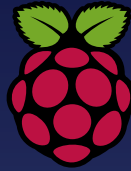


pi4ooREXX



—
—
—
—
—

An introduction to programming the Raspberry Pi
with ooRexx and BSF4ooRexx



TABLE OF CONTENTS

01

Introduction

02

Raspberry Pi

03

Required
Software

04

Examples

05

Conclusio





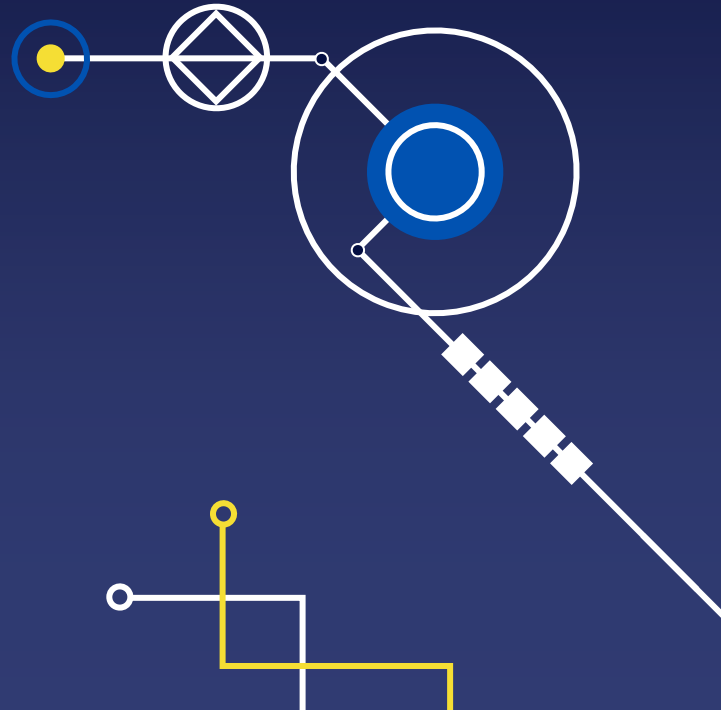
01

Introduction



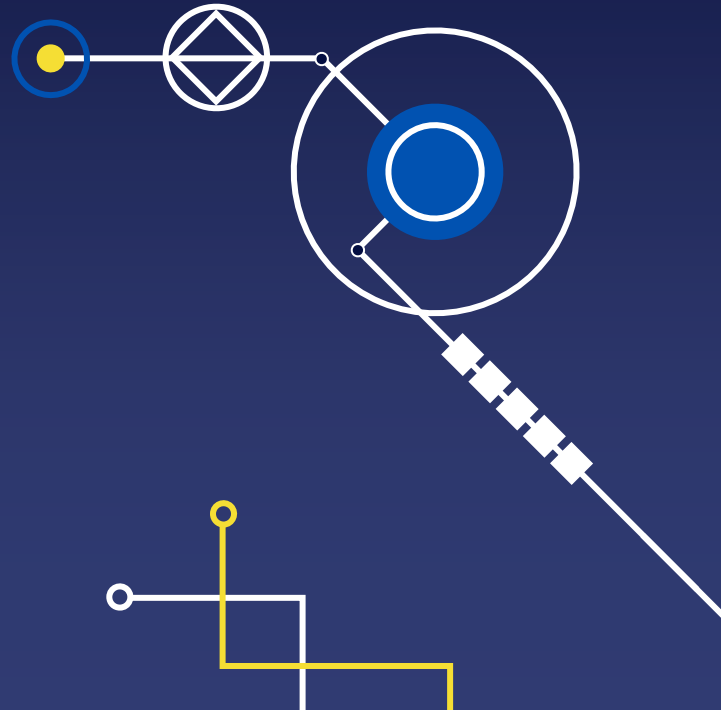
About me - Marcel Dür

- RPA Developer @ Wien Energie
- automate business processes
- Power Automate (Desktop)/ PowerApps
- Python, Java
- maybe soon ooRexx too?



Motivation

- Already experience with Raspberry Pi programming
- get to know ooRexx in BP 1 & 2
- 32nd Rexx Symposium
- combine ooRexx and Raspberry Pi

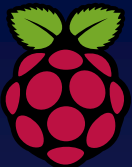




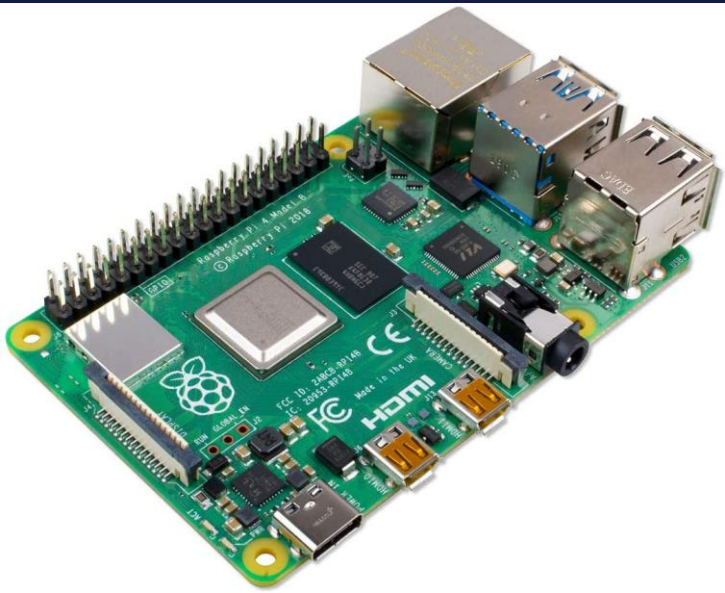
02

Raspberry Pi





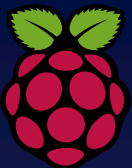
Raspberry Pi 1



Raspberry Pi 4B



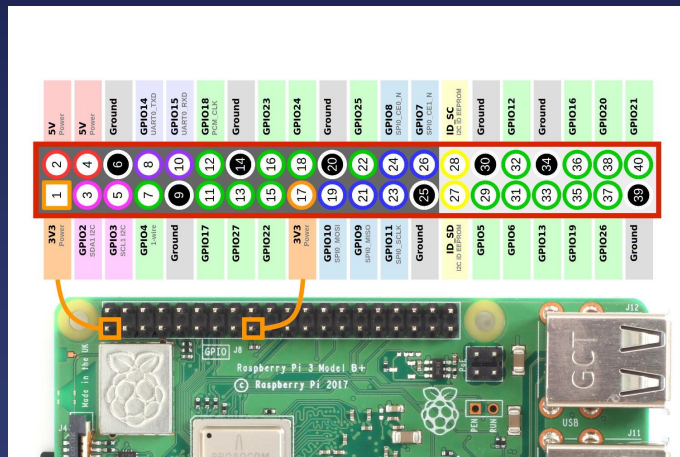
Raspberry Pi Zero 2WH



Raspberry Pi 2

GPIO Interface(Header):

