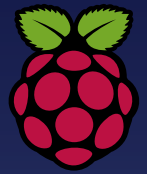
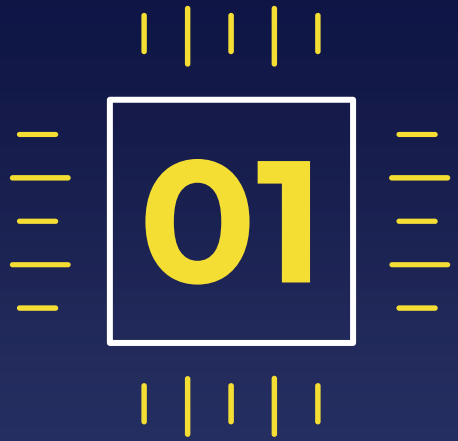
pi4oorexx



An introduction to programming the Raspberry Pi
with ooRexx and BSF4ooRexx

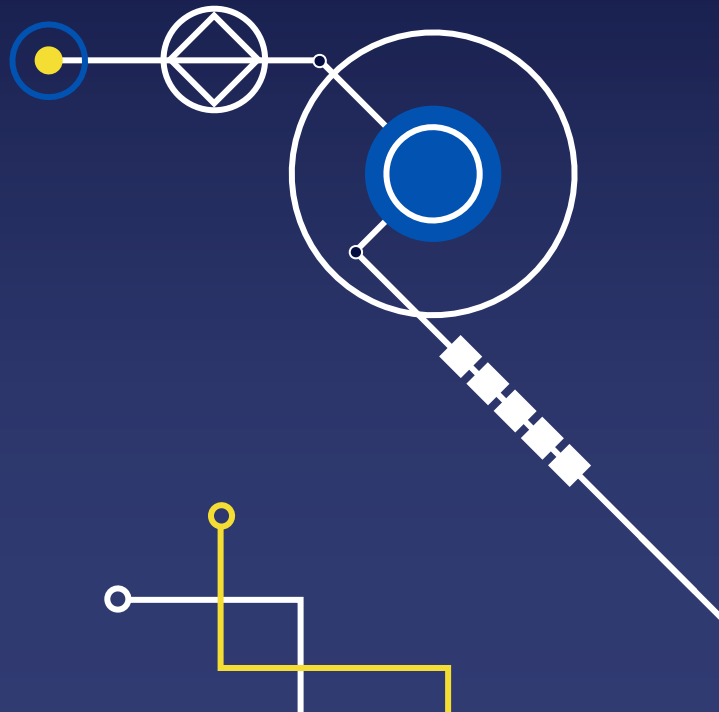
TABLE OF CONTENTS



Introduction

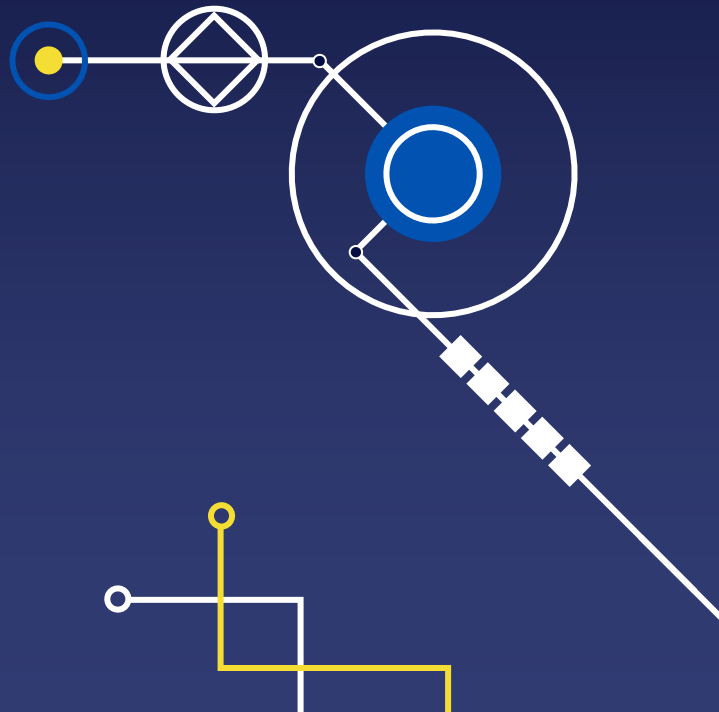
About me - Marcel Dür

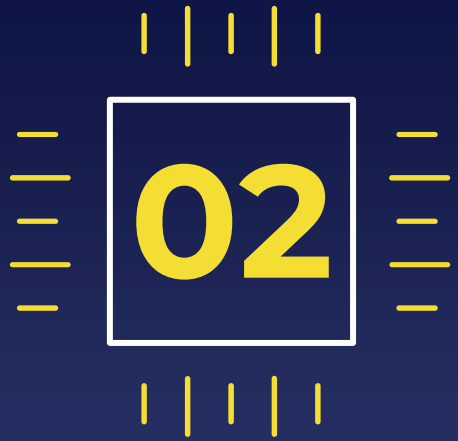
- RPA Developer @ Wien Energie
- automate business processes
- Power Automate (Desktop)/ PowerApps
- Python, Java
- maybe soon ooRexx too?



Motivation

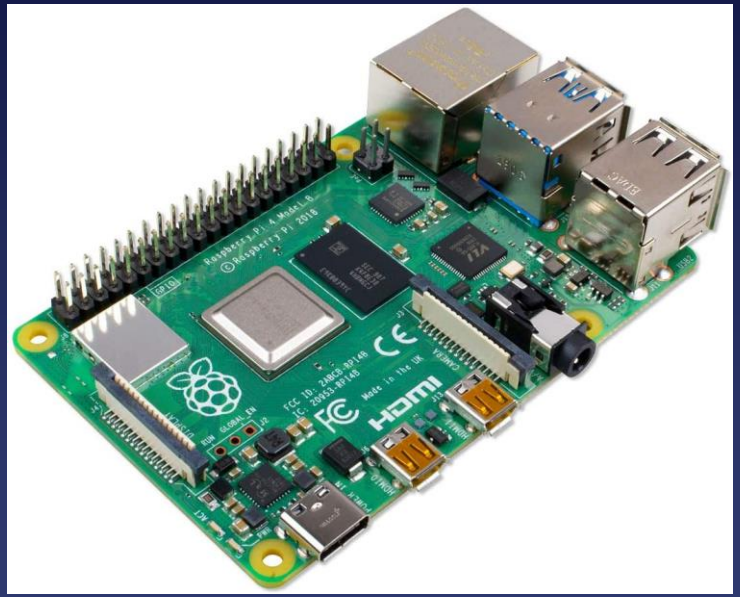
- Already experience with Raspberry Pi programming
- get to know ooRexx in BP 1 & 2
- 32nd Rexx Symposium
- combine ooRexx and Raspberry Pi





