Technische Dokumentation

Mr. Tea

HAW Hamburg
Fakultät DMI

Projekt durchgeführt und Dokumentation verfasst von:

Artur Tahiraj	[2009542]
Stefanie Kohl	[2011097]
Irena Becker	[2238833]
Oliver Völling	[2248958]

Bei Prof. Dr. Thorsten Edeler und Prof. Dr. Plaß
Im Fach IT-Systeme und Mobile Systeme

Abgabe am 20.07.2017

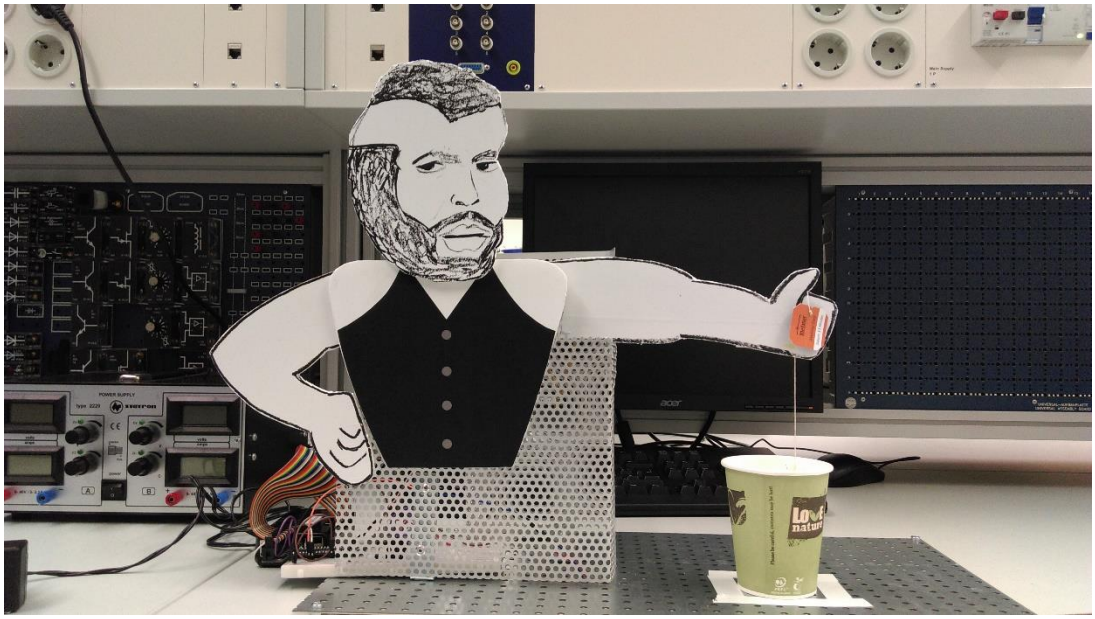
Inhalt

Ergebnis des Projekts

Wir haben die Ziele, die wir im Konzept festgelegt haben, erreicht. Unser Endprodukt, kann mittels einer App gesteuert werden und brüht eigenständig Tee, bis die vorgegebenen Ziehzeit erreicht wurde. Dazu muss Wasser aufgekocht und in Behältnis gegossen und der Teebeutel an den Arm der Maschine befestigt werden. Das Thermometer wird jedoch etwas anders verwendet. Anstatt, dass gewartet wird bis die Wassertemperatur ideal für die jeweilige Teesorte ist, wird der Teebeutel sofort in das heiße Wasser eingetaucht. Sobald der Tee fertig gezogen ist, liefert das Thermometer die Temperaturdaten und wartet solange bis der Tee trinkbereit ist, dann wird eine Benachrichtigung auf das Handy geschickt.

Das Design wurde an den Schauspieler Laurence Tureaud (Mr. T) angelehnt, der eine typisch englische Weste trägt. Er hält den Teebeutel in seiner linken Hand, an der eine Messingstange befestigt ist, die wiederrum an einen Servomotor geschraubt wurde. Das Thermometer befindet sich unter der Fläche, an welcher die Tasse oder der Becher platziert wird. Es ist elastisch auf einem Schwamm gebunden und ragt ca. 1 cm aus der Metallfläche heraus, damit sichergegangen wird, dass Kontakt zwischen dem Behältnis und dem Thermometer besteht.

