

Hardware 2: Project Exercises

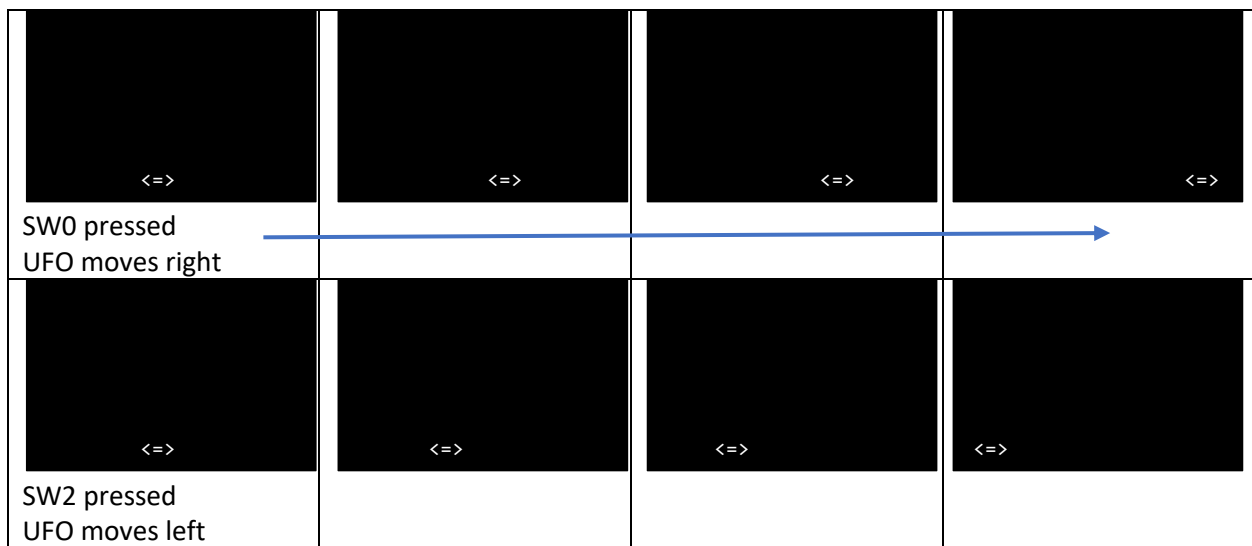
Metropolia University of Applied Sciences
2024

Week 1: OLED

Task 1.1

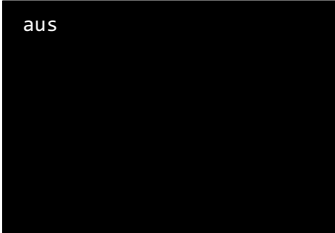
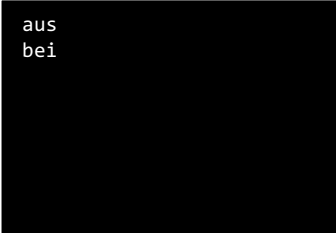
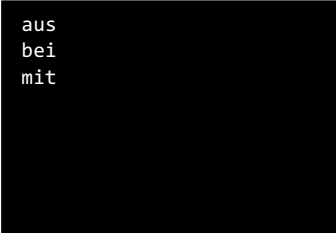
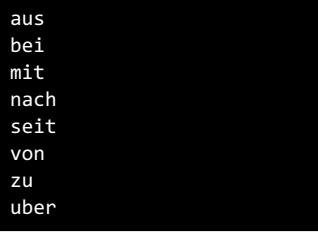
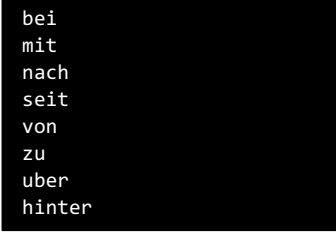
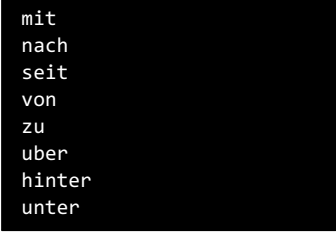
Implement a program that uses the two development board buttons to control a “UFO”. The UFO is shown at the bottom of the screen with characters “<=>”. SW0 and SW2 are used to move the UFO left and right. The program must stop the UFO at the edges of the screen so that it is completely visible. When UFO is at left edge it must only be possible to move the UFO right and vice versa.

The font size is 8x8 pixels.







Task 1.2

Implement a program that reads user input from the keyboard in an infinite loop. The input is typed in Thonny Shell window while the program is running. The user input is drawn to the OLED screen starting from the top of the screen. Each input is drawn below the previous one. When the screen is full the display is scrolled up by one line and then new text is drawn at the bottom of the screen.

		
User keeps entering text until the screen becomes full. Then lines start to scroll up.		
		

