Displaying DHT11 information with LED matrix, Lab 4

Embedded System Design - CS3813301
Group 6
Tobias Vera - tobiasvera2011@gmail.com (F10915126),
Gerardo Fisch - gerarfisch@gmail.com (F10915108)
November 23rd, 2021

I. Introduction

The purpose of this lab is to introduce the students to a how shift registers (74HC595) as IO expanders by displaying DHT11 collected information via an 8x8 LED dot matrix (1588BS) display.

II. Procedure

- Connect the 1588BS dot matrix display to Arduino, use a series resistor at least 220-ohm per LED (row or column, not both).
- Read the DHT11 temperature and relative humidity.
- Use assembly or C to decode the DHT11 data, verify the checksum, and drive the LED dot matrix display.
- Program the Arduino.

III. Experimental Data

A. Schematic of the Circuit

The schematic of the circuit can be seen on the Figure 1. From the schematic it can be seen that the pins used for this circuit are the pins from 0 to 7 for the 8x8 LED Display, the pin 13 for the DHT11 and the pins 8,9 and 10 for the 74HC595 Shift Register. From the schematic of the Arduino UNO[1], it can be seen that the only output ports used on the circuit are the PORT B and PORT D, since the pins from 0 to 7 correspond to the PORT D and the pins from 8,9 and 10 correspond to the PORT B and the only input/output port used is the pin 13 which is from the PORTB. In addition, the GND and 5V pins were used to power the DHT11 and the 74HC595 Shift Register.

B. Photo of the wired-up circuit

Based on the schematic of the circuit, the same circuit was reproduced. The circuit can be seen on Figure 2.

C. Results

The Figures 4, 5, 6 and 7 show the final result.

IV. Questions

1) What are the pros and cons of using a shift register as an IO expander?
Pros:
- We can use shift registers as IO expanders.
- Very quick when you want to convert data from serial to parallel or vice versa. They are faster than normal serial to parallel converter circuits.

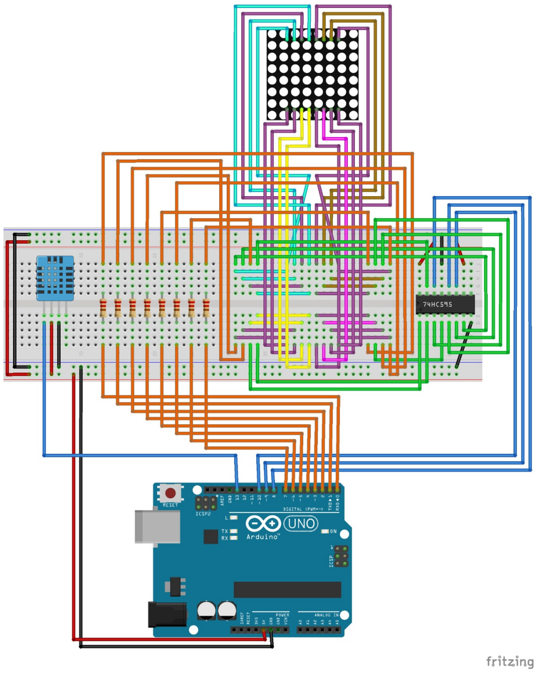

Fig. 1. Schematic of the circuit

Cons:
- The strength of the output current coming from a shift register is not so strong.
- There could be some synchronization problem to deal with, but they are very fast for the general use.

2) What rate did you send data out to the shift register to make it appear seamless to humans? What would happen if you transmitted data too slowly?
The rate at which the data was sent to the shift register was 50us. The shift register was used as a tool for the sequence which determine which column of the matrix to turn on. Therefore, in case the data is transmitted too slowly, will appear to be flickering.