Im Folgenden werden auf nähere technischen Details und die Programmierung der Applikation zum Bedienen der Maschine eingegangen.

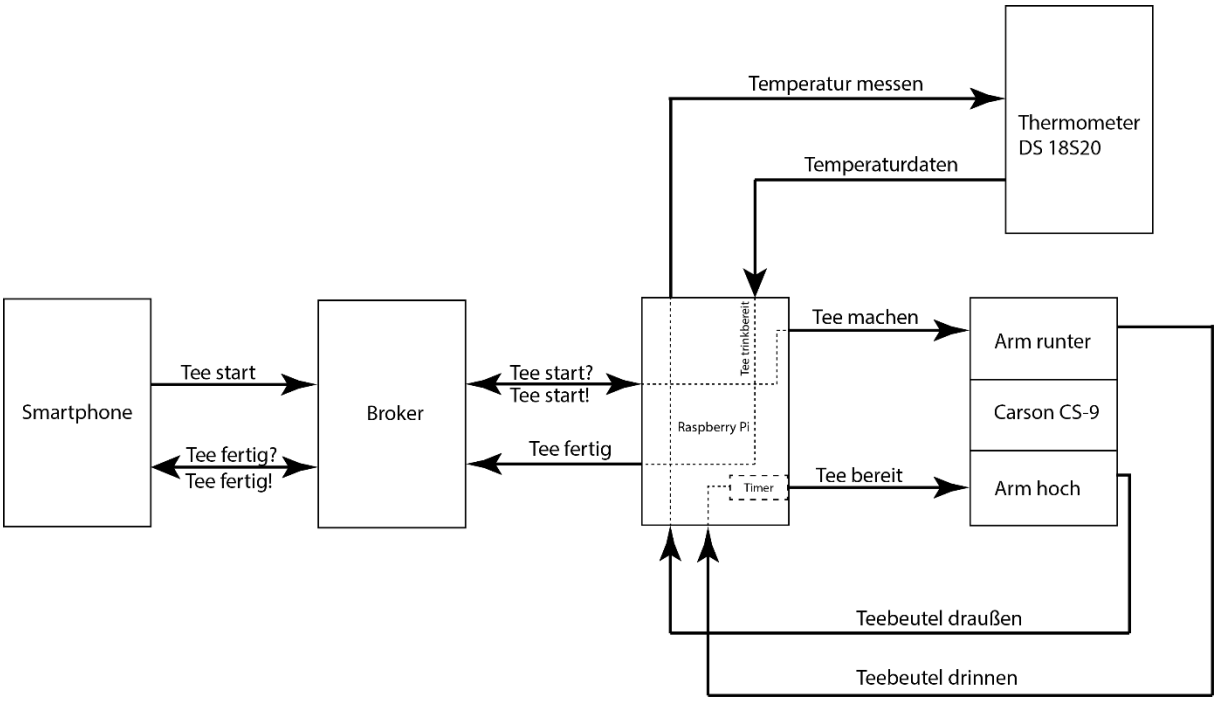


Technische Umsetzung der Hardware mit dem Raspberry Pi

Zur technischen Umsetzung von Mr. Tea benötigen wir einen Servomotor, ein Thermometer, einen 4700 Ohm Widerstand, eine sechs Volt Spannungsquelle und einen Raspberry Pi. Unsere gewählten Bauteile waren dafür:

- Servomotor: Carson CS-9
- Thermometer: Maxis DS 18S20
- Spannungsquelle: 4 mal 1,5V Batterien in Reihe geschaltet
- Raspberry Pi 3 [raspi06.local]

Signalfluss und Skript



Da wir die Kommunikation zwischen Raspberry Pi und Smartphone App mittels Broker und Topics realisiert haben wird auf Seiten des Raspberry Pi innerhalb eines Python 3 Skriptes abgefragt, ob etwas in dem abonnierten Topic verändert. Nach Auswahl des Tees in der App wird die entsprechende Teesorte auf den Broker übertragen und der Raspberry Pi liest die Daten aus. Dabei ist die Ziehzeit für die jeweilige Teesorte im Python 3 Skript gespeichert und wird dort abgefragt. Der Raspberry Pi gibt das Signal an den Servomotor, sodass dieser sich Schrittweise in einem ca. 30 Grad Winkel senkt und somit eine Messingstange in Bewegung setzt, an deren Ende ein Teebeutel in die Tasse taucht.

```
while on:
    for i in range (88, 75, -1):
        p.ChangeDutyCycle(i/10)
        time.sleep(0.1)
    p.ChangeDutyCycle(0)
    time.sleep(0.1)
```

Dabei startet im Python 3 Skript ein Timer, der der Ziehzeit entspricht. Nachdem dieser abgelaufen ist und niemand in der App auf den Stopp Button gedrückt hat, sendet der Raspberry Pi wieder ein Signal zu dem Motor, sodass sich der Motor mit derselben Schrittweite nach oben bewegt und wieder auf seiner Ursprungsposition ist.

