

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 643924



D3.4

Test sets and results with EoT prototypes



Copyright © 2015 The EoT Consortium

The opinions of the authors expressed in this document do not necessarily reflect the official opinion of EOT partners or of the European Commission.

1. DOCUMENT INFORMATION

Deliverable Number	D3.4
Deliverable Name	Test sets and results with EoT prototypes
Authors	N. Vallez (UCLM), J. M. Rico (UCLM) , J. L. Espinosa-Aranda (UCLM), A. Dehghani (MOVIDIUS), S. Krauss (DFKI), R. Reiser (DFKI), A. Pagani (DFKI)
Responsible Author	Oscar Deniz (UCLM) e-mail: Oscar.Deniz@uclm.es phone: +34 926295300 Ext.6286
Keywords	EoT Firmware, Tests, EoT Prototype
WP	WP3
Nature	R
Dissemination Level	PU
Planned Date	01.06.2016
Final Version Date	30.05.2016
Reviewed by	O. Deniz (UCLM)
Verified by	C. Fedorzcak (THALES)

3. ABSTRACT

In this deliverable, we describe all software tests which we performed in EoT so far. These tests are divided into: 1) Firmware tests, i.e. unit tests for the different modules developed as part of EoT's firmware, 2) Prototype tests, i.e. test sets applied to the hardware prototypes and 3) Integration tests, i.e. non-unit tests that check that major modules can work together.

4. TABLE OF CONTENTS

1.	Document Information.....	2
2.	Document History	3
3.	Abstract.....	4
4.	Table of Contents.....	5
5.	Firmware Test Set.....	6
6.	Prototype Test Set	8
6.1.	Introduction	8
6.2.	PowerActiveBaseline	8
6.3.	DisparityMap	10
6.4.	UsbVideo208	11
6.5.	BMP180	12
6.6.	BMX055	12
6.7.	DipSwitchGpio	13
6.8.	PmicLedBlink	14
6.9.	RemoteControl.....	14
6.10.	I2CTestR4CAMBOnI2C0AndI2C1.....	14
6.11.	ts_host_test501_bulkout.....	15
6.12.	RevisionWriterEEPROM.....	15
6.13.	MV0198PowerMeasure	17
6.14.	SimpleRTEMS_sdCard	18
6.15.	SimpleRTEMS_HTTPClient.....	19
6.16.	powerMeasurementsShavesIslands.....	20
6.17.	SPI_WiFi_test_TI_C33XX.....	21
6.18.	WiFi_AP_test_TI_CCXX	23
6.19.	WiFi_Get_Time_test_TI_CCXX	23
6.20.	Motor_Control_GPIO/EN/PWN/I2C_test	24
6.21.	InitClocksAndMemory	25
6.22.	SPI_flash_test.....	25
6.23.	NanEye0 External Interface Test (Mode B).....	26
6.24.	Results.....	28
7.	Integration tests	30
7.1.	Introduction	30
7.2.	Integration Tests on the MV0182 board-based prototype	30
7.3.	Integration Tests on Rev1 board	34
8.	Conclusions.....	37
9.	Glossary	38

5. FIRMWARE TEST SET

In this section we briefly summarise the unit tests developed as part of EoT's firmware. Note that the tests were already described in detail in a previous deliverable (D3.3).

Unit test	Operations tested
WiFiFunctions	Save and restore a Wi-Fi profile, restore a Wi-Fi profile when no profile has been saved previously, generate an Access Point using the default configuration, generate an Access Point using a saved profile, change the Wi-Fi device to Station mode (mode necessary to connect to other access point) and change the Wi-Fi device to access point mode (mode necessary to generate an access point).
Camera Interface	There are not specific tests for this module.
Video streaming	There are no automatic tests for testing the functionalities of this module.
Input buttons/DIP switches	The functions of this module cannot be tested automatically, since they require user input or feedback.
SDCard Management	Mounting and unmounting functionality, accessing and manipulating directories and Unit test for the crypto module.
Motor control	These tests change the status of the GPIO pins and send I2C messages to validate the required hardware components.
TimeFunctions	It tests to set a time.
Flash	Opening not existing files, multiple file creation at the same time, multiple file handler creation pointing to one file, write data, append data, peek data, read data, It tests if flash memory is empty, position, available space, remove data, rename file, It tests if a file exists.
Power management	There are not automatic tests for testing the functionalities of this module.
Audio Input & Output	There are no automatic tests for testing the functionalities of this module.
Sparse optical flow with OpenCV	There are no unit tests.

