

Messen & Steuern mit Raspberry Pi

Bundesseminar für Elektro - Elektrotechnik, 24. - 26.02.2016

“Geschwindigkeitsmessung & Methodenvergleich”

Erstellt von

Ing. Manfred Wallner & Markus Gombocz

Inhaltsverzeichnis:

[Versuchsbeschreibung](#)

[Raspberry Pi](#)

[Geschwindigkeitsmessung](#)

[Materialliste](#)

[Aufgabe 1](#)

[Aufgabe 2](#)

[Aufgabe 3](#)

[Aufgabe 1: Messung mit Lichtschranken](#)

[Schaltbild](#)

[Programmierung \(Pin-polling\)](#)

[Programmierung \(ISR\)](#)

[Implementierung der Geschwindigkeitsberechnung](#)

[Aufgabe 2: Messung mit Ultraschallsensor](#)

[Schaltbild](#)

[Aufgabe 3: Zusammenführen von Aufgabe 1 & 2](#)

[Schaltbild](#)

[Programmierung](#)

[Weiterführende Aufgaben](#)

[Referenzen](#)

[Anhang](#)

[C-Programme auf der RaspberryPi](#)

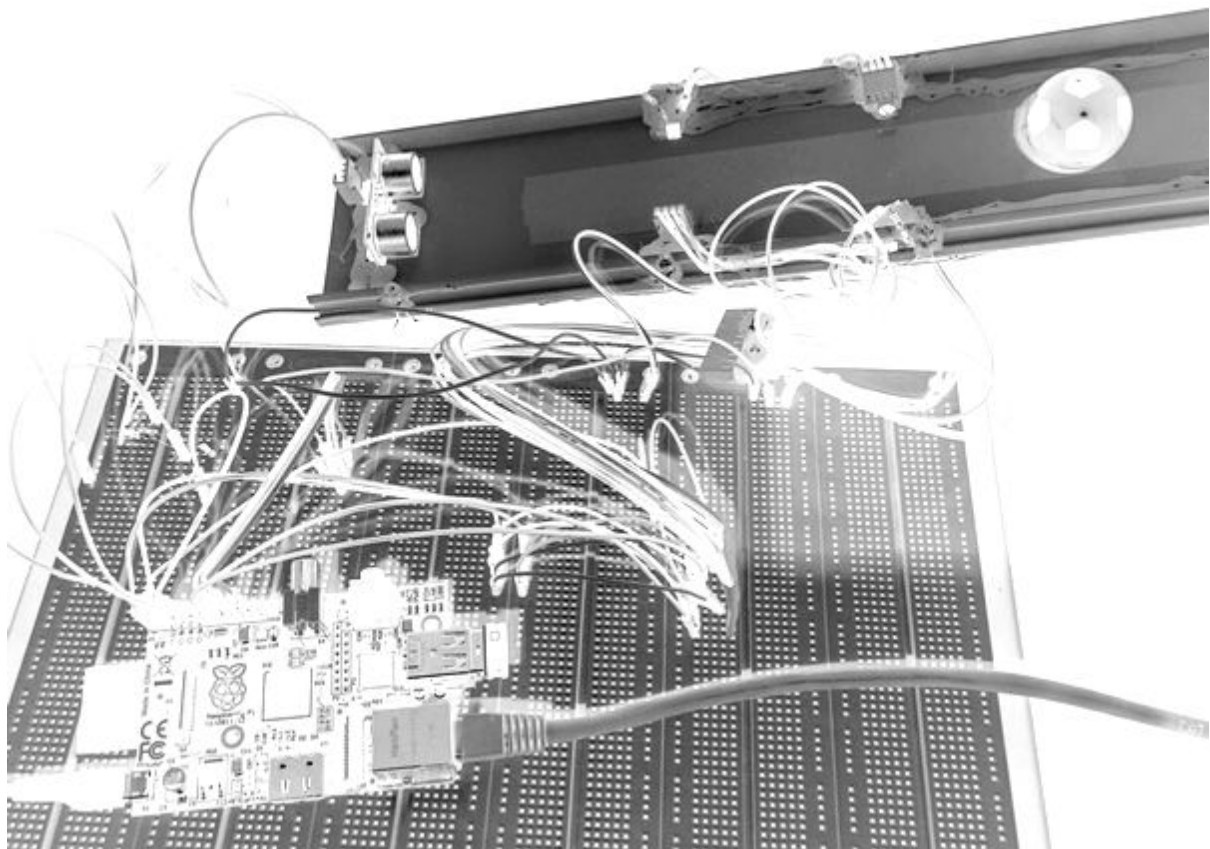
[Lizenz](#)

[Haftungsausschluss](#)

Versuchsbeschreibung

In dieser Anleitung werden mit wenigen elektrischen Bauteilen zwei einfache Möglichkeiten zur Geschwindigkeitsbestimmung von sich bewegenden Objekten erläutert.

Grundverständnis der Elektronik sowie Programmierung in C sind hilfreich bei der Umsetzung der gezeigten Beispiele, jedoch keine Notwendigkeit sofern die Anleitung strikt eingehalten wird.



Der Autor übernimmt keine Haftung für Schäden materieller oder ideeller Natur - die Benutzung dieser Anleitung erfolgt ausdrücklich auf eigenes Risiko.