- 1-Wire
- I²C
- SPI
- UART
- DI /DO
- PWM

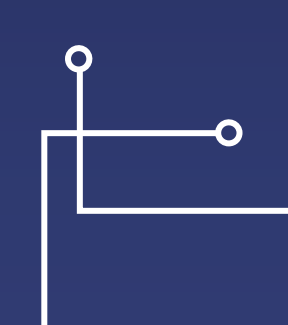


3V3 Power	1	2	5V Power
GPIO2 SDA1 I2C	3	4	5V Power
GPIO3 SCL1 I2C	5	6	Ground
GPIO4 1-wire	7	8	GPIO14 UART0_TXD
Ground	9	10	GPIO15 UART0_RXD
GPIO17	11	12	GPIO18 PCM_CLK
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3 Power	17	18	GPIO24
GPIO10 SPI0_MOSI	19	20	Ground
GPIO9 SPI0_MISO	21	22	GPIO25
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CEO_N
Ground	25	26	GPIO7 SPI0_CEO_N
ID SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPIO5	29	30	Ground
GPIO6	31	32	GPIO12
GPIO13	33	34	Ground
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
Ground	39	40	GPIO21





03



**Required
Software**



Required Software 1

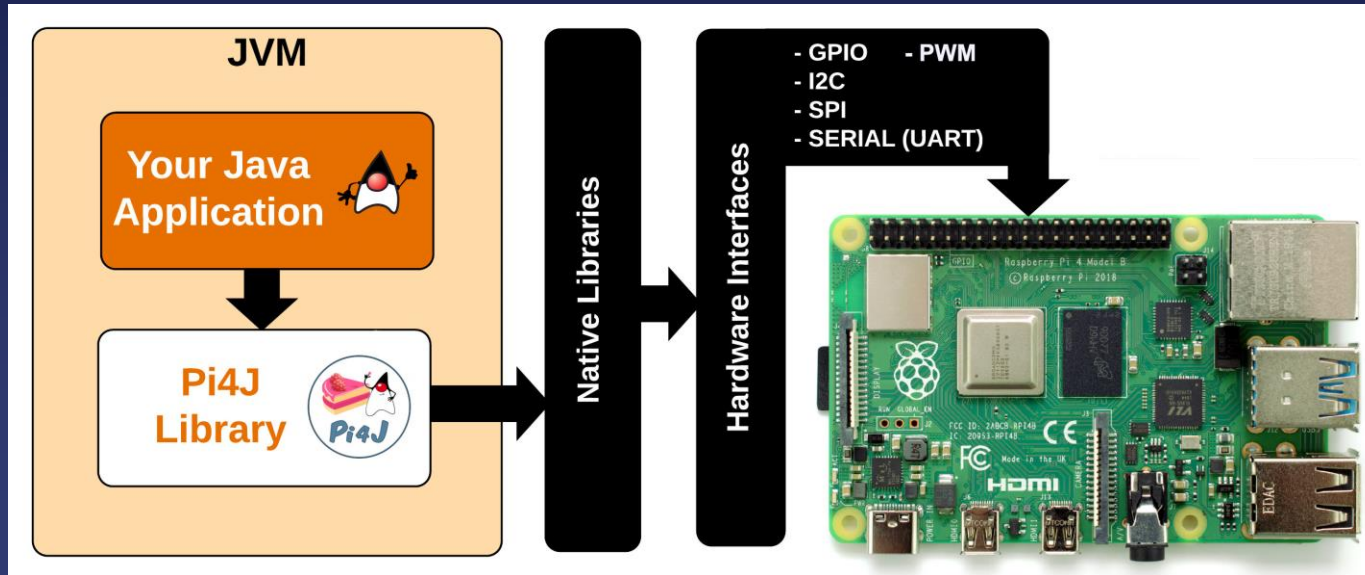


- Raspberry Pi OS
- Wiring Pi
- I2C Tools for Linux
- pi4oorex

Required Software 2



Pi4J:

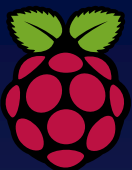




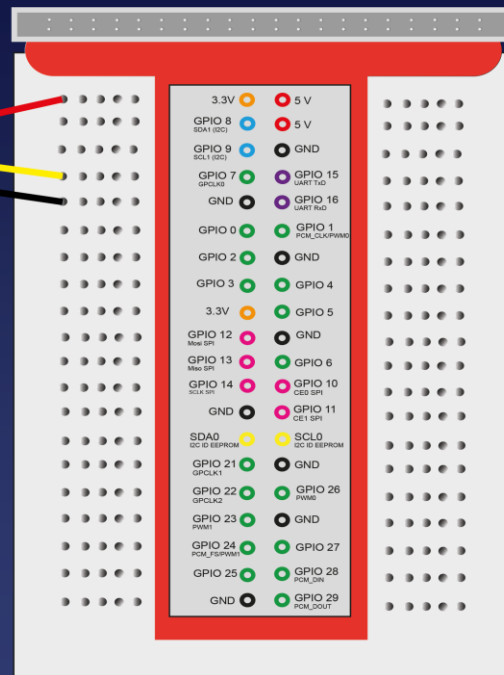
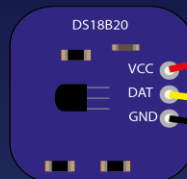
04