vTrack	It tests the Gaussian image and corners generated by the pixelPipe component, It tests if the pixelPipe finds corners correctly, It tests if the featureMaintenance is able to correctly pass these features to the output buffers, It tests if the pixelPipe is finding corners correctly, It tests if the featureMaintenance is able to correctly pass these features to the output buffers, It tests the number of points tracked, average runtime of vTrack, average runtime of each module, average tracking error of a feature, It tests if the corner coordinates are correct and if it can be tracked by optical flow correctly.
Colour histogram matching	It tests to create two histograms and the Earth Movers Distance operation.
Keypoint matching	
Rotation-invariant face detector	There are no automatic tests for testing the functionalities of this module.
OpenCV	Canny edge detector, Haar-based Cascade Face Detector, K-means algorithm, contours, Histogram/LUT, Delaunay Triangulation, testPyramidSegmentation, testSquareDetector, Watershed, Morphological Operations, Fourier, Inpainting, Min Area Rectangle, Lucas Kanade, DOG, Dilation, Harris Corners, Median Filter
OpenCV in the cloud	This unittest sends a POST request to the REST API of the server, including a picture which is read from the SD card. The server example returns its width and height in pixels.
Libccv	Canny edge detector, Pedestrian cascade detector using Integral Channel Features, Matrix addition operation, Sobel operations, Otsu threshold calculation, Image contrast modification, Perspective transform on a picture, Histogram of gradients calculation, Scale Invariant Feature Transform, CBLAS matrix multiplication, Convolutional network of 11x11 on 225x225 with uniform weights, Convolutional network of 5x5 on 27x27 with non-uniform weights, Convolutional network of 5x5x4 on 27x27x8 partitioned by 2, Stroke Width Transform (text detection) and Face Cascade Detector
Quirc	It tests recognition of different QR codes (different size, colour, quantity)

6. PROTOTYPE TEST SET

Introduction

In this section of the document, we describe the set of tests applied to the Rev1 board (also known as 'DevBoard'). These tests were applied after the board bring-up. Most of the tests were provided by Movidius and these check that low-level functionality works as in the MV0182 board-based prototype that was used at the start of the project. Note that other unit tests were also applied to check that the software modules developed continued to work in this board. These tests were described in the previous section.

PowerActiveBaseline

6.2.1. Software description

This example measures the temperature and the power consumption for three states:

1. Static power with minimal clocks, with DDR disabled.
2. Static power with minimal clocks, with DDR enabled.
3. Static power with DMA transactions. All transfers are checked.

The sampling of 4 rails of current take ~5 ms and the DMA transfer takes ~6.67 ms. So, to measure all rails, we need to repeat the transfer for 4 times. The DMA transfer rate is limited to 12,000 MB/s because the transfer should take more than 5 ms.

For each state, all the current rails are measured and power consumption is displayed. For the consistency of the measurements, the measurements are done for ten times.

6.2.2. Hardware requirements

This software should run on MV182 board (DevBoard). Require mv0198 board connected to the DevBoard.

