Rafał Łokuciejewski
Jens Krauth
Sven Andresen

Internettechnologien Projekt - Doku 1

Raspberry Pi
Bei dem Raspberry Pi handelt es sich um ein Einplatinencomputer, der über einen ARM-Mikroprozessor verfügt und dessen Größe ca. einer Kreditkarte entspricht. Auf dem von uns benutzten Pi wurde eine Linux-Distribution installiert, nämlich der Rasobian Jessie Lite, welches auf Debian basiert. Der weitere Text beschäftigt sich mit der Inbetriebnahme des Geräts.

Um den Raspberry Pi nutzen und konfigurieren zu können, wird es an ein Bildschirm mittels HDMI-Kabels angeschlossen. Da das Betriebssystem keine grafische Oberfläche zur Verfügung stellt, ist es nötig, die Befehlsbasierte Eingabe zu nutzen.

Der erste Schritt bestand daraus, sich bei dem Betriebssystem anzumelden. Um die nötigen Rechte zu erhalten, wurde die Command-Line $sudo eingegeben und danach startete man mit $raspi -config die Konfiguration des Computers. Als erstes wurden Spracheinstellungen angepasst und SSH aktiviert, sowie das Passwort verändert.

Um die am PC geschriebene Software auf den Pi übertragen zu können, braucht man vorzugsweise eine WLAN-Verbindung. Diese wurde auch konfiguriert, indem man 2 Dateien (telekom.crt und wpa_supplicant.conf) modifizierte, die im WLAN-Verzeichnis lagen. Anschließend musste man die Dateien als root in das WPA-Supplicant Verzeichnis kopieren. Nachdem das WLAN konfiguriert wurde, folgte die Aktualisierung des Pi:

- $ sudo apt-get update
- $ sudo apt-get upgrade
- $ sudo apt-get dist-upgrade

Um mit der Tabulator-Taste eine Vervollständigung der bisherigen Eingabe zu ermöglichen, wurden die BASH-Vervollständigung aktiviert. Dies geschiecht, indem man die Datei **/etc/bash.bashrc** als root öffnet und entsprechende Zeilen auskommentiert. Zusätzlich, um die Verwaltung und Beschaffung der Software zu ermögliche, wurde auf dem Pi ein Repository (Git) installiert: **$sudo apt-get install git-core**.

Der Raspberry Pi verfügt über sog. General Purpose Input Output (GPIO), d.h. programmierbare Ein- und Ausgänge für allgemeine Zwecke. Diese liegen in Form von Lötpunkten (Pins) vor und fungieren als Schnittstelle zu anderen Systemen oder Schaltungen. Um die GPIOs zu konfigurieren und zu steuern, wurde ein Framework WiringPi installiert:

- $ git clone git://git.drogon.net/wiringPi
- $ cd wiringPi
- $ ./build

Das Framework ist eine in C geschriebene Library und kann von den meisten gängigen Programmiersprachen benutzt werden (in unserem Fall: Java). Mit der Command **$read all** kann man dann die aktuelle Konfiguration der Pins sehen. Zusätzlich kann man die auch als Input oder Output deklarieren, sowie ihren Zustand lesen bzw. setzen, was bei der Programmierung nötig ist.

Wie bereits erwähnt, wird im Rahmen des Projektes Java verwendet, wobei die Software an einem getrennten PC geschrieben wird und dann in ein Git-Repository hochgeladen. Dann wird mit die Software vom Pi aus vom Git gepullt. Um eine Java-Programm ausführen zu können, ist es nötig, eine Java-Version auf dem Pi zu installieren, nämlich die Oracle-Version:

- $ sudo apt-get install oracle-java8-jdk

Mit **$java -version** kann man die aktuelle Java-Version verifizieren.
Um eine Schnittstelle von Java zum Pi zu erhalten wurde die Library PI4J benutzt. Zusätzlich wird Maven als Projektverwaltungstool verwendet.

Um eine serielle Kommunikation für Java-Programm zu ermöglichen, wurde die Library RXTX installiert.