Examples

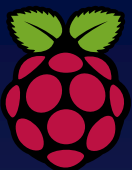




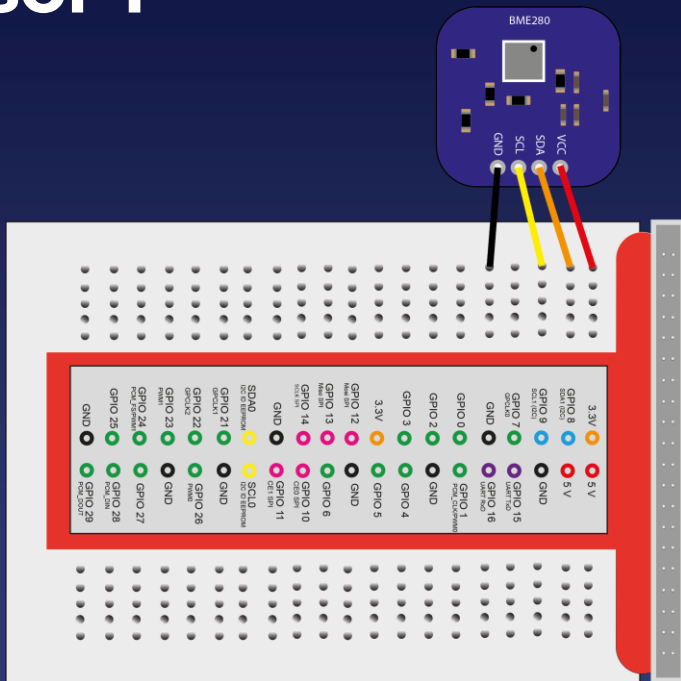
1-Wire DS18B20 Temperature Sensor

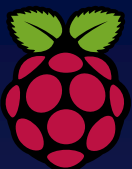


```
16 ::Routine ds18b20GetTemp public
17 Sensors = "ls /sys/bus/w1/devices"
18 sen=.array~new
19 address system sensors with output using (sen)
20 ds18b20 = sen[1]
21 stream=.stream~new("/sys/bus/w1/devices/"||ds18b20||"/driver/"
    ||ds18b20||"/temperature")~open
22 return stream~lineIn/1000
```



I²C BME 280 temp/hum/pres Sensor 1



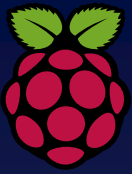


I²C BME 280 temp/hum/pres Sensor 2



```
pi@raspberrypi:~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20: 20 21 -- 23 24 -- -- 27 -- -- -- -- -- -- -- -- -- --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  -- 77
pi@raspberrypi:~ $ █
```

```
195 /*load all required classes and make the i2c connection available locally*/
196 ::routine setupBme public
197 pkgLocal=.context~package~local -- get package local directory
198 device = bsf.loadClass("com.pi4j.io.i2c.I2CDevice")
199 i2cbus = bsf.loadClass("com.pi4j.io.i2c.I2CBus")
200 bus = bsf.loadClass("com.pi4j.io.i2c.I2CFactory")~getInstance(i2cbus~BUS_1)
201 pkgLocal~device = bus~getDevice(119) --0x77 = hex -> int
202 return
203
```