6.2.3. Expected output

The results consist in printf seen using the debugger. For each state, are printed the power and temperature. The power numbers will vary slightly.

```
Ma2150
=====
UART: Static power measurement with minimal clocks (no DDR)
UART: Core mW = 180.420, DDR mW = 19.830, Temp=34.7C
UART: Core mW = 180.420, DDR mW = 19.830, Temp=34.7C
UART: Core mW = 180.600, DDR mW = 19.830, Temp=34.7C
UART: Core mW = 180.420, DDR mW = 19.830, Temp=34.5C
UART: Core mW = 180.420, DDR mW = 19.830, Temp=34.3C
```



```
UART: Core mW = 180.420, DDR mW = 19.830, Temp=34.5C
UART: Core mW = 180.600, DDR mW = 19.830, Temp=33.9C
UART: Core mW = 180.600, DDR mW = 19.920, Temp=33.7C
UART: Core mW = 180.600, DDR mW = 19.920, Temp=33.9C
UART: Core mW = 180.600, DDR mW = 19.830, Temp=34.1C
UART:
UART: Static power measurement with minimal clocks (DDR)
UART: Core mW = 276.180, DDR mW = 36.510, Temp=34.1C
UART: Core mW = 276.270, DDR mW = 36.510, Temp=34.3C
UART: Core mW = 277.590, DDR mW = 36.510, Temp=34.5C
UART: Core mW = 277.770, DDR mW = 36.510, Temp=34.3C
UART: Core mW = 277.590, DDR mW = 36.510, Temp=34.5C
UART: Core mW = 277.770, DDR mW = 36.510, Temp=34.5C
UART: Core mW = 277.770, DDR mW = 36.510, Temp=34.1C
UART: Core mW = 277.590, DDR mW = 36.510, Temp=34.3C
UART: Core mW = 277.770, DDR mW = 36.510, Temp=34.5C
LRT: Leon entered error mode.
UART: Core mW = 277.680, DDR mW = 36.510, Temp=34.5C
UART:
UART: Static power measurement with DMA transactions
UART: Core mW = 325.940, DDR mW = 204.240, Temp=34.5C
UART: Core mW = 327.478, DDR mW = 204.360, Temp=35.2C
UART: Core mW = 327.298, DDR mW = 204.420, Temp=36.0C
UART: Core mW = 327.523, DDR mW = 204.390, Temp=35.4C
UART: Core mW = 328.971, DDR mW = 204.630, Temp=35.4C
UART: Core mW = 327.523, DDR mW = 204.810, Temp=36.3C
UART: Core mW = 327.433, DDR mW = 204.810, Temp=36.0C
UART: Core mW = 328.837, DDR mW = 204.930, Temp=36.0C
UART: Core mW = 328.836, DDR mW = 204.900, Temp=36.0C
UART: Core mW = 329.016, DDR mW = 204.960, Temp=36.3C
UART:
UART: moviUnitTest:PASSED
```

Ma2100

=====

```
UART: Static power measurement with minimal clocks (no DDR)
UART: Core mW = 192.060, DDR mW = 20.310
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.150, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART: Core mW = 192.060, DDR mW = 20.220
UART:
UART: Static power measurement with minimal clocks (DDR)
UART: Core mW = 285.300, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.210, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
```

```
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.120, DDR mW = 38.790
UART: Core mW = 285.030, DDR mW = 38.790
UART:
UART: Static power measurement with DMA transactions
LRT: Leon entered error mode.
UART: Core mW = 354.285, DDR mW = 170.550
UART: Core mW = 355.687, DDR mW = 171.090
UART: Core mW = 355.777, DDR mW = 170.970
UART: Core mW = 355.687, DDR mW = 170.970
UART: Core mW = 355.687, DDR mW = 171.000
UART: Core mW = 357.178, DDR mW = 171.090
UART: Core mW = 355.777, DDR mW = 171.150
UART: Core mW = 357.178, DDR mW = 171.240
UART: Core mW = 357.178, DDR mW = 170.910
UART: Core mW = 357.178, DDR mW = 171.090
UART:
UART: moviUnitTest:PASSED
```

DisparityMap

6.3.1. Software description

Compute the Census Disparity Map from two stereo images.

The algorithm consists in the following steps:

1. (LOS) Initialize USB and start LRT
2. (LRT) Initialize board and imx208 sensors. Resolution used is VGA 640x480.
3. (SHAVE 0) Pre-processing step: compute the inverse of Homography and (LRT) compute rectified coordinates for right image
4. (LRT) Initialize two SIPP pipelines, one for input conversion and the other for median filtering
5. (LRT) Capture RAW10 images from the cameras and (SHAVE 0) convert them to YUV400 for further processing
6. (HW block) Rectify right image
7. (LRT) Split images into 4 patches
8. (SHAVES 0-3) Start 4 SHAVEs in order to process the 4 patches in parallel. The following filters are applied:
 - Census Transform 5x5 on left image -> the result is a binary number to encode local spatial structure
 - Census Transform 5x5 on right image -> the result is a binary number to encode local spatial structure
 - Census Matching of left and right images -> matches one pixel from the left image with 64 pixels from the right image
 - Census Minimum -> chooses the closest match out of the 64 presumed matches (Winner Takes All)
9. (LRT) Reconstruct image from patches
10. (HW block) Apply median filtering on disparity map
11. (SHAVE 0) Post-processing step: convert resulted disparity map from YUV400 to YUV422

12. (LOS) Stream disparity map on USB

6.3.2. Hardware requirements

This example works on Myriad2 ma2100 and ma2150 silicon.