Raspberry Pi

“Der Raspberry Pi ist ein Einplatinencomputer, der von der britischen Raspberry Pi Foundation entwickelt wurde. Der Rechner enthält ein Ein-Chip-System von Broadcom mit einem ARM-Mikroprozessor, die Grundfläche der Platine entspricht etwa den Abmessungen einer Kreditkarte. Der Raspberry Pi kam Anfang 2012 auf den Markt; sein großer Markterfolg wird teils als Revival des bis dahin weitgehend bedeutungslos gewordenen Heimcomputers zum Programmieren und Experimentieren angesehen. Der im Vergleich zu üblichen Personal Computern sehr einfach aufgebaute Rechner wurde von der Stiftung mit dem Ziel entwickelt, jungen Menschen den Erwerb von Programmier- und Hardwarekenntnissen zu erleichtern. Entsprechend niedrig wurde der Verkaufspreis angesetzt, der je nach Modell etwa 5 bis 35 USD beträgt.” - https://de.wikipedia.org/wiki/Raspberry_Pi am 22.02.2016



Arbeiten auf der Raspberry Pi:

Prinzipiell reicht die Leistung einer Raspberry aus um “direkt” auf dieser zu entwickeln. D.h. sofern ein Monitor mit HDMI-Anschluss sowie USB-Tastatur & Maus verfügbar sind, können diese direkt an die RaspberryPi angeschlossen werden und diese dann wie ein normaler PC verwendet werden.

In dieser Anleitung werden grundlegende Einrichtungsarbeiten, wie das Installieren eines Betriebssystems übersprungen. Sollte tiefergreifende Info dazu notwendig sein, finden Sie weiter hinten eine Referenzenliste.

Wichtige Anmerkungen zum Raspberry Pi:

Die Grundversorgungsspannungen, die beim Raspberry Pi für Schaltungen zur Verfügung steht, beträgt 3,3 V und 5 V. Intern bzw. die GPIO (Generelle Ein- und Ausgänge) arbeitet mit einer Spannung von 3,3 V. Beim verdrahten von Bauteilen ist immer darauf zu achten, das keine Kurzschlüsse passieren und Vorwiderstände verwendet werden. Bitte auf keinen Fall irgendwelche GPIO Pins kurzschließen oder mit der 5 V Versorgung verbinden. Dies kann zu einer Beschädigung bzw. Zerstörung des Raspberry Pi's führen.... so, jetzt kann es losgehen! :-)

Geschwindigkeitsmessung

Wir unterscheiden prinzipiell zwischen drei Arten der Geschwindigkeitsmessung:

1. Zeitmessung einer Wegstrecke

Bei diesem Messverfahren wird die Zeit t gemessen, die ein Objekt für einen bestimmten Weg x benötigt. $v = \frac{x}{t}$

2. Wegmessung in festen Zeitabständen

Wenn die Entfernungen (x_1, x_2, \dots) zu einem Anhaltspunkt in mindestens zwei Zeitpunkten (t_1, t_2, \dots) bekannt sind, errechnet sich die Geschwindigkeit mit $v = \frac{x_2 - x_1}{t_2 - t_1} = \frac{\Delta x}{\Delta t}$.

3. Integration der Beschleunigung

Wenn die Beschleunigung a bekannt ist, kann mittels Integration die Geschwindigkeit

errechnet werden. $v = v_0 + \int_0^t a \, dt$.

Materialliste

Für alle Aufgaben in dieser Anleitung wird zwingend ein RaspberryPi mit Raspbian Image benötigt. Dieses beinhaltet großteils alle Software-Komponenten um die Programme für das Experiment zu erstellen. Zusätzlich wird wiringPi benötigt um auf die GPIO-Schnittstellen des RaspberryPi's einfach zugreifen zu können.

Eine Anleitung zur Installation der Software findet sich auf folgenden Seiten:

- Raspbian-Jessie-Image: <https://www.raspberrypi.org/downloads/raspbian/>
- wiring Pi: <https://github.com/WiringPi/WiringPi>
- Projektsammlung auf GitHub: https://github.com/mwallner/pi_E_samples

Folgende Komponenten sowie Raspberry Pi's, SD-Karten, Stromversorgung und nötiges Experimentier-Equipment (Steckboard, Kabelverbindungen etc.) bekommt man meist bei Elektronik-Händler wie Neuhold-Elektronik Graz, Conrad Elektronik, ...

Aufgabe 1

- 2x IR-Empfänger OPN6042-36-CUT
- 2x IR-Sendediode 1.3 ... 1.7 V ~ 100mA
- 2x 68 Ω Widerstand
- 2x 160 Ω Widerstand
- 2x Standard LED 10-20mA

Aufgabe 2

- 1x HC-SR04 Ultraschallsensor
- 1x 220 Ω Widerstand
- 1x 430 Ω Widerstand

Aufgabe 3

- Kabelkanal (ca 1m)
- Heißkleber
- Matchboxautos

Alle weiteren Aufgaben beziehen sich ausschließlich auf die Programmierung bzw. möglicher Erweiterungen zu den Aufgaben 1 bis 3.

Aufgabe 1: Messung mit Lichtschranken

Die einfachste Art der Geschwindigkeitsmessung ist die so genannte 2-Punkt Messung. Zwei Lichtschranken werden in bekanntem Abstand montiert und die Durchgangszeit gemessen (Annahme: konstante Geschwindigkeit!) - dadurch lässt sich die Geschwindigkeit eines Objektes ableiten:

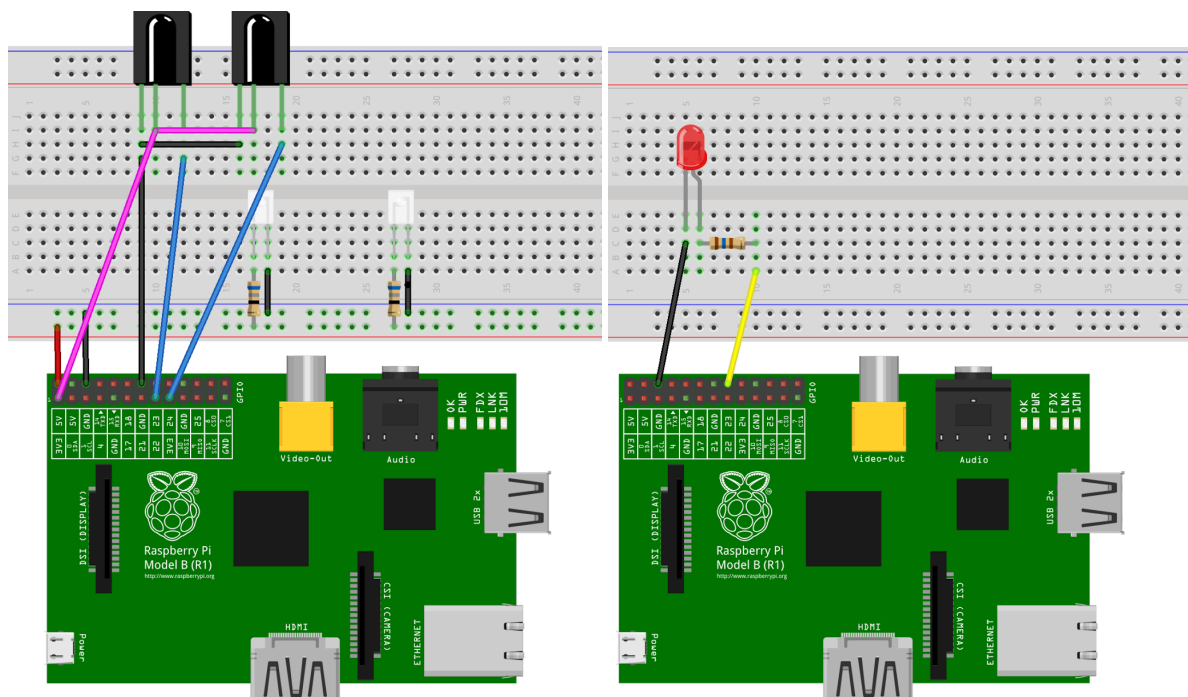
$$v = \frac{\Delta x}{\Delta t} = \frac{\text{distance}}{T2 - T1}$$

Der Zeitpunkt der Unterbrechung der ersten Lichtschranke ist als t_1 , der Zeitpunkt der Unterbrechung der zweiten Lichtschranke als t_2 definiert. Die Distanz zwischen den Lichtschranken muss mit einem Lineal gemessen werden.

Schaltbild

ACHTUNG!

- In diesen Schaltbildern wird ein anderer IR-Receiver (Infrarotempfänger) verwendet!
- Stellen Sie sicher, dass dieser wie im Datenblatt beschrieben verkabelt wird!



Im ersten Schaltbild wird die Verkabelung der Lichtschranken (IR-LED & IR-Receiver) gezeigt, im zweiten Schaltbild die Verkabelung einer einfachen LED. Diese Schaltbilder sollen kombiniert werden - bei entsprechender Programmierung zeigt die LED die Lichtschranken-Durchgänge an.

Programmierung (Pin-polling)

Das erste Programm soll den Status eines GPIO-Pins in einer Programm-Schleife überprüfen und Bildschirmausgaben machen sobald sich der Status dieses Pins ändert. Weiters wird bei Statusänderung eine LED aktiviert.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wiringPi.h> // Include WiringPi library!

int main(int argc, char* argv[]) {
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    int nPinGate = atoi(argv[1]); //first argument is pin of IR-receiver
    int nPinLED = atoi(argv[2]); //second argument is pin of LED to activate

    printf("listening on bcm pin %d, activating LED on pin%d\n", nPinGate, nPinLED);

    pinMode(nPinGate, INPUT);
    pinMode(nPinLED, OUTPUT);

    int nState = 0, nLastState = 0;
    while (true) {
        nState = digitalRead(nPinGate);
        if (nState != nLastState) {
            printf("state changed! -> %d\n", nState);
            digitalWrite(nPinLED, 1);
            usleep(1000000);
            digitalWrite(nPinLED, 0);
            nLastState = digitalRead(nPinGate);
        }
        usleep(10);
    }
    return 0;
}
```

Um diesen Sourcecode auszuführen muss er zuerst kompiliert (in ein ausführbares Programm umgewandelt) werden. (Details siehe *Anhang/C-Programme auf der RaspberryPi*)

Zum Beispiel:

```
pi@raspberrypi:~/workspace $ gcc -Wall -o gate gate.cpp -lwiringPi
pi@raspberrypi:~/workspace $ sudo ./gate 27 22
listening on bcm pin 27, activating LED on pin22
state changed! -> 1
state changed! -> 0
^Cpi@raspberrypi:~/workspace $
```


Programmierung (ISR)

Da die vorige Pin-polling Methode eine sehr aufwändige und zudem recht ungenaue Methode ist um Statusänderungen zu verfolgen, wird in diesem Beispiel eine Interrupt(Unterbrecher)-Service-Routine (kurz ISR) verwendet. Sobald ein Pin aktiv wird, unterbricht der Prozessor seine übliche Ausführung und springt in eine vordefinierte Funktion (Beispiel ISR_GATE)

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>
#include <wiringPi.h> // Include WiringPi library!

int nPinGate;
int nPinLED;

void ISR_GATE(void) {
    digitalWrite(nPinLED, 1);
    usleep(1000000);
    digitalWrite(nPinLED, 0);
}

int main(int argc, char* argv[]) {
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    nPinGate = atoi(argv[1]); //first argument is pin of IR-receiver
    nPinLED = atoi(argv[2]); //second argument is pin of LED to activate

    printf("listening on bcm pin %d, activating LED on pin%d\n", nPinGate, nPinLED);

    pinMode(nPinGate, INPUT);
    pinMode(nPinLED, OUTPUT);

    if (wiringPiISR(nPinGate, INT_EDGE_BOTH, &ISR_GATE) < 0) {
        printf("Unable to setup ISR for GPIO %d (%s)\n\n", nPinGate, strerror(errno));
        exit(1);
    }

    while (true) {
        usleep(100);
    }
    return 0;
}
```

Anmerkung: diese Methode ist deswegen zu bevorzugen, da Statusänderungen von Pins sofort registriert werden. Bei der Polling-Methode wird eine Schleife immer wieder durchlaufen, was Rechenleistung und vor allem Zeit kostet.