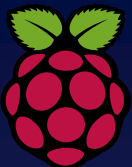
I²C BME 280 temp/hum/pres Sensor 3



```

48 -- Example
49
50 call_setupBme
51
52 say getTemp()
53 say getPressure()
54 say getHumidity()
55
56 exit 0
57
58 -----
59
60 /returns temp/
61
62 ::routine getTemp public
63 temp = calcValues()
64 return temp-Temp
65
66 -----
67
68 /return Pressure/
69
70 ::routine getPressure public
71 p = calcValues()
72 return p-pressure
73
74 -----
75
76 /return humidity/
77
78 ::routine getHumidity public
79 h = calcValues()
80 return h-humidity
81
82 -----
83
84 /read and calculates the values/
85
86 ::routine calcValues
87 -- read compensations parameter
88 data = bsf.createByteArray("byte.class",24)
89 .device-read(136,data,0,24)
90
91 -- humidity coefficients
92
93 -- convert Data sheet Table 16
94 -- Temperature coefficients
95
96 dig_T1 = (0x7f(data[1])) + ((0x7f(data[2]))*256)
97
98 dig_T2 = (0x7f(data[3])) + ((0x7f(data[4]))*256)
99 if dig_T2 > 32767 then dig_T2 = dig_T2 - 65536
100
101 dig_T3 = (0x7f(data[5])) + ((0x7f(data[6]))*256)
102 if dig_T3 > 32767 then dig_T3 = dig_T3 - 65536
103
104 -----
105 -- pressure coefficients
106
107 dig_P1 = (0x7f(data[7])) + ((0x7f(data[8]))*256)
108
109 dig_P2 = (0x7f(data[9])) + ((0x7f(data[10]))*256)
110 if dig_P2 > 32767 then dig_P2 = dig_P2 - 65536
111
112 dig_P3 = (0x7f(data[11])) + ((0x7f(data[12]))*256)
113 if dig_P3 > 32767 then dig_P3 = dig_P3 - 65536
114
115 dig_P4 = (0x7f(data[13])) + ((0x7f(data[14]))*256)
116 if dig_P4 > 32767 then dig_P4 = dig_P4 - 65536
117
118 dig_P5 = (0x7f(data[15])) + ((0x7f(data[16]))*256)
119 if dig_P5 > 32767 then dig_P5 = dig_P5 - 65536
120
121 dig_P6 = (0x7f(data[17])) + ((0x7f(data[18]))*256)
122 if dig_P6 > 32767 then dig_P6 = dig_P6 - 65536
123
124 dig_P7 = (0x7f(data[19])) + ((0x7f(data[20]))*256)
125 if dig_P7 > 32767 then dig_P7 = dig_P7 - 65536
126
127 dig_P8 = (0x7f(data[21])) + ((0x7f(data[22]))*256)
128 if dig_P8 > 32767 then dig_P8 = dig_P8 - 65536
129
130 dig_P9 = (0x7f(data[23])) + ((0x7f(data[24]))*256)
131 if dig_P9 > 32767 then dig_P9 = dig_P9 - 65536
132
133 -- Read dig_H1 from 0xA1 -> 161
134
135 data_H1 = bsf.createByteArray("byte.class",1)
136 .device-read(161,data_H1,0,1)
137
138 dig_H1 = (0x7f(data_H1[1]))
139
140 -- Read 7 Bytes from 0xA1 -> 225
141
142 data2 = bsf.createByteArray("byte.class",7)
143 .device-read(225,data2,0,7)
144
145 -- humidity coefficients
146
147 dig_H2 = (0x7f(data2[1])) + (data2[2]*256)
148 if dig_H2 > 32767 then dig_H2 = dig_H2 - 65536
149
150 dig_H3 = 0x7f(data2[3])
151
152 dig_H4 = ((0x7f(data2[4])*516) + (0x7f(data2[5])))
153 if dig_H4 > 32767 then dig_H4 = dig_H4 - 65536
154
155 dig_H5 = ((0x7f(data2[6])/16) + (0x7f(data2[6])/16))
156 if dig_H5 > 32767 then dig_H5 = dig_H5 - 65536
157
158 dig_H6 = (0x7f(data2[7]))
159 if dig_H6 > 127 then dig_H6 = dig_H6 - 256
160
161 -----
162
163 com1 = bsf.createByteArray("byte.class",1)
164 com1-put(box("byte.class",1),1) -- 0x01 = 1
165 .device-write(242,com1) --0xF2 = 242
166
167 --select control measurement register
168
169 com2 = bsf.createByteArray("byte.class",1)
170 com2-put(box("byte.class",39),1) --0x27 = 39
171 .device-write(246,com2) --0xF4 = 246
172
173 --select config register
174
175 com3 = bsf.createByteArray("byte.class",1)
176 com3-put(box("byte.class",90),1) -- 0xA0 -> 90 (byte) -->Java Byte
177 .device-write(242,com3) --0xF5 = 245
178
179 cell sysSleep 1 -- phase
180
181 --read measured data from 0xF7 -> 247 8 Byte
182
183 meas = bsf.createByteArray("byte.class",8)
184 .device-read(247,meas,0,8)
185
186 -- convert pressure and temp
187
188 adc_p = ((0x7f(meas[1])*65536) + (0x7f(meas[2])*256) + (0x7f(meas[3])/16)
189 adc_t = ((0x7f(meas[1])*65536) + (0x7f(meas[2])*256) + (0x7f(meas[3])/16)
190
191 -- convert humidity data
192
193 adc_h = (0x7f(meas[7])*256)+(0x7f(meas[8]))
194
195 --Temp offset calculation
196
197 var1 = ((adc_t / 16384) - (dig_T1 / 1024)) + dig_T2
198 var2 = ((adc_t / 131072) - (dig_T1 / 8192)) + ((adc_t / 131072) - (dig_T1 / 8192)) * dig_T3
199
200 t_line = var1 + var2
201 temp = (t_line) / 5120
202
203 -- pressure offset calculation
204
205 var3 = (t_line / 2) - 64000
206 var4 = var3 * var3 * dig_P0 / 32768
207 var4 = var4 + var3 * dig_P1 * 2
208 var4 = (var4 / 4) + (dig_P4 + 65536)
209 var3 = (dig_P3 + var3 * var3 / 524288 + dig_P2 + var3) / 524288
210 var3 = (1 + var3 / 32768) * dig_P1
211
212 p = 1046576 - adc_p
213 p = (p-(var4/4096))*16384 / var3
214 var3 = dig_P9 + p * p / 2147483648
215 var4 = p * dig_P8 / 32768
216
217 pressure = (p - (var3 + var4 * dig_P7) / 16 ) / 100
218
219 -----
220
221 -- humidity offset calculation
222
223 var_H = t_line - 76800
224 var_H = (adc_h - (dig_H2 + 64 + dig_H5/16384 + var_H))* (dig_H2 / 65536 + ( 1 + dig_H6 / 67108864 + var_H * ( 1 +
225 humidity = var_H * ( 1 - dig_H1 + var_H / 524288)
226
227 -----
228
229 DataColl = .directory-new
230 DataColl--temp = temp
231 DataColl--pressure = pressure
232 DataColl--humidity = humidity
233
234 return DataColl
235
236 -----
237
238 /!load all required classes and make the i2c connection available locally!/
239
240 ::routine setupBme public
241 pkgLocalContext-package-local -- get package local directory
242 device = bsf.loadClass("com.pi4j.io.i2c.I2CDevice")
243 i2cBus = bsf.loadClass("com.pi4j.io.i2c.I2Cbus")
244 bus = bsf.loadClass("com.pi4j.io.i2c.I2CFactory").getInstance(i2cBus-BUS_1)
245 pkgLocal-device = bus-getDevice(119) --0x77 = hex -> int
246 return
247
248 -----
249
250 /*: the 0x7f routine checks if the number is negative and if it
251 is the value is made positive with +256
252 -- V & 0xFF -- in Java */
253
254 ::routine 0x7f
255 use arg v
256
257 if v < 0 then
258 do
259 v = v+256
260 return v
261 end
262 else return v
263
264 -----
265
266 /*: the 0xf routine returns a value between 0-15.
267 V & 0xf in Java
268 */
269
270 ::routine 0xf
271 use arg v
272 return 0xf(v)//16
273
274 -----
275
276 ::requires bsf.cls
277

```

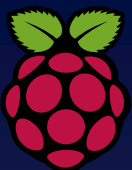


I²C BME 280 temp/hum/pres Sensor 4



```
52 data = bsf.createJavaArray("byte.class",24)
53 .device~read(136,data,0,24)
54
55 -- convert Data      Datasheet Table 16
56 -- temperature coeffocients
57
58 dig_T1 = (0xff(data[1])) + ((0xff(data[2]))*256)
59 |
60 dig_T2 = (0xff(data[3])) + ((0xff(data[4]))*256)
61 if dig_T2 > 32767 then dig_T2 = dig_T2 - 65536
62
63 dig_T3 = (0xff(data[5])) + ((0xff(data[6]))*256)
64 if dig_T3 > 32767 then dig_T3 = dig_T3 - 65536
65
66 -- pressure coefficients
67
68 dig_P1 = (0xff(data[7])) + ((0xff(data[8]))*256)
69
```

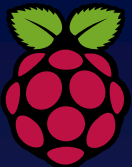
```
174 p = 1048576 - adc_p
175 p = (p-(var4/4096))*6250 / var3
176 var3 = dig_P9 * p * p / 2147483648
177 var4 = p * dig_P8 / 32768
178
179 pressure = (p+ (var3 + var4 + dig_P7) / 16 ) / 100
180
181 -- humidity offset calculation
182
183 var_H = t_fine - 76800
184 var_H = (adc_h - (dig_H4 * 64 + dig_H5/16384 * var_
185 humidity = var_H * ( 1 - dig_H1 * var_H / 524288)
186
187 DataColl = .directory~new
188 DataColl~~temp = temp
189 DataColl~~pressure = pressure
190 DataColl~~humidity = humidity
191
192 return DataColl
193
```