A MV182 board with Myriad2 chip.

A MV202 board with IMX208 sensors connected to the CAMB connector of the MV182.

A USB cable connection from MV182 to a PC.

6.3.3. Expected output

a) Profiling information and the message "PIPE: LRT: moviUnitTest: PASSED"

b) The displayed output can be seen with a player that can output the USB streaming (i.e. AmCap, VLC, Skype, Cheese)

■ UsbVideo208

6.4.1. Software description

Send the stream data from imx208 camera through USB bus.

This application uses the Leon code and a SIPP pipeline running on shaves. Basically, these are the few steps made during the application:

1. Start the USB DataPump on LeonOS
2. General configurations of the board (internal clocks, external clock generator for sensors, GPIOs, ...)
3. Configure the camera sensors and the in chip datapath (MIPI, SIPP)

6.4.2. Hardware requirements

A MV182 board and an USB cable connection from MV182 to a PC host.

6.4.3. Expected output

The debug console should show the following messages:

```
UART:
UART: Configuring the SIPP pipeline
UART: Configuring imx208 dual cameras and datapath
UART: +UsbPumpRtems_UsbPumpInit_ServiceTask:
UART:
```

The displayed video will be 640 x 480. When the application starts, the PC host should identify the camera and Skype can be used to display camera output.

Other players that can be used to view the streaming:

- Windows: amcap, vlc, skype
- Linux: cheese, guvcview, xawtv, skype

■ BMP180

6.5.1. Software description

The application verifies the BMP180 id, reads the pressure and the temperature. The values (pressure and temperature) are checked if they are in a selected range.

The range for pressure and temperature via next defines from main.c:

```
TEMPERATURE_LOWER_LIMIT  
TEMPERATURE_UPPER_LIMIT  
PRESURE_LOWER_LIMIT  
PRESURE_UPPER_LIMIT
```

The waiting time after BMP180 was configured in a mod (temperature or pressure) can be selected via next define

```
TIME_TO_WAIT_AFTER_CONFIG
```

The test:

- read and verify id
- read and verify pressure
- read and verify temperature
- set BMP180 to give temperature, read and verify, set BMP to give pressure, read and verify at every iteration

The numbers of these iterations can be chosen for each test via next defines

```
NUMBER_OF_READING_ONLY_ID  
NUMBER_OF_READING_ONLY_PRESSURE  
NUMBER_OF_READING_ONLY_TEMPERATURE  
NUMBER_OF_READ_SET_BOTH_TEMP_PRES
```

The frequency of the I2C bus is 100Khz and 400Khz.

6.5.2. Expected output

The results consist in printf seen using the debugger. The value of the temperature and pressure are display:

```
UART: Test start  
UART:  
UART: moviUnitTest: PASSED  
LOS: Application terminated successfully.
```

■ BMX055

6.6.1. Software description

An application that reads from the Bosch BMX055.

The application will dump an .bin file that contains the readings from the sensor. Compare "example.bin" with generated file to check the output of sensor. The data is packed in a 20 bytes structure as follows:

- 8 bytes timestamp (system clock ticks)
- 4 bytes imu type enum (unsigned)
- 2 bytes DataX (signed)
- 2 bytes DataY (signed)
- 2 bytes DataZ (signed)
- 2 bytes DataR (signed) (reserved/unused)
- 4 bytes for alignment

6.6.2. Hardware requirements

This software should run on the mv0182 with a breakout board connected to the Bosch bmx055 sensor.

6.6.3. Expected output

The results consist in printf seen using the debugger:

```
UART: I2C test: 0
UART: I2C test result:0 = IMU I2C SUCCES!
UART: test = 6
UART: 5 sec pass
DEBUG:dumpMemoryToFile():(7018FCC01:000BB800)[000_7018FCC0_000BB800_
LE.bin]
UART: dump memory done
UART: Test end.
```

"000_7018FCC0_000BB800_LE.bin" is the name of output file - may differ from one test to other. "example.bin" is an example of output generated by a good sensor (mv182101 board).

DipSwitchGpio

6.7.1. Software description

The application checks the state of the switches and print the changes on the screen. Switches that are tested are 5,6,7,8.