3) Why can you only turn on one column / row of LEDs at a time?
If more than one column / row is turned on at a time, they will have the same pattern. So, if more than one column / row are going to be used to display the same pattern, they can be turned on at the same time, otherwise, a sequence with some delay would have to be applied in order to turn

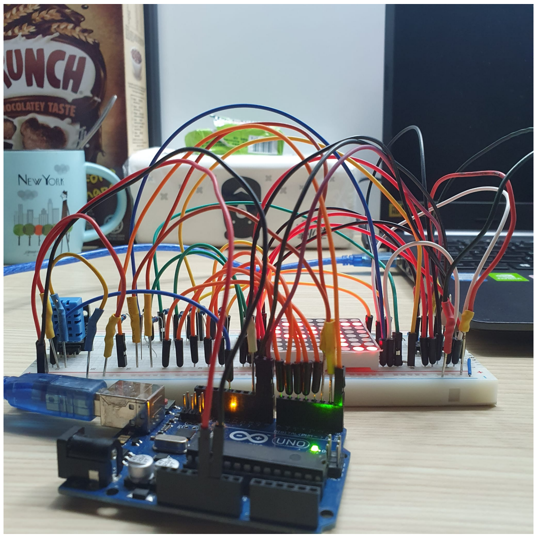

Fig. 2. Wired-up circuit

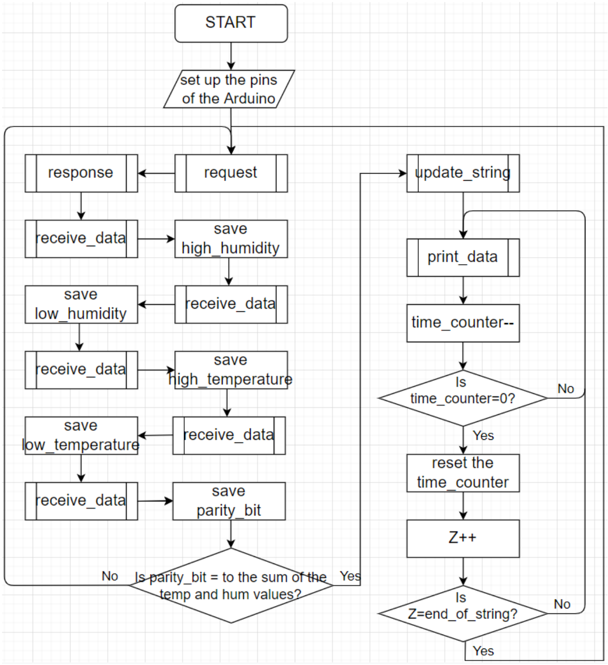

Fig. 3. Flow Diagram of the Program

on columns / rows very quickly and display different things.
4) What happens when you turn on multiple LEDs?
If multiple LEDs are turned on the brightness of them is going to be affected. As more LEDs are turned on, the less bright they are.

V. DISCUSSION

The use of a shift register is a useful technique when the Arduino is shortened out of input/output pins. In this case, it was used as a tool for the sequent which determine which column to turn on, saving a lot of pins from the Arduino board. Doing some testing before the experiment, it turned out that it's also a useful tool for converting data from serial to parallel.

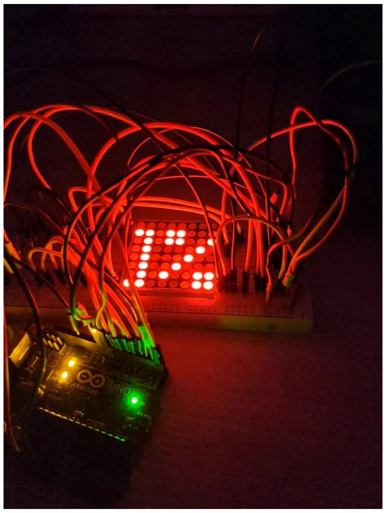

Fig. 4. Example of the 8x8 LED showing %.