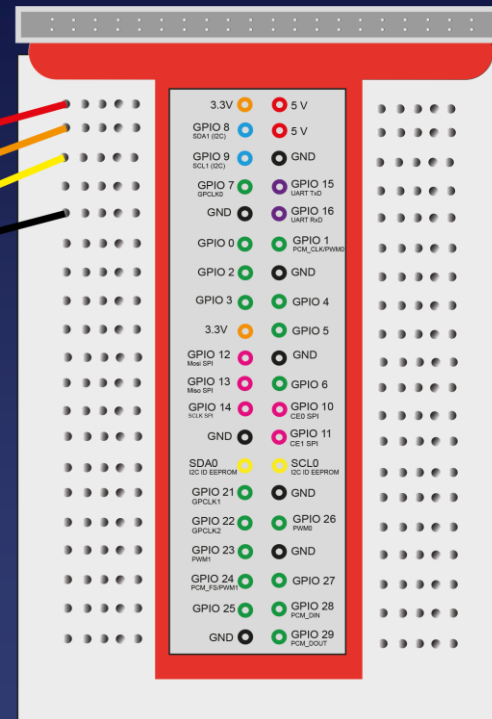
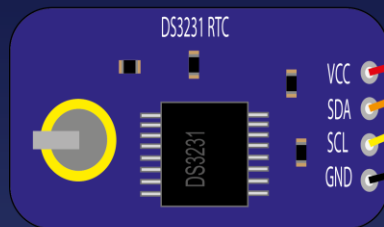
I²C BME 280 temp/hum/pres Sensor 5 with pi4oorex



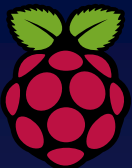
```
11 b = bsf.loadClass("at.pi4oorex.bme280.BME280") -- load requires Java Class
12 b~help -- use the help function to learn more about the available routines
13 b~measure -- start measurement
14 say b~getPressure -- get Pressure
15 say b~getTempCelcius -- get Temperature in degree Celcius
16 say b~getTempFahrenheit -- get Temperature in degree Fahrenheit
17 say b~getHumidity -- get Humidity in %
18 ::requires bsf.cls -- get Java Support
19 |
```



DS3231 Real Time Clock 1



```
79  /* get access to all required classes */
80  ::routine setupDs3231 public
81  pkglocal=.context~package~local -- get package local directory
82  device = bsf.loadClass("com.pi4j.io.i2c.I2CDevice")
83  i2cbus = bsf.loadClass("com.pi4j.io.i2c.I2CBus")
84  bus = bsf.loadClass("com.pi4j.io.i2c.I2CFactory")~getInstance(i2cbus~BUS_1)
85  pkglocal~device = bus~getDevice(104)
86
```

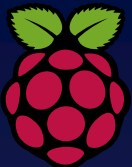


DS3231 Real Time Clock 2



```
59  /*routine to set the desired time */
60  ::routine setTime public
61  use arg hh,mm,ss
62  .device~write(0,BSFRawBytes(ss~x2c))
63  .device~write(1,BSFRawBytes(mm~x2c))
64  .device~write(2,BSFRawBytes(hh~x2c))
65  return
66
```

```
33  ::routine getTime public
34  read = bsf.CreateJavaArray("byte.class",1)
35  .device~read(0,read,0,1)
36  s= read[1]~d2x
37  .device~read(1,read,0,1)
38  m= read[1]~d2x
39  .device~read(2,read,0,1)
40  h= read[1]~d2x
41  time = h":"m":"s
42  return time
43
```

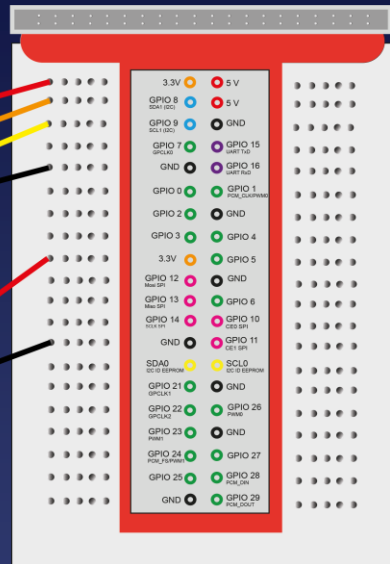
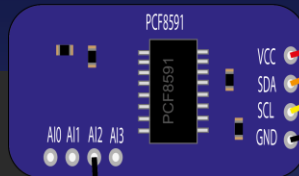


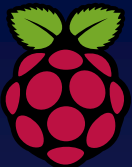
PCF8591 8-Bit ADC

```

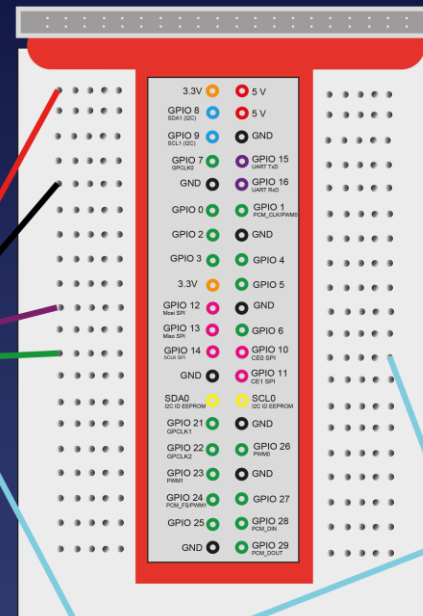
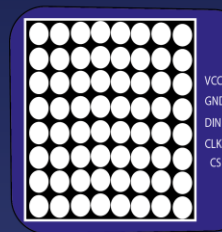
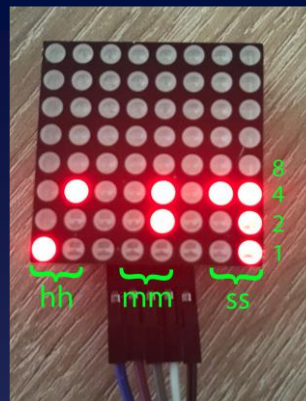
19  say getAnalogInput(2)
20  exit
21
22
23  ::routine getAnalogInput public
24  use arg input
25
26  if input = 0 then addr = "0x40"
27  else if input = 1 then addr = "0x41"
28  else if input = 2 then addr = "0x42"
29  else if input = 3 then addr = "0x43"
30  else say "falsche eingabe"
31
32  value = .array~new
33  address system "i2cget -y 1 0x48 " addr with output append using(value) -- dummy query
34  address system "i2cget -y 1 0x48 " addr with output append using(value)
35
36  return x2d(substr(value[2] ,3)) -- 0xf5 -> f5
37