Implementierung der Geschwindigkeitsberechnung

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>

#define BCM_FIRST_GATE 27
#define LED_FIRST_GATE 25
#define BCM_SECOND_GATE 22
#define LED_SECOND_GATE 24
#define GATE_DISTANCE 75.0 /*mm*/

struct timeval starttime, endtime;
int _ACTIVE_MEASUREMENT_ = 0;

void calcSpeed() {
    double secs_elapsed, speed;
    secs_elapsed = (endtime.tv_sec - starttime.tv_sec) + (endtime.tv_usec -
starttime.tv_usec)/1000000.0;
    speed = GATE_DISTANCE/secs_elapsed;
    printf("\n");
    printf("distance: %lf mm\n", GATE_DISTANCE);
    printf("seconds elapsed: %lf sec\n", secs_elapsed);
    printf("speed: %lf mm/sec\n", speed);
    printf("\n");
}

void reset() {
    usleep(200000);
    digitalWrite(LED_FIRST_GATE, 0);
    digitalWrite(LED_SECOND_GATE, 0);
    _ACTIVE_MEASUREMENT_ = 0;
}

void FIRST_GATE_PASS(void) {
    if (_ACTIVE_MEASUREMENT_) {
        return;
    }
    _ACTIVE_MEASUREMENT_ = 1;
    digitalWrite(LED_FIRST_GATE, 1);
    struct timeval now;
    gettimeofday(&now, NULL);
    starttime = now;
    printf("FIRST_GATE_PASS\n");
}

void SECOND_GATE_PASS(void) {
    if (!_ACTIVE_MEASUREMENT_) {
        return;
    }
    digitalWrite(LED_SECOND_GATE, 1);
    struct timeval now;
    gettimeofday(&now, NULL);
    endtime = now;
    printf("SECOND_GATE_PASS\n");
    calcSpeed();
    reset();
}
```

```

int main(int argc, char* argv[]) {
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    pinMode(BCM_FIRST_GATE, INPUT);
    pinMode(LED_FIRST_GATE, OUTPUT);
    pinMode(BCM_SECOND_GATE, INPUT);
    pinMode(LED_SECOND_GATE, OUTPUT);

    if (wiringPiISR(BCM_FIRST_GATE, INT_EDGE_BOTH, &FIRST_GATE_PASS) < 0) {
        printf("Unable to setup ISR for GPIO %d (%s)\n\n", BCM_FIRST_GATE,
strerror(errno));
        exit(1);
    }
    if (wiringPiISR(BCM_SECOND_GATE, INT_EDGE_BOTH, &SECOND_GATE_PASS) < 0) {
        printf("Unable to setup ISR for GPIO %d (%s)\n\n", BCM_SECOND_GATE,
strerror(errno));
        exit(1);
    }