6.7.2. Expected output

The results consist in printf seen using the debugger. Change the switches and the change should be printed. Application detect transition up to down. Sometimes, transition down to up can be recognized because of noise.

```
UART: Test run
UART: Act the switches and check if the change is detected
UART: Switch 7 changed
```

```
UART: Switch 7 changed
UART: Switch 8 changed
UART: Switch 7 changed
UART: Switch 7 changed
UART: Switch 6 changed
UART: Switch 6 changed
UART: Switch 5 changed
```

■ PmicLedBlink

6.8.1. Software description

Test to make PM LEDs blink, one after another. The application sets LED1 and resets LED2 and vice-versa by writing in the respective registers of the PMIC slave.

6.8.2. Expected output

Test passes if the LEDs blink one after another

■ RemoteControl

6.9.1. Software description

The application verifies if the infrared signal sent from the remote controller is received by the sensor.

6.9.2. Expected output

The application should display a confirm message for any button pressed by the user

```
UART: NECRemoteControlInit
UART: Test start
UART: Infrared signal received
UART: Infrared signal received
UART: Infrared signal received
```

■ I2CTestR4CAMBOnI2C0AndI2C1

6.10.1. Software description

Self-checking test for changing I2C for CAM_B from I2C0 to I2C1 and back.

This application performs a number of tests on I2C connections with its slaves, by reading from and writing to the respective registers.

First test is for IMX208 on I2C0, second is for WM8325 on I2C2, after sets WM8325's GPIO3 on high and tests IMX208 on I2C0 (shouldn't work), after tests IMX208 on I2C1 and should work.

Sets again the default value for WM8325 GPIO3 to low and tests again that IMX208 works on i2c0.

At the end of I2C test block the user will be informed if the test passed or failed.

6.10.2. Expected output

```
UART: Board initialise with status 0
UART: Revision detected: mv182 R4
UART: Test #1
UART: I2C 208 on I2C0 test PASS
UART: I2C 208 on I2C0 test PASS
UART: Test #2
UART: I2C WM8325 on I2C2 test PASS
UART: I2C WM8325 on I2C2 test PASS
UART: Set PMIC GPIO3 High PASS
UART:
UART: moviUnitTest:PASSED
LOS: Application terminated successfully.
```

ts_host_test501_bulkout

6.11.1. Software description

The test case provided is a HOST test case. It expects to be connected to a USB Device (e.g. a memory stick).

6.11.2. Expected output

You should see the test complete and you should be able to see the traffic transmitted on the console of your USB sniffer.

RevisionWriterEEPROM

6.12.1. Software description

The application used to write to the EEPROM.

On PC side:

- Simple application that creates a binary file with the image of revision details.

On board side:

- If WRITE_AND_READ is defined in myriad/main, application read binary file (created with pc app) and write EEPROM memory (via I2C)
- If WRITE_AND_READ is undefined in myriad/main, application read binary file, verify checksum and display revision details write.scr script, in first stage will define the serial number in makefile and start myriad application in order to write EEPROM. In second stage will delete the define used for write operation from Makefile and will start application to read and check the EEPROM. After that, the Makefile is restored

6.12.2. Expected output

```
mdk/testApps/BoardTests/mv182/RevisionWriterEEPROM$ bash write.scr -  
pcbRevision=4 -elecRevision=0 -serialNumber=000003 MV_SOC_REV=ma2100  
-ethmac=11:12:13:14:15:16
```

Binary file created with success:

OFFSET	NAME	VALUE	HEX
0x00	ID	ID	0x4944
0x02	EEPROM FORMAT	1	0x01
0x03	REM. BYTES	28	0x0000001c
0x07	BOARD NAME	MV0182	0x4d5630313832
0x0D	PCB REV	4	0x04
0x0E	MEC REV	0	0x00
0x0F	ELEC REV	2	0x02
0x10	SERIAL NR	182004	0x0002c6f4
0x14	ETH MAC	0:00:00:00:00:00	0x000000000000
0x1A	FEATURE FLAGS	0x00000000	0x00000000
0x1E	REZERVED	0x00000000	0x00000000
0x22	CHECKSUM	38	0x26

Press any key to continue

Cleaning all built files from the MDK distribution and all project built files.

```
Batch execute: <echo $run_opt  
>
```

run

```
Batch execute: <$run_opt>
```

```
Batch execute: <echo $exit_opt>
```

```
Batch execute: <$exit_opt>
```

UART: Start

```
DEBUG: loadMemFromFile() : o:000000 l:000035 -> 0x701BFDB8
```

```
[../input/182004.bin]
```