Aufgabe 1.1

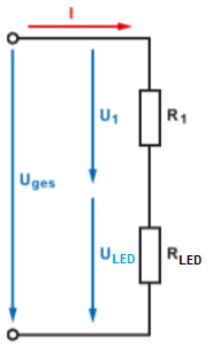
Die erste Aufgabe bestand daraus, eine LED Mittels Raspberry Pi zum leuchten zu bringen. Dazu wurde ein Steckboard benutzt, welches über 3.3V verfügte. Die Versorgung kam von dem GPIO 21 (Laut BCM Benennung), welches als Out-Pin konfigureirt wurde und dessen Wert auf 1 gesetzt wurde:

- $gpio -g mode 21 out

- $gpio -g write 21 1

Da eine LED mit 2.5V betrieben werden soll, ein GPIO aber 3.3V leistet, war es nötig, einen Vorwiederstand anzuschließen. Da es sich bei dieser Schaltung um eine einfache Serien Schaltung mit Versorgung, Widerstand und LED handelt, kann der Vorwiederstand wie folgt bestimmt werden: 3.3V-2.5V=0.8V, d.h. am Widerstand sollen 0.8V abfallen $0.8V = R * 0.016A => R = 50\ Ohm$. Die entsprechende Schaltung ist in folgender Abbildung zu sehen.

Rafał Łokuciejewski
Jens Krauth
Sven Andresen



Aufgabe 1.2

Bei dieser Aufgabe sollte man zusätzlich den Fotowiederstand (LDR) in die Schaltung einbauen. Auch hier war es nötig, den ensprechenden Vorwiderstand zu berechnen, da ein GPIO eine logische 0 dann sieht, wenn die am ihm anliegende Spannung unter 0.8V liegt und eine 1 sobald die Spannung über 2.0V steigt. D.h. für die beiden Werte berechnet man den Vorwiderstand und nimmt den Mittelwert, der zwischen den beiden liegt: die Berechnung sieht dann wie folgt aus:

$$\frac{3,3V}{R_{ges}} = \frac{1,3V + 2,0V}{R + 10000\Omega} \Rightarrow R = \frac{1,3V * 10000\Omega}{2,0V} = 6500\Omega \quad (1)$$

und

$$\frac{3,3V}{R_{ges}} = \frac{2,5V + 0,8V}{R + 1000\Omega} \Rightarrow R = \frac{2,5V * 10000\Omega}{0,8V} = 3125\Omega \quad (2)$$

Das heißt ein passender Widerstand wäre bei 5000 Ω für diese Schaltung. Der Widerstand wird vor dem Fotosensor angeschlossen. In einem Pinslot im Steckbrett, der zwischen Vorwiderstand und Sensor liegt, kann nun der logische Wert an einen GPIO (Input) vom Pi weitergeleitet werden. Dieser kann über die Pi Console ausgegeben werden.
Als nächstes sollte der analoge Wert des Fotowiderstands über einen Arduino und dessen A/D-Wandler bereitgestellt werden. Dazu wurde der Arduino so konfiguriert, dass er über eine serielle Schnittstelle (TXRX) in äquidistanten Intervallen das gemessene Signal an den Pi überträgt. Der Arduino besitzt einen 10 Bit AD-Wandler, das heißt er liefert werte zwischen 0 und 1023.
Zunächst wird die serielle Schnittstelle initialisiert mit *Serial.begin(baudrate)*, die Baudrate wurde auf 9600 festgelegt. Also 9600 Bits/s.

Anschließend kann mit dem Befehl *analogRead(Pin)* der analoge Wert eingelesen werden und mit *Serial.print()* der Wert auf die serielle Schnittstelle geschrieben. Damit später in Java detektiert werden kann, wo eine Nachricht beginnt und endet, wird jedem Wert noch das Trennzeichen $ hinzugefügt. Dieses Szenario wird in einer Schleife immer wiederholt.
Die Verarbeitung der eingehenden Werte wurde mittels Java gelöst. Es wird gewartet bist das Trennzeichen $ gelesen wird, anschließend werden die gelesenen Bytes in einen String

konkateniert, bis ein erneutes Trennzeichen $ gelesen wird. Dies verhindert, dass abgeschnittene Werte bearbeitet werden.