Raspberry Pi

Raspberry Pi 1



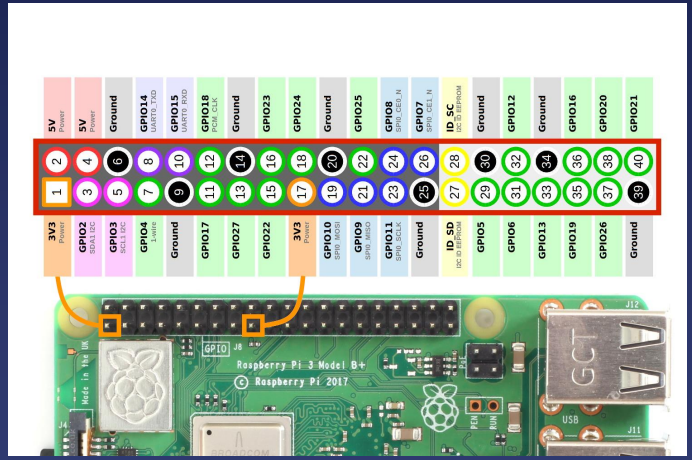
Raspberry Pi 4B

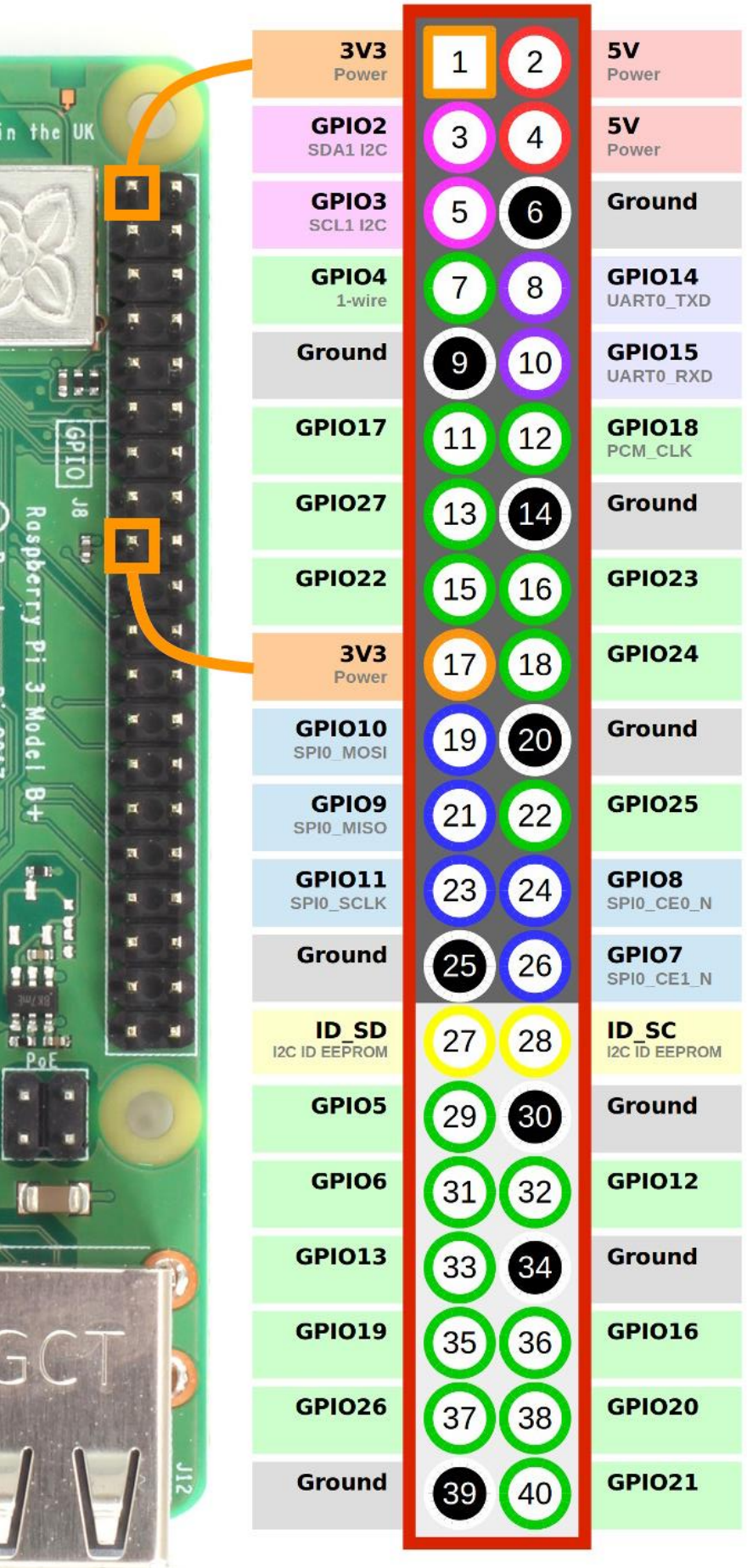
Raspberry Pi Zero 2WH

Raspberry Pi 2

GPIO Interface(Header):

