Lab 4 – FYS4240

Video and AI DAQ with LabVIEW

Jan Kenneth Bekkeng, 28.3.207

Lab 4

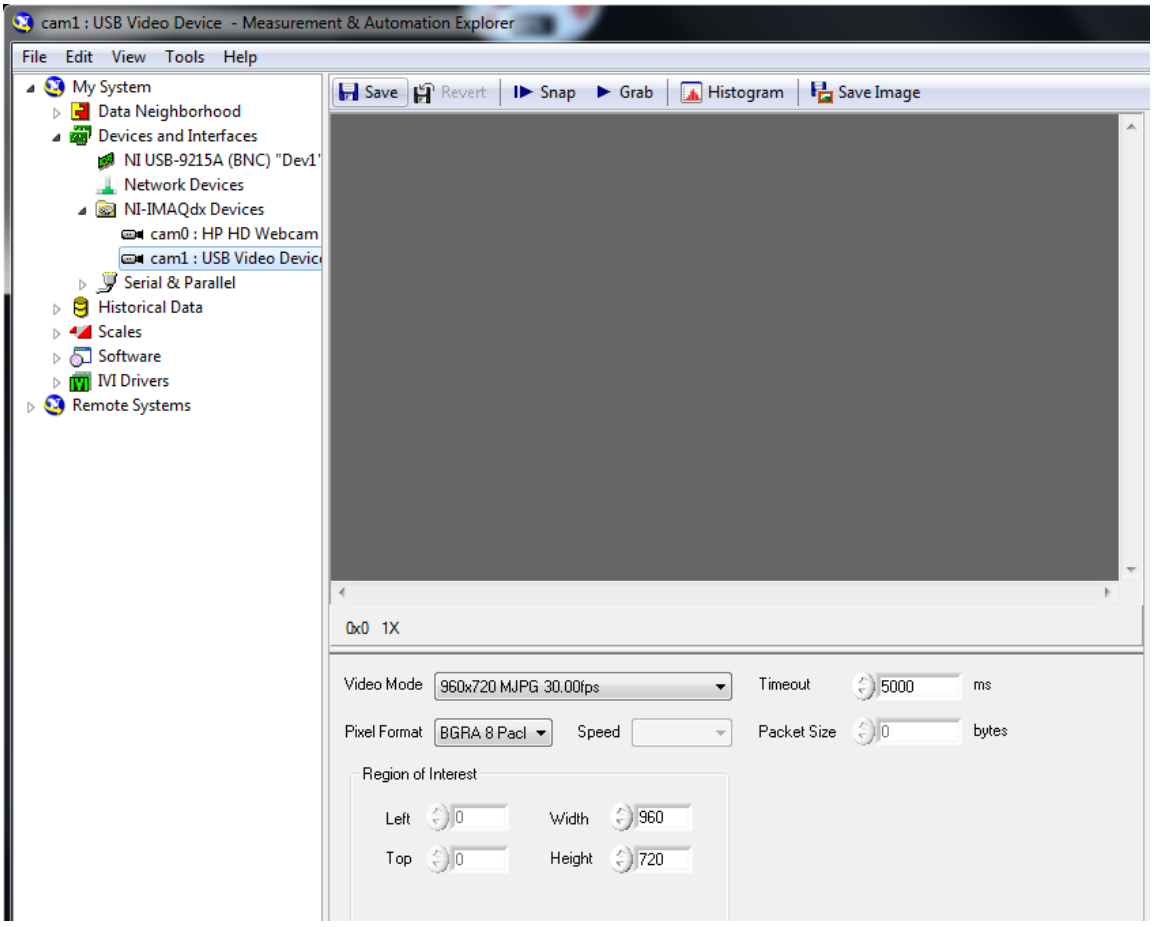
In this lab you will make the LabVIEW program **DataSender.vi** to simultaneously acquire video data from a USB web camera and analog input data from a USB DAQ-box (NI USB-9162). Since the video and the analog signals have different sampling speed they will be acquired in different loops, and data must be transferred between loops using queues. The video data must be possible to save in an AVI-file and a binary file. (If AVI save is selected and compression is used this will limit the while loop speed) . The analog data are to be saved in a TDMS file. In addition the analog data must be transmitted over Ethernet using the UDP protocol, and displayed in another program called **DataReceiver.vi** running on the same or a different computer.

Some VIs (functions) have been made available.

To make the programming easier no advanced GUI programming with Event loop is required. The "file open" and "file close" can be done outside the while loops. This means that to save a new file the program must be stopped and then run (with the white arrow in LabVIEW) again.

Configure the USB camera on your PC using MAX:

- Set resolution to **960 x 720 and 30 fps**
- Press **save** to save current settings to (a configuration) file.