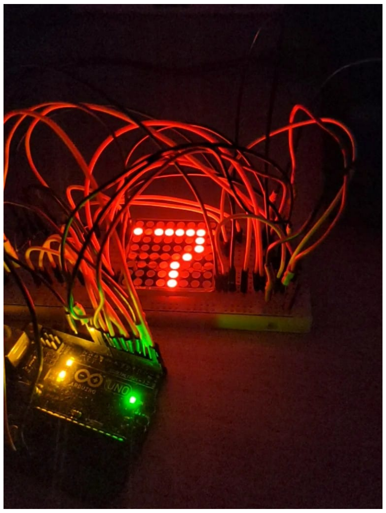

Fig. 5. Example of the 8x8 LED showing 7.

The first version of this experiment was done without using a shift register, which took a lot of digital IO pins. Doing that, it was discovered that all port pins can act as digital IO, even though the Arduino board indicates another function.
The use of the 8x8 LED dot matrix was very interesting and similar to the 4-digit-7-segment display in the way it works, except that here it uses rows instead of segments and columns instead of digits. If the columns are to display the same pattern, they can be turned on at the same time, but if the columns are going to display different patterns, then they have to be turned on sequentially applying some delay.

VI. CONCLUSION

The use of a shift register as an IO expander is a useful tool that saves a lot of port pins from the board, giving the opportunity to do more complex projects. Also, the port pins of the Atmel328p are configurable and they can be used as IO pins even when the board indicates another thing.
The 8x8 LED dot matrix is very useful for displaying a

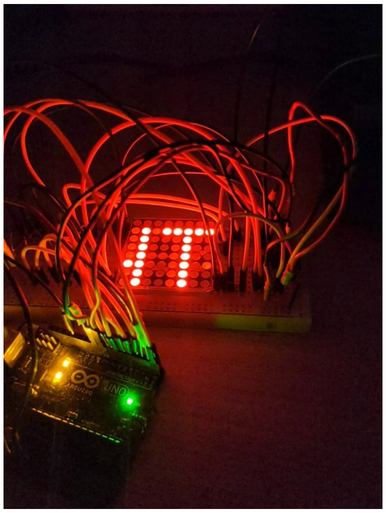

Fig. 6. Example of the 8x8 LED showing T.

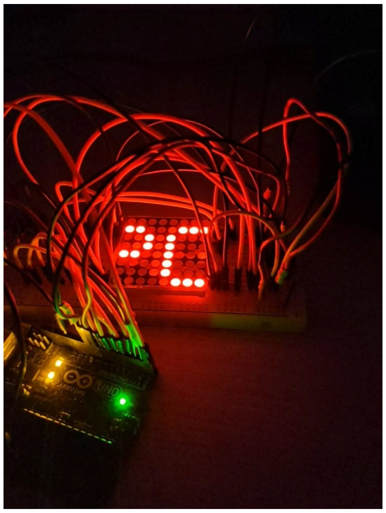

Fig. 7. Example of the 8x8 LED showing °C.

wide variety of characters and custom pictures, similar to the components we already used in other labs.
The experiment gave satisfactory results and displays the sentence as required by the professor.

