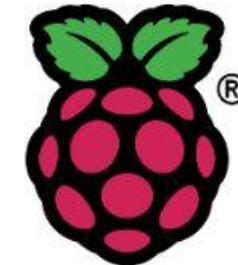




IIT

Industrielle Informationstechnologien
4. / 5. Jahrgang

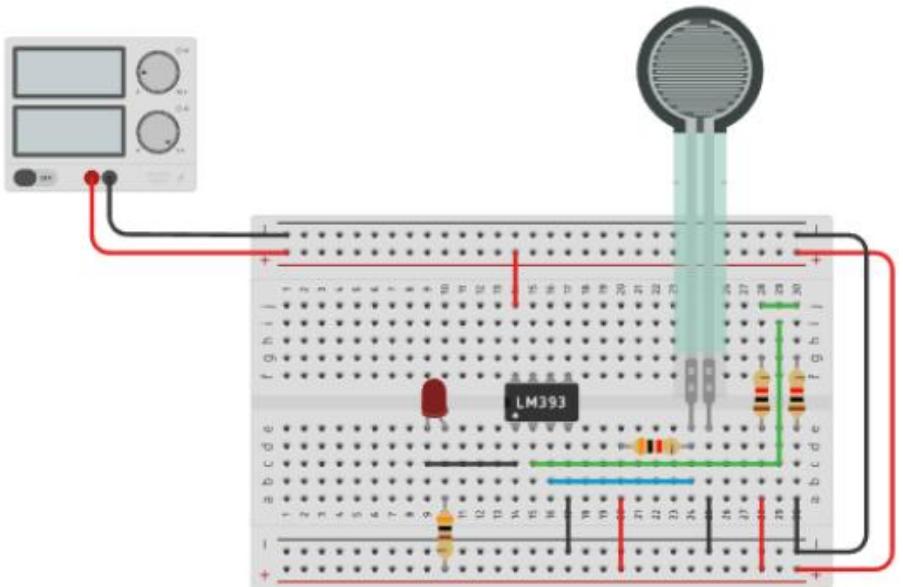


Inhalte

- » Mikrocontroller- und Single Board Systeme
 - grundlegender Aufbau und Funktionsweise am Beispiel Arduino & Raspberry Pi
- » Prozessdatenverarbeitung & Visualisierung
 - Anbindung von Sensoren, Aktuatoren, Displays, ...
- » Kommunikation und industrielle Bussysteme
 - Anwendung gängiger Schnittstellen (UART, I²C, SPI, ...)
 - Funktionsweise industrieller Bussysteme (CAN-Bus, RS485)
- » Embedded Systems & industrielle Programmiermethoden
 - Theoretische Grundlagen und hardwarenahe Programmierung / Registermanipulation