DataSender.vi

It must be possible to select the USB web camera, a folder for saving all the data (*.bin, *.AVI and *.TDMS), and a test number (for the file names). The program needs a button to **save data** to file (when it is pressed), and a **stop button** to stop the program. It should also be separate buttons to enable saving of video data to an AVI file and/or a binary file. A display for the video must also be shown. See example in Figure 1. A counter e.g. for the number of saved video frames is nice to have.

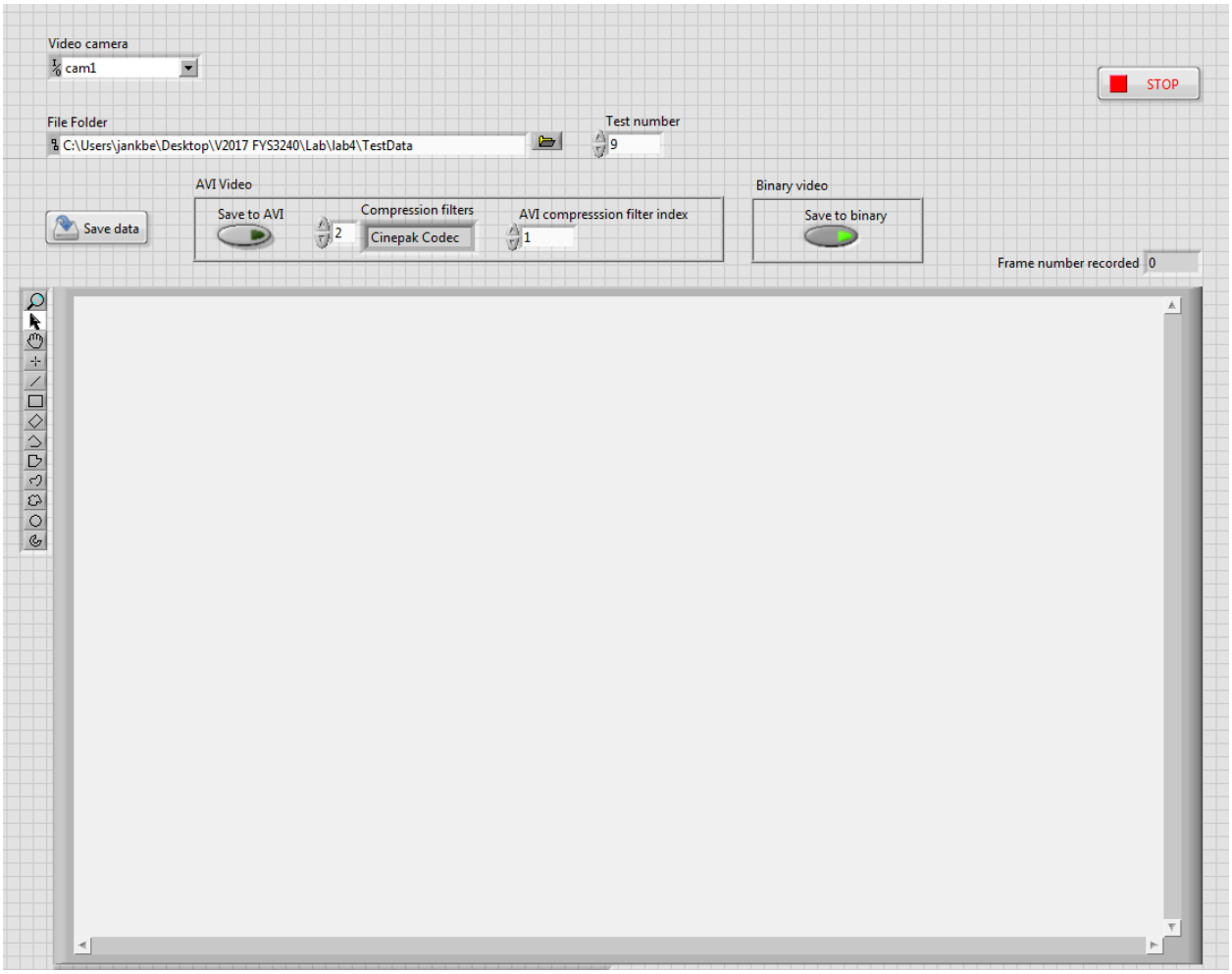


Figure 1: GUI example

The program must use producer-consumer architecture with queues to send measurement data between loops. The program must use five loops; see Figure 2 and the description below:

- Video data acquisition loop (producer)
  - Read video data from the USB web camera.
  - Add video data to video Queue.
  - Add video data to a Notifier (A Notifier do not have buffer such as a queue).

- Video save loop (consumer)
  - Read video data from the video Queue.
  - Save to AVI file if enabled (use case).
  - Save to binary file if enabled (use case).