```

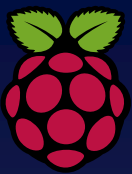




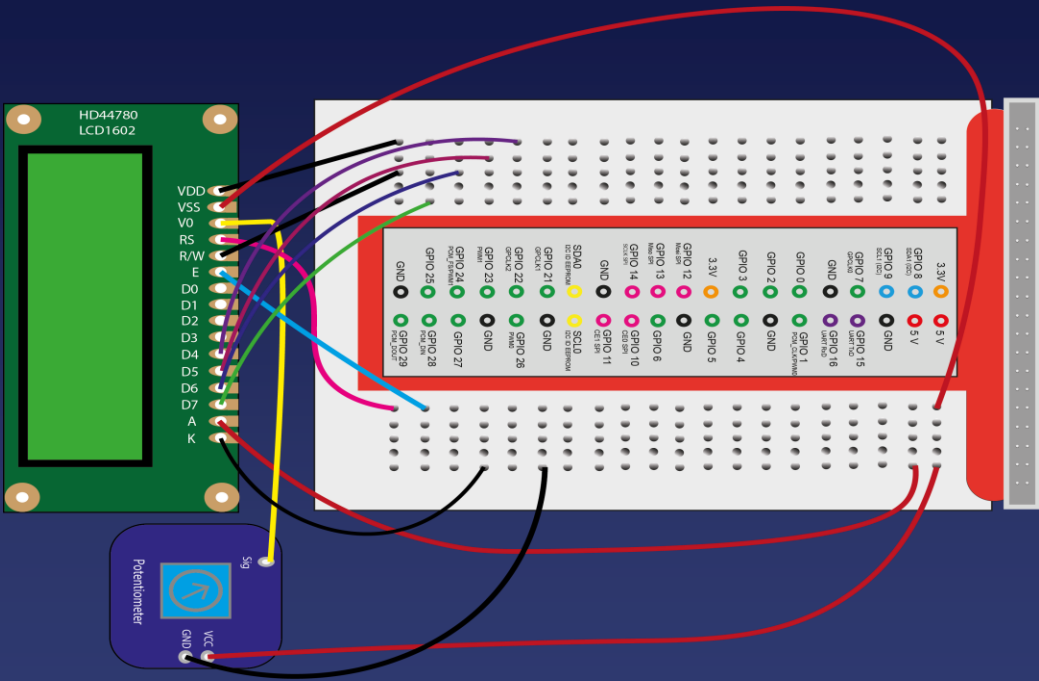
SPI Binary Clock



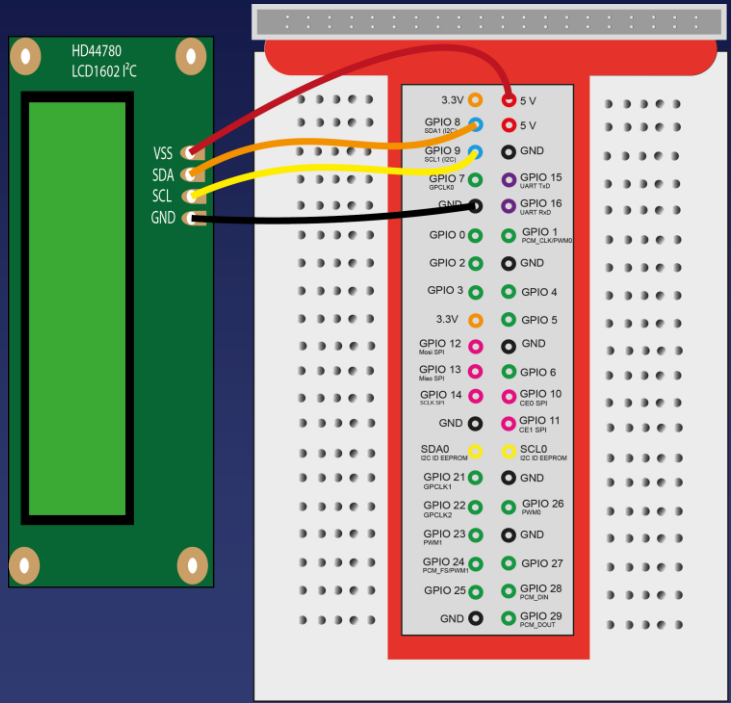
```
11 -- get SPI Support
12 call setupLED
13 call sysssleep 0.5
14 -- initialize the chip so that it can accept data
15 call init
16 call sysssleep 0.5
17 /* create an array with the available registers and
18  convert the numbers into characters*/
19 reg= .array~of(1~x2c,2~x2c,3~x2c,4~x2c,5~x2c,6~x2c,7~x2c,8~x2c)
20
21 do forever
22   time = TIME()           --get system Time
23   /*formats the read-in time for the output on the display
24   colons are replaced by zeros and the string is inverted*/
25   time = time~replaceAT("0",3)~replaceAT(":",6)~reverse
26   /* console output in the usual format*/
27   say time~reverse~replaceAT(":",3)~replaceAT(":",6)
28   /*output the current time on the display. Byte by byte*/
29   do i = 1 to 8
30     call write reg[i] , time[i]~d2c
31     call sysssleep 0.02
32   end
33   call sysssleep 0.1
34 end
35 exit
36
37 ::requires "LEDMatrixDriver.rex"  --load Driver fir LED Matrix Modul
```



