Studie 1

Stan Debakker

Inhoudsopgave

1 inleiding **3**

2 LCD scherm **3**

3 IR afstandssensor **4**

4 Lijn sensor **4**

5 Knoppen **5**

6 Motoren **5**

7 Hoofd **8**
7.1 hoofd driver 8
7.1.1 RGB leds 9
7.1.2 Leds 9
7.1.3 Lichtsensor 9
7.2 Ultrasone sensor 10

8 De servomotoren **10**

9 Muziekkaart **12**
9.1 Versie (Version) 21
9.2 Instellingen (SeTtings) 21
9.2.1 EOL als 21
9.2.2 Schrijf timeout 21
9.2.3 List mode 21
9.2.4 Bitrate 21
9.2.5 Commando's die ik niet gebruik 23
9.3 Speel commando's (Play Commands) 23
9.3.1 Speel bestand 23
9.3.2 Volume 23
9.3.3 Herhaal 23
9.3.4 Jump 24
9.3.5 Versnellen 24
9.3.6 Speel toon 24

9.3.7 Boost 24
9.3.8 Simple comandos 24
9.3.9 Spectrum analysator 25
9.4 Bestands commando's (File Commands) 25
9.4.1 Sluit bestand 25
9.4.2 Tijd wijzigen 25
9.4.3 Wissen 25
9.4.4 Bestand verwerker 25
9.4.5 Bestand info 25
9.4.6 Jump 25
9.4.7 Kaart info 25
9.4.8 Lijst bestanden 26
9.4.9 Maak map 26
9.4.10 Verander naam 26
9.4.11 Open 26
9.4.12 Volume gebruik 26
9.4.13 Lees 26
9.4.14 Lees lijn 27
9.4.15 Trunkeren 27
9.4.16 Schrijf 27
9.4.17 Schrijf lijn 27
9.4.18 Bestand status 27
9.5 Errors 28

10 Besluit **28**

11 Bibliografie **29**

1 inleiding

Hieronder volgt een korte beschrijving van de verschillende onderdelen van de Elektor Robot. Al deze onderdelen werden nagekeken op hun werking en getest dat alles nu correct functioneert. De juiste codes voor de respectievelijke componenten wordt bijgehouden in een bibliotheek. Dit voor later gebruik bij het verder uitwerken van de GRAP. Voor het programmeren en testen wordt gebruik gemaakt van een Arduino Mega. Via de usb-poort is het makkelijker programmeren en te testen.

2 LCD scherm

Het scherm werkt nu. Er stond eerst een verkeerd I^2C adres, in de datasheet van de robot die ik vond, maar heb dit kunnen aanpassen.
De code in bibliotheek voor het LCD scherm is dus al in orde.
C++ code

```
1  #include <Arduino.h>
2  #include <Wire.h>
3  #include "LCD.h"
4
5  #define LCD_ADDR 0x63
6
7  //zet de cursor posietie=X+20Y
8  void set_cursor(uint8_t positie){
9    Wire.beginTransmission(LCD_ADDR);
10   Wire.write(0);
11   Wire.write(2);
12   Wire.write(positie);
13   Wire.endTransmission();
14 }
15 //wist het hele scherm
16 void clear_screen(){
17   Wire.beginTransmission(LCD_ADDR);
18   Wire.write(0);
19   Wire.write(12);
20   Wire.endTransmission();
21 }
22 //zet de backlight (aan of af) volgens vairalble aan
23 void backlight(bool aan){
24   Wire.beginTransmission(LCD_ADDR);
25   Wire.write(0);
26   Wire.write(aan?19:20);
27   Wire.endTransmission();
28 }
```

```
29 //print 1 karacter op het scherm
30 void print_char(char karacter){
31   Wire.beginTransmission(LCD_ADDR);
32   Wire.write(0);
33   Wire.write(karacter);
34   Wire.endTransmission();
35 }
36 //print een lijst met karacters of string
37 void print_text(char* text){
38   while (*text!=0){
39     print_char(*text);
40     text++;
41   }
42 }
```

3 IR afstandssensor

Ik heb een aantal metingen gedaan met de afstandssensoren. Er zijn 3 sensoren: rechts, center en links. In de tabel hieronder vind je de analoge waardes die gemeten heb:

afstand (cm)	19	28,5	38	47,5	57	66,5	76
rechts (analog)	240	160	120	80	70	50	40
center (analog)	210	150	180	150	135	110	85
links (analog)	235	185	135	105	80	70	60

Uit deze data en met behulp van geogebra en de datasheet ben ik tot de constatatie gekomen dat $l(A) = \frac{C_1}{A} + C_2$ (in het Nederlands betekent dit de lengte in functie van de Analoge(0-1023) waarde)
De optimale waarden voor de sensoren zijn (r staat voor de correlatiecoëfficiënt hoe dichter bij ± 1 hoe beter):

	C_1	C_2	r
rechts	4286.8927	3.6032954	0.9909
center	7894.2012	-11.045445	0.8223
links	2679.1287	12.849024	0.9831

We kunnen besluiten dat deze afstandssensoren werken, behalve dat de center sensor een probleem heeft rond 38cm.