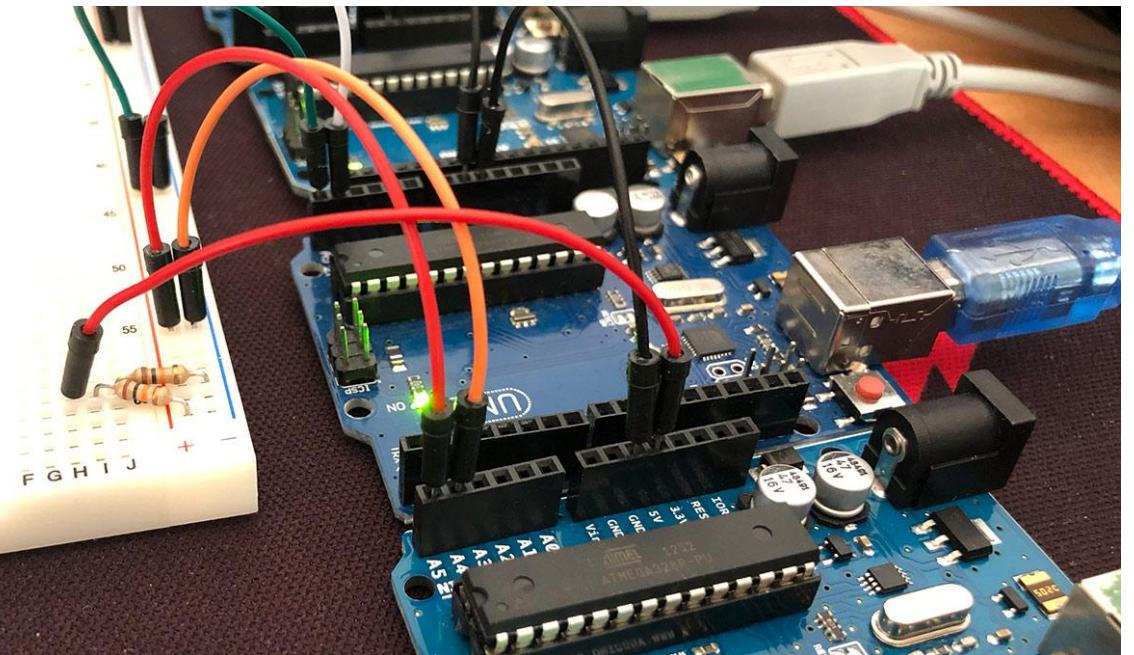
Inhalte

» Umsetzung von Hard- & Software



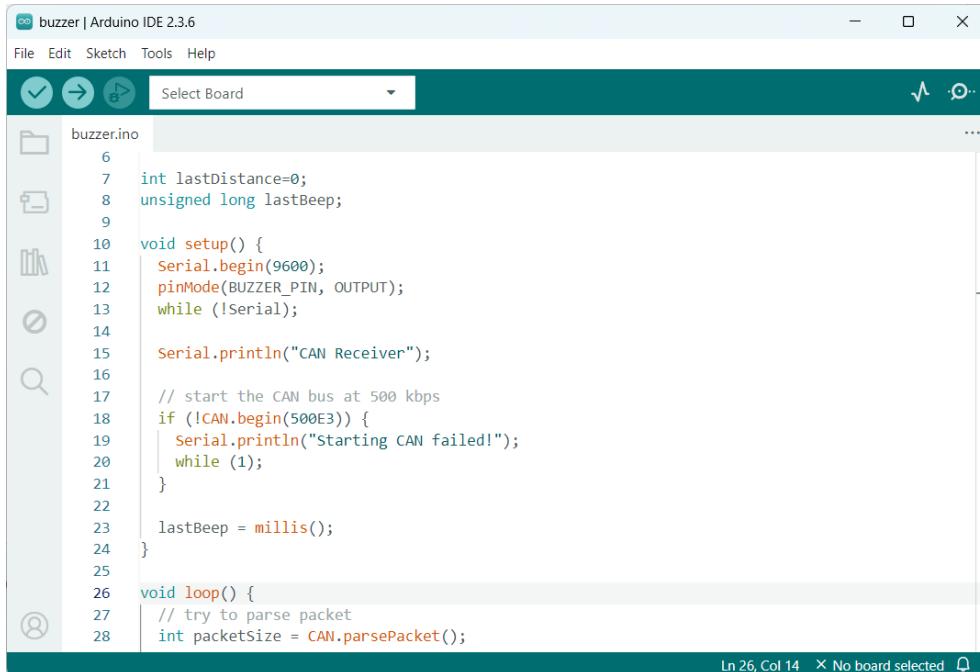
Schaltungsaufbau im Emulator

Umsetzung von Schaltungen im Labor



Inhalte

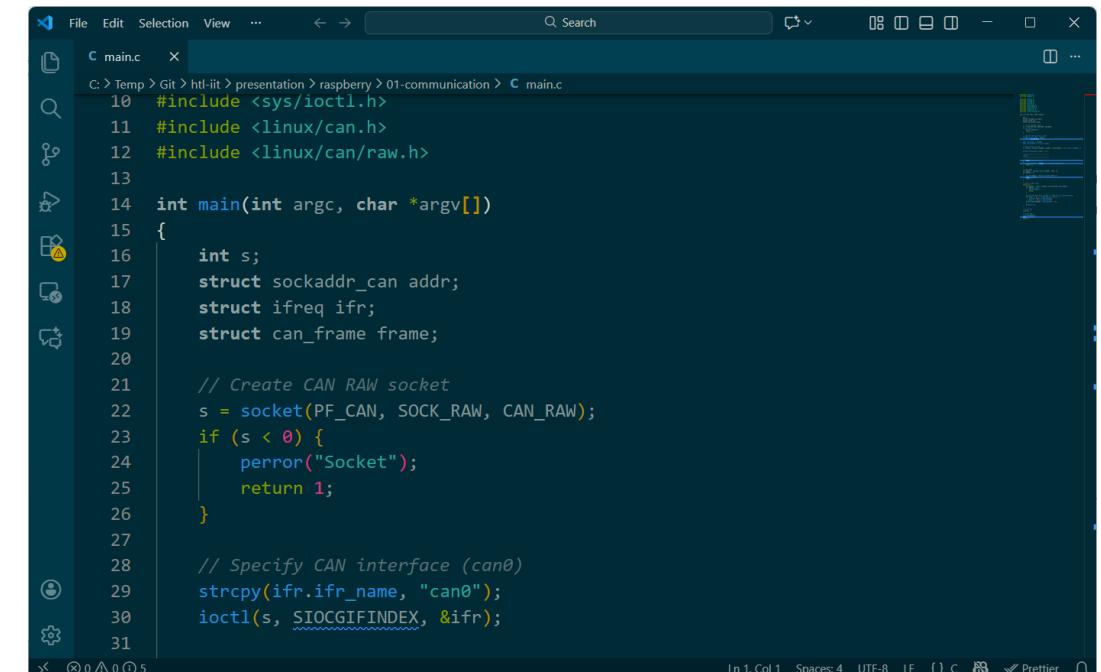
» Umsetzung von Hard- & Software



The screenshot shows the Arduino IDE interface with the file "buzzer.ino" open. The code implements a CAN receiver using the ICAN library. It initializes the CAN bus at 500 kbps and prints "CAN Receiver" to the serial monitor. It then enters a loop where it tries to parse a packet from the CAN bus.

```
File Edit Sketch Tools Help
Select Board
buzzer.ino
6
7 int lastDistance=0;
8 unsigned long lastBeep;
9
10 void setup() {
11   Serial.begin(9600);
12   pinMode(BUZZER_PIN, OUTPUT);
13   while (!Serial);
14
15   Serial.println("CAN Receiver");
16
17 // start the CAN bus at 500 kbps
18 if (!ICAN.begin(500E3)) {
19   Serial.println("Starting CAN failed!");
20   while (1);
21 }
22
23 lastBeep = millis();
24 }
25
26 void loop() {
27   // try to parse packet
28   int packetSize = CAN.parsePacket();
```

Ln 26, Col 14 No board selected



The screenshot shows a terminal window with the file "main.c" open. The code uses raw sockets to interact with a CAN interface named "can0". It creates a socket, specifies the interface, and then performs an ioctl operation to set up the interface.

