechen der Verbindung oder bei einem gewollten abkoppeln des Clients soll der Server auf der mobilen Einheit weiter auf neue Clients warten.

Da die Studenten in der Vorlesung überwiegend selbstständig mit den Robotern arbeiten sollen, ist es wichtig, dass eine Stabilität des Systems gegeben ist, damit nicht die meiste Zeit in den Roboter investiert wird, sondern in die eigentliche Aufgabe.

### Modularität

Modularität ist ein Hauptaspekt des zu erstellenden Systems, da der Roboter, der später an die mobile Einheit angeschlossen wird noch nicht bekannt ist. Deshalb sollten sämtliche Schnittstellen einfach erweiterbar sein, um Funktionen hinzuzufügen, die Sensoren oder Aktoren bedienen, die jetzt noch nicht berücksichtigt werden können. Wenn diese Anforderung nicht erfüllt werden würde, müsste das gesamte System nochmal neu aufgesetzt werden, sobald der Roboter bekannt ist. Da dies nicht die Absicht ist, wird es notwendig sein, die Software so zu entwerfen, dass so wenig wie möglich Eingriffe in den schon geschriebenen Code gemacht werden müssen, um den neuen Roboter zu integrieren.

Um diese Modularität voll ausnutzen zu können, muss eine gute Dokumentation vorhanden sein.

### Datenrate

Kamerabilder zu übertragen benötigt viele Daten und hat einen hohen Rechenaufwand, aufgrund der zu schreibenden und analysierenden Menge an Bytes. Diese Anforderung soll erfüllt werden, damit auf dem fernsteuernden Gerät später eine Übertragung der mobilen Einheit zu sehen ist und der Roboter gelenkt werden kann ohne, dass er gesehen werden muss.

Um eine flüssige Wiedergabe zu gewährleisten, muss die benötigte Datenrate bestimmt werden und getestet werden, ob das System die Anforderungen erfüllen kann.