Zugriff auf den mBot per ESP-Now

Aud dem mBot wird ein Programm aufgespeilt, welches in der Lage ist, seriell Daten mit eine ESP-Modul auszutauschen, welches statt dem Bluetooth- oder dem WLAN-Modull auf dem mCore-Controller aufgesteckt ist.

Die Daten vom ESP-Modul in Richtung des mBot sind in erster Linie verschiedene Befehle an:

- die beiden Motoren
- die beiden Onboard-RGB-LEDs
- die LED-Matrix,
- den Summer
- den IR-Sender
- weitere Aktoren, die zusätzlich an freien Port des mBot angeschlossen werden können.

Der mBot kann in umgekehrter Richtung die Daten der verschiedenen Sensoren an das ESP-Modul übertragen. Das sind sind die Werte:

- des Helligkeitssensor
- des Linienverfolgers
- des Helligkeitssensor
- von Sensoren, die zusätzlich an freien Port des mBot angeschlossen werden können.

Das Programm auf den mBot wird einmalig der aktuellen Aktoren- und Sensorenkonfiguration angepasst und bleibt danach unverändert.

Das seriell verbundene ESP-Modul übermittelt die seriell übertragenen Daten per ESP-Now an ein zweites ESP-Modul. Dieses Programm wird nur einmalig aufgespielt.

Das zweite ESP-Modul soll entweder

- als Fernsteuerung dienen, die auch die Auswertung der Sensordaten und die entsprechenden Reaktionen das mBot darauf beinhaltet,
- über die seriellen Verbindung zu einer Python-IDE auf dem PC die Steuerung des mBot durch ein Pythonprogramm ermöglichen.

Für die Entwicklung des Programmes des mit dem mBot seriell verbundenen ESP-Moduls wird ein „ESP8266 12E“-Modul auf einem Steckbrett genutzt. Später kann dieses durch eine kleineres „ESP8266-01S“ oder „ESP8266-12F (D1 Mini) ersetzt werden, welches auf das mCore-Board gesteckt wird.