```
C main.c
C: > Temp > Git > htlt-it > presentation > raspberry > 01-communication > C main.c
10 #include <sys/ioctl.h>
11 #include <linux/can.h>
12 #include <linux/can/raw.h>
13
14 int main(int argc, char *argv[])
15 {
16   int s;
17   struct sockaddr_can addr;
18   struct ifreq ifr;
19   struct can_frame frame;
20
21   // Create CAN RAW socket
22   s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
23   if (s < 0) {
24     perror("Socket");
25     return 1;
26   }
27
28   // Specify CAN interface (can0)
29   strcpy(ifr.ifr_name, "can0");
30   ioctl(s, SIOCGIFINDEX, &ifr);
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF { } C ↻ Prettier

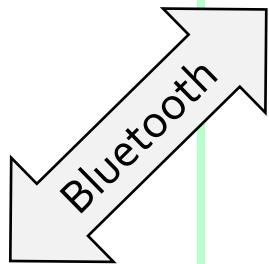
Implementierung in modernen Entwicklungsumgebungen

Versuchsaufbau



Touch UI (Android)

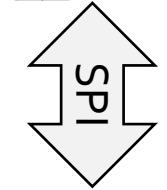
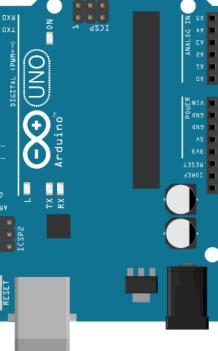
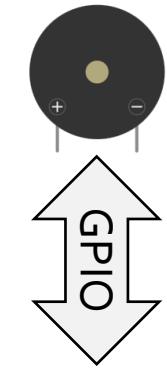
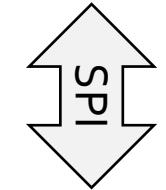
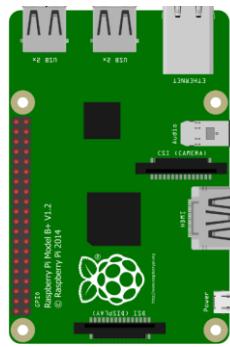
Android Jetpack
Compose Entwicklung
(SEW4./5. JG)



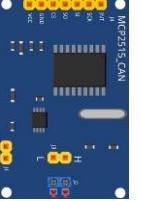
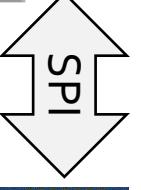
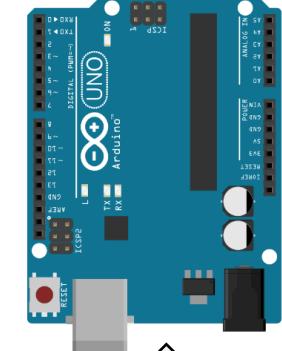
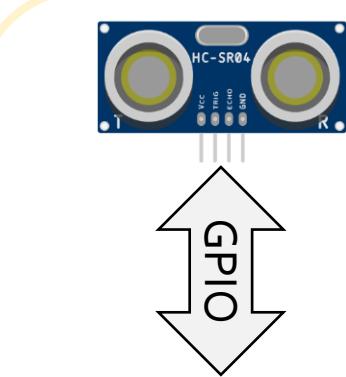
BT-Modul



zentrales Steuergerät (Raspberry Pi)



Audio Modul (Arduino)



MCP2515

Abstandssensor (Arduino)