Sobald der Motor wieder oben ist, liest das Python 3 Skript das Thermometer aus. Das DS 18S20 sendet die Daten konstant an den Raspberry Pi, der die Informationen in einem Unterverzeichnis speichert. Dieses Verzeichnis wird dabei ausgelesen und die Temperatur in einer Variable gespeichert.

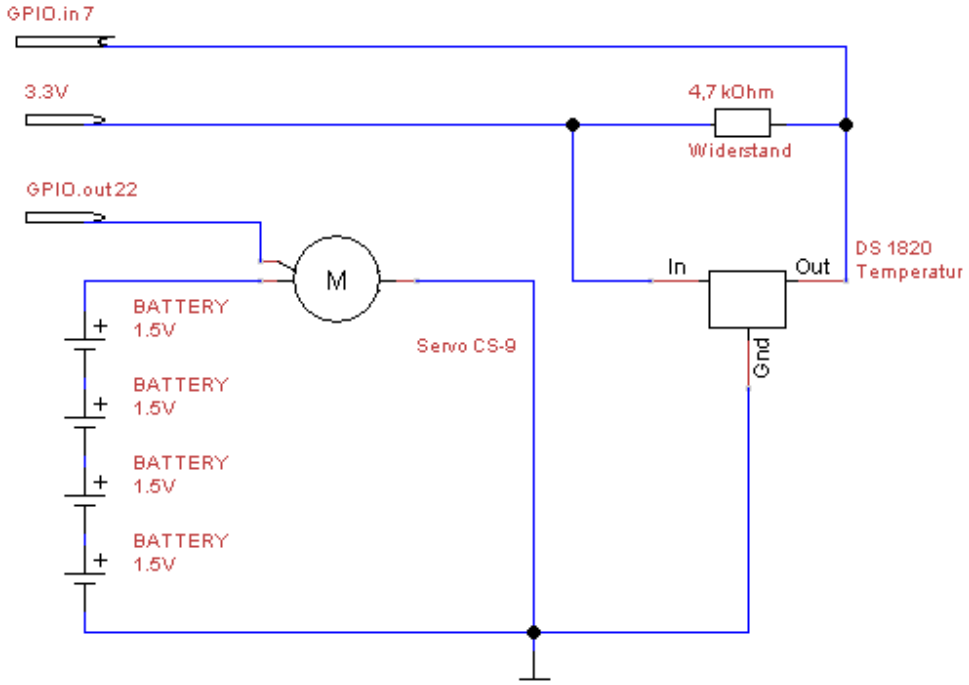
```
file = open('/sys/bus/w1/devices/' + '10-0008032a18bc' + '/w1_slave')

filecontent = file.read()
file.close()
stringvalue = filecontent.split("\n")[1].split(" ")[9]
temperature = float(stringvalue[2:]) / 1000
print("%5.3f °C" % temperature)
return

sys.exit(0)
```

Sollte die Temperatur unter 37 Grad fallen, gilt der Tee als trinkbereit. Die Temperatur wird alle 5 Minuten abgefragt. Wenn der Tee trinkbereit ist, wird an den Broker gesendet, dass der Tee nun trinkbar ist.

Elektrische Schaltung und Signalwege



Zur elektrischen Realisierung für die 6 Volt für den Motor wurden 4 1,5 Volt Batterien in Reihe geschaltet und ebenfalls in Reihe mit dem Motor. Die Minus Leitung des Motors wurde zusätzlich mit Masse des Raspberry Pis verbunden. Der GPIO Pin 22 im BCM Setup des Raspberry Pis dient als Output und überträgt die 3,3 Volt für die Digitale Signalübertagung. Dazu wurde eine Pulsweitenmodulation mit 50 Hertz benutzt, um das Signal an den Motor zu übertragen. Der Motor benötigt für die Ausführung seiner Bewegung 1,2 Ampere.