References

[1] https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

APPENDIX

```
.equ high_humidity = 0x200                                ; memory spaces where the data received from
.equ low_humidity = 0x201                                 ; the dht11 is saved
.equ high_temperature = 0x202
.equ low_temperature = 0x203
.equ parity_bit = 0x204

.equ dht11_port = 5 ; (13 - 8 = 5)

.equ latchPin = 1                ; Latch pin of 74HC595 is connected to pin 9 (9 - 8 = 1)
.equ clockPin = 2                ; Clock pin of 74HC595 is connected to pin 10 (10 - 8 = 2)
.equ dataPin = 0                 ; (8 - 8 = 0)

.equ humidity_decimal = 0x41              ; 0x141 is where the humidity decimal starts
.equ humidity_unit = 0x47                 ; 0x147 is where the humidity decimal starts
.equ humidity_fractional = 0x4F           ; 0x14F is where the humidity decimal starts
.equ temperature_decimal = 0x6D           ; 0x16D is where the humidity decimal starts
.equ temperature_unit = 0x73              ; 0x173 is where the humidity decimal starts
.equ temperature_frational = 0x7B         ; 0x17B is where the humidity decimal starts

.equ param1 = 0x17C                  ; memory space that will be used as a parameter
                                     ; for "functions" (call)
.def end_of_string = r29

.equ U_0 = 0b01111110
.equ U_1 = 0b00000001
.equ U_2 = 0b00000001
.equ U_3 = 0b00000001
.equ U_4 = 0b01111110

.equ P_0 = 0b01111111
.equ P_1 = 0b01000100
.equ P_2 = 0b01000100
.equ P_3 = 0b01000100
.equ P_4 = 0b00111000

.equ T_0 = 0b01000000
.equ T_1 = 0b01000000
.equ T_2 = 0b01111111
.equ T_3 = 0b01000000
.equ T_4 = 0b01000000

.equ N_0 = 0b01111111
.equ N_1 = 0b00010000
.equ N_2 = 0b00001000
.equ N_3 = 0b00000100
.equ N_4 = 0b01111111

.equ S_0 = 0b00110010
.equ S_1 = 0b01001001
.equ S_2 = 0b01001001
.equ S_3 = 0b01001001
.equ S_4 = 0b00100110

.equ R_0 = 0b01111111
.equ R_1 = 0b01000100
.equ R_2 = 0b01000100
.equ R_3 = 0b01000100
.equ R_4 = 0b00111011

.equ H_0 = 0b01111111
.equ H_1 = 0b00001000
.equ H_2 = 0b00001000
.equ H_3 = 0b00001000
.equ H_4 = 0b01111111

.equ C_0 = 0b00111110
.equ C_1 = 0b01000001
.equ C_2 = 0b01000001
.equ C_3 = 0b01000001
.equ C_4 = 0b00100010

.equ n0_0 = 0b01111111
.equ n0_1 = 0b01000001
.equ n0_2 = 0b01000001
```

```
.equ n0_3 = 0b01000001
.equ n0_4 = 0b01111111

.equ n1_0 = 0b00000000
.equ n1_1 = 0b00100000
.equ n1_2 = 0b01111111
.equ n1_3 = 0b00000000
.equ n1_4 = 0b00000000

.equ n2_0 = 0b01001111
.equ n2_1 = 0b01001001
.equ n2_2 = 0b01001001
.equ n2_3 = 0b01001001
.equ n2_4 = 0b01111001

.equ n3_0 = 0b01001001
.equ n3_1 = 0b01001001
.equ n3_2 = 0b01001001
.equ n3_3 = 0b01001001
.equ n3_4 = 0b01111111

.equ n4_0 = 0b01111000
.equ n4_1 = 0b00001000
.equ n4_2 = 0b00001000
.equ n4_3 = 0b00001000
.equ n4_4 = 0b01111111

.equ n5_0 = 0b01111001
.equ n5_1 = 0b01001001
.equ n5_2 = 0b01001001
.equ n5_3 = 0b01001001
.equ n5_4 = 0b01001111

.equ n6_0 = 0b01111111
.equ n6_1 = 0b01001001
.equ n6_2 = 0b01001001
.equ n6_3 = 0b01001001
.equ n6_4 = 0b01001111

.equ n7_0 = 0b01000000
.equ n7_1 = 0b01000000
.equ n7_2 = 0b01000111
.equ n7_3 = 0b01001000
.equ n7_4 = 0b01110000

.equ n8_0 = 0b01111111
.equ n8_1 = 0b01001001
