VILNIUS UNIVERSITY
ŠIAULIAI ACADEMY

BACHELOR PROGRAMME SOFTWARE ENGINEERING

Programming of Embedded Systems

Laboratory work No. 2

Student : Hanna Asiadouskaya

Lecturer: dr. Nerijus Ramanauskas

Šiauliai, 2025

Aim of the laboratory work:

The aim of the practical work is to understand the work of DAC2, ADC pin on the solder bridges, and SW button, using STM32 Nucleo F756ZG board, through receiving digital signal and conversion to analog output, and then to given measuring range.

Variant No 2.

No	ADC pin	AO Sensor type	Measuring range
2	A2	CO2, ppm	0 - 2000

Code and Comments:

Importing module to convert DAC, importing library to control and read from microcontroller pins, another one to read analog value and another to work with delays.

Configure SW as an input, define pin for analog input (ADC), object to read analog value, and define DAC on channel 2 to generate analog voltage output.

For simple creation of the while loop, I decided to define global variable and dac_val to ease the counting process.

```
1  from pyb import DAC
2  from machine import Pin
3  from machine import ADC
4  import time
5
6  pin_SW = Pin("SW", Pin.IN, Pin.PULL_DOWN)
7  A2 = Pin("A2", Pin.ANALOG)
8  adc = ADC(A2)
9  dac = DAC(2)
10 global dac_val
11 dac_val = 0
```

Here is while true statement, to keep the code running, to update DAC value and read ADC.

If the button pressed (value equal 1), then increment dac_val to 10, until it reach 250. (This value I chose as it's the one step before it overflow, and start the count from the beginning. Overflow may happen as it can also read the noise)

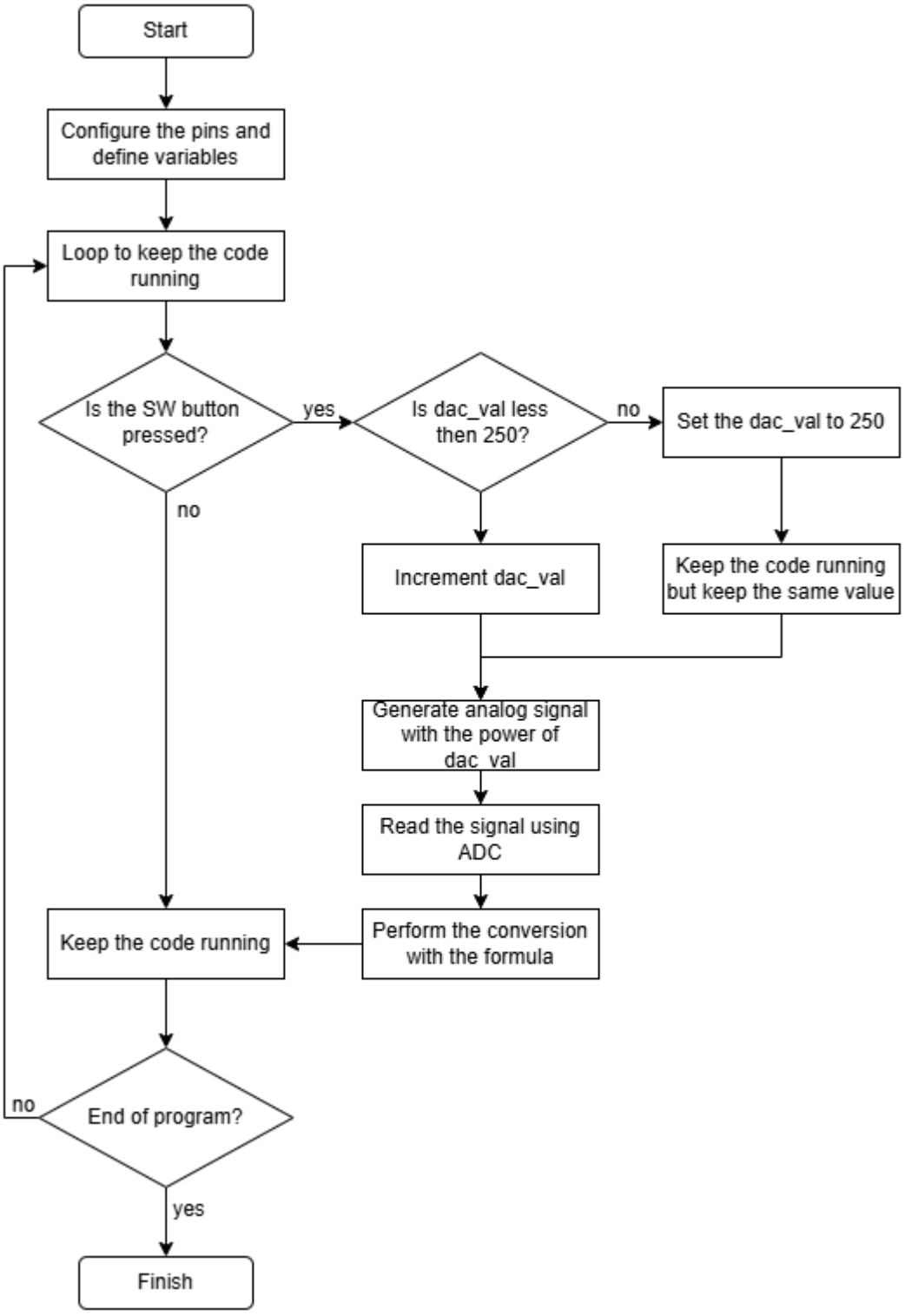
```
13 while True:
14     if pin_SW.value()!=0:
15 
16         if dac_val < 250:
17             dac_val += 10
18         else:
19             dac_val = 250
```

So, next we write current dac_value, then read the analog value from pin that was given.

Then we print output, and as my task was to use range for CO2, I use formula in line 24.

```
20         dac.write(dac_val)
21         val = adc.read_u16()
22         print(f"DAC value: {dac_val}")
23         print(f"Source value: {val}")
24         sens_val = (val/65536 * 2000)
25         print(f"CO2 value: {sens_val} ppm\n")
26 
27         time.sleep_ms(200)
```

Algorithm:


Start
Configure the pins and define variables
Loop to keep the code running
Is the SW button pressed?
yes
Is dac_val less then 250?
no
Set the dac_val to 250
no
Increment dac_val
Keep the code running but keep the same value
Generate analog signal with the power of dac_val
Read the signal using ADC
Perform the conversion with the formula
Keep the code running
End of program?
no
yes
Finish

Conclusion:

In this study, we successfully practiced programming on the STM32 Nucleo F756ZG microcontroller, explored hardware configurations like soldier bridges and pins like DAC2 and ADC, learned the configuration of SW button, and how to handle voltage output with noise. Understand the process of receiving output, like receiving digital signal through DAC2 and converting to analog on ADC.

Code as a text:

```
from pyb import DAC
from machine import Pin
from machine import ADC
import time


pin_SW = Pin("SW", Pin.IN, Pin.PULL_DOWN)
A2 = Pin("A2", Pin.ANALOG)
adc = ADC(A2)
dac = DAC(2)
global dac_val
dac_val = 0
while True:
    if pin_SW.value()!=0:
        if dac_val < 250:
            dac_val += 10
        else:
            dac_val = 250
        dac.write(dac_val)
        val = adc.read_u16()
        print(f"DAC value: {dac_val}")
        print(f"Source value: {val}")
        sens_val = (val/65536 * 2000)
        print(f"CO2 value: {sens_val} ppm\n")


        time.sleep_ms(200)
```