4 Lijn sensor

De robot heeft 3 lijnsensoren.
Bij ongeveer 0V is de ondergrond wit, bij 5V is het zwart, en de rechtse sensor heeft een witte ondergrond 2V in plaats van 0V.
C++ code

```
1 #include <Arduino.h>
```

```
2  #include "IR_afstandsensor.h"
3
4  #define IR_L 0
5  #define IR_C 1
6  #define IR_R 2
7
8  //stuurt de afstand van de afstandsensor tot een object terug (~10-80 cm)
9  //analoge_pin is de pin waarop de sensor is bevestigd
10 //sensor is de sensor (links:IR_L/0,midden:IR_C/1,rechts:IR_R/2)
11 //stuurt 0 terug als sensor fout is ingeheven
12 float read_IR_sensor(uint8_t analoge_pin,uint8_t sensor){
13   int16_t data=0;
14   for (int8_t i=0;i<16;i++){
15     data+=analogRead(analoge_pin);
16     delay(10);
17   }
18   //stuur de afstand terug volgens de calibratie
19   switch (sensor){
20     case IR_R:
21       return 16*4286.8927/float(data)+3.6032954;
22     case IR_C:
23       return 16*7894.2012/float(data)-11.045445;
24     case IR_L:
25       return 16*2679.1287/float(data)+12.849024;
26     default:
27       return 0;
28   }
29 }
```

5 Knoppen

De robot heeft 2 NO (normaal open) knoppen.

6 Motoren

De robot heeft 2 12V motoren met een I^2C driver.
De driver kan ook de stroom door de motoren en de spanning erover meten.
Ook kan hij de posietie van de motoren.
C++ code

```
1  #include <Wire.h>
2  #include <Arduino.h>
3  #include "motor.h"
4
```

```
5  #define driver_ADDR 0x58
6
7  //https://www.robot-electronics.co.uk/htm/md25i2c.htm
8  // get data
9    int32_t get_encoder1(){
10     return get_4byte_command(0x2);
11   }int32_t get_encoder2(){
12     return get_4byte_command(0x6);
13   }
14   uint8_t get_snelheid1(){
15     return get_1byte_command(0x0);
16   }uint8_t get_snelheid2(){
17     return get_1byte_command(0x1);
18   }uint8_t get_volt(){
19     return get_1byte_command(0xA);
20   }uint8_t get_stroom1(){
21     return get_1byte_command(0xB);
22   }uint8_t get_stroom2(){
23     return get_1byte_command(0xC);
24   }uint8_t get_versnelling(){
25     return get_1byte_command(0xE);
26   }uint8_t get_mode(){
27     return get_1byte_command(0xF);
28   }
29
30   int32_t get_4byte_command(uint8_t command){
31     int32_t data=0;
32     Wire.beginTransmission(driver_ADDR);
33     Wire.write(command);
34     Wire.endTransmission(false);
35     Wire.requestFrom(driver_ADDR,4,true);
36     while(Wire.available()){
37       data=(data<<8)+Wire.read();
38     }
39     return data;
40   }
41   uint8_t get_1byte_command(uint8_t command){
42     uint8_t data;
43     Wire.beginTransmission(driver_ADDR);
44     Wire.write(command);
45     Wire.endTransmission(false);
46     Wire.requestFrom(driver_ADDR,1,true);
47     while(Wire.available()) {
48       data=Wire.read();
49     }
50     return data;
```

```
51    }
52  //sent commando's
53    void set_acceleration(uint8_t versnelling){
54      Wire.beginTransmission(driver_ADDR);
55      Wire.write(14);
56      Wire.write(versnelling);
57      Wire.endTransmission();
58    }
59    void set_mode(uint8_t mode){
60      Wire.beginTransmission(driver_ADDR);
61      Wire.write(15);
62      Wire.write(mode);
63      Wire.endTransmission();
64    }
65    void set_regulator(bool set_to){
66      Wire.beginTransmission(driver_ADDR);
67      Wire.write(16);
68      if (set_to){
69        Wire.write(0x31);
70      }else{
71        Wire.write(0x30);
72      }
73      Wire.endTransmission();
74    }
75    void set_time_out(bool set_to){
76      Wire.beginTransmission(driver_ADDR);
77      Wire.write(16);
78      if (set_to){
79        Wire.write(0x33);
80      }else{
81        Wire.write(0x32);
82      }
83      Wire.endTransmission();
84    }
85    void reset_encoders(){
86      Wire.beginTransmission(driver_ADDR);
87      Wire.write(16);
88      Wire.write(0x20);
89      Wire.endTransmission();
90    }
91    void set_snelheid1(int8_t snelheid){
92      Wire.beginTransmission(driver_ADDR);
93      Wire.write(0);
94      Wire.write(snelheid);
95      Wire.endTransmission();
96    }
```

```
97    void set_snelheid2(int8_t snelheid){
98      Wire.beginTransmission(driver_ADDR);
99      Wire.write(1);
100     Wire.write(snelheid);
101     Wire.endTransmission();
102   }
```