UART: Start to write:

UART: OFFSET	NAME	VALUE	HEX
UART: 0x00	ID	ID	0x4944
UART: 0x02	EEPROM FORMAT	1	0x01
UART: 0x03	REM. BYTES	28	0x0000001c
UART: 0x07	BOARD NAME MV0182		0x4d5630313832
UART: 0x0D	PCB REV	4	0x04
UART: 0x0E	MEC REV	0	0x00
UART: 0x0F	ELEC REV	2	0x02
UART: 0x10	SERIAL NR	182004	0x0002c6f4
UART: 0x14	ETH MAC	0:00:00:00:00:00	0x000000000000
UART: 0x1A	FEATURE FLAGS	0x00000000	0x00000000
UART: 0x1E	REZERVED	0x00000000	0x00000000
UART: 0x22	CHECKSUM	38	0x26

UART: EERPRON was written with success. Please hit 'q' key and 'enter'

LOS: Application terminated successfully.

P0:ALOS>q

moviDebug: INFO: Debugger exited.

Cleaning all built files from the MDK distribution and all project built files.


```
Batch execute: <echo $run_opt
>
run
Batch execute: <$run_opt>
Batch execute: <echo $exit_opt>

Batch execute: <$exit_opt>
UART: Start
UART: Start to read:
UART: EEPROM checksum is valid. Data from EEPROM:
UART: OFFSET      NAME          VALUE          HEX
UART: 0x00         ID            ID              0x4944
UART: 0x0          EEPROM FORMAT 1              0x01
UART: 0x03         REM. BYTES    28             0x0000001c
UART: 0x07         BOARD NAME MV0182          0x4d5630313832
UART: 0x0D         PCB REV       4              0x04
UART: 0x0E         MEC REV       0              0x00
UART: 0x0F         ELEC REV     2              0x02
UART: 0x10         SERIAL NR    182004         0x0002c6f4
UART: 0x14         ETH MAC      0:00:00:00:00:00 0x000000000000
UART: 0x1A         FEATURE FLAGS 0x00000000     0x00000000
UART: 0x1E         REZERVED     0x00000000     0x00000000
UART: 0x22         CHECKSUM     38             0x26
UART: Read was done with success. Please hit 'q' key and 'enter'
LOS: Application terminated successfully.
P0:ALOS>q
moviDebug: INFO: Debugger exited.
```

MV0198PowerMeasure

6.13.1. Software description

This example demonstrates how the power measure works. It runs applications on all processors:

- LOS (RTEMS) starts LRT and wait for board initialize.
- LRT perform next operations:
 - *wait 100 ms | no shaves is running
 - starts 1 SHAVES and wait for shave to finish | 1 shave is running
 - *wait 100 ms | no shaves is running
 - starts next 3 SHAVES and wait for shaves to finish | 3 shaves are running
 - *wait 100 ms | no shaves is running
 - starts next 6 SHAVES and wait for shaves to finish | 6 shaves are running
 - *wait 100 ms | no shaves is running
 - waits (100 ms) is used only to see in TraceProfiler the values when shaves doesn't run
- LOS start a thread that measure all rails

This application uses both the Leon and the Shave code.

Basically, these are the few steps made during the application:

1. The Clock Power reset module is initialized. The memory is also properly set.
2. The Shaves are started and stopped 5 times.
3. All rails are polled and recorded in ~25 ms interval. All shave events (e.g. RUN, RESET) are also recorded.

6.13.2. Expected output

The binary trace buffer are saved into the output folder:

- tracebuff_los.bin - contains the power measurement event
- tracebuff_lrt.bin - contains the shave events

These are pulled in automatically when the ./trace/TraceProfiler.mtrd file is opened in Eclipse as a log events graphics.

SimpleRTEMS_sdCard

6.14.1. Software description

The example initializes the SDIO Driver in Posix_Init and then creates and starts a thread that mounts a filesystem, creates a file, writes 5 MB data to the file, synchronizes, reads back 5 MB, verifies data, and finally unmounts the filesystem.

6.14.2. Hardware requirements

SDHC Cards - V2.0 onwards (High Speed supported). Cards must be formatted FAT32, maximum 32K Clusters for maximum performance.