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

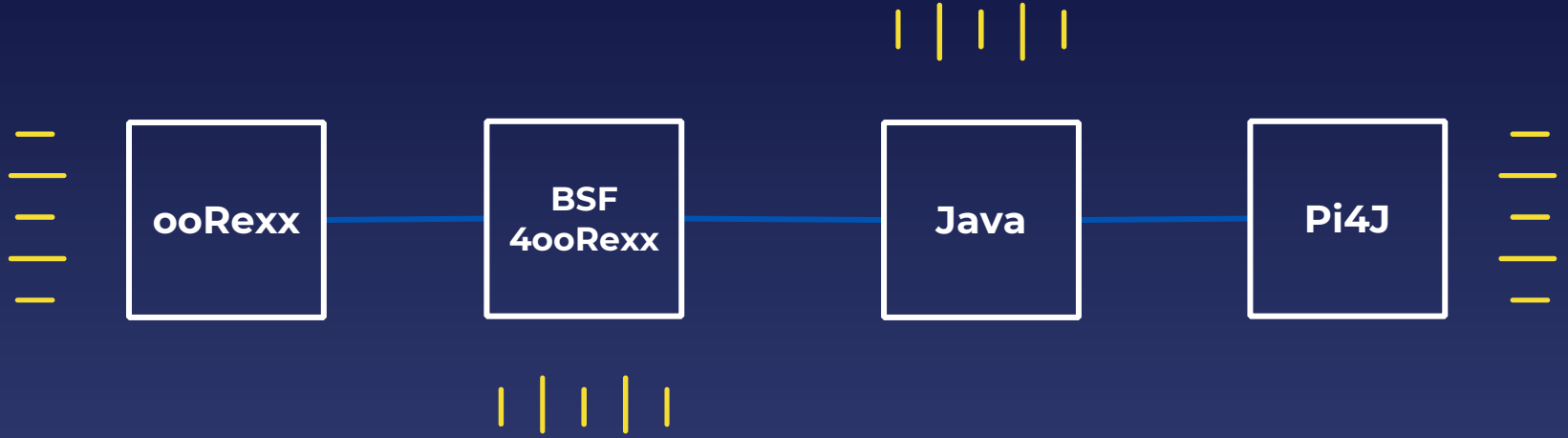



3V3 Power
1
2
5V Power
GPIO2 SDA1 I2C
3
4
5V Power
GPIO3 SCL1 I2C
5
6
Ground
GPIO4 1-wire
7
8
GPIO14 UART0_TXD
Ground
9
10
GPIO15 UART0_RXD
GPIO17
11
12
GPIO18 PCM_CLK
GPIO27
13
14
Ground
GPIO22
15
16
GPIO23
3V3 Power
17
18
GPIO24
GPIO10 SPI0_MOSI
19
20
Ground
GPIO9 SPI0_MISO
21
22
GPIO25
GPIO11 SPI0_SCLK
23
24
GPIO8 SPI0_CE0_N
Ground
25
26
GPIO7 SPI0_CE1_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

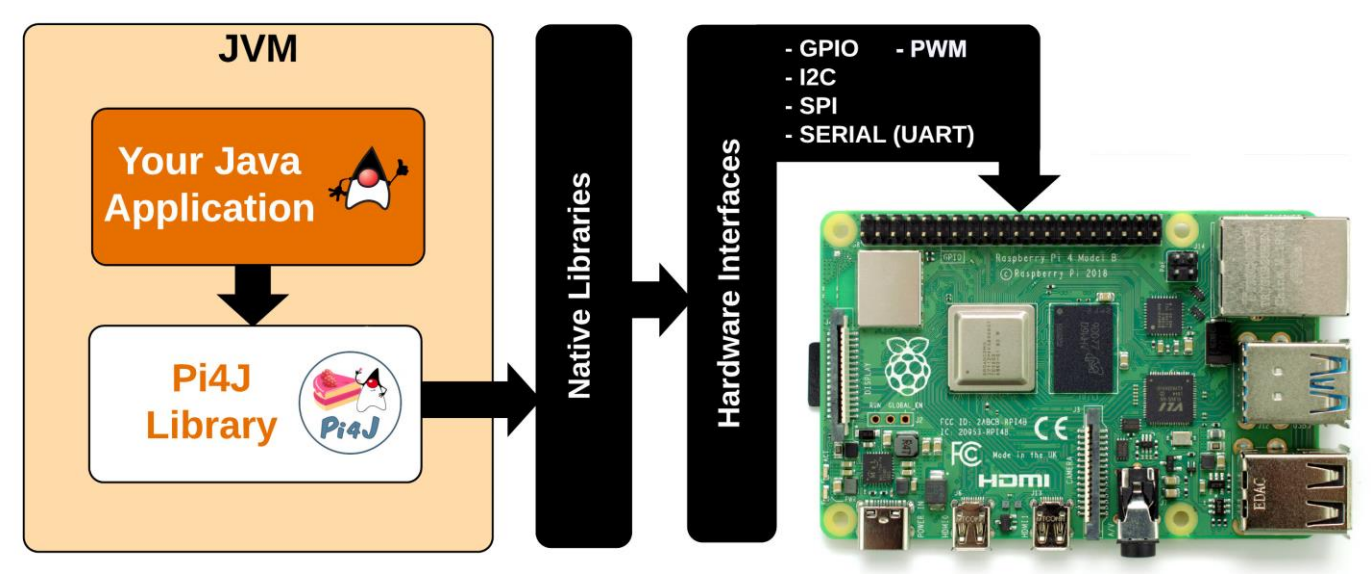
Required Software 1



- Raspberry Pi OS
- Wiring Pi
- I2C Tools for Linux
- pi4oorexx

Required Software 2

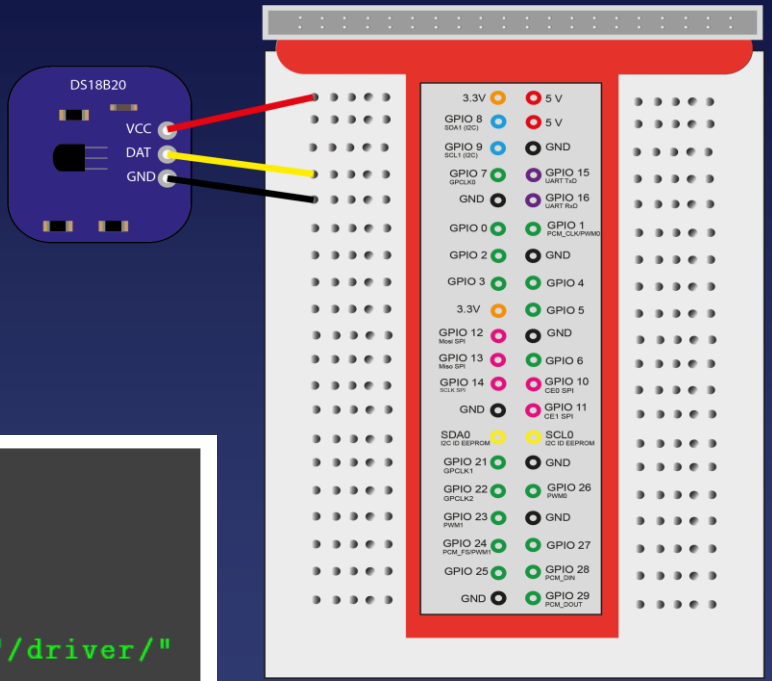
Pi4J:



04

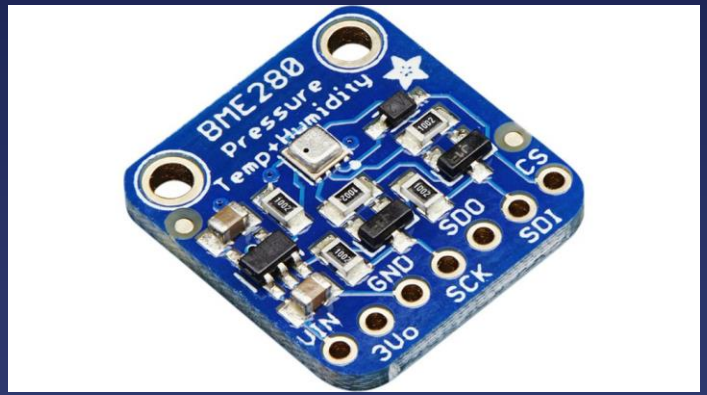
Examples

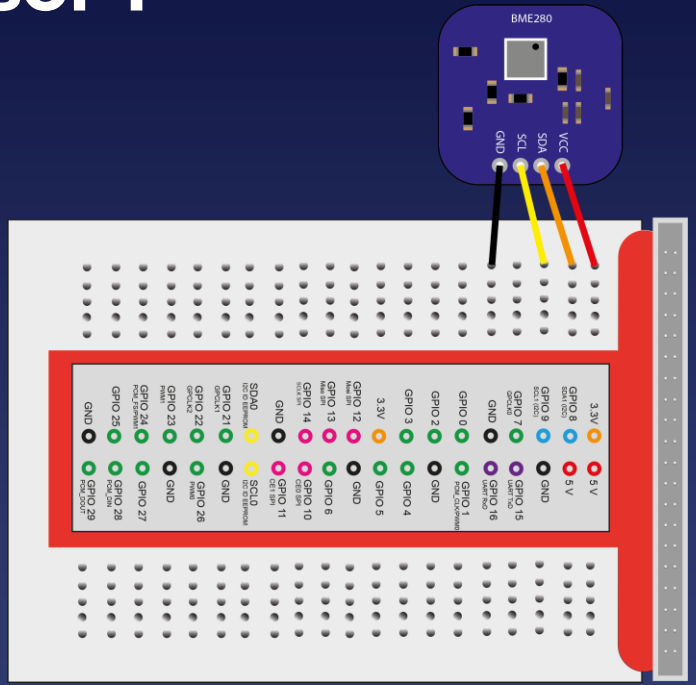
1-Wire DS18B20 Temperature Sensor




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

I²C BME 280 temp/hum/pres Sensor 1



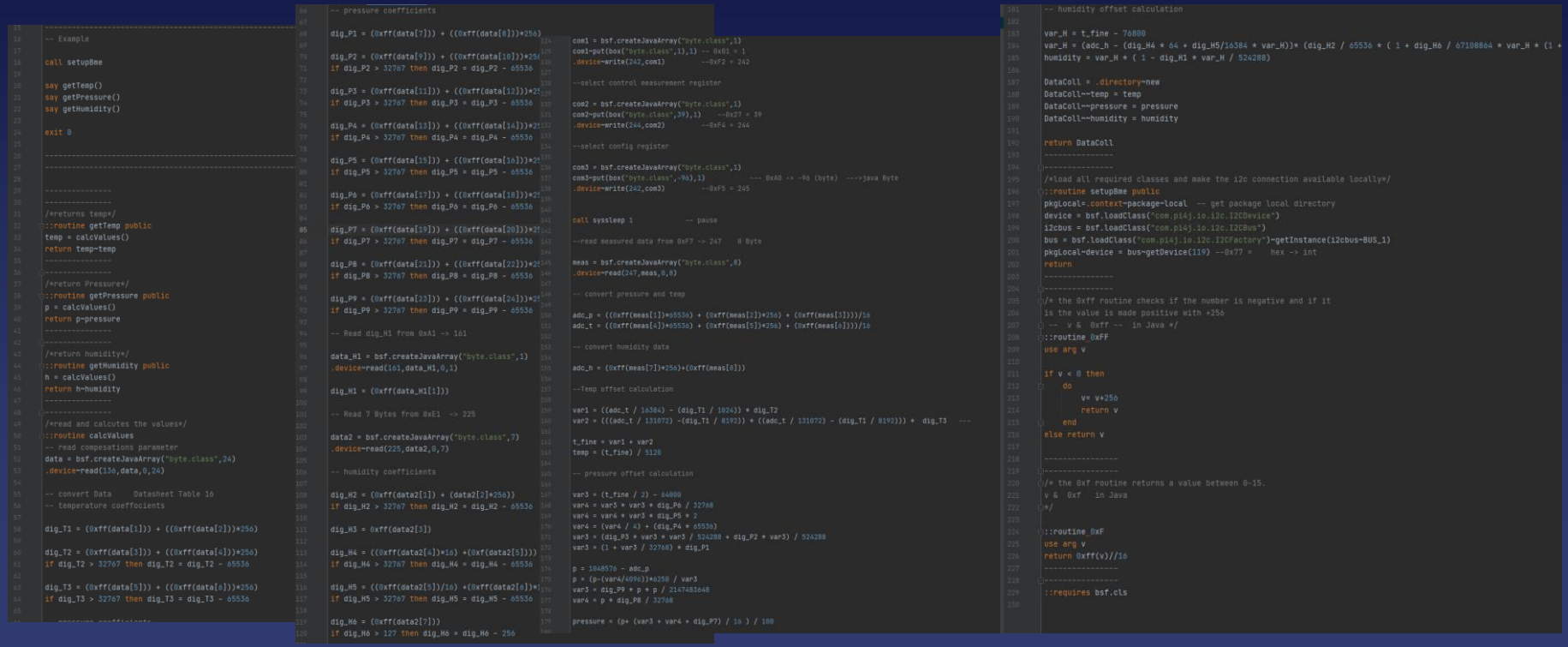


I²C BME 280 temp/hum/pres Sensor 2

```
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
```