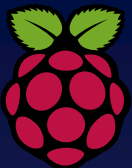
LC Display



parallel



I²C

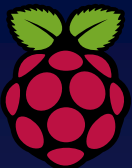


LC Display parallel

```
59 ::routine lcdByte
60 use arg bits, mode
61
62 /*Decide whether to write to the command or data register*/
63 if mode = .COMMANDREGISTER then do
64     .rs~LOW
65 end
66 else do
67     .rs~HIGH
68 end
69
70 /*Set data lines to low*/
71 .d4~LOW
72 .d5~LOW
73 .d6~LOW
74 .d7~LOW
75
76 /*Convert the passed character into an 8 bit string*/
77 bits = bits~c2x~x2b
78
79 /*create upper nibble*/
80 do i=1 to 4
81
82 if i = 1 then do
83     if bits[i] = 1 then .d7~high
84     end
85
86 else if i = 2 then do
87     if bits[i] = 1 then .d6~high
88     end
89
90 else if i = 3 then do
91     if bits[i] = 1 then .d5~high
92     end
93
94 else
95     if bits[i] = 1 then .d4~high
96 end
97
98 /*Write data to the register*/
99 call sysleep 0.001
100 .e~high
101 call sysleep 0.001
102 .e~low
```

```
171 ::Routine LCDprint public
172 use arg zeile, String
173
174 if zeile = 1 then call lcdByte .LCD_ROW_1 , .COMMANDREGISTER
175 else if zeile = 2 then call lcdByte .LCD_ROW_2 , .COMMANDREGISTER
176 else if zeile = 3 then call lcdByte .LCD_ROW_3 , .COMMANDREGISTER
177 else call lcdByte .LCD_ROW_4 , .COMMANDREGISTER
178 string = substr(string,1,16)
179 do j=1 to string~length
180     call lcdByte String[j] , .DATAREGISTER
181 end
182 return
```

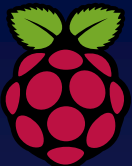
```
31 call setup --load all required connections
32 call init --Initialize display
33
34 say "Clock is running" --outputs time and date
35 do forever
36     call lcdprint 1 ,TIME()
37     call lcdprint 2 , DATE()
38 end
39 exit
40 ::requires BSF.CLS --Get Java Support
41
```



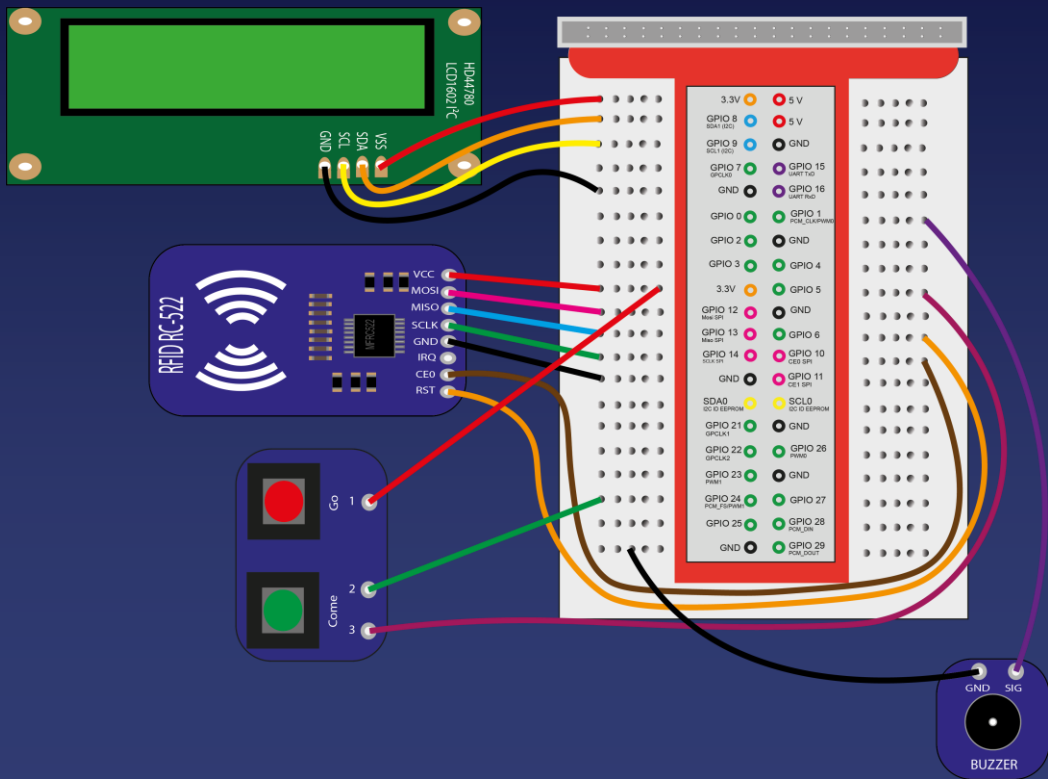
LC Display I²C 2

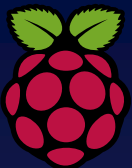
```
110 ::routine write
111   use arg byte, m
112
113   if m = 1 then do           --Datamode
114     mode   = 1101
115     mode_  = 1001
116   end
117   else do                   --Commandmode
118     mode   = 1100
119     mode_  = 1000
120   end
121
122   -- create upper Nibble
123
124   upper_Nibble = substr(byte,1,4)
125   by_un_en     = toByte((upper_Nibble||mode)~b2x~x2d)
126   by_un_en_    = toByte((upper_nibble||mode_)~b2x~x2d)
127
128   -- create lower Nibble
129   lower_Nibble = substr(byte,5,4)
130
131   by_ln_en     = toByte((lower_Nibble||mode)~b2x~x2d)
132   by_ln_en_    = toByte((lower_nibble||mode_)~b2x~x2d)
133
134   --put the bytes into a Java Byte array
135
136   out=bsf.createJavaArray("byte.class", 4)
137   out~put(by_un_en,1)
138   out~put(by_un_en_,2)
139   out~put(by_ln_en,3)
140   out~put(by_ln_en_,4)
141
142   /*write byte by byte to the controller*/
143   do i= 1 to 4
144     .device~write(out[i])
145   end
146   return
```

```
171 ::routine setup public
172   pkgLocal=.context~package~local -- get package local directory
173   /*Load required classes*/
174   i2cbus = bsf.loadClass("com.pi4j.io.i2c.I2CBus")
175   /*get Instance of I2CFactory on I2C Bus 1 */
176   bus = bsf.loadClass("com.pi4j.io.i2c.I2CFactory")~getInstance(i2cbus~BUS_1)
177   /*get Device on I2C Address 0x27 -> 39 decimal*/
178   pkgLocal~device = bus~getDevice(39)
179   return
```