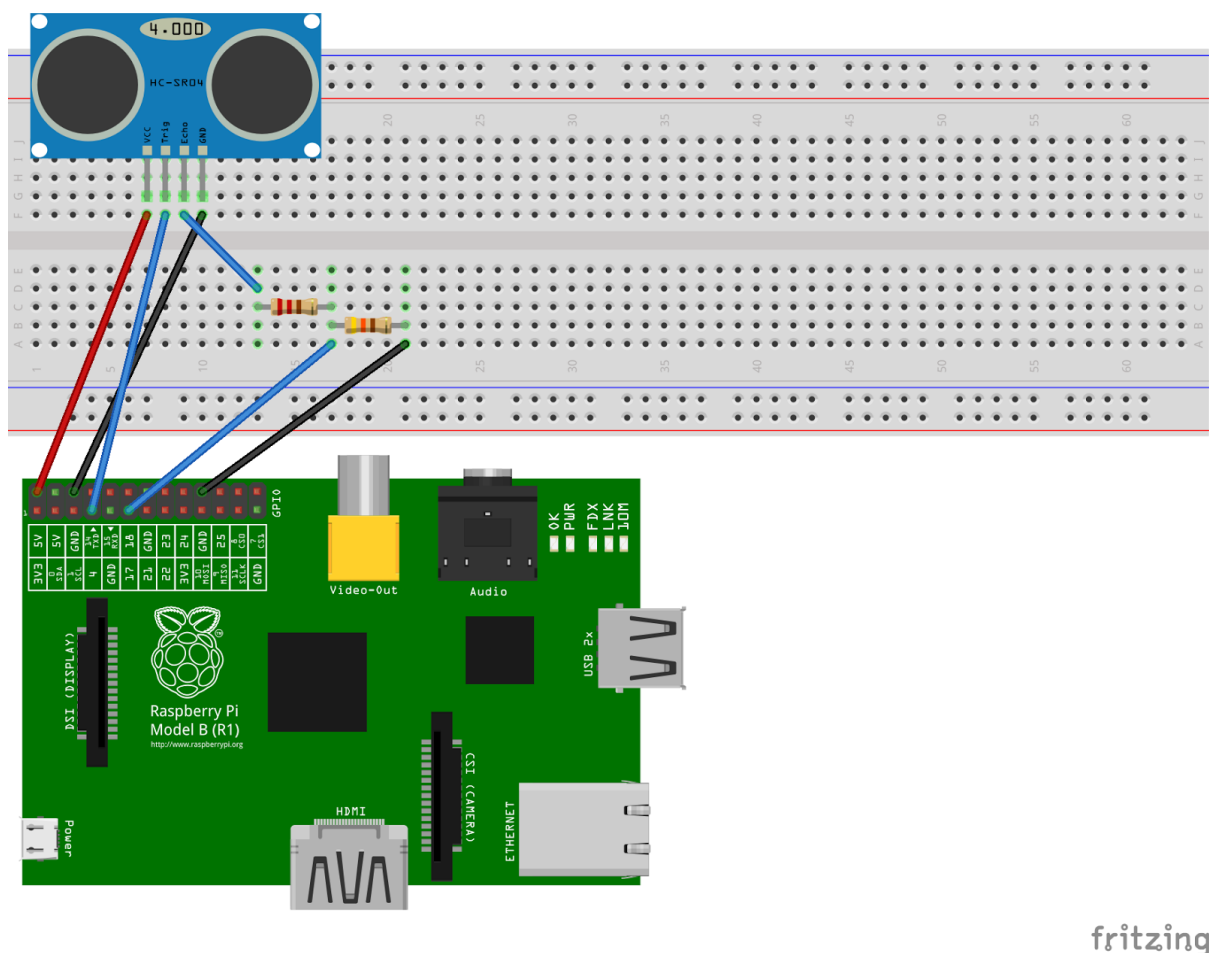
    while (true) {
        usleep(100);
    }
    return 0;
}

```

Aufgabe 2: Messung mit Ultraschallsensor

Der Ultraschallsensor ist sehr empfindlich. Auch kleine Objekte werden als Hindernisse registriert, deshalb ist es sauberer, wenn sich keine Objekte wie Leds, Sensoren, Kabel..etc. in der Messbahn befinden. Sollte dies nicht möglich sein, muss die Messung so ausgeführt werden, dass nur im freien Bereich gemessen wird. Der Sensor benötigt ebenso einen Mindestabstand von 2,5 cm. Die Messung muss also bei $< 2,6$ cm gestoppt werden. Der Sensor arbeitet mit 5 V, deshalb muss für den GPIO Eingang eine Pegelanpassung über einen Spannungsteiler durchgeführt werden.

Schaltbild



Programmierung

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>

#define BCM_ULTRASONIC_TRIGGER 17
#define BCM_ULTRASONIC_ECHO 4

const double SPEED_SOUND = 343.0; // m/s

int us_nCounterEdgeDown = 0;
int us_nCounterEdgeUp = 0;
struct timeval us_starttime, us_endtime;

void ISR_ECHO(void) {
    struct timeval now;
    int nInputState;

    gettimeofday(&now, NULL);
    nInputState = digitalRead(BCM_ULTRASONIC_ECHO);
    if (HIGH == nInputState) {
        us_starttime = now;
        us_nCounterEdgeUp++;
    } else {
        us_endtime = now;
        us_nCounterEdgeDown++;
    }
}

double getDistanceMM() {
    digitalWrite(BCM_ULTRASONIC_TRIGGER, HIGH);
    usleep(5000);
    digitalWrite(BCM_ULTRASONIC_TRIGGER, LOW);
    usleep(20000);

    double secs_elapsed = (us_endtime.tv_sec - us_starttime.tv_sec) + (us_endtime.tv_usec -
us_starttime.tv_usec) / 1000000.0;
    double distance = SPEED_SOUND * 100 * 10 * secs_elapsed / 2; //mm
    printf("distance: %.1f mm (time elapsed: %g s)\n", distance, secs_elapsed);
    return distance;
}

int main(int argc, char* argv[]) {
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

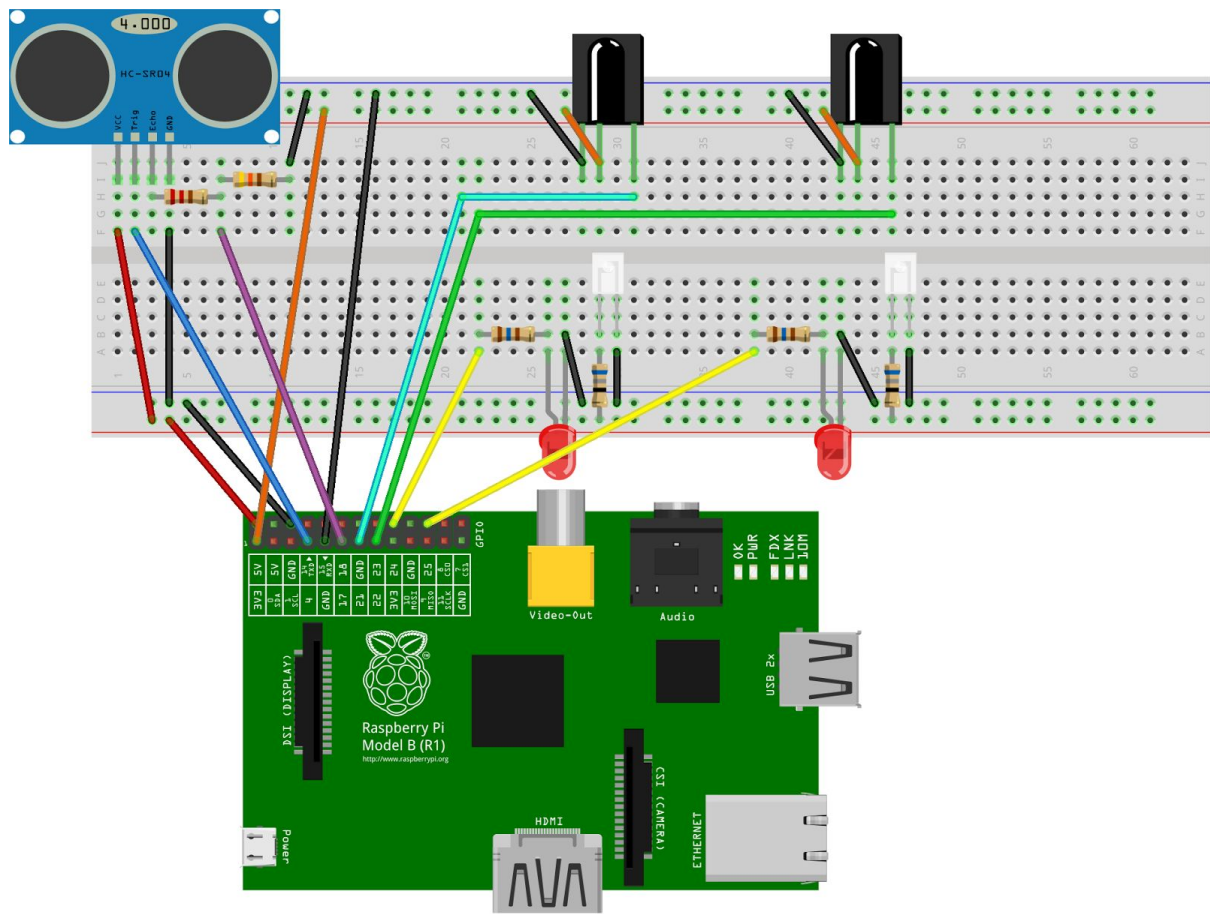
    pinMode(BCM_ULTRASONIC_ECHO, INPUT);
    pinMode(BCM_ULTRASONIC_TRIGGER, OUTPUT);

    if (wiringPiISR(BCM_ULTRASONIC_ECHO, INT_EDGE_BOTH, &ISR_ECHO) < 0) {
        printf("Unable to setup ISR (ULTRASONIC) for GPIO %d (%s)\n\n", BCM_ULTRASONIC_ECHO,
strerror(errno));
        exit(1);
    }
    while (true) {
        usleep(100000);
        getDistanceMM();
    }
    return 0;
}
```