7 Hoofd

7.1 hoofd driver

In het hooft zit er in micro controler verbonden via I^2C met de hooft prosessor.
deze stuurt volgende zaken aan.
C++ code

```
1  #include <Arduino.h>
2  #include <Wire.h>
3  #include "hoofd.h"
4
5  #define licht_sensor_L 0
6  #define licht_sensor_R 1
7  #define geluids_sensor 2
8
9  #define hoofd_ADDR 0x88
10
11 void set_RGB_LEDs(bool rechts, uint8_t Rood, uint8_t Groen, uint8_t Blauw){
12   Wire.beginTransmission(hoofd_ADDR);
13   if (rechts){
14     Wire.write(4);
15     Wire.write(Rood);
16     Wire.write(Groen);
17     Wire.write(Blauw);
18   }else{
19     Wire.write(1);
20     Wire.write(Blauw);
21     Wire.write(Groen);
22     Wire.write(Rood);
23   }
24   Wire.endTransmission();
25 }
26 uint8_t get_sensor_value(uint8_t sensor){
27   Wire.beginTransmission(hoofd_ADDR);
28   switch (sensor){
29     case licht_sensor_L:
```

```
30          Wire.write(8);
31          break;
32        case licht_sensor_R:
33          Wire.write(9);
34          break;
35        case geluids_sensor:
36          Wire.write(10);
37        default:
38          Wire.endTransmission();
39          return 0;
40      }
41      Wire.endTransmission(false);
42      uint8_t data=0;
43      Wire.requestFrom(hoofd_ADDR,1,true);
44      while(Wire.available()) {
45        data=Wire.read();
46      }
47      return data;
48  }
49  void set_LEDs(uint8_t LEDs){
50      Wire.beginTransmission(hoofd_ADDR);
51      Wire.write(7);
52      Wire.write(LEDs);
53      Wire.endTransmission();
54  }
55  void set_gevoeligheid (uint8_t gevoeligheid){
56      Wire.beginTransmission(hoofd_ADDR);
57      Wire.write(11);
58      Wire.write(gevoeligheid);
59      Wire.endTransmission();
60  }
```

7.1.1 RGB leds

Het hoofd heeft 2 sets van PWM (puls with modulation) gestuurde RGB (0-255 per kleur) leds.

7.1.2 Leds

Het hoofd heeft 8 leds die apart gestuurd kunnen worden.

7.1.3 Lichtsensor

Het hoofd heeft 2 lichtsensoren.

7.2 Ultrasone sensor

Het hoofd heeft een ultrasone sensor.
C++ code

```
#include <Wire.h>
#include <Arduino.h>
#include "US_afstands_sensor.h"

#define US_ADDR 0x70

uint16_t US_afstands_sensor(){
  uint16_t data;
  Wire.beginTransmission(US_ADDR);
  Wire.write(0);
  Wire.write(0x51);
  Wire.endTransmission();
  delay(70);
  Wire.beginTransmission(US_ADDR);
  Wire.write(2);
  Wire.endTransmission(false);
  Wire.requestFrom(US_ADDR,2);
  while(Wire.available()) {
    data=data<<8;
    data+=Wire.read();
  }
  return data;
}
```