- Video display loop (consumer)
    - Use the **Wait on Notification** function to read data from the notifier, and display video data.
- AI DAQ loop (Producer)
    - Use the **DAQ Assistant (express VI)** to configure continuous acquisition from the two channels ai0 and ai1, with a sample rate of 10 kHz and a buffer size of 1000. The input range should be +/- 5 V.
    - Add analog input (AI) data to a queue (different queue with different name than used for video data!)
- AI UDP send & AI save loop (Consumer)
    - Read from the queue and send AI data over Ethernet using UDP. You can use the string “localhost” to address the machine you are using; this gives the IP address 127.0.0.1 Then use the **String To IP Function** to get the correct decimal address required by the **UDP write** function.
    - Save AI data to a TDMS file with ***group name*** set to Lab4, and ***channel names*** set to ch1 and ch2.

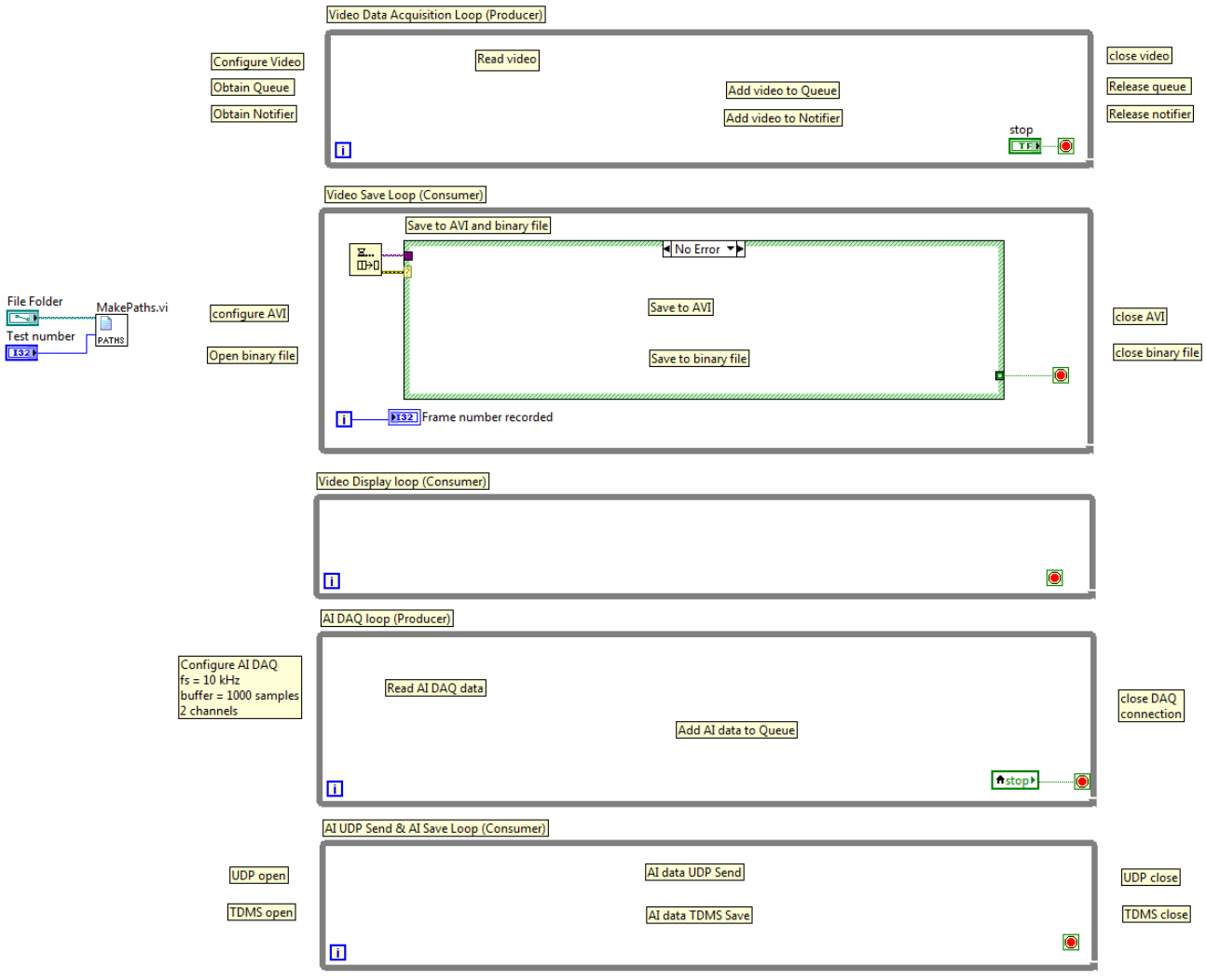


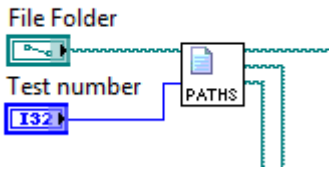
Figure 2: Program structure - five while loops