Aufgabe 3: Zusammenführen von Aufgabe 1 & 2

Um die Lichtschranken bzw. Ultraschall-Messmethode automatisiert zu vergleichen, müssen die Aufbauten aus Aufgabe 1 und Aufgabe 2 kombiniert werden. Aufgrund von akutem Platzmangel auf Übungs-Steckboards wird empfohlen die Lichtschranken mit Heißkleber in einem Kabelkanal zu befestigen. Dieser Kabelkanal dient als "Teststrecke" in der die Geschwindigkeit von rutschenden & rollenden Objekten vermessen wird. Der Ultraschallsensor kann entweder wie eine Radarpistole am Ende, oder am Anfang des Kanals befestigt werden. Zusätzlich zu den Sensoren können "Status-LEDs" im Kanal befestigt werden - diese ermöglichen ein visuelles Feedback zur Sensoraktivität.

Schaltbild



fritzing

Programmierung

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#include <sys/time.h>
#include <unistd.h>

#define BCM_FIRST_GATE 27
#define LED_FIRST_GATE 25
#define BCM_SECOND_GATE 22
#define LED_SECOND_GATE 24

#define BCM_ULTRASONIC_TRIGGER 17
#define BCM_ULTRASONIC_ECHO 4

const double SPEED_SOUND = 343.0; // m/s
#define GATE_DISTANCE 75.0 /*mm*/

int _ACTIVE_MEASUREMENT_ = 0;

////////// ULTRASONIC_SENSOR //////////
int us_nCounterEdgeDown = 0;
int us_nCounterEdgeUp = 0;
struct timeval us_starttime, us_endtime;

void ISR_ECHO(void) {
    if (!_ACTIVE_MEASUREMENT_) {
        return;
    }
    struct timeval now;
    int nInputState;

    gettimeofday(&now, NULL);
    nInputState = digitalRead(BCM_ULTRASONIC_ECHO);
    if (HIGH == nInputState) {
        us_starttime = now;
        us_nCounterEdgeUp++;
    } else {
        us_endtime = now;
        us_nCounterEdgeDown++;
    }
}

double _getDistanceMM(double *pDTimeWastedMSec) {
    digitalWrite(BCM_ULTRASONIC_TRIGGER, HIGH);
    usleep(5000);
    digitalWrite(BCM_ULTRASONIC_TRIGGER, LOW);
    usleep(20000);
    if (pDTimeWastedMSec)
        *pDTimeWastedMSec += 25;

    double secs_elapsed = (us_endtime.tv_sec - us_starttime.tv_sec) + (us_endtime.tv_usec -
us_starttime.tv_usec) / 1000000.0;
    double distance = SPEED_SOUND * 100 * 10 * secs_elapsed / 2; //mm
    if (distance < 30) {
        printf("sensor returned invalid distance!\n");
        return 0;
    }
    printf("distance: %.1f mm (time elapsed: %g s)\n", distance, secs_elapsed);
    return distance;
}
```

```

double getUltrasonicSpeed() {
    double dMea1 = _getDistanceMM(0);
    double dTimeBetweenMeasurementMSec = 10;
    usleep(dTimeBetweenMeasurementMSec*1000);
    double dTimeWastedMSec = 0.0;
    double dMea2 = _getDistanceMM(&dTimeWastedMSec);
    double dDistance = dMea1 - dMea2;
    if (dDistance < 0) {
        printf("invalid measurement from ultrasonic sensor!\n");
        return 0.0;
    }
    double dTimeTotal = dTimeWastedMSec/1000.0 + dTimeBetweenMeasurementMSec/1000.0;
    printf("us_distance: %lf mm\n", dDistance);
    printf("us_seconds elapsed: %lf sec\n", dTimeTotal);
    double dSpeed = dDistance/dTimeTotal;
    printf("us_speed: %lf mm/sec\n", dSpeed);

    return dSpeed;
}

////////// IR-GATES //////////
struct timeval starttime, endtime;

double getGateSpeed() {
    double secs_elapsed, speed;
    secs_elapsed = (endtime.tv_sec - starttime.tv_sec) + (endtime.tv_usec -
starttime.tv_usec)/1000000.0;
    speed = GATE_DISTANCE/secs_elapsed;
    printf("gate_distance: %lf mm\n", GATE_DISTANCE);
    printf("gate_seconds elapsed: %lf sec\n", secs_elapsed);
    printf("gate_speed: %lf mm/sec\n", speed);
    return speed;
}