.equ n8_2 = 0b01001001
.equ n8_3 = 0b01001001
.equ n8_4 = 0b01111111

.equ n9_0 = 0b01111001
.equ n9_1 = 0b01001001
.equ n9_2 = 0b01001001
.equ n9_3 = 0b01001001
.equ n9_4 = 0b01111111

.equ perc_0 = 0b01100010
.equ perc_1 = 0b01100100
.equ perc_2 = 0b00001000
.equ perc_3 = 0b00010011
.equ perc_4 = 0b00100011

.equ dash = 0b00001000
.equ point = 0b00000001
.equ space = 0b00000000

.equ degree_0 = 0b00110000
.equ degree_1 = 0b01001000
.equ degree_2 = 0b01001000
.equ degree_3 = 0b00110000

setup:
  ldi r16, 0x00
  ldi r17, 0xFF
```

```
out DDRD, r17       ; set up the output ports
out PORTD, r16

sbi DDRB, dht11_port
sbi DDRB, latchPin
sbi DDRB, clockPin
sbi DDRB, dataPin

sbi PORTB, dataPin      ; datapin = 1
sbi PORTB, clockPin     ; the shifter receice '1' eight times
nop                     ; due to the 8 instructions of nop
nop                     ; (8 cycles)
nop
nop
nop
nop
nop
cbi PORTB, clockPin     ; the shifter stop receiving data
; this was done to have the shifter register equals 0xFF


; The string will be stored into the memory starting at 0x100
ldi r31, 0x01
ldi r30, 0x00

ldi r16, space
st Z+, r16
st Z+, r16
st Z+, r16
st Z+, r16
st Z+, r16
st Z+, r16
st Z+, r16

  ldi r16, U_0
st Z+, r16
ldi r16, U_1
st Z+, r16
ldi r16, U_2
st Z+, r16
ldi r16, U_3
st Z+, r16
ldi r16, U_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, P_0
st Z+, r16
ldi r16, P_1
st Z+, r16
ldi r16, P_2
st Z+, r16
ldi r16, P_3
st Z+, r16
ldi r16, P_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, T_0
st Z+, r16
ldi r16, T_1
st Z+, r16
ldi r16, T_2
st Z+, r16
ldi r16, T_3
st Z+, r16
ldi r16, T_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, P_0
st Z+, r16
```

```
ldi r16, P_1
st Z+, r16
ldi r16, P_2
st Z+, r16
ldi r16, P_3
st Z+, r16
ldi r16, P_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, dash
st Z+, r16
st Z+, r16
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, N_0
st Z+, r16
ldi r16, N_1
st Z+, r16
ldi r16, N_2
st Z+, r16
ldi r16, N_3
st Z+, r16
ldi r16, N_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, T_0
st Z+, r16
ldi r16, T_1
st Z+, r16
ldi r16, T_2
st Z+, r16
ldi r16, T_3
st Z+, r16
ldi r16, T_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, U_0
st Z+, r16
ldi r16, U_1
st Z+, r16
ldi r16, U_2
st Z+, r16
ldi r16, U_3
st Z+, r16
ldi r16, U_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, S_0
st Z+, r16
ldi r16, S_1
st Z+, r16
ldi r16, S_2
st Z+, r16
ldi r16, S_3
st Z+, r16
ldi r16, S_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, T_0
st Z+, r16
ldi r16, T_1
st Z+, r16
ldi r16, T_2
```

```
st Z+, r16
ldi r16, T_3
st Z+, r16
ldi r16, T_4
st Z+, r16
ldi r16, space
st Z+, r16

subi r30, 251  ; r30 = r30 + 5, here is where the decimal digit of humidity will be stored
ldi r16, space
st Z+, r16

subi r30, 251  ; r30 = r30 + 5, here is where the unit digit of humidity will be stored
ldi r16, space
st Z+, r16

ldi r16, point
st Z+, r16
ldi r16, space
st Z+, r16

subi r30, 251  ; r30 = r30 + 5, here is where the fractional digit of humidity will be stored
ldi r16, space
st Z+, r16

ldi r16, n0_0
st Z+, r16
ldi r16, n0_1
st Z+, r16
ldi r16, n0_2
st Z+, r16
ldi r16, n0_3
st Z+, r16