I²C BME 280
temp/hum/pres Sensor 3



I²C BME 280 temp/hum/pres Sensor 4

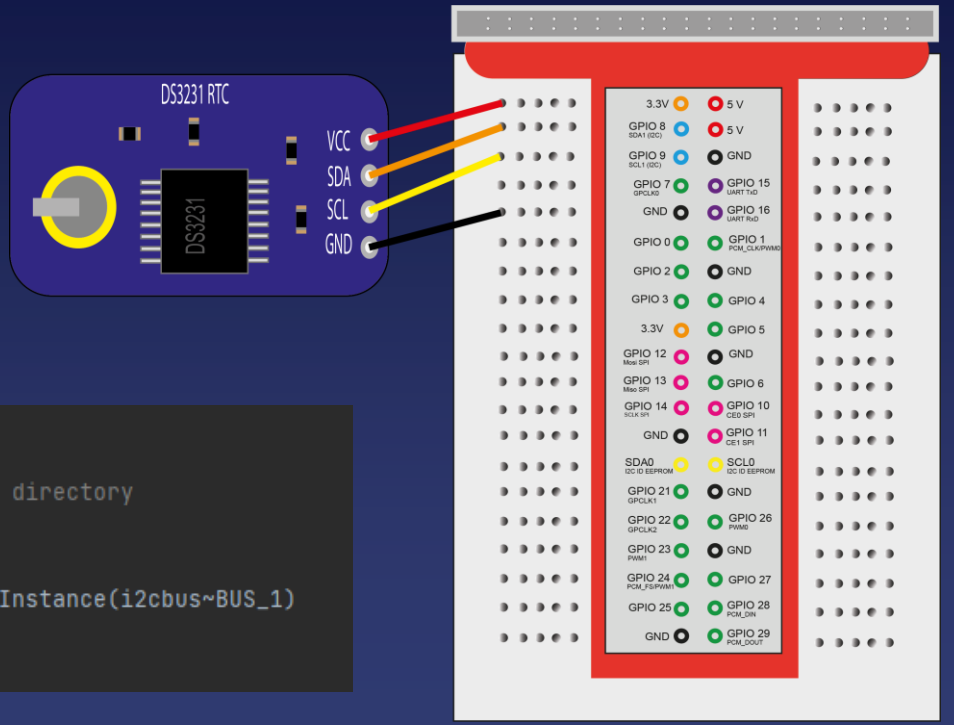
```
52  data = bsf.createJavaArray("byte.class",24)
53  .device~read(136,data,0,24)
54
55  -- convert Data     Datasheet Table 16
56  -- temperature coeffocients
57
58  dig_T1 = (0xff(data[1])) + ((0xff(data[2]))*256)
59
60  dig_T2 = (0xff(data[3])) + ((0xff(data[4]))*256)
61  if dig_T2 > 32767 then 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
```

I²C BME 280 temp/hum/pres Sensor 5 with pi4oorexx

```
11  b = bsf.loadClass("at.pi4oorexx.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
```

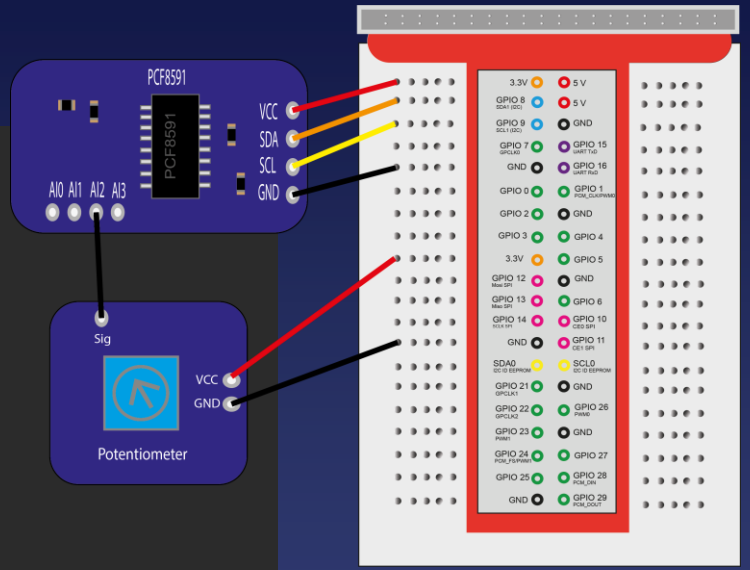
DS3231 Real Time Clock 2

```
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
```

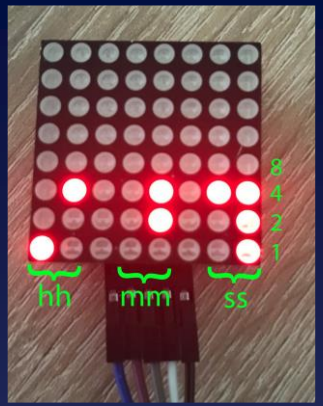
PCF8591 8-Bit ADC

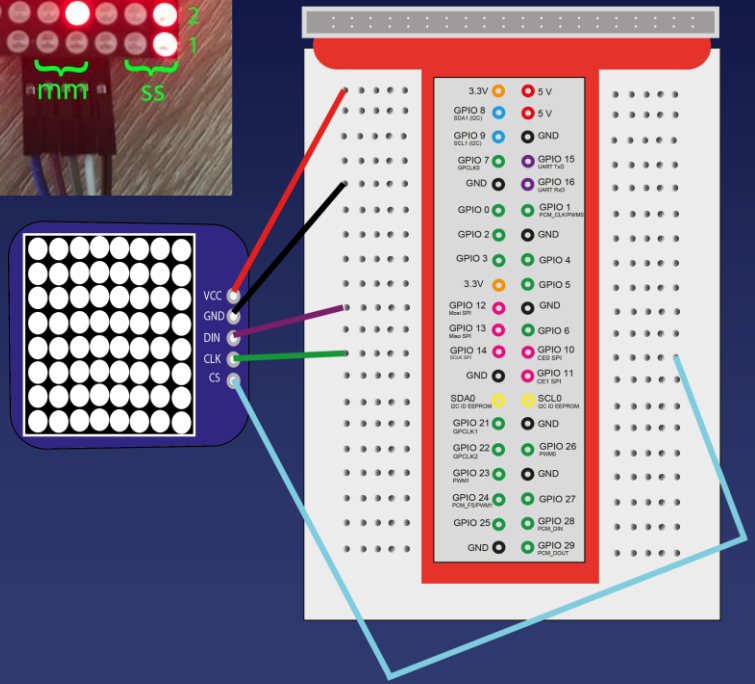
```
19  say getAnalogInput(2)
20  exit
21
22
23  ::routine getAnalogInput public
24  use arg input
25
26  if input = 0 then addr = "0x40"
27  else if input = 1 then addr = "0x41"
28  else if input = 2 then addr = "0x42"
29  else if input = 3 then addr = "0x43"
30  else say "falsche eingabe"
31
32  value = .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
```