Simplifications of the program:

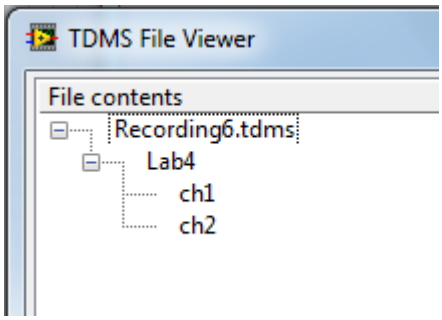
- You can open/create the files (binary, AVI and TDMS) before you go into the while loops, and close the files outside the loops (when the while loops are finished). This simplifies the program, but it means that:
    - Files are always created when you run the program.
    - You need to select the folder for saving the data and a test number before you run the program.
    - In order to make a new recording to a new file you need to stop the program, select a new test number, and then restart the program.
    - To be sure to save all the remaining data in a queue to file when you press the stop button you would need to **flush** the queues (or find the number of remaining elements in the queues) and then write all these remaining elements to file inside a for or a while loop. However, queue flushing is not required to be added in this lab.
        - Note that if compression is used for AVI files this take some time. That means that if you press the save data button again to stop the saving, and immediately after you press "stop" in the program, several image frames can be remaining in the queue waiting to be written to file, and they can be lost (not saved to file) without queue flushing.

Recommended workflow:

1. Make a VI called **MakePaths.vi**
    a. Input (control) is a file path and a test number, the output (indicators) are the full file paths to the three different files. E.g. if the file folder selected is "C:\Test" and the Test number is 1 the output should be "C:\Test\Recording1.avi", "C:\Test\Recording1.bin" and "C:\Test\Recording1.tdms".
    b. These three outputs are used by the "Video save loop" and the "AI UDP send & AI save loop" to set the correct file names and file paths.

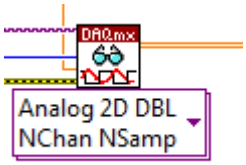


2. Program and test the video daq first (first three while loops)
    a. Use the provided program **Bin_Video_Viewer.vi** to test that you managed to write the video correctly to a binary file.
3. Then add the AI daq (two last while loops)
    a. Test that you made the TDMS file correctly using the provided program **TDMS_Viewer.vi.** You should get a result like this:

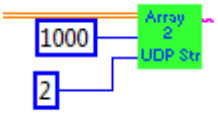


Additional help:

- AI DAQ loop:
  - Select 2D DBL (array) as output from the **DAQmx Read** function.



- Convert data to string before sending over UDP:
  - Use the provided **function 2Darray2string_UDP.vi** as shown below.
  - This function converts from DBL (double precision) to SGL (single precision) before UDP transmission, in order to save bandwidth.



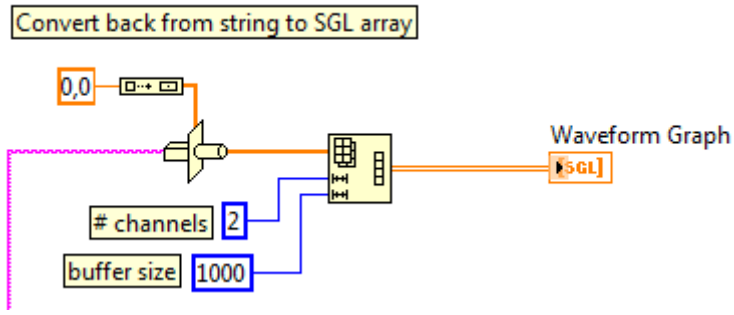
- To create an uncompressed AVI file you set the "Compression Filter" input on the "IMAQ AVI Create VI" to an empty string constant. You can also add compression to the AVI files if you like (optional).

- In addition to the LabVIEW lectures and LabVIEW examples in Lecture 7, 8 and 9 you can find help for:
  - Producer-consumer structure for data: see the template under "File-new".

- Reading video from a USB camera: see the provide program **testvideo.vi**
- Reading from the DAQ box: See lecture 8.
- UDP send: See lecture 7 and the example UDP Sender.vi
- Use “Help – Find Examples” and search for keywords such as “AVI”, “TDMS”, “Write binary” and “Notifier / *General Notifier Example.vi*”

DataReceiver.vi

- Read analog DAQ-data sendt over UDP.
- Display the data in a **graph**
- The program can use only a single while loop.
- **Hints**:
  - Search for UDP under “Help – Find Examples”, and look at the example ***UDP Receiver.vi.***
  - Convert UDP string back to array of SGL (single precision float) in the following way:



Testing

- Start **DataSender.vi,** then start **DataReceiver.vi**