Das Thermometer DS 1820 besitzt drei Anschlüsse. Es benötigt einen Input von 3,3 Volt, welcher vom Raspberry Pi direkt gespeist wird. Die Masseleitung des Thermometers wird mit der Masse des Raspberry Pis verbunden. Der Output des Thermometers bzw. dessen Datenleitung wird mit den GPIO Pin Nummer 7 verbunden. Dieser wird als Eingang festgelegt. Zwischen der Datenleitung und der Inputleitung befindet sich ein 4700 Ohm Widerstand. Über diesen Widerstand wird die abfallende Spannung gemessen und je nach Temperatur ändert sich diese. Diese Änderung wird vom Raspberry Pi ausgewertet und mithilfe einer Funktion innerhalb des Skripts als Temperatur interpretiert.

Programmierung und Gestaltung der App

Die App ermöglicht zunächst die Bedienung des Roboterarms. Außerdem wird der Nutzer über die App über den aktuellen Stand der Teezubereitung informiert und erhält eine entsprechende Mitteilung, wenn der Tee auf die trinkbereite Temperatur abgekühlt ist.

Die Interaktion über ein Smartphone bietet sich an dieser Stelle an, da im aktuellen digitalen Zeitalter in jedem Haushalt mindestens ein Smartphone vorhanden ist. Die Kommunikation über das Smartphone bietet dem Nutzer die Möglichkeit, sich weiteren Verpflichtungen zu widmen und übernimmt die Aufgabe der Teezubereitung.

Um die Applikation so plattformunabhängig zu implementieren, entschieden wir uns für eine Umsetzung in HTML und JavaScript.

Im ersten Schritt programmierten wir eine Webanwendung. Als die Funktionalität und die Verbindung zum Server/Broker implementiert waren, gestalteten wir das Frontend nach unseren Vorstellungen weiter aus. Zuletzt wandelten wir unsere Webapplikation mit Hilfe von *Cordova* und *PhoneGap* in eine native App für Android Smartphones um. Durch unsere Webanwendung haben nun auch Nutzer von iPhones und WindowPhones die Möglichkeit unseren Roboter zu bedienen.

Beim Start der App wird noch keine Verbindung zum Broker aufgebaut. Erst wenn eine Teesorte ausgewählt wird, verbindet sich die App mit dem Broker, submittet den zuvor definierten Raum und sendet die Information über die gewählte Teesorte an den Broker.

Da die Schritte 1) subscriben, 2) publishen und 3) der Wechsel auf den neuen Screen zeitlich genau in dieser Reihenfolge erfolgen sollen, haben wir einen Delay von jeweils 100 Millisekunden zwischen die einzelnen Vorgänge integriert.

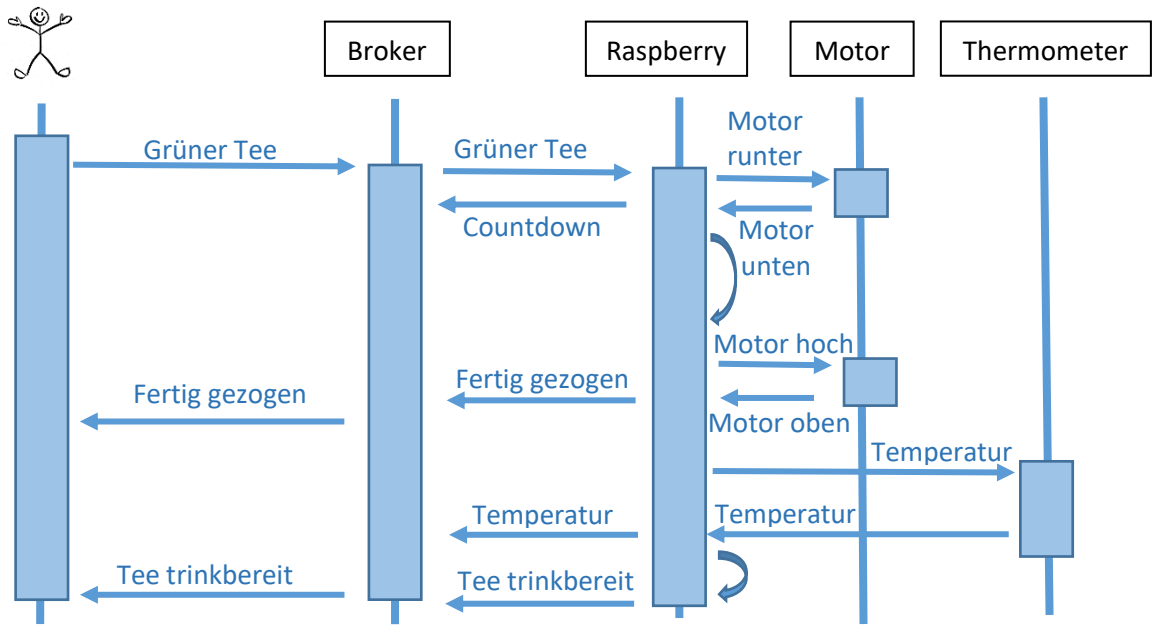
```
var delayMillis = 100; /

setTimeout(function() {
    client.subscribe('haw/dmi/mt/its/ss17', {qos: 2}); //y
}, delayMillis);
setTimeout(function() {
    publish('Schwarztee','haw/dmi/mt/its/ss17',2); //your
}, delayMillis);
setTimeout(function() {
    window.location.replace('schwarz.html'); //your code
}, delayMillis);
```