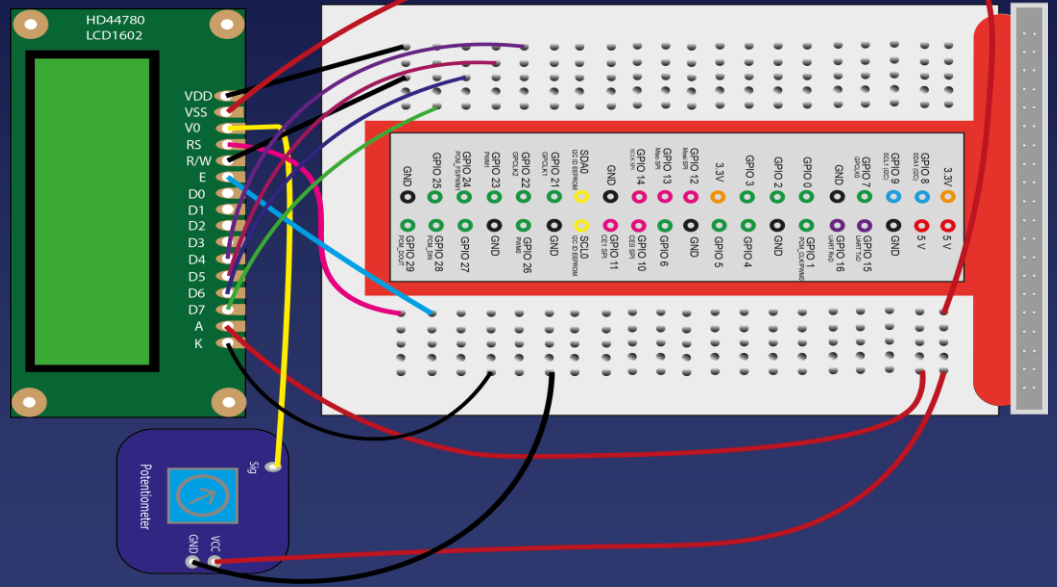
SPI Binary Clock

```
-- get SPI Support
call setupLED
call syssleep 0.5
-- initialize the chip so that it can accept data
call init
call syssleep 0.5
/* create an array with the available registers and
 convert the numbers into characters*/
reg= .array~of(1~x2c,2~x2c,3~x2c,4~x2c,5~x2c,6~x2c,7~x2c,8~x2c)

do forever
    time = TIME()           --get system Time
    /*formats the read-in time for the output on the display
    colons are replaced by zeros and the string is inverted*/
    time = time~replaceAT("0",3)~replaceAT("0",6)~reverse
    /* console output in the usual format*/
    say time~reverse~replaceAT(":",3)~replaceAT(":",6)
    /*output the current time on the display. Byte by byte*/
    do i = 1 to 8
        call write reg[i] , time[i]~d2c
        call syssleep 0.02
    end
    call syssleep 0.1
end
exit

::requires "LEDMatrixDriver.rex"   --load Driver fir LED Matrix Modul
```

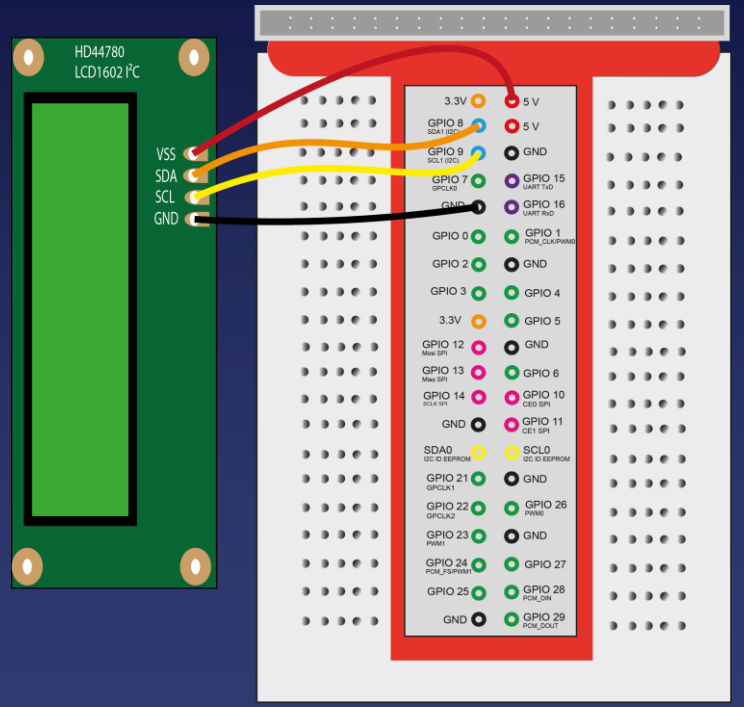




LC Display



parallel



I²C

LC Display parallel

```
::routine lcdByte
use arg bits, mode

/*Decide whether to write to the command or data register*/
if mode = .COMMANDREGISTER then do
    .rs~LOW
end
else do
    .rs~HIGH
end

/*Set data lines to low*/
.d4~LOW
.d5~LOW
.d6~LOW
.d7~LOW

/*Convert the passed character into an 8 bit string*/
bits = bits~c2x~x2b

/*create upper nibble*/
do i=1 to 4

if i = 1 then do
        if bits[i] = 1 then .d7~high
    end

else if i= 2 then do
        if bits[i] = 1 then .d6~high
    end
else if i = 3 then do
        if bits[i] = 1 then .d5~high
    end
 else
        if bits[i] = 1 then .d4~high
end

/*Write data to the register*/
call syssleep 0.001
.e~high
call syssleep 0.001
.e~low
```

```
::Routine LCDprint public
use arg zeile, String

if zeile = 1 then call lcdByte .LCD_ROW_1 , .COMMANDREGISTER
else if zeile = 2 then call lcdByte .LCD_ROW_2 , .COMMANDREGISTER
else if zeile = 3 then call lcdByte .LCD_ROW_3 , .COMMANDREGISTER
else call lcdByte .LCD_ROW_4 , .COMMANDREGISTER
string = substr(string,1,16)
do j=1 to string~length
    call lcdByte String[j] , .DATAREGISTER
end
return
```