RFID Attendance System 1





RFID Attendance System 2

```
⚙️::routine setup
  pkgLocal=.context~package~local -- get package local directory
  --- initialzie MFRC522
  rc522clientimpl = bsf.loadClass("at.pi4oorexx.mfrc522.rc522.RC522ClientImpl")
  card = bsf.loadClass("at.pi4oorexx.mfrc522.model.card.Card")
  pkgLocal~rc522client = rc522clientimpl~createInstance
  --- initialize Button and Buzzer

  gpio = bsf.loadClass("com.pi4j.io.gpio.GpioFactory")~getInstance
  RaspiPin = bsf.loadClass("com.pi4j.io.gpio.RaspiPin")
  PinPullDown = bsf.loadClass("com.pi4j.io.gpio.PinPullResistance")~PULL_DOWN
  pkgLocal~pinCome = gpio~provisionDigitalInputPin(RaspiPin~GPIO_05,PinPullDown)
  pkgLocal~pinGo = gpio~provisionDigitalInputPin(RaspiPin~GPIO_24,PinPullDown)
  pinstate = bsf.loadClass("com.pi4j.io.gpio.PinState")
  pkgLocal~pinBuzzer = gpio~provisionDigitalOutputPin(RaspiPin~GPIO_01,pinstate~high)

  --- initialize LC- Display

  device = bsf.loadClass("com.pi4j.io.i2c.I2CDevice")
  lcd = bsf.import("at.pi4oorexx.lcd.I2CLCD")
  i2cbus = bsf.loadClass("com.pi4j.io.i2c.I2CBus")
  bus = bsf.loadClass("com.pi4j.io.i2c.I2CFactory")~getInstance(i2cbus~BUS_1)
  device = bus~getDevice(box('int',39)) --0x27 = 39 hex -> int
  pkgLocal~screen =lcd~new(device)
  return
```

```
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

do while emp~lines<> 0
  data = emp~linein
  parse var data tag " ," name " ," perso
  if tag = tagID then do
    --say tagid passt
    file =.stream~new("attendance.csv")
    file~open("write")
    if cg = "come" then do
      file~lineout(date() " ," Time() " ," name " ," COME)
      say come
      .screen~clear
      .screen~display_string_pos("card detected" ,1,0)
      .screen~display_string_pos(("Hi" name) ,2,0)
    end
  else do
    file~lineout(date() " ," Time() " ," name " ," GO)
    say go
    .screen~clear
    .screen~display_string_pos("card detected" ,1,0)
    .screen~display_string_pos("Bye" name,2,0)
  end
  file~close
  call beep 1
  call sysleep (2) ---> damit man Ausgabe von "Hallo" lesen kann
  i=10 --> damit abfrage beendet wird
  readOK = 1
end
end
```



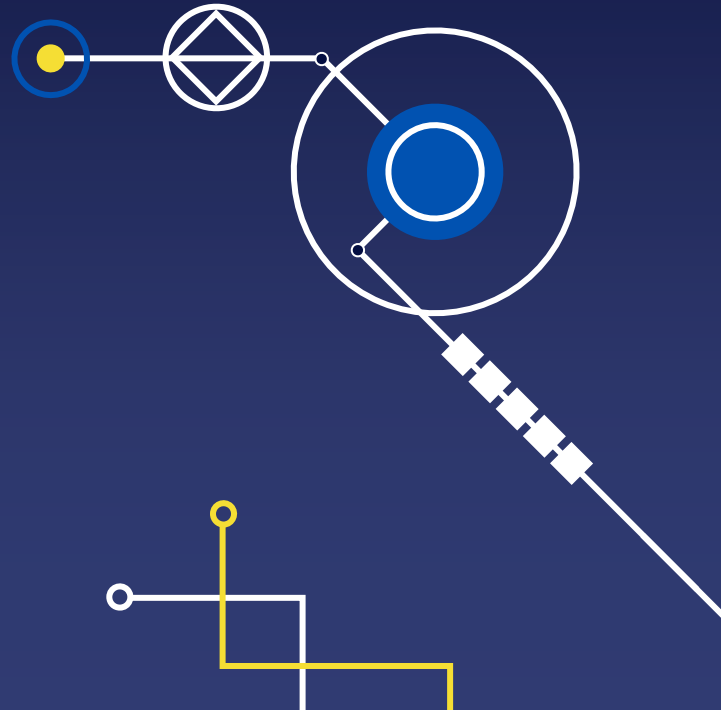
05

Conclusio



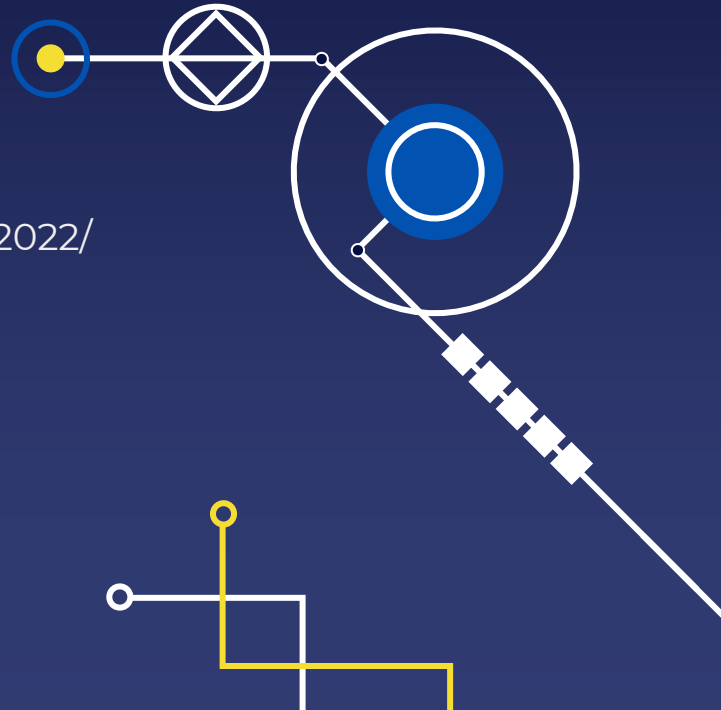
Conclusio:

- The programming is simplified clearly and can be used so by a beginner.
- Goal: to do without Pi4j



Links:

- <https://github.com/pi4oorexx/pi4oorexx>
- https://wi.wu.ac.at/rgf/diplomarbeiten/BakkStuff/2022/202204_Duer_pi4oorexx.pdf



THANKS!

Do you have any questions?

Marcel.duer@gmail.com

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik

Please keep this slide for attribution