void calcSpeed() {
    printf("\n");
    double dGateSpeed = getGateSpeed();
    printf("\n");
    double dUSSpeed = getUltrasonicSpeed();
    printf("\n");
    printf("diff: %lf\n", dUSSpeed-dGateSpeed);
    printf("\n");
}

void reset() {
    usleep(200000);
    digitalWrite(LED_FIRST_GATE, 0);
    digitalWrite(LED_SECOND_GATE, 0);
    _ACTIVE_MEASUREMENT_ = 0;
}

void FIRST_GATE_PASS(void) {
    if (_ACTIVE_MEASUREMENT_) {
        return;
    }
    _ACTIVE_MEASUREMENT_ = 1;
    digitalWrite(LED_FIRST_GATE, 1);
    struct timeval now;
    gettimeofday(&now, NULL);
    starttime = now;
    printf("FIRST_GATE_PASS\n");
}

void SECOND_GATE_PASS(void) {
    if (!_ACTIVE_MEASUREMENT_) {
        return;
    }
    digitalWrite(LED_SECOND_GATE, 1);
    struct timeval now;
    gettimeofday(&now, NULL);
    endtime = now;
    printf("SECOND_GATE_PASS\n");
    calcSpeed();
    reset();
}

```



```

int main(int argc, char* argv[]) {
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    pinMode(BCM_FIRST_GATE, INPUT);
    pinMode(LED_FIRST_GATE, OUTPUT);
    pinMode(BCM_SECOND_GATE, INPUT);
    pinMode(LED_SECOND_GATE, OUTPUT);
    pinMode(BCM_ULTRASONIC_ECHO, INPUT);
    pinMode(BCM_ULTRASONIC_TRIGGER, OUTPUT);

    if (wiringPiISR(BCM_FIRST_GATE, INT_EDGE_BOTH, &FIRST_GATE_PASS) < 0) {
        printf("Unable to setup ISR (FIRST_GATE) for GPIO %d (%s)\n\n", BCM_FIRST_GATE,
strerror(errno));
        exit(1);
    }
    if (wiringPiISR(BCM_SECOND_GATE, INT_EDGE_BOTH, &SECOND_GATE_PASS) < 0) {
        printf("Unable to setup ISR (SECOND_GATE) for GPIO %d (%s)\n\n", BCM_SECOND_GATE,
strerror(errno));
        exit(1);
    }
    if (wiringPiISR(BCM_ULTRASONIC_ECHO, INT_EDGE_BOTH, &ISR_ECHO) < 0) {
        printf("Unable to setup ISR (ULTRASONIC) for GPIO %d (%s)\n\n", BCM_ULTRASONIC_ECHO,
strerror(errno));
        exit(1);
    }

    while (true) {
        usleep(100);
    }
    return 0;
}

```

Weiterführende Aufgaben

Die in dieser Anleitung erstellten Programme sind sehr einfach gehalten. Folgende Aufgaben wären sinnvolle Erweiterungen die relativ einfach umgesetzt werden können:

1. Messergebnisse (live oder aus CSV) sollen auf Plausibilität überprüft werden.
2. Messergebnisse sollen im CSV (comma-seperated-values)-Format gespeichert werden (benutze Standard FILE IO <http://www.cplusplus.com/reference/cstdio/FILE/>).
3. Es soll permanent, bzw. nach jeder Messung, die 'schnellste', 'langsamste' und durchschnittliche Geschwindigkeit aller Messungen ausgegeben werden.
4. Es kann eine 'automatische Kalibrierung' des Abstands der Lichtschranken mittels Ultraschallsensor implementiert werden.

Referenzen

Raspberry Pi: <https://www.raspberrypi.org/>

Raspbian: <https://www.raspbian.org/>

wiringPi: <http://wiringpi.com/>

fritzing: <http://fritzing.org/home/>

C Grundlagen: https://de.wikibooks.org/wiki/C-Programmierung:_Grundlagen

C/C++ Referenz: <http://www.cplusplus.com/>

Alle Beispielprogramme, Schaltbilder und vieles mehr sind auch im Netz verfügbar:

https://github.com/mwallner/pi_E_samples

Anhang

C-Programme auf der RaspberryPi

C-Programme müssen, nachdem diese geschrieben wurden, in Maschinencode übersetzt werden. Dieser Vorgang wird *kompilieren* genannt. Zusätzlich verwenden Programme fast immer Bestandteile von anderen Programmen - um diese zu verwenden muss ein Programm zusätzlich gegen dieses Programm oder Bibliothek *gelenkt* werden.

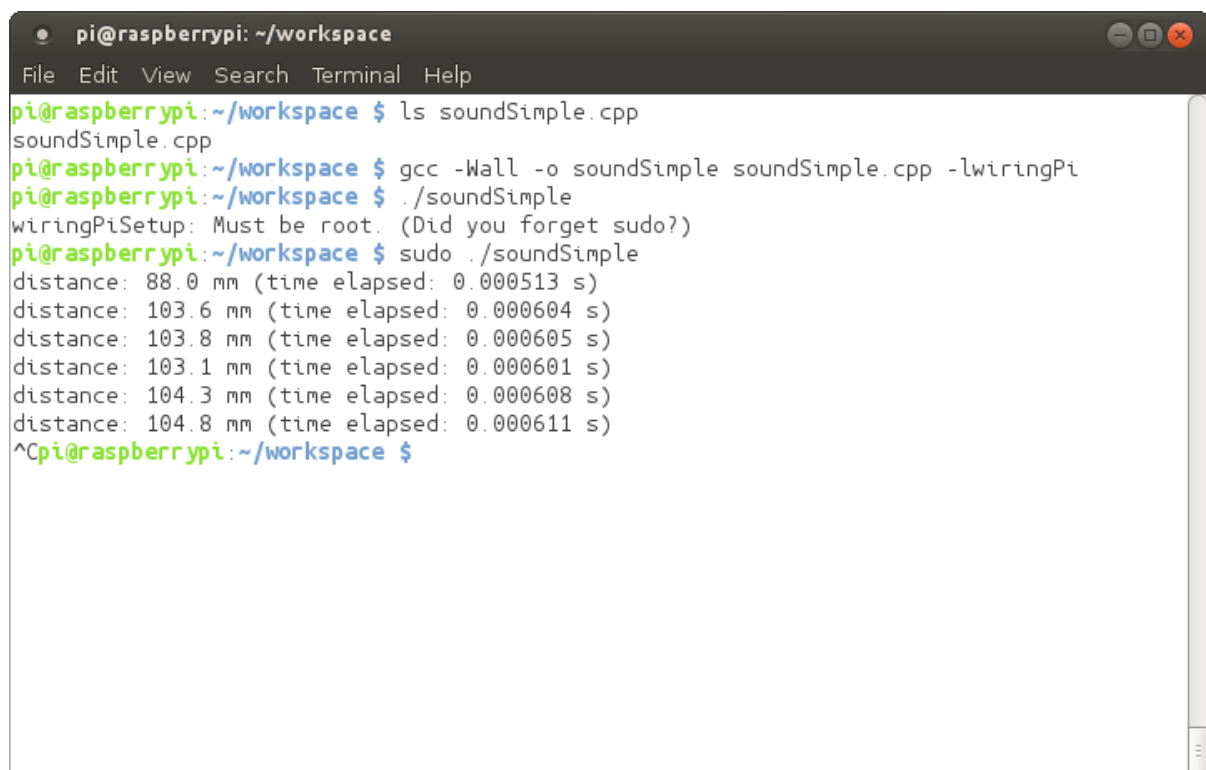
Prinzipiell wird empfohlen ein Programm wie folgt zu erstellen:

Die c/cpp-Datei muss kompiliert werden (wir verwenden den GNU-C-Compiler gcc)