8 De servomotoren

De robot heeft 4 servomotoren. 2 voor de "handjes" en 2 om het hoofd te besturen.
De beide "handjes" zijn apart bestuurbaar tussen 0= volledig rechts en 255= links.
De eerste motor bedient het hoofd horizontaal: 0 volledig links en 255 rechts horizontaal.
De tweede motor bedient het hoofd verticaal, hierbij is 0 volledig naar beneden en 255 volledig omhoog.
C++ code

```
#include <Wire.h>
#include <Arduino.h>
#include "servo.h"
```

```
4
5  #define Servo_ADDR 0x55
6
7  void init_servo(uint8_t mode){
8    Wire.beginTransmission(Servo_ADDR);
9    Wire.write(0);
10   Wire.write(mode);
11   Wire.endTransmission();
12 }
13 bool set_pos(uint8_t servo,uint8_t posietie){
14   Wire.beginTransmission(Servo_ADDR);
15   switch (servo){
16     case Hooft_horizontaal:
17       Wire.write(1);
18       break;
19     case Hooft_vertikaal:
20       Wire.write(2);
21       break;
22     case Grip_L:
23       Wire.write(3);
24       break;
25     case Grip_R:
26       Wire.write(4);
27       break;
28     default:
29       Wire.endTransmission();
30       return false;
31   }
32   Wire.write(posietie);
33   Wire.endTransmission();
34   return true;
35 }
36 bool set_ofset(uint8_t servo,uint8_t ofset){
37   Wire.beginTransmission(Servo_ADDR);
38   switch (servo){
39     case Hooft_horizontaal:
40       Wire.write(5);
41       break;
42     case Hooft_vertikaal:
43       Wire.write(6);
44       break;
45     case Grip_L:
46       Wire.write(7);
47       break;
48     case Grip_R:
49       Wire.write(8);
```

```
50          break;
51        default:
52          Wire.endTransmission();
53          return false;
54      }
55      Wire.write(ofset);
56      Wire.endTransmission();
57      return true;
58  }
```