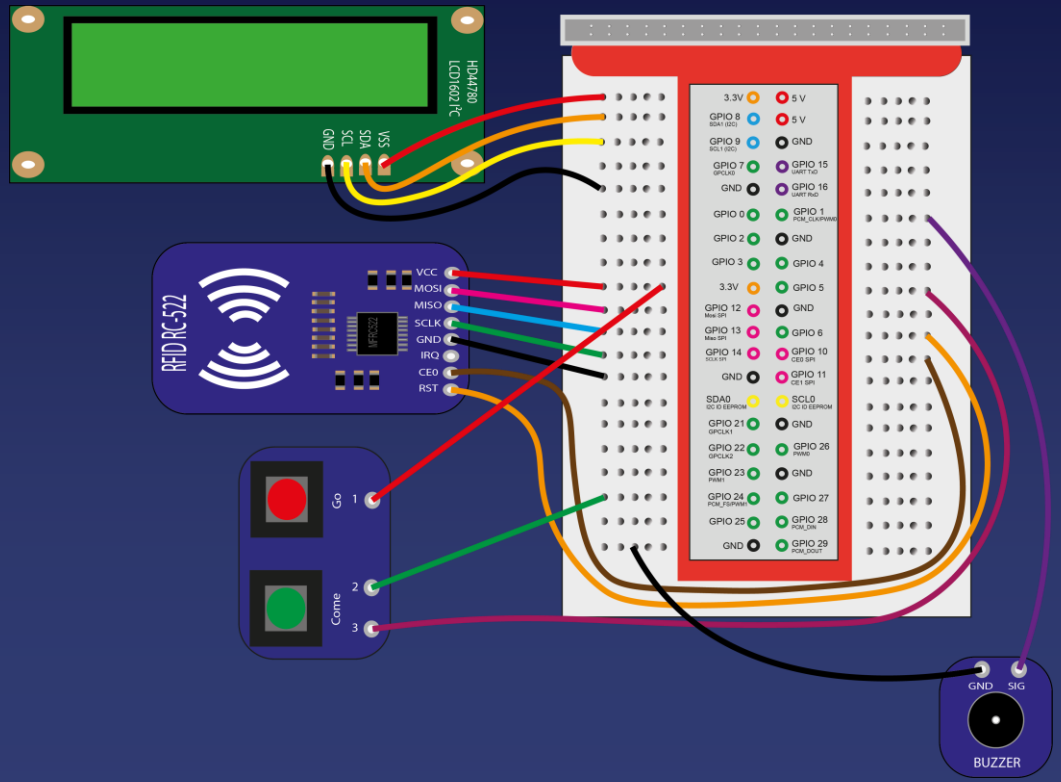
```
call setup              --load all required connections
call init               --Initialize display

say "Clock is running"              --outputs time and date
do forever
    call lcdprint 1 ,TIME()
    call lcdprint 2 , DATE()
end
exit
::requires BSF.CLS  --Get Java Support
```

LC Display I²C 2

```
::routine write
use arg byte, m

if m = 1 then do                --Datamode
    mode    = 1101
    mode_   = 1001
end
else do                         --Commandmode
    mode    = 1100
    mode_   = 1000
end

-- create upper Nibble

upper_Nibble = substr(byte,1,4)
by_un_en     =   toByte((upper_Nibble||mode)~b2x~x2d)
by_un_en_    =   toByte((upper_nibble||mode_)~b2x~x2d)

-- create lower Nibble
lower_Nibble = substr(byte,5,4)

by_ln_en     =   toByte((lower_Nibble||mode)~b2x~x2d)
by_ln_en_    =   toByte((lower_nibble||mode_)~b2x~x2d)

--put the bytes into a Java Byte array

out=bsf.createJavaArray("byte.class", 4)
out~put(by_un_en,1)
out~put(by_un_en_,2)
out~put(by_ln_en,3)
out~put(by_ln_en_,4)

/*write byte by byte to the controller*/
do i= 1 to 4
    .device~write(out[i])
end
return
```

