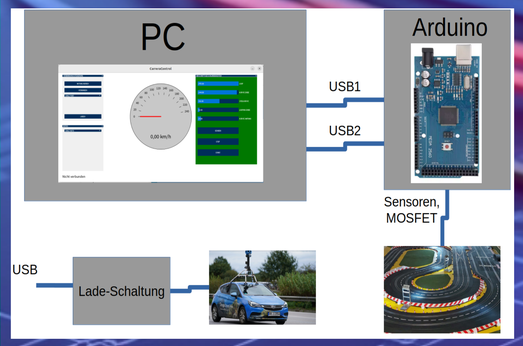
CarreraBot:
Der autonome Rennfahrer

Rafael Heinitz (11), Lukas Herb(11)

Elektronik



Gesamtsystem

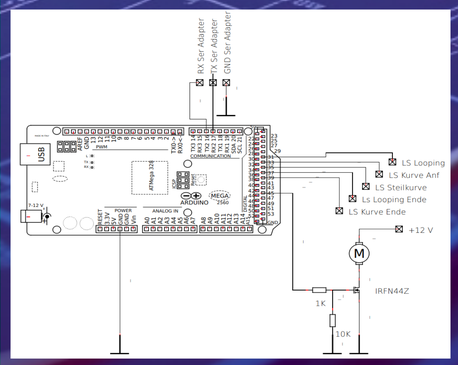
Das Gesamtsystem besteht aus einem Arduino mit entsprechender Steuerungssoftware, PC mit Processing Program, Carrera-Bahn mit Sensoren und Motorsteuerung und einem Kamera-Auto.

-Die Einstellungen werden über Processing an den Arduino übertragen.

-Sensoren entlang der Straße helfen dabei, die Position des Autos zu bestimmen.

-Die Geschwindigkeit des Fahrzeugs wird über einen Power-MOSFET geregelt.

-Ein Kamerawagen ist unabhängig vom restlichen System und verfügt über einen eigenen LiPo-Akku zur Stromversorgung der Kamera.

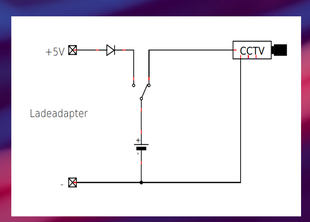


Arduino

Es wird ein Arduino Mega eingesetzt, da er zahlreiche Anschlüsse besitzt. Wir hatten ursprünglich geplant, deutlich mehr Sensoren anzuschließen. Außerdem verfügt der Mega über mehrere echte serielle Schnittstellen. Zwei davon verwenden wir: eine zum Programmieren und eine für die Kommunikation zwischen Arduino und dem Processing-Programm.

Ein Power-N-Kanal-MOSFET vom Typ IRFZ44N steuert mittels PWM die Leistung des Motors. Der 10k-Widerstand dient als Pull-Down-Widerstand und sorgt dafür, dass der Transistor ohne Signal vom Arduino nicht schaltet. Der 1k-Widerstand schützt lediglich den Ausgang des Arduinos.

Als Lichtschranken verwenden wir fertige Reflexionslichtschranken für Arduino von Amazon. In QElectroTech gab es keine Schaltsymbole für diese Bauelemente, daher haben wir nur die Anschlüsse gezeichnet.



Kamera-Auto

Ein Auto haben wir zum Kamera-Auto umgebaut. Es ist mit einer FPV-Kamera ausgestattet, die über einen LiPo-Akku mit Strom versorgt wird. Der Akku hält jedoch nicht lange, daher haben wir eine Schaltung gebaut, um ihn direkt im Auto laden zu können.

Über einen Umschalter wird der Akku entweder vom Ladeadapter geladen oder er versorgt die Kamera. Die Ladespannung beträgt 5 V – das ist zu hoch für die Kamera. Eine Diode sorgt dafür, dass der Akku nicht beschädigt wird, falls versehentlich Plus und Minus vertauscht werden.