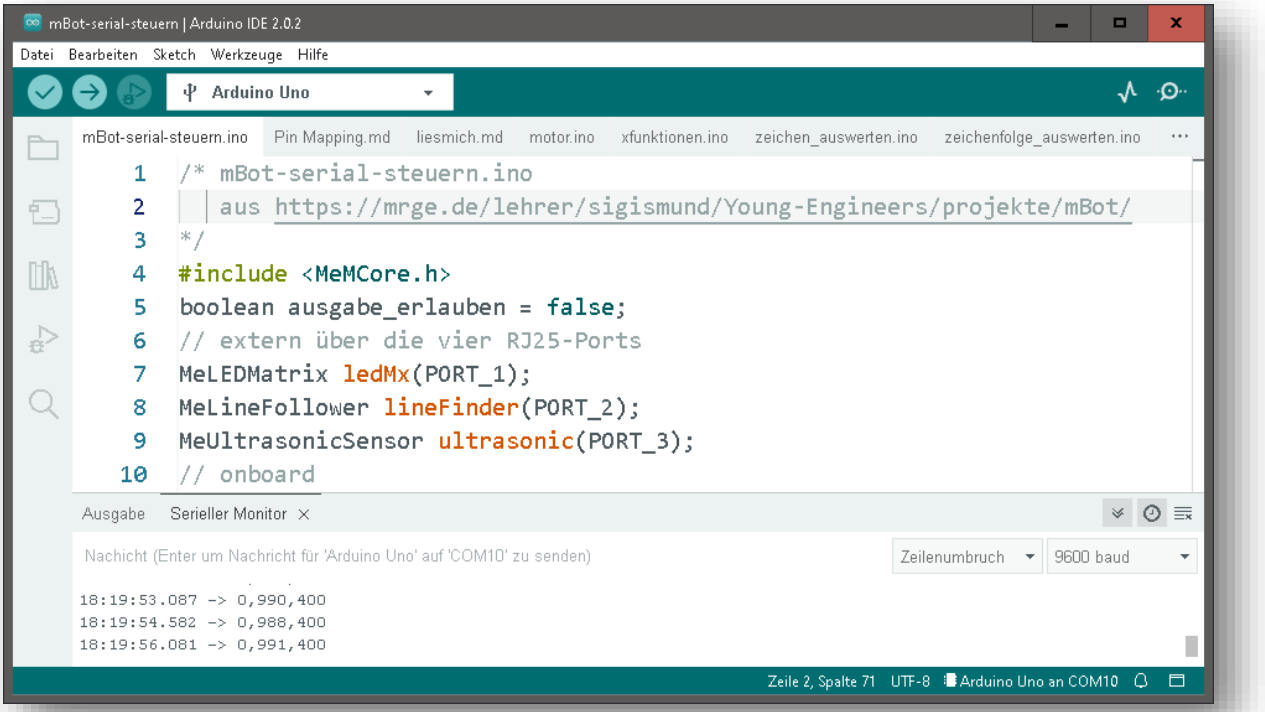
Das mBot-Programm

Der mBot wird mit der Arduino-IDE verbunden, auf der die Bibliothek “MakeBlockDrive“ installiert ist.
https://mrge.de/lehrer/sigismund/makeblock/arduino-ide/

Damit ist die Programmierung des mBot leicht möglich.

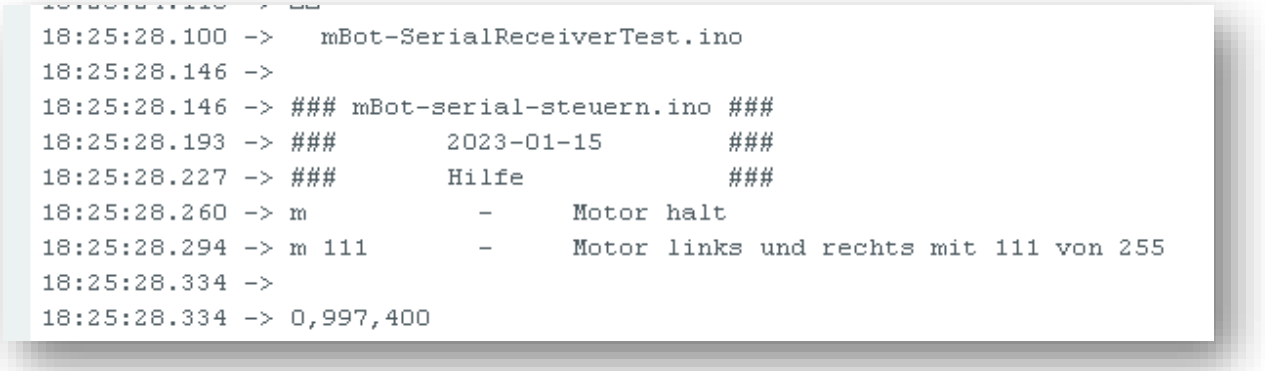
Die Programmentwicklung auf dem mCore erfolgt über die Arduino IDE. Der Datenaustausch erfolgt zum Testen über den Seriellen Monitor.

https://mrge.de/lehrer/sigismund/Young-Engineers/projekte/mBot/



Alle Dateien befinden sich im Ordner „mBot-serial-steuern“ und die Datei, die auf den mCore übertragen wird, ist die „mBot-seriial-steuern.ino“.

Ist das Programm auf dem mCore hochgeladen und mit der Arduino IDE verbunden, meldet es sich bei Neustart auf dem „Seriellen Monitor“ der Arduino IDE.



Die Messwerte des Linien-, Helligkeit- und Entfernungssensors werden danach auf den „Seriellen Monitor“ übertragen. Über die Eingabezeile können verschiedene Kurzbefehle an den mBot gesendet werden.
Die entsprechenden Dateien im Ordner „mBot-serial-steuern“ können bei Bedarf erweitert werden

Die vorgesehene Hilfefunktion kann nicht im vollen Umfang bereitgestellt werden. Nach dem Compilieren und Hochladen startet der mCore-Controller nicht.