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

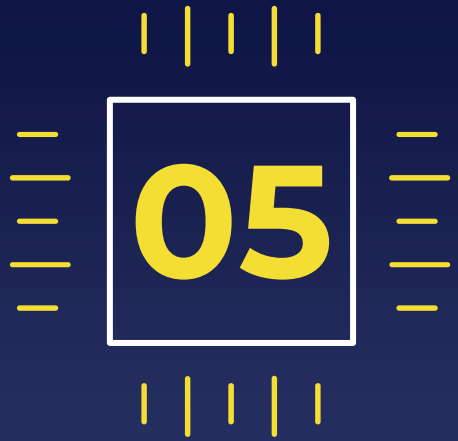
RFID Attendance System 1



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
```

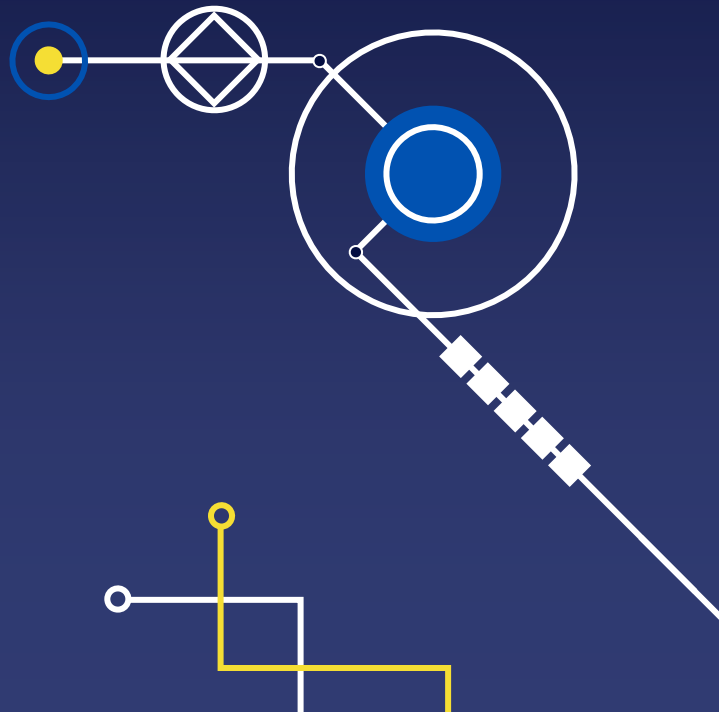
```
47          do while emp~lines<> 0
48              data = emp~linein
49              parse var data tag "," name "," perso
50              if tag = tagID then do
51                 --say tagid passt
52                 file =.stream~new("attendance.csv")
53                 file~open("write")
54                 if cg = "come" then do
55                 file~lineout(date() ", "Time() "," name "," COME)
56                  say come
57                      .screen~clear
58                      .screen~display_string_pos("card detected" ,1,0)
59                      .screen~display_string_pos(("Hi" name) ,2,0)
60                 end
61                 else do
62                 file~lineout(date() ", "Time() "," name "," GO)
63                     say  go
64                      .screen~clear
65                      .screen~display_string_pos("card detected" ,1,0)
66                      .screen~display_string_pos("Bye" name,2,0)
67                 end
68                 file~close
69                 call beep 1
70                 call syssleep (2)   ---> damit man Ausgabe von "Hallo" lesen kann
71                  i=10  --> damit abfrage beendet wird
72                  readOK = 1
73              end
74          end
```



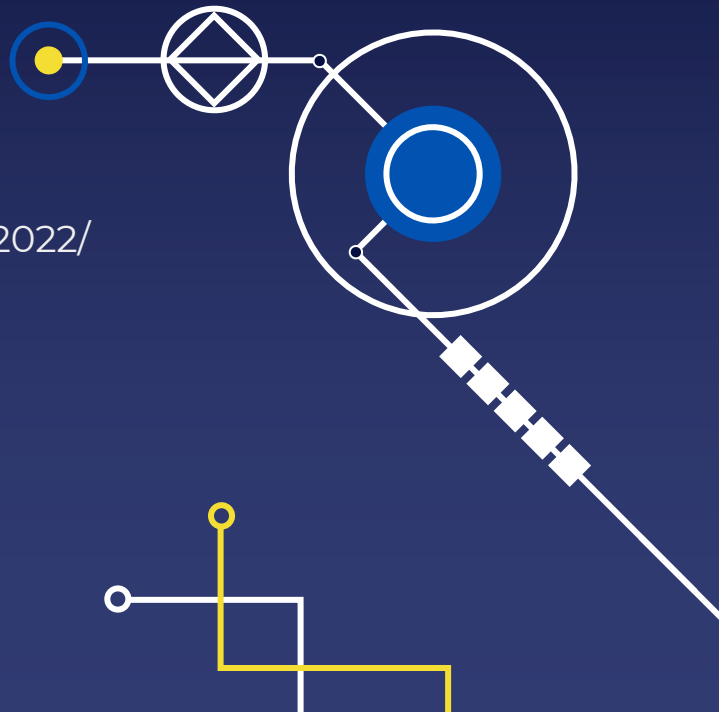
Conclusio

Conclusio:

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



Links:

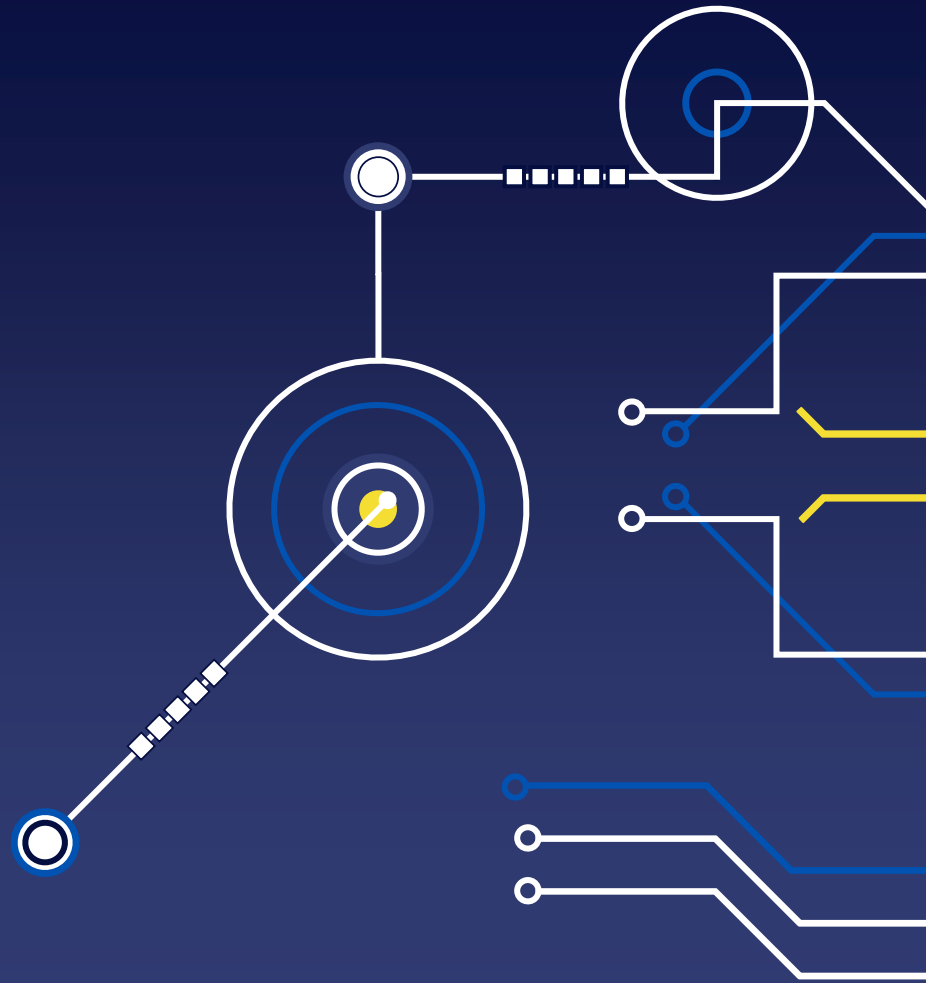


- https://github.com/pi4oorexx/pi4oorexx
- https://wi.wu.ac.at/rgf/diplomarbeiten/BakkStuff/2022/202204_Duer_pi4oorexx.pdf

THANKS!

Do you have any questions?

Marcel.duer@gmail.com




slidesgo