Software

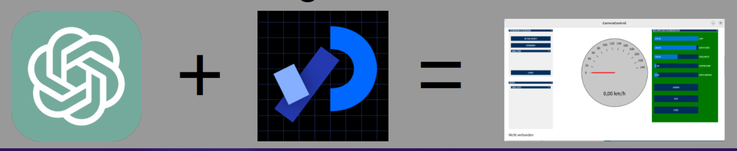
```
const int Sensoren[] =            {31,39,33,37,35};          // Arduino Pins, wo die Sensoren angeschlossen sind
volatile byte LS_Werte[] =        { 0, 0, 0, 0, 0};          // Werte der Sensoren (an/aus)
volatile byte Abschnitt_speed[] = { 0, 0, 0, 0, 0};          // Geschwindigkeit, die nach den jeweiligen Sensor gesetzt wird
volatile int AktSpeed = 0;                                   // Aktuelle Geschwindigkeit
```

```
void loop() {                                                // Arduino Endlosschleife
  while (Serial2.available() > 0) {                          // Falls Daten von Prozessing ...
    static String NachrichtVonProcessing = "";               // Eine Textvariable zum Speichern der Nachricht
    char receivedChar = Serial2.read();                      // Zeichen aus Serial2 lesen
    if (receivedChar == '\n') {                              // Wenn Return...
      speedwerte_von_processing(NachrichtVonProcessing);     // ...dann ist Nachricht komplett -> bearbeiten, interpretieren
      NachrichtVonProcessing = "";                           // alte Nachricht löschen
    } else {                                                 // Noch kein Return-Zeichen?
      NachrichtVonProcessing += receivedChar;                // Zeichen zu der Nachricht-Variable hinzufügen
    }
  }
  Serial2.println(AktSpeed);                                 // Sende aktuelle Geschwindigkeit an Processing
}
```

```
void speedwerte_von_processing(String input) {               //Funktion interpretiert die Nachrichten von Processing
  if (input.startsWith("speeds :")) {
    //Liste der Geschwindigkeiten wird in Abschnitt_speed[] geschrieben
  }
  else if (input.startsWith("stop")) {
    //Stop-Wunsch wird nach Looping Sensor ausgeführt
  }
  else if (input.startsWith("start")) {
    //Anfangsgeschwindigkeit bis zum ersten Sensor, bis die richtige Abschnittsgeschwindigkeit gesetzt wird
  }
}
```

```
// Timer-Interrupt. Diese Funktion wird vom Arduino automatisch jede Millisekunde aufgerufen
ISR(TIMER1_COMPA_vect) {
  digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN)); // Mit eingebauten LED jedes Mal blinken
  FahrtWatchDog++;                                       // Watch-Dog Zähler erhöhen
  if ( FahrtWatchDog >=2000 ){                           // Nach 2 Sekunden Bahn ausschalten, falls WD-Zähler überläuft
    AktSpeed = 0;
  }
  for (int i = 0; i < SensorenAnz; i++) {                // Prüfe alle Sensoren
    LS_Werte[i] = digitalRead(Sensoren[i]);
    if (LS_Werte[i] == 1){                               // Wenn Lichtschranke AN
      FahrtWatchDog = 0;                                 // Watch-Dog Zähler reset
      if ( StopWunsch && (i==StopLS) ){                  // Wenn Stop-Wunsch und entsprechende Lichtschranke   ...
        AktSpeed = 0;
        Gestoppt = true;                                 //  -> Stop
      }
      else{
        if (!Gestoppt){                                  //wenn nach stoppwunsch nicht gestoppt
          AktSpeed = Abschnitt_speed[i];                 // übernehe die Abschnittsgeschwindigkeit
        }
      }
    }
    analogWrite(BahnPin, AktSpeed);                      // Setze PWM
  }
}
```

GUI in Processing erstellt mit Hilfe von KI



Idee

Umgesetzt

- Carrera-Bahn als Testumgebung für autonomes Fahren genutzt
- Auto fährt sicher und schnell ohne menschliche Steuerung
- Geschwindigkeit an fünf Streckenpunkten mit IR-Sensoren erfasst
- Geschwindigkeit abhängig von Streckenabschnitt angepasst
- PWM-Signal steuert Motorleistung über Arduino
- Timer-Interrupt liest alle Sensoren regelmäßig aus
- Auto mit FPV-Kamera
- Projekt verbindet Physik, Technik und Programmierung

Probleme	Lösungsansätze
1) Wackelkontakte	→ Alle Kontakte verlötet
2) Kurzschlüsse, MOSFET-Ausfall	→ Starke Auto-Lampe in Reihe
3) Störungen der Lichtsensoren	→ Abdeckung der Sensoren
4) Unterschiedliche Fahrverhalten	→ Eigene Einstellung pro Auto

Weitere Ideen

MOSFET durch H-Brückenschaltung für Fahrtrichtungswechsel/Bremsen ersetzen * Kamera-Auto direkt über die Schiene laden * Automatische Anpassung der Fahrparameter pro Abschnitt * Tatsächliche Geschwindigkeit präzise messen * Startsignal automatisch oder manuell auslösen * VR-Ansicht live auf Monitor übertragen * Finish-Signal bei Ziellinie auslösen * GUI mit Rundenzähler integrieren * Highscore-Anzeige für Spielergebnisse * Musikuntermalung während des Rennens * Erkennung von Unfällen bei „Mensch-Auto“ * Beschleunigungssensor im Auto * Eigenes Fahrzeug entwickeln und bauen