ldi r16, n0_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, perc_0
st Z+, r16
ldi r16, perc_1
st Z+, r16
ldi r16, perc_2
st Z+, r16
ldi r16, perc_3
st Z+, r16
ldi r16, perc_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, R_0
st Z+, r16
ldi r16, R_1
st Z+, r16
ldi r16, R_2
st Z+, r16
ldi r16, R_3
st Z+, r16
ldi r16, R_4
st Z+, r16
ldi r16, space
st Z+, r16

ldi r16, H_0
st Z+, r16
ldi r16, H_1
st Z+, r16
ldi r16, H_2
st Z+, r16
ldi r16, H_3
st Z+, r16
ldi r16, H_4
st Z+, r16
ldi r16, space
```

```
  st Z+, r16

  subi r30, 251  ; r30 = r30 + 5, here is where the decimal digit of temperature will be stored
  ldi r16, space
  st Z+, r16

  subi r30, 251  ; r30 = r30 + 5, here is where the unit digit of temperature will be stored
  ldi r16, space
  st Z+, r16

  ldi r16, point
  st Z+, r16
  ldi r16, space
  st Z+, r16

  subi r30, 251  ; r30 = r30 + 5, here is where the fractional digit of temperature will be stored
  ldi r16, space
  st Z+, r16

  ldi r16, n0_0
  st Z+, r16
  ldi r16, n0_1
  st Z+, r16
  ldi r16, n0_2
  st Z+, r16
  ldi r16, n0_3
  st Z+, r16
  ldi r16, n0_4
  st Z+, r16
  ldi r16, space
  st Z+, r16

  ldi r16, degree_0
  st Z+, r16
  ldi r16, degree_1
  st Z+, r16
  ldi r16, degree_2
  st Z+, r16
  ldi r16, degree_3
  st Z+, r16
  ldi r16, space
  st Z+, r16

  ldi r16, C_0
  st Z+, r16
  ldi r16, C_1
  st Z+, r16
  ldi r16, C_2
  st Z+, r16
  ldi r16, C_3
  st Z+, r16
  ldi r16, C_4
  st Z+, r16
  ldi r16, space
  st Z+, r16

  mov r29, r30

  ldi r16, space
  st Z+, r16
  st Z+, r16
  st Z+, r16
  st Z+, r16
  st Z+, r16
  st Z+, r16
  st Z+, r16

get_data:
  call request                 ; a request to the dht11 is made
  call response                ; the response of the dht11 is received
  call receive_data            ; data is received 5 times and saved in 5 different registers
  sts high_humidity, r18
  call receive_data
  sts low_humidity, r18
  call receive_data
  sts high_temperature, r18
  call receive_data
  sts low_temperature, r18
  call receive_data
  sts parity_bit, r18

  lds r16, high_humidity
  lds r17, low_humidity
  add r16,r17
  lds r17, high_temperature
  add r16,r17
  lds r17, low_temperature
  add r16,r17
  cp r16, r18             ; compare if parity_bit == low_hum + high_hum + low_temp + high_temp
  brne get_data           ; if is not equal go to get_data
  call update_string      ; else go to update_string

ldi r30, 0
ldi r25, 255  ; registers used as time
ldi r26, 255  ; counters

loop:
  call print_data

  subi r25, 1
  sbci r26, 0
  cpi r25,0x00             ; Compare time_counter with 0
  breq shift_by_one        ; if time_counter == 0 go to shift_by_one
  rjmp loop                ; else repeat the loop

  shift_by_one:
    ldi r26, 255           ; reset the time counter
    inc r30                ; update the Z pointer (This shift the output by 1)
    cp r30, end_of_string  ; compare Z with end_of_string
    breq get_data          ; if equal go back to receive data from the dht11
    rjmp loop              ; else continue printing

request:
  sbi DDRB, dht11_port     ; set the dht11 as an output port to send the request
  cbi PORTB, dht11_port    ; set the dht11 as low