Die Realisierung kann aus dem folgenden Codesnippet entnommen werden:

```
if(arg instanceof byte[]) {
    try {
        byte[] array = (byte[]) arg;
        //parse to string
        String arrayString = new String(array);
        //append to buffer
        buffer += arrayString;
        //find first occurence of Delimeter
        String xbuffer = buffer.substring(buffer.indexOf('$') + 1);
        if(xbuffer.contains("$")) { //If we find another Delimeter we have a full message
            xbuffer = xbuffer.substring(0, xbuffer.indexOf('$'));
        } else
            return;
        buffer = ""; // clear the buffer
        System.out.println(xbuffer); //output the message from the serial Port
        if(xbuffer.length() != 0 && Integer.parseInt(xbuffer) > 500) {//if greate than 500 turn led on
            pinLed.high();
        } else
            pinLed.low();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Der konkatenierte String wird anschließend auf der Konsole ausgegeben.

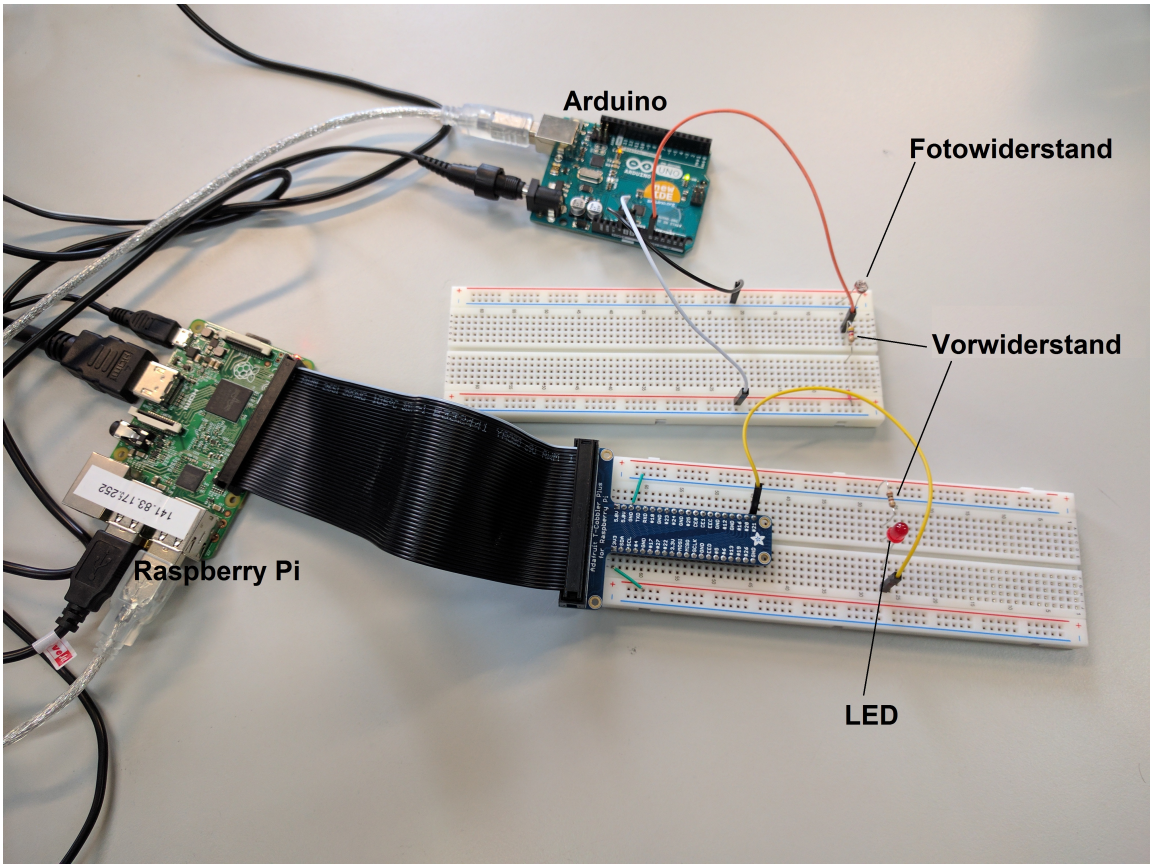
Die letzte Aufgabe war es, abhängig von dem analogen Wert eine Led zum leuchten zu bringen. Dies ist der letzten if Bedingung aus dem Codesnippet zu entnehmen. Mit dem Befehl *pinLed.high();* bzw *pinLed.low();* kann die LED an bzw. ausgeschaltet werden. Dafür müssen die Pins aber zuerst initialisiert mit den folgenden Befehl initialisiert werden. Der Threshold wurde experimentell auf 500 festgelegt.

- pinLed = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_29, "MyLed", PinState.LOW);

- pinLed.setShutdownOptions(true, PinState.LOW);

Das Ergebnis der Gesamtschaltung kann in folgender Abbildung gesehen werden.

Rafał Łokuciejewski
Jens Krauth
Sven Andresen



Für den nächsten Meilenstein ist es zusätzlich wichtig, dass der analoge Wert in den entsprechenden Luxwert umgerechnet wird. Dafür sind folgende Formeln gegeben

$$U_{LDR} = \frac{analogValue \cdot U_{in}}{1023} \tag{3}$$

$$R_{LDR} = \frac{R_V \cdot U_{LDR}}{U_{IN} - U_{LDR}} \tag{4}$$

$$E_v = R_{LDR}^{-1.31022} \cdot 210.910430 \tag{5}$$