mBot-serial-steuern.ino | Pin Mapping.md | liesmich.md | motor.ino | xfunktionen.ino | zeichen_auswerten.ino | zeichenfolge_auswerten.ino

```
1  void hilfe() {
2    Serial.println("### mBot-SerialReceiverTest.ino ###");
3    Serial.println("###         2023-01-15          ###");
4    Serial.println("###           Hilfe             ###");
5    String msg="";
6    // //              1         2         3         4         5         6
7    // //     12345678901234567890123456789012345678901234567890123456789 - 60 Zeichen
8    msg+= "h            -     Hilfe über Seriellen Monitor ausgeben\n";
9    msg+= "i            -     Programminfo über Seriellen Monitor ausgeben\n";
10   msg = "m            -     Motor halt\n";
11   msg+= "m 111        -     Motor links und rechts mit 111 von 255  \n";
12   Serial.println(msg);
13   // msg = "m 111 -222   -     Motor links mit 111 und rechts mit -222 \n";
14   // msg+= "p            -     Summer an für 100 ms mmit 600 Hz        \n";
15   // msg+= "t            -     Ausgabe der Zeit in mm:ss auf der LED-Matrix\n";
16   // msg+= "x            -     LED-Matrix löschen\n";
17   // Serial.println(msg);
18   // msg = "x txt        -     LED-Matrix: txt ausgeben\n";
19   // msg+= "x zahl        -    LED-Matrix: zahl ausgeben\n";
20   // Serial.println(msg);
21 }
```

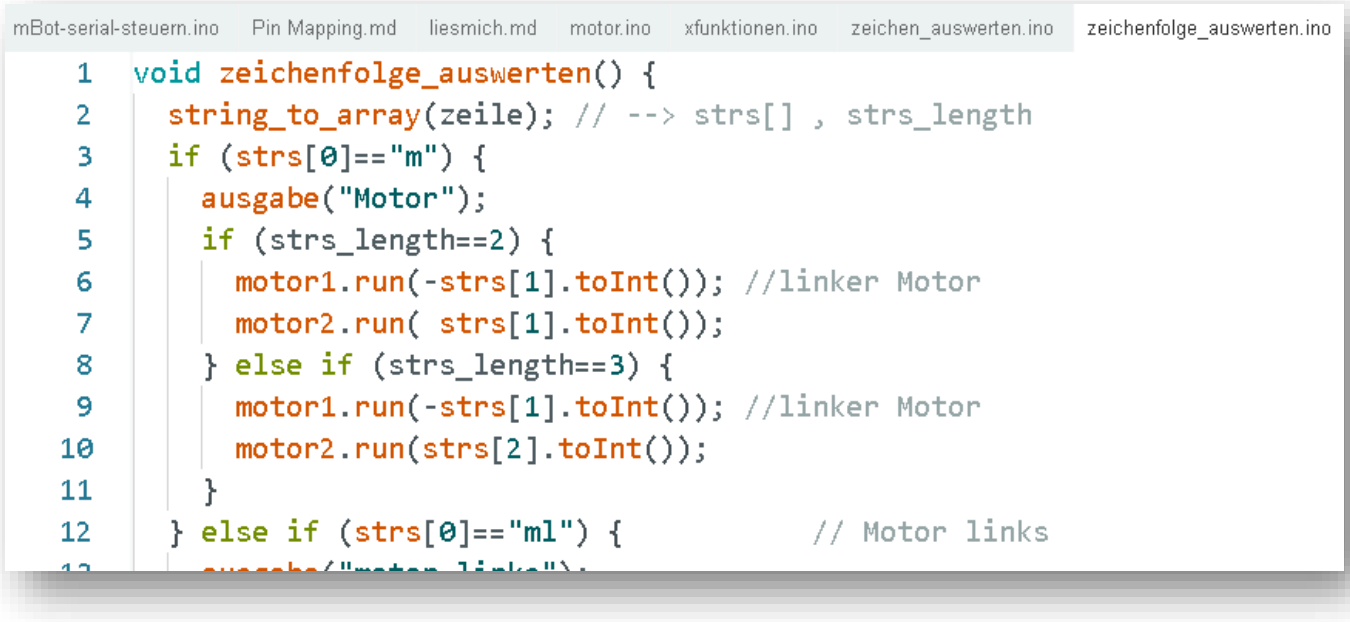
Mit den auskommentierten Programmzeilen funktioniert das Programm. (Als Ursache wird der umfangreiche Text vermutet. Die Länge der Zeichenketten scheint auf 256 Byte begrenzt.