9 Muziekkaart

de tijdelijke code (nog niet alles is getest wegens een te grote SD kaart of omgezet in code)

```
1  #define settings 0
2  #define playback_commands 1
3  #define file_commands 2
4
5  #define EOL_CR 0
6  #define EOL_LF 1
7  #define EOL_CRLF 2
8
9  const uint32_t temp[]={9600,19200,38400,57600,115200,2400,4800,230400,460800};
10
11 struct play_back_info {
12   uint8_t error=0;  //enkel as er een error is gebeurt
13   uint16_t position;
14   uint8_t samplerate;
15   uint16_t bitrate;
16   char channels;
17 };
18 struct play_back_status{
19   uint8_t error=0;  //enkel as er een error is gebeurt
20   uint16_t tijd;
21   uint8_t aantal_keer;
22   char status;  //P: speelt,S: gestopt,D: gepouzeert
23 };
24 struct versie {
25   uint8_t main_versie;
26   uint8_t sub_versie;
27   bool beta;
28   uint8_t beta_vesie;
29   char model[5];
```

```
30 };
31 struct free_handeler{
32   uint8_t error=0;
33   uint8_t data=0;
34 }
35 struct file_info{
36   uint8_t error=0;
37   uint32_t huidige_positie=0;
38   uint32_t grote=0;
39 };
40
41 void setup() {
42   delay(500);
43   Serial.begin(115200);
44   /*for (uint8_t i=0;i<8;i++){
45     Serial.print(temp[i]);
46     Serial1.begin(temp[i]);
47     Serial.print('\t');
48     Serial.println(set_bitrate(4),HEX);
49     delay(1000);
50   }*/
51   uint32_t bit_rate=temp[4];
52   Serial.println(bit_rate);
53   Serial1.begin(bit_rate);
54   Serial1.write('\r');
55   Serial1.write('V');
56   Serial1.write('\r');
57 }
58
59 void loop() {
60   if (_available()){
61     Serial.print(char(Serial1.read()));
62   }
63   if (Serial.available()>0){
64     _write_char(char(Serial.read()));
65   }
66 }
67
68 void sincronize_transmission(){
69   _write_char('\r');
70   char c=0;
71   while (_available()&&c=='>') {
72     c=_read_char();
73   };
74 }
75 //instellingen
```

```
76    uint8_t set_EOL_as(uint8_t mode){
77      char data;
78      switch (mode){
79        case EOL_CR:
80          data='0';
81          break;
82        case EOL_LF:
83          data='1';
84          break;
85        case EOL_CRLF:
86          data='2';
87          break;
88        default:
89          return 6; //command formating error
90      }
91      _send_command('E',data,settings);
92      return _get_return_error();
93    }
94    uint8_t set_write_timeout(uint8_t tijd){
95      if (tijd!=255){
96        char data[4];
97        sprintf(data,"%dn",tijd);
98        _send_command('D',data,settings);
99        return _get_return_error();
100     }else{
101       return 6; //command formating error
102     }
103   }
104   uint8_t set_List_mode(bool grote){
105     if (grote){
106       _send_command('L','0',settings);
107     }else{
108       _send_command('L','1',settings);
109     }
110     return _get_return_error();
111   }
112   uint8_t _set_bitrate(uint8_t waarde){
113     if (waarde<8){
114       char data[2]={waarde+'0','\n'};
115       _send_command('D',data,settings);
116       return _get_return_error();
117     }else{
118       return 6; //command formating error
119     }
120   }
121   /*uint8_t _set_hardware_bussy(bool actief){  //kan niet worden gebruikt niet g
```

```
122       _send_command('H',(actief?"0":"1"),settings);
123       return _get_return_error();
124     }
125     uint8_t _input_style(uint8_t mode){
126       if (mode<3){
127         char data[2]={mode+'0','\n'};
128         _send_command('S',data,settings);
129         return _get_return_error();
130       }else{
131         return 6; //command formating error
132       }
133     }
134     uint8_t _set_prompt_char(char karakter){
135       char data[4];
136       sprintf(data,"%dn",karakter);
137       _send_command('D',data,settings);
138       return _get_return_error();
139     }*/
140   //speel commando's
141     uint8_t play_file(char* path){
142       _send_command('F',path,playback_commands);
143       return _get_return_error();
144     }
145     uint8_t set_volume(uint8_t links,uint8_t rechts){
146       if (links!=255 & rechts!=255){
147         if (links==rechts){
148           char data[4];
149           sprintf(data,"%d\n",links);
150           _send_command('V',data,playback_commands);
151           return _get_return_error();
152         }else{
153           char commando[8];
154           sprintf(commando,"%d-%d\n",links,rechts);
155           _send_command('V',commando,playback_commands);
156           return _get_return_error();
157         }
158       }else{
159         return 6; //command formating error
160       }
161     }
162     uint8_t herhaal(uint8_t aantal_keer){ //0 is een oneindig aantal keer
163       char data[4];
164       sprintf(data,"%d\n",aantal_keer);
165       _send_command('O',data,playback_commands);
166       return _get_return_error();
167     }
```

```
168   uint8_t jump(uint16_t tijd){
169     char data[6];
170     sprintf(data,"%d\n",tijd);
171     _send_command('J',data,playback_commands);
172     return _get_return_error();
173   }
174   uint8_t speed(uint8_t persentage){
175     if (persentage<=250&persentage>=90){
176       char commando[4];
177       sprintf(commando,"%d\n",persentage);
178       _send_command('X',commando,playback_commands);
179       return _get_return_error();
180     }
181     return 6; //command formating error
182   }
183   uint8_t play_tone(uint8_t base_frequency_index,uint8_t d){
184     //tone_frequency=base_frequency_index*d
185     //base_frequency_index=0: 334,53125
186     //base_frequency_index=1: 375
187     //base_frequency_index=2: 250
188     //base_frequency_index=3: 172,265625
189     //base_frequency_index=4: 187,5
190     //base_frequency_index=5: 125
191     //base_frequency_index=6: 86,328125
192     //base_frequency_index=7: 93,75
193     if (base_frequency_index<=8){
194       if (d<=31){
195         char data[4];
196         sprintf(data,"%d\n",((base_frequency_index<<5)+d));
197         _send_command('T',data,playback_commands);
198         return _get_return_error();
199       }
200     }
201     return 4;
202   }
203   uint8_t boost(int8_t treble_amplitude,uint8_t treble_frequency,int8_t bass_amp
204     if (treble_amplitude>=-8&&treble_amplitude<=7){
205       if (treble_frequency<=15){
206         if (bass_amplitude>=-8&&bass_amplitude<=7){
207           if (bass_frequency<=15){
