mipi-gstreamer-2

February 10, 2021

1 Using MIPI Sensors with GStreamer

1.1 Part 2

Import and initialize the GStreamer first.

```
[ ]: import gi
     gi.require_version("Gst", "1.0")
     from gi.repository import Gst

     Gst.init(())
```

A pipeline can than be created that converts the video format and causes the appsink to use CPU memory this way.

The *nvvidconv* element can convert between the different memory types. By just specifying *video/x-raw* (and leaving the *memory:NVMM*-part out), the pipeline converts the buffers to CPU memory:

```
[ ]: pipeline = Gst.parse_launch("nvarguscamerasrc ! nvvidconv ! video/x-raw !␣
      ↪appsink name=sink max-buffers=1")

     # Start the pipeline, then wait for it to run
     pipeline.set_state(Gst.State.PLAYING)
     pipeline.get_state(Gst.CLOCK_TIME_NONE)

     # Get our sink element to work with
     sink = pipeline.get_by_name("sink")
```

Now let's see if we get a buffer with the right memory this time!

```
[ ]: sample = sink.emit("try-pull-sample", 1 * Gst.SECOND)
     print("Caps:", sample.get_caps().to_string())
     buffer = sample.get_buffer()
     print("Got a buffer of size: %d Bytes" % (buffer.get_size()))
```

Note that the buffer has the correct size this time. To access the data in the buffer, it needs to be mapped first. When we are done working with the data, we need to unmap it again to avoid a memory leak.

[ ]:
```
buffer = sample.get_buffer()
result, mapinfo = buffer.map(Gst.MapFlags.READ)
print ("Buffer contains %d bytes of data. The 3rd data byte contains the value␣
 ↪0x%x" % (mapinfo.size, mapinfo.data[3]))
buffer.unmap(mapinfo)
```

Note that this still gives us only limited access to the data. Since the buffer is immutable in Python, it could only be mapped as a read-only buffer. We can not write to the buffer directly.

Shut down the pipeline to release the camera:

[ ]:
```
pipeline.set_state(Gst.State.NULL)
```

[ ]:
```

```