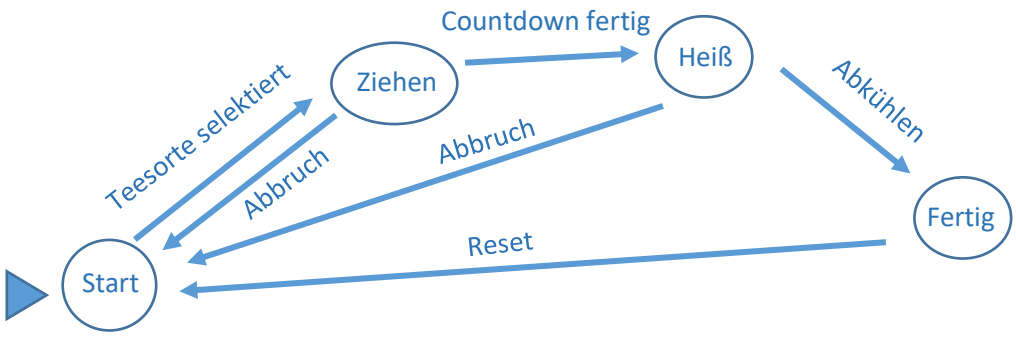
Im weiteren Verlauf „wartet“ die App auf die Information „fertig gezogen“ im Broker und ändert in diesem Fall den Screen. Der Nutzer wird darüber informiert, dass der Tee zwar fertig, jedoch zu heiß und noch nicht trinkbereit ist. Wenn das Thermometer die vordefinierte Temperatur an den RaspberryPi ünergibt, wird die Information „Tee trinkbereit“ in den Broker gegeben. Die Anwendung liest die Information aus und gibt sie an den Nutzer weiter. Die Anwendung kann zu jeder Zeit abgebrochen werden. Dabei wird die Last-Will Nachricht "abbr“ an den Broker gesendet und die Verbindung zum Broker wird zurückgesetzt.

Abbildungen: *Benutzerscreens mit 1) Wahl des Grünen Tees, 2) Abwarten der Ziehzeit, 3) Abwarten der Abkühlung auf trinkbare Temperatur, 4) Information über die Fertigstellung des Tees*

Ein Sequenzdiagramm veranschaulicht die zeitliche Abfolge der Vorgänge folgendermaßen:



Folgende Zustände ergeben sich hieraus:



Zusammenfassend können wir festhalten, dass die Umsetzung der App mit HTML und JavaScript eine übersichtliche Aufteilung in Frontend und Logik ermöglichte. Auch unser erstmaliger Einsatz von Cordova und PhoneGap erwies sich als nutzerfreundlich und unkompliziert. Um die Ladezeiten der Screens zu verringern hätten wir von Anfang an unsere App nach dem Prinzip einer Single-Page Anwendung implementieren können. Außerdem werden wir in Zukunft Push-Nachrichten anzeigen lassen, statt aufwändige Screenwechsel für gewünschte Brokerinhalte zu programmieren.