208             uint16_t boost_o=(treble_amplitude&0xf)<<12|(treble_frequency&0xf)<<
209             char data[6];
210             sprintf(data,"%d\n",boost);
211             _send_command('B',data,playback_commands);
212             return _get_return_error();
213           }
```

```
214           }
215         }
216       }
217       return 6; //command formating error
218     }
219     uint8_t pouse(){
220       _send_command('P',"\n",playback_commands);
221       return _get_return_error();
222     }
223     uint8_t stop(){
224       _send_command('S','\n',playback_commands);
225       return _get_return_error();
226     }
227     uint8_t reset_audio(){
228       _send_command('R','\n',playback_commands);
229       return _get_return_error();
230     }
231     play_back_info get_play_back_info(){
232       sincronize_transmission();  //sincronizeer de bus
233       play_back_info info;
234       _send_command('I','\n',playback_commands);
235       while(!_available()){}
236       if (_peek_char()=='E'){
237         info.error=_get_return_error();
238       }else{
239         info.position=_get_nummer(DEC);
240         info.samplerate=_get_nummer(DEC);
241         info.bitrate=_get_nummer(DEC);
242         info.channels=_get_nummer(DEC);
243       }
244       return info;
245     }
246     play_back_status get_play_back_status(){
247       sincronize_transmission();  //sincronizeer de bus
248       play_back_status info;
249       _send_command('I','\n',playback_commands);
250       while(!_available()){}
251       if (_peek_char()=='E'){
252         info.error=_get_return_error();
253       }else{
254         info.status=_read_char();
255         info.tijd=_get_nummer(DEC);
256         info.aantal_keer=_get_nummer(DEC);
257       }
258       return info;
259     }
```

```
260 //file commando's
261   uint8_t close_file(uint8_t bestand){
262     if (bestand<=4){
263       if (bestand==0){
264         _send_command('C','\n',file_commands);
265       }else{
266         char data[2]={bestand+'0','\n'};
267         _send_command('C',data,file_commands);
268       }
269       return _get_return_error();
270     }
271     return 6; //command formating error
272   }
273   uint8_t change_time_stamp(uint16_t jaar,uint8_t maand,uint8_t dag,uint8_t uur,
274     if(jaar<2000){
275       return 6; //command formating error
276     }else if(maand==0|maand>12){
277       return 6; //command formating error
278     }else if(dag==0|dag>31){
279       return 6; //command formating error
280     }else if(minuten==0|minuten>60){
281       return 6; //command formating error
282     }else if(seconden==0|seconden>60){
283       return 6; //command formating error
284     }
285     char data[20+sizeof(path)];
286     sprintf(data,"%d-%d-%d-%d-%d-%d-%s\n",jaar,maand,dag,uur,minuten,seconden,pa
287     _send_command('D',data,file_commands);
288     return _get_return_error();
289   }
290   uint8_t erase_file(char* path){
291     _send_command('E',path,file_commands);
292     return _get_return_error();
293   }
294   free_handeler get_free_handeler(){
295     free_handeler data;
296     _send_command('F','\n',file_commands);
297     if (_peek_char()=='E'){
298       data.error=_get_return_error();
299     }else{
300       data.data=_get_nummer(HEX);
301     }
302     return data;
303   }
304   file_info bestands_info(){
305     file_info data;
```

```
306       _send_command('E','\n',file_commands);
307       if (_peek_char()=='E'){
308         data.error=_get_return_error();
309       }else{
310         data.huidige_positie=_get_nummer(DEC);
311         data.grote=_get_nummer(DEC);
312       }
313
314     }
315     uint8_t jump(uint32_t addr){
316       char data[11];
317       sprintf(data,"%d\n",addr);
318       _send_command('J',data,file_commands);
319       return _get_return_error();
320     }
321     uint8_t jump_end(){
322       _send_command('J',"E\n",file_commands);
323       return _get_return_error();
324     }
325     uint8_t change_name(char* path,char* old_name,char* new_name){
326       char data[4+2*sizeof(path)+sizeof(old_name)+sizeof(new_name)];
327       sprintf(data,"%s/%s-%s/%s")
328     }
329   //private functions
330     bool _available(){
331       return Serial1.available();
332     }
333     char _peek_char(){
334       while (!_available()){}
335       return Serial1.peek();
336     }
337     char _read_char(){
338       while (!_available()){}
339       return Serial1.read();
340     }
341     void _write_char(char data){
342       Serial1.write(data);
343     }
344     void _send_command(char instelling,char* opperands,uint8_t comando){
345       switch (comando){
346         case settings:
347           _write_char('S');
348           _write_char('T');
349           break;
350         case playback_commands:
351           _write_char('P');
```

```
352           _write_char('C');
353           break;
354         case file_commands:
355           _write_char('F');
356           _write_char('C');
357           break;
358       }
359       _write_char(instelling);
360       while (*opperands!='\n'){
361         _write_char(*opperands);
362       }
363       _write_char('\r');
364     }
365     uint8_t _get_return_error(){
366       char c;
367       while(true){
368         c=_read_char();
369         if (c=='>'){
370           return 0; //alles goed
371         }else if (c=='E'){
372           return _get_nummer(HEX);
373         }else if (c=='-'){
374           while (_available()) _read_char();  //data (clear de buffer)
375           return 0; //alles goed
376         }
377       }
378     }
379     int32_t _get_nummer(uint8_t base){
380       bool negatief=false;
381       char c;
382       int32_t getal;
383       c=_read_char();
384       if (c=='-'){
385         negatief=true;
386         c=_read_char();
387       }
388       while (isHexadecimalDigit(c)){
389         if(isDigit(c)){
390           c-='0'; //maar er een getal van
391         }else if (isLowerCase(c)){
392           c-='a'; //maak er een getal van
393         }else{
394           c-='A'; //maak er een getal van
395         }
396         if (c>=base){
397           break;
```

```
398         }
399         getal=getal*base+c;
400         c=_read_char();
401       }
402       return (negatief? -getal:getal);
403     }
```