Task 1.3

Implement a program that uses the three development board buttons to control line drawing. When the program starts, it starts to draw pixels from the left side of the screen halfway between top and bottom of the screen and constantly moves towards the right edge of the screen. When the drawing reaches the right edge of the screen, the drawing is wrapped back to the left side. Buttons SW0 and SW2 are used to move the pixels towards the top or bottom of the screen, so that by pressing buttons you can draw lines at different heights. Pressing SW1 clears the screen and continues drawing from middle left side.

			
SW0 and Sw2 pressed to move pixels up and down	Pixels wrap back to left side	Pixels wrap back to left side	SW1 clears the screen and drawing starts again from middle left side.

Week 2: Finding peaks, plotting with Thonny and Creating a repository

In the following exercises, you will work with test data sampled at 250Hz rate. That means that you get 250 data points per one second of signal. The test data is read from a file so it is not live data but you can still use the sample count as a measure of time. For example, when you have read 250 samples you know that you have one second of data or if the distance between two positive peaks is 186 samples you know that time between the peaks is $186/250$ seconds = 0,744 seconds.

Copy the three example files: capture_250Hz_01.txt, capture_250Hz_02.txt, and capture_250Hz_03.txt to the root directory of your Pico.

Use filefifo to read samples from a file. Filefifo class is installed to your Pico when you set it up with the example project (<https://gitlab.metropolia.fi/lansk/pico-test>). Filefifo is a mock implementation of a class that we will use later as temporary storage of live data. At this point you only need to know how to open a file and read data using the class.

The example below shows how to open a file and read data using Filefifo. Filefifo works in looping mode by default so you can read “infinite” amount of data. When the data ends filefifo rewinds back to the beginning of the file and starts over. The code below creates a Filefifo instance and then reads 100 data points from the filefifo. The first parameter (10) is a dummy parameter that is there to keep the interface of Filefifo as similar to the one that we will use later with live data. The second parameter is the name of the file to read data from.

```
from filefifo import Filefifo

data = Filefifo(10, name = 'capture_250Hz_01.txt')

for _ in range(100):
    print(data.get())
```

Task 2.1

Implement a program that finds positive peaks from a test signal using slope inspection. The test signal(s) contain pure sine wave so the peaks can be found by inspecting the slope of the signal without using a threshold. The peak is at a point where the slope turns from positive to negative.

Your program must print at least three peak-to-peak intervals both in number of samples and seconds and to calculate the frequency of the signal.

Task 2.2

Implement a program that reads test signal from the file, scales it to range 0 – 100 and prints the scaled values to console. Remember to enable Plotter in Thonny to see the graph.

Start by reading two seconds of data and find minimum and maximum values from the data. Then use min and max values to scale the data so that minimum value is printed as zero and maximum value as 100. Plot 10 seconds of data with the scaling.

Task 2.3

Create a repository for your project using pico-lib as a submodule. Your own repository must have the same functionality as the example project i.e., it must be possible to setup a Pico with an empty filesystem using the repo.

Week 3: Rotary encoder and interrupts

Task 3.1

Implement a program that uses the rotary encoder to control LED brightness. The encoder button is used to toggle the LED on/off and turning the encoder adjusts the brightness if the LED is on-state. If the LED is off, then the encoder turns are ignored. The program must use interrupts for detecting encoder turns and a fifo to communicate turns to the main program. The interrupt handler may not contain any LED handling logic. Its purpose is to send turn events to the main program.

The encoder button must be polled in the main program and filtered for switch bounce.

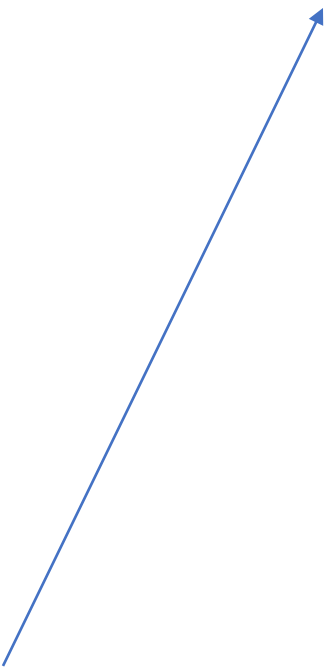
Task 3.2

Implement a program that uses the rotary encoder to select an item from a menu. The menu has three options: LED1, LED2, LED3. Encoder turns move the selection (arrow, highlight, etc.) and pressing the button activates the selection. Activation turns toggles the selected LED on/off. The state of the LED must be updated in the menu.

Use an interrupt for both turn detection and encoder button. The turn and press event must be sent to the main program through a fifo. All of the menu logic must be in the main program.

The encoder button does not have hardware filtering so switch bounce filtering must be performed. Bounce filtering should be done in the interrupt handler with the help of time.ticks_XXX-functions. Filtering is done by taking a millisecond time stamp on each detected press and comparing that to the timestamp of previous press. The new press is ignored if it is too close, for example less than 50 ms away from the previous press.