Es können einzelne Zeichen als Befehle übertragen und ausgewertet werden.
Dabei ist die alphabetische Reihenfolge in der switch-case-Anweisung zu beachten.

mBot-serial-steuern.ino | Pin Mapping.md | liesmich.md | motor.ino | xfunktionen.ino | zeichen_auswerten.ino | zeichenfolge_auswerten.ino

```
1  void zeichen_auswerten() {
2    ausgabe("Zeichen_auswerten");
3    char zeichen = zeile[0];
4    switch (zeichen) {  // Zeichen müssen alphaptisch sortiert sein
5      case '0':
6        ausgabe("RGB aus"); led.setColorAt(0, 0, 0, 0); led.setColorAt(1,
7      case 'b':
8        ausgabe("blau"); led.setColorAt(1, 0, 0, 55); led.show(); break;
9      case 'h':
10       hilfe(); break;
11     case 'm':
```

Die Auswertung von durch Leerzeichen getrennte Zeichenketten erlaubt Befehle mit den zugehörigen Parametern zu empfangen und auszuwerten.



Die beiden Dateien mit der Endung „.md“ sind MarkDown-Dateien und können zur zusätzlichen Dokumentation des Programmes genutzt werden.

Über den „Seriellen Monitor“ können alle Funktionen des mBot getestet werden.

Im nächsten Schritt erfolgt die serielle Kopplung des mCore mit einem µController, der WLAN- bzw. WSP-Now-fähig sind. Dafür sind ein ESP8266 oder ESP32 vorgesehen.

Das Programm auf dem ESP8266

Dieses Programm wird ebenfalls über die Arduino IDE entwickelt, da wir ESP-Now und Python auf diesen Modulen nicht beherrschen.

Wegen der unterscheidlichen Signalspannungen von 5V und 3.3V ist der TX des mCore und dem Rx des ESP8266 ein Spannungsteiler über eine Widerstandsschaltung zu verbinden.
TX des mCore und Rx des ESP-Moduls können direkt verbunden werden.

mCore-serial ←→ ESP8266-softserial

Während der Programmentwicklung erfolgt die Verbindung zwischen mCore und ESP-Modul über die seriellen Pins TX (D1) und RX (D0) und eine software-serielle Schnittstelle Tx/D1 und Rx/D2 am ESP-Modul.

```
mBot TX    →    grau    -  Spannungsteiler
                                gelb      →    D2 (Rx)

mBot RX    ←    lila    -         grün      ←    D1 (Tx)
```

So kann der serielle Datenaustausch das ESP-Moduls mit dem mCore und später die Kommunikation des ESP8266 mit einem zweiten ESP-Modul per ESP-Now überprüft werden.

Nach einem Reset des mCore meldet sich das startende Programm über die serielle Schnittstelle mit seinem Programmnamen und die Sensordaten des mCore werden regelmäßig (z. B. aller 2 Sekunden) auch gesendet.
Auf dem seriell gekoppelten ESP-Modul sollen jetzt diese Daten über auf dem Seriellen Monitor der mit diesem Modul gekopplten Arduino IDE angezeigt werden.

mCore-serial ←→ ESP8266-softserial-ESPNow ←→ ESP32-ESPNow-serial

Nach erfolgreicher serieller Verbindung von mCore und ESP8266 erfolgt das Hinzufügen Kommunikation über ESP-Now auf dem ESP8266.

Über diese drahtlose Verbindung werden die vom mCore an das ESP8266-Modul gesendeten Daten an ein ESP32-Modul weitergeleitet, welcher diese Daten in einem Pythonprogramm verarbeitet.

Umgekehrt sendet das ESP32-Modul seine Befehle über ESP-Now an das ESP8266-Modul, welches diese seriell an den mCore weiterleitet. Diese Befehle können auf dem ESP32 per Nutzereingaben oder programmgesteuert über ein Pythonprogramm gesendet werden. Unsere ESP-Fernsteuerungen müssen jetzt nur noch für den Befehlssatz des mBot angepasst werden oder ein Pythonprogramm sendet die gewünschten Befehle an den mCore.

SoftSerial-espnow-sender-receiver.ino version-1.md

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <SoftwareSerial.h>

#define MYPORT_TX 5  // D1  -> green    -> RXD der Gegenstelle
#define MYPORT_RX 4  // D2 <-  yellow  <-  TXD der Gegenstelle
SoftwareSerial myPort;


uint8_t receiverAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

const byte nachricht_length = 64;
typedef struct struct_message {
  char nachricht[nachricht_length];
} struct_message;

struct_message myData;
struct_message myData_in;

String nachricht[nachricht_length];
```