9.1 Versie (Version)

print V≪cr≫
geeft de versie van de kaart
formaat: mmm.nn[-bxxx]≪sp≫SN:RMP1-sss...sss
mmm: grote updates
nn: kleindere updates
xxx: beta update (indien het beta is)
sss...sss: serie nummer
mijn versie: 100.01 SN:RMP1-OEM

9.2 Instellingen (SeTtings)

ieder commando begint met ST[instelling][parameter(optioneel)]≪cr≫

9.2.1 EOL als

instelling:'E'
parameter:0(EOL word ≪cr≫) 1(EOL word ≪lf≫) 2(EOL word ≪cr≫≪lf≫)

9.2.2 Schrijf timeout

instelling:'T'
parapeter:0-254
als 0(wacht onijnig) ander waarde * 10ms

9.2.3 List mode

instelling:'L'
parameter:0-1

9.2.4 Bitrate

instelling:'D'
parameter:0-8

index	bitrate (b/s)
0	9600
1	19200
2	38400
3	57600
4	115200
5	2400
6	4800
7	230400
8	460800

9.2.5 Commando's die ik niet gebruik

Hardware Bussy .
instelling:'H'
parameter:0,1
0: incatief
1: actief (set "D"hoog als er iets is aan het spelen)
"Dïs niet verbonden dus dit comando doet niets

input style .
instelling:'S'
parameter:0,1,2
0 "no input style (imput interface ignored)"
1 "8 button/Switch interdace"
2 "7 Bit plus Trigger interface"

prompt character .
instelling:'P'
parameter:0-255
zet het caracter op het einde de transmissie
default '>' of 62

9.3 Speel commando's (Play Commands)

ieder commando begint met PC[instelling][parameter(s) (optioneel)]≪cr≫

9.3.1 Speel bestand

instelling:'F'
parameter:[path]
speel het bestand met path:[path]

9.3.2 Volume

instelling:'V'
parameter:"[rechts]≪sp≫[links]" of [algemeen] of niets
algemeen: zet het volume voor links en rechts
rechts,links:0-255 geluid hoe groter het nummer hoe stiller (255 geen geluid)
niets: de huidige instelling terug
rechts is het geluid dat uit de speaker komt

9.3.3 Herhaal

instelling:'O'
parameter:0-255 aantal keer je iets wil herhalen (0: oneindig aantal keer)

9.3.4 Jump

instelling:'J'
parameter: naar waar je wil gaan (tijd)

9.3.5 Versnellen

instelling:'X'
parameter:90-250 het percentage dat je wil dat de geluidsopname word afspeelt

9.3.6 Speel toon

instelling:'T'
parameter:0-255 ("base frequentie index"*32+d)
met $1 \leq d < 32$ en $1 \leq$"base frequentie index"< 8

base frequentie index	frequentie (Hz)
0	334,53125
1	375
2	25
3	172,265625
4	187,5
5	125
6	86,328125
7	93,75

de frequentie die word gespeeld=base frequentie $\cdot$ d

9.3.7 Boost

instelling:'B'
parameter:0-65535 ((treble amplitude $\cdot$ 4096) + (treble freuentie index $\cdot$ 256) +
(bass amplitude $\cdot$ 16) + (bass frequentie index))
treble frequentie index (0-15) = treble frequentie / 1000
bass frequentie index (0-15) = bass frequentie / 10
amplidude (-8 tot 7)(gebruik de binaire waarde van het 2 complement)
de code die dit uitvoert

```
1  uint16_t boost_o=(treble_amplitude&0xf)<<12|(treble_frequency&0xf)<<8|(bass_ampl
```

9.3.8 Simple comandos

geen parameter
geen return
pauze: 'P'
stop: 'S'
reset: 'R'

9.3.9 Spectrum analysator

instelling:'Y'
dit werkt niet

9.4 Bestands commando's (File Commands)

ieder commando begint met FC[instelling][parameter(s) (optioneel)]≪cr≫

9.4.1 Sluit bestand

instelling:'C'
parameters: (geen sluit alles) of (een specifiek bestand 1-4)

9.4.2 Tijd wijzigen

instelling: 'D'
parameters: (jaar 4 digit)≪sp≫(maand 1-12)≪sp≫(dag 1-31)≪sp≫(uur 0-23)≪sp≫(minutien 0-59)≪sp≫(seconden 0-59)≪sp≫(path(van het bestand))

9.4.3 Wissen

instelling:'E'
parameters:(path van het bestand of de map (moet leeg zijn voor je het kan verwijderen))

9.4.4 Bestand verwerker

instelling:'F'
parameter:geen
return: volgende beschikbare slot terug indien geen stuurt hij een error(E03>)

9.4.5 Bestand info

instelling:'E'
parameter:(bestand slot 1-4)
return:(huidige positie)/(grote)

9.4.6 Jump

instelling:'J'
parameters:(bestand slot 1-4)≪sp≫(E:einde of addr)

9.4.7 Kaart info

instructie:'K'
parameter: geen of 'B' of 'I'
geen: stuurt "CSD"(16 bytes) en "CID"(16 bytes) "raw"door (niet ascii)