```
"gcc -Wall -o output_datei input_datei -lwiringPi"
```

Die Parameter haben folgende Bedeutung:

<code>gcc</code>	- ruft den Compiler auf
<code>-Wall</code>	- es werden alle Warnungen & Fehler angezeigt
<code>output_datei</code>	- der Name des erstellten Programms
<code>input_datei</code>	- die Quelldatei (beinhaltet den Sourcecode)
<code>-lwiringPi</code>	- diese Anweisung teilt dem Compiler mit die Anwendung gegen die wiringPi-Bibliothek zu linken.



```
pi@raspberrypi: ~/workspace
File Edit View Search Terminal Help
pi@raspberrypi:~/workspace $ ls soundSimple.cpp
soundSimple.cpp
pi@raspberrypi:~/workspace $ gcc -Wall -o soundSimple soundSimple.cpp -lwiringPi
pi@raspberrypi:~/workspace $ ./soundSimple
wiringPiSetup: Must be root. (Did you forget sudo?)
pi@raspberrypi:~/workspace $ sudo ./soundSimple
distance: 88.0 mm (time elapsed: 0.000513 s)
distance: 103.6 mm (time elapsed: 0.000604 s)
distance: 103.8 mm (time elapsed: 0.000605 s)
distance: 103.1 mm (time elapsed: 0.000601 s)
distance: 104.3 mm (time elapsed: 0.000608 s)
distance: 104.8 mm (time elapsed: 0.000611 s)
^Cpi@raspberrypi:~/workspace $
```

* Um ein Programm auszuführen, das direkt mit der Hardware (GPIO-Pins) interagiert, sind oft Administrator-Rechte ("root") notwendig. Es muss das Kommando `sudo` vorangestellt werden um Programme mit Administrator-Rechte zu starten.

Lizenz



Sie dürfen:

- **Teilen** — das Material in jedwedem Format oder Medium vervielfältigen und weiterverbreiten
- **Bearbeiten** — das Material remixen, verändern und darauf aufbauen
- Der Lizenzgeber kann diese Freiheiten nicht widerrufen solange Sie sich an die Lizenzbedingungen halten.

Unter folgenden Bedingungen:

- **Namensnennung** — Sie müssen angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Diese Angaben dürfen in jeder angemessenen Art und Weise gemacht werden, allerdings nicht so, dass der Eindruck entsteht, der Lizenzgeber unterstütze gerade Sie oder Ihre Nutzung besonders.
- **Nicht kommerziell** — Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.
- **Weitergabe unter gleichen Bedingungen** — Wenn Sie das Material remixen, verändern oder anderweitig direkt darauf aufbauen, dürfen Sie Ihre Beiträge nur unter derselben Lizenz wie das Original verbreiten.
- **Keine weiteren Einschränkungen** — Sie dürfen keine zusätzlichen Klauseln oder technische Verfahren einsetzen, die anderen rechtlich irgendetwas untersagen, was die Lizenz erlaubt.

Haftungsausschluss

Die Benutzung dieser Anleitung und die Umsetzung der darin enthaltenen Informationen erfolgt ausdrücklich auf eigenes Risiko. Haftungsansprüche gegen den Autor für Schäden materieller oder ideeller Art, die durch die Nutzung oder Nichtnutzung der Informationen bzw. durch die Nutzung fehlerhafter und/oder unvollständiger Informationen verursacht wurden, sind grundsätzlich ausgeschlossen. Rechts- und Schadensersatzansprüche sind daher ausgeschlossen. Das Werk inklusive aller Inhalte wurde unter größter Sorgfalt erarbeitet. Der Autor übernimmt jedoch keine Gewähr für die Aktualität, Korrektheit, Vollständigkeit und Qualität der bereitgestellten Informationen. Druckfehler und Falschinformationen können nicht vollständig ausgeschlossen werden.