  delay_20ms:
    ldi r26, 147
    ldi r27, 160
    ldi r28, 2
    dloop_20ms:
      dec r26
      brne dloop_20ms
      dec r27
      brne dloop_20ms
      dec r28
      brne dloop_20ms

  sbi PORTB, dht11_port    ; set the dht11 as high
  ret

response:
  cbi DDRB, dht11_port     ; set the dht11 as an input port to wait for a response

  loop_response_1:         ; while (PIN of the dht11 == 1)
    sbic PINB, dht11_port
    rjmp loop_response_1
  loop_response_2:         ; while (PIN of the dht11 == 0)
    sbis PINB, dht11_port
    rjmp loop_response_2
  loop_response_3:         ; while (PIN of the dht11 == 1)
    sbic PINB, dht11_port
    rjmp loop_response_3
  ret

receive_data:
  ldi r16, 0x00
  ldi r18, 0x00
  rec_data_loop1:           ; loop used to receive the 8 bits
    rec_data_loop1_1:       ; while (PIN of the dht11 == 0)
```

```
      sbis PINB, dht11_port
      rjmp rec_data_loop1_1

    call delay_50us

    in r17, PINB
    andi r17, 1 << dht11_port
    cpi r17, 0x00
    brne r17_is_not_0
    lsl r18               ; if (PINB == 0)
      rjmp end_if
    r17_is_not_0:
      lsl r18             ; if (PINB == 1)
      ori r18, 0x01
    end_if:

    rec_data_loop1_2:     ; while (PIN of the dht11 == 1)
      sbic PINB, dht11_port
      rjmp rec_data_loop1_2

    inc r16
    cpi r16, 0x08
    brne rec_data_loop1
  ret

delay_50us:
  ldi  r27, 2
    ldi  r28, 9
  L1: dec  r28
    brne L1
    dec  r27
    brne L1
    ret

update_string:  ; this function changes temp and hum values of the string to the ones read by the dht11
  lds r16, high_humidity
  ldi r17, 0
  div_by_10_1:
    inc r17
    subi r16, 10
    brcc div_by_10_1    ; branch if C is zero
  dec r17             ; once too many
  subi r16, 246       ; param1 += 1, add back to get the remainder

  sts param1, r16
  ldi r30, humidity_unit
  call set_digit

  sts param1, r17
  ldi r30, humidity_decimal
  call set_digit

  lds r17, low_humidity
  sts param1, r17
  ldi r30, humidity_fractional
  call set_digit

  lds r16, high_temperature
  ldi r17, 0
  div_by_10_2:
    inc r17
    subi r16, 10
    brcc div_by_10_2    ; branch if C is zero
  dec r17             ; once too many
  subi r16, 246       ; param1 += 1, add back to get the remainder

  sts param1, r16
  ldi r30, temperature_unit
  call set_digit

  sts param1, r17
  ldi r30, temperature_decimal
  call set_digit

  lds r17, low_temperature
  sts param1, r17
  ldi r30, temperature_fractional
  call set_digit

  ret

print_data:
  ld r18, Z
  out PORTD, r18

  cbi PORTB, dataPin
  sbi PORTB, clockPin    ; upload '0' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b01111111

  call delay_50us

  ldd r18, Z+1
  out PORTD, r18

  sbi PORTB, dataPin
  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b10111111

  call delay_50us

  ldd r18, Z+2
  out PORTD, r18

  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b11011111

  call delay_50us

  ldd r18, Z+3
  out PORTD, r18

  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b11101111

  call delay_50us

  ldd r18, Z+4
  out PORTD, r18

  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b11110111

  call delay_50us

  ldd r18, Z+5