'B': stuurt "CSD"(16 bytes) en "CID"(16 bytes) ascii HEX door
'I': stuurt kaart type (1:MMC,2:SD V1,4:SDV2,12:SDHC) en stuurt "CSD", "CID", "OCR"door ascii HEX door met de naam: gescheiden

9.4.8 Lijst bestanden

instelling:'L'
parameters:(geen,'C','S','I')≪sp≫(path of patroon)
geen: als "L"setting=0 grote bestand, als "L"setting=1 aanmaakdatum
'C': aantal bestanden dat in het path of patroon zit
'S': geeft alles wat er in de map zit alleen maar een path te gebruiken, geen patroon
'I': gaat over alles wat kan in het patroon (je moet er wel over lopen, als er geen meer zijn krijg je "E07" oftewel einde bestand)

9.4.9 Maak map

instelling:'M'
parameters:(path naar de map)

9.4.10 Verander naam

instelling:'N'
parameters:(oude path naam)—(nieuwe path naam)

9.4.11 Open

instelling:'O'
parameters:(bestand verwerker) (mode: 'R','W','A',RW') (path)
mode: 'R' allen lezen(path moet al bestaan), 'W' alleen schrijven (path mag nog niet bestaan), 'RW'/'A' schrijven en lezen (path moet nog niet bestaan) ('RW' start in het beging,'A' start op het einde)

9.4.12 Volume gebruik

instelling:'Q'
parameters: geen of 'A'
(als 'A' stuurt FAT(type::"12","16","32")≪cr≫) (vrije ruimte)/(totale ruimte)

9.4.13 Lees

instelling:'R'
parameters:(bestand verwerker) (optioneel aantal bytes worden gelezen max 512) (indien aatal bytes is ingegeven, het start adders)

9.4.14 Lees lijn

instelling:"RL"
parameters: hetzelfde als lees
stopt als hij een EOL (end of line) tegen komt

9.4.15 Trunkeren

instelling:'U'
parameters:(bestand verwerker)
verwijdert alles achter de huidige positie

9.4.16 Schrijf

instelling:'W'
parameters:(bestand verwerker) (optioneel aantal bytes dat wordt geschreven max 512)

9.4.17 Schrijf lijn

instelling:'WL'
parameters: hetzelfde als schrijf
op het einde van de schrijf operatie wordt er een EOL (end of line) geschreven

9.4.18 Bestand status

instelling:'Z'
parameters: (optioneel bestand verwerker)
indien alles goed return:≪sp≫ anders error

9.5 Errors

code	beschrijving
EO2	commando te lang
E03	bestandsverwerker vol
E04	ongekend commando
E05	kaart initialisatie error
E06	commando slecht geformateerd
E07	end of file (EOF)
E08	geen SD kaart
E09	SD kaart reset gefaald
E0A	SD kaart fysiek schrijf beschermd (controleer de schakelaar op de kaart)
EE5	geen map
EE6	alleen lezen bestand
EE7	geen bestand - commando verwachte map
EE8	schrijf fout
EE9	mode incorrect voor schrijven
EEA	geen vrije ruimte op de SD kaart
EEB	bestand verwerker nog niet open
EEC	mode incorrect voor lezen
EED	onbekende mode om een bestand te openen
EEF	file systeem is kapot (test op PC)
EF1	bestand verwerker al open
EF2	bestand bestaat niet
EF3	kan geen map maken
EF4	bestand bestaat al
EF5	bestand is niet correct (controleer spelling en vreemde tekens)
EF6	verkeerde handeling gespecifieerd

10 Besluit

Uit dit blijkt dat alles werkt maar 2 sensoren hebben kleine afwijkingen, en de muziekkaart kon ik nog niet testen omdat ik geen SD kaart heb in FAT32 of lagen maar mijn SD kaart is exFAT.

11 Bibliografie

robot-electronics. (z.d.). SRF02 Ultrasonic range finder. Geraadpleegd op 4 november 2024, van https://robot-electronics.co.uk/htm/srf02techI2C.htm
Vishay. (2009, 17 augustus). TCRT5000. Geraadpleegd op 4 november 2024, van https://www.vishay.com/docs/83760/tcrt5000.pdf
Sharp. (z.d.). gp2Y0A21YK0F. Geraadpleegd op 4 november 2024, van https://global.sharp/products/device/lineup/data/pdf/datasheet/gp2y0a21yk$_e$*.pdf*
Elektor, amp; Huyskens, Bart. (2011, 3 april). Elektor Proton Robot. Geraadpleegd op 4 november 2024, van https://e2cre8.be/wp-content/uploads/2015/12/Elektor-proton-robot-programming-guide-V1$_1$ − *.pdf*