

RC-Glied Batterie Spannungs Messung

Wichtig zu wissen, das ganze geht nur wegen der Vernachlässigung von V_{C1} ! Diese Methode ist relativ ungenau, bietet aber einen groben Überblick ob eine Batterie fast leer ist.

Formeln

$$V_{C1} = \frac{I_1 \cdot t}{C_1}$$

$$I_1 = \frac{V_{Bat} - V_{C1}}{R_1}$$

$$V_{Bat} \gg V_{C1}$$

$$I_1 = \frac{V_{Bat}}{R_1}$$

$$V_{C1} = \frac{V_{Bat} * t}{C_1 * R_1}$$

$$V_{Bat} = \frac{V_{C1} * C_1 * R_1}{t}$$

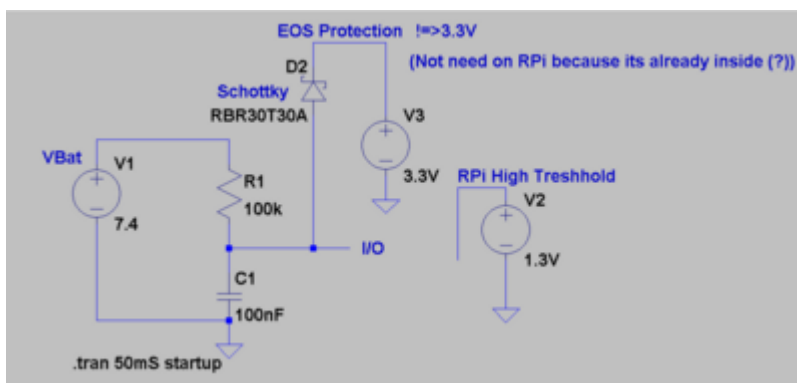
$$V_{C1} \approx 1.3V_{RPI3}$$

$$R_1 \approx 100k$$

$$C_1 \approx 100nF$$

Schaltplan

Die 3.3V Spannungsquelle soll die „Referenzspannung“ der I/Os angeben. Im Falle des RaspberryPis ist das 3.3V. Im Falle eines 5V Arduinos 5V. Diese Spannungsquelle ist die Betriebsspannung des CPUs.



Code - Arduino

[sketch.ino](#)

```
#include <Arduino.h>

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
}

const int rcpin = 2;
const float vih_voltage = 2.85;
const float capacity = 100 * pow(10, -9);
const float resistor = 100000;
const int dischargeDelay = 200;

float measureVoltage(){
    //Pin auf Low setzen zum entladen des Kondensators
    pinMode(2, OUTPUT);
    digitalWrite(rcpin, LOW);
    delay(dischargeDelay); //Warten bis Kondensator leer
    long startMicros = micros(); //Startzeit festlegen
    pinMode(2, INPUT); //Pin auf Input legen um zu detektieren wann Pin
    High
    while(!digitalRead(rcpin)){ //Warte solange Pin nicht High

    }
    long endMicros = micros(); //Neue Zeit speichern
    long finalMicros = endMicros - startMicros; //Dauer des Vorgangs
    berechnen
    //Serial.println(finalMicros);

    //Berechnen der Spannung
    float vbat = (vih_voltage * capacity * resistor)
    /((float)(finalMicros * pow(10, -6)));
    return vbat;
}

void loop() {
    Serial.println(measureVoltage());
}
```

Code - Java RaspberryPi

RC_measure.java

```
package rc_measure;

import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalInput;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.Pin;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
import static java.lang.Math.pow;
import java.util.concurrent.TimeUnit;

/**
 *
 * In einer echten Java Anwendung würde man das nicht so machen, das
 * dient wirklich nur als Beispiel.
 */
public class RC_measure {

    /**
     * @param args the command line arguments
     */
    static Pin rcPin = RaspiPin.GPIO_00;
    static final int r = 100000;
    static final float c = (float) (100 * pow(10, -9));
    /**
     * Vih = 0.625*Vdd
     * http://www.farnell.com/datasheets/1835725.pdf
     */
    static final float vih = (float) 2.976473;
    public static void main(String[] args) throws InterruptedException
    {
        GpioController gpio = GpioFactory.getInstance();
        gpio.unexportAll();
        final GpioPinDigitalOutput outPin =
gpio.provisionDigitalOutputPin(rcPin);
        outPin.setShutdownOptions(true, PinState.LOW);
        outPin.low(); //Entlade Kondensator;

        TimeUnit.SECONDS.sleep(1);
        //gpio.shutdown();
        gpio.unprovisionPin(outPin);

        long startTime = System.nanoTime();
        final GpioPinDigitalInput inPin =
gpio.provisionDigitalInputPin(rcPin);
```

```
while(inPin.isLow()){ //warte
}
long estimatedTime = System.nanoTime() - startTime;
gpio.unprovisionPin(inPin);
System.out.println(estimatedTime);
System.out.println("Voltage: " +
measureRCVoltage(estimatedTime, r, c, vih));
}
static float measureRCVoltage(long t, float r, float c, float
vih_voltage){
float vbat = (vih_voltage * c * r) /((float)(t * pow(10, -9)));
return vbat;
}
}
```

From:
<https://wiki.hackerspace-bremen.de/> - Hackerspace Bremen e.V.

Permanent link:
https://wiki.hackerspace-bremen.de/projekte/hardware_tricks/batterie_messungen/rc_glied_messung

Last update: 2022-11-17 22:34