<pre>[LED1 - OFF] LED2 - OFF LED3 - OFF <turn encoder> LED1 - OFF [LED2 - OFF] LED3 - OFF <press button> LED1 - OFF [LED2 - ON] LED3 - OFF <turn encoder> LED1 - OFF LED2 - ON [LED3 - OFF] <press button> LED1 - OFF LED2 - ON [LED3 - ON] <turn encoder></pre>	<pre>LED1 - OFF [LED2 - ON] LED3 - ON <press button> LED1 - OFF [LED2 - OFF] LED3 - ON</pre>
--	---



Task 3.3

Implement a program that uses the rotary encoder to scroll data that is read from a file

- Read 1000 values from a file to a list.
 - It is easiest to use filefifo to read the values.
- Find minimum and maximum values from the list
- Use rotary encoder to scroll a window of 128 values on the screen.
 - Program must prevent scrolling backwards past the first sample.
 - Program must prevent scrolling past the last sample that still fills the screen (last index - 128)
- Use an interrupt and fifo to send turns to the main program.

Note that your program needs an instance of both filefifo and fifo.

Week 4: Processing and displaying data from a file

All the following assignments use the same data that is read from file(s) using the filefifo class.

There are three files that contain the captured data. All files have been captured using 250 Hz sampling frequency which means that time interval between each sample is 4 ms. The quality of the signal increases with the number of the file. Capture01 contains data that contains signal clipping, motion artifacts, and large variations in the amplitude of the signal. Capture02 and capture03 have amplitude variations but no motion artifacts. You should start your assignment with capture02 or capture03, and when they produce satisfactory results then move on to test with Capture01.

Below is a plot of the files. Note that horizontal scaling is not the same in all plots. Capture03 has fewer data points than the other two files.

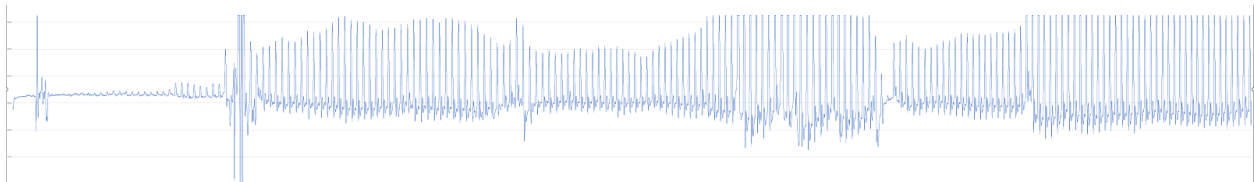


Figure 1 Plot of capture01_250Hz

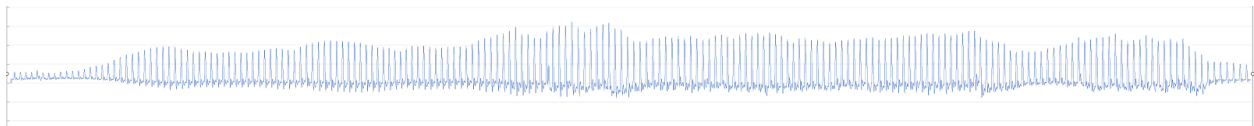


Figure 2 Plot of capture02_250Hz



Figure 3 Plot of capture03_250Hz

Task 4.1

Implement a program that uses one of the algorithms from lecture 3 (pulse detection) to detect heart rate from data in a file. Use filefifo class to read data points from the file. The program must output at least 20 heart rate values to the console (Thonny console).

Task 4.2

Implement a program that plots the signal from the file on the OLED screen. The program must scale the values automatically so that minimum and maximum values of the previous 250 samples will be the minimum and maximum values used for scaling the next 250 samples to be plotted. Plotting is to scaled

so that minimum value plots a pixel at the bottom of the screen and maximum value at the top of the screen. If scaled value would be outside the screen the value is capped to the limits.

- Less than 0 \rightarrow 0
- Greater than 63 \rightarrow 63

There are 250 data points per second and if you consider a typical heart rate in the range of 60-80 BPM there will be 185 – 250 samples between peaks. This means that if you plot all samples, you will not be able to see two beats on the screen because the screen is 128 pixels wide. To implement horizontal scaling plot an average of five successive samples per pixel. That way the display will show $128 \times 5 \times 4\text{ms} = 2560\text{ ms}$ of samples which is enough to display at least two peaks even at very low heart rates.

Task 4.3

Implement a program that uses the rotary encoder to scroll data that is read from a file

- When SW_1 is pressed read 1800 values from a file.
 - Use filefifo to read the values.
 - Put average of 5 successive samples to the list so that the read 1800 values will create a list of 360 values to scroll
- Use rotary encoder to scroll a window of 128 values on the screen
 - Program must prevent scrolling backwards past the first sample
 - Program must prevent scrolling past the last sample that still fills the screen (last index - 128)
 - If no data is available, the program displays “Press SW1”
- Pressing SW_2 while turning the encoder changes scaling of the values.
- Pressing SW_0 while turning the encoder changes the offset. Offset is the value that is subtracted from the value before scaling. Offset determines the vertical position of the signal on the screen

Note that your program needs an instance of both filefifo and fifo.