  out PORTD, r18

  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin    ; latch register = 0b11111011

  call delay_50us

  ldd r18, Z+6
  out PORTD, r18

  sbi PORTB, clockPin    ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin    ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin     ; latch register = 0b11111101

  call delay_50us

  ldd r18, Z+7
  out PORTD, r18

  sbi PORTB, clockPin     ; upload '1' to the shifter
  cbi PORTB, clockPin
  cbi PORTB, latchPin     ; the content of the  Shift Register is copied into the Storage/Latch Register
  sbi PORTB, latchPin     ; latch register = 0b11111110

  call delay_50us
  ret

set_digit:
  lds r16, param1
  cpi r16, 0x00     ; compare param1 with values between 0-9 to decide
  breq print_0      ; which number to show
  cpi r16, 0x01
  breq print_1
  cpi r16, 0x02
  breq print_2
  cpi r16, 0x03
  breq print_3_long_jump
  cpi r16, 0x04
  breq print_4_long_jump
  cpi r16, 0x05
  breq print_5_long_jump
  cpi r16, 0x06
  breq print_6_long_jump
  cpi r16, 0x07
  breq print_7_long_jump
  cpi r16, 0x08
  breq print_8_long_jump
  cpi r16, 0x09
  breq print_9_long_jump
  print_3_long_jump:
    jmp print_3
  print_4_long_jump:
    jmp print_4
  print_5_long_jump:
    jmp print_5
  print_6_long_jump:
    jmp print_6
  print_7_long_jump:
    jmp print_7
  print_8_long_jump:
    jmp print_8
  print_9_long_jump:
    jmp print_9
  print_0:
    ldi r16, n0_0
    st Z+, r16
    ldi r16, n0_1
    st Z+, r16
    ldi r16, n0_2
    st Z+, r16
    ldi r16, n0_3
    st Z+, r16
    ldi r16, n0_4
    st Z+, r16
    rjmp switch_end
  print_1:
    ldi r16, n1_0
    st Z+, r16
    ldi r16, n1_1
    st Z+, r16
    ldi r16, n1_2
    st Z+, r16
    ldi r16, n1_3
    st Z+, r16
    ldi r16, n1_4
    st Z+, r16
    rjmp switch_end
```

```
print_2:
  ldi r16, n2_0
  st Z+, r16
  ldi r16, n2_1
  st Z+, r16
  ldi r16, n2_2
  st Z+, r16
  ldi r16, n2_3
  st Z+, r16
  ldi r16, n2_4
  st Z+, r16
  rjmp switch_end
print_3:
  ldi r16, n3_0
  st Z+, r16
  ldi r16, n3_1
  st Z+, r16
  ldi r16, n3_2
  st Z+, r16
  ldi r16, n3_3
  st Z+, r16
  ldi r16, n3_4
  st Z+, r16
  rjmp switch_end
print_4:
  ldi r16, n4_0
  st Z+, r16
  ldi r16, n4_1
  st Z+, r16
  ldi r16, n4_2
  st Z+, r16
  ldi r16, n4_3
  st Z+, r16
  ldi r16, n4_4
  st Z+, r16
  rjmp switch_end
print_5:
  ldi r16, n5_0
  st Z+, r16
  ldi r16, n5_1
  st Z+, r16
  ldi r16, n5_2
  st Z+, r16
  ldi r16, n5_3
  st Z+, r16
  ldi r16, n5_4
  st Z+, r16
  rjmp switch_end
print_6:
  ldi r16, n6_0
  st Z+, r16
  ldi r16, n6_1
  st Z+, r16
  ldi r16, n6_2
  st Z+, r16
  ldi r16, n6_3
  st Z+, r16
  ldi r16, n6_4
  st Z+, r16
  rjmp switch_end
print_7:
  ldi r16, n7_0
  st Z+, r16
  ldi r16, n7_1
  st Z+, r16
  ldi r16, n7_2
  st Z+, r16
  ldi r16, n7_3
  st Z+, r16
  ldi r16, n7_4
  st Z+, r16
  rjmp switch_end
print_8:
  ldi r16, n8_0
  st Z+, r16
  ldi r16, n8_1
  st Z+, r16
  ldi r16, n8_2
  st Z+, r16
  ldi r16, n8_3
  st Z+, r16
  ldi r16, n8_4
  st Z+, r16
  rjmp switch_end
print_9:
  ldi r16, n9_0
  st Z+, r16
  ldi r16, n9_1
  st Z+, r16
  ldi r16, n9_2
  st Z+, r16
  ldi r16, n9_3
  st Z+, r16
  ldi r16, n9_4
  st Z+, r16
switch_end:
  ret
```