6.14.3. Expected output

```
UART:
UART: RTEMS POSIX Started
UART:
UART: osBoard0182Initialise 0
UART:
UART: Sdio driver initialising
UART:
UART: OsDrvSdioInit sc RTEMS_SUCCESSFUL
UART:
UART: rtems_bdpart_register_from_disk sc RTEMS_SUCCESSFUL
UART:
UART: Mounting File System RTEMS_SUCCESSFUL
UART:
UART: Thread 1 created
UART:
UART: Creating file /mnt/sdcard/myfile
UART:
UART: Writing 5242880 bytes to file
UART:
UART: Perform fsync
UART:
```

```
UART: Closing file
UART:
UART: Opening file /mnt/sdcard/myfile
UART:
UART: Read 5242880 characters
UART:
UART: Verifying data...
UART:
UART: Card successfully unmounted
UART:
```

SimpleRTEMS_HTTPClient

6.15.1. Software description

The example initializes the network stack in POSIX_Init, prints out networks statistics and finally creates and starts a thread. Part of this initialization involves getting an IP address from the network DHCP and obtaining DNS servers and default gateway.

The thread will resolve an address and create a TCP connection against a www.google.com. It will issue a HTTP GET request and wait for the response. The request response will be printed out.

6.15.2. Hardware requirements

MV182 must be connected to a network with a DHCP server available.

6.15.3. Expected output

```
UART:
UART: rtems_leon_greth_gbit_driver_setup RTEMS_SUCCESSFUL
UART: dhcpc: gr_eth: inet: 192.168.86.16   mask: 255.255.252.0
UART:                srv: 192.168.85.30   gw: 192.168.85.1
UART:
UART: Resolving name to IP addresses:
UART:
UART:                *Host Address 0: 216.58.211.132
UART:
UART: Connection is successful to 216.58.211.132.
UART: HTTP request to www.google.com: 18 bytes sent
UART: HTTP response:
UART:
UART:
*****
****
UART:
UART: HTTP/1.0 302 Found
UART: Location:
http://www.google.ie/?gws_rd=cr&ei=tvrhVKbZDITa7gaZp4E4
UART: Cache-Control: private
UART: Content-Type: text/html; charset=UTF-8
UART: Set-Cookie:
PREF=ID=467f11172c49746a:FF=0:TM=1424095926:LM=1424095926:S=
```

```
UART: DPjklbce3hTvcVT4; expires=Wed, 15-Feb-2017 14:12:06 GMT;
path=/; domain=
UART: .googlee3hTvcVT4; expires=Wed, 15-Feb-2017 14:12:06 GMT;
path=/; domain=.google.com
UART: Set-Cookie: NID=67=AsrUNOt28wvuY_UutIfhd3pvGd-
_fUvvWSa0jokUjbbbyVcJptFt2m
UART: IWLXcCF5qoOPD-
HgOMf8KlrDv3ar8rFvVBbgsscizamXmxo841zBo3iV6r0bSXQCu941RIUru
UART: k0f; expires=Tue, 18-Aug-2015 14:12:06 GMT; path=/;
domain=.google.com; Http expires=Tue, 18-Aug-2015 14:12:06 GMT;
path=/; domain=.google.com; HttpOnly
UART: P3P: CP="This is not a P3P policy! See
http://www.google.com/support/acc
ounts/bin/answer.py?hl=en&answer=151657 for more info."
UART: Date: Mon, 16 Feb 2015 14:12:06 GMT
UART: Server: gws
UART: Content-Length: 256
UART: X-XSS-Protection: 1; mode=block
UART: X-Frame-Options: SAMEORIGIN
UART: Alternate-Protocol: 80:quic,p=0.08
UART:
UART: <HTML><HEAD><meta http-equiv="content-type"
content="text/html; charset=u
UART: tf-8">
UART: <TITLE>302 Moved</TITLE></HEAD><BODY>
UART: <H1>302 Moved</H1>
UART: The document has moved
UART: <A
HREF="http://www.google.ie/?gws_rd=cr&ei=tvrrhVKbZDITa7gaZp4E4">h
e
UART: re</A>.
UART: </BODY></HTML>
UART:
UART:
*****
****
UART:
```

powerMeasurementsShavesIslands

6.16.1. Software description

The tests is measuring a list of voltage and current rails, it compares them with present values (ranges) and give a final report related to these measurements.

6.16.2. Hardware requirements

These tests require a MV198 board connected to the PowerMon connector of the MV182 board.

6.16.3. Expected output

On debug console there will be printed a list of 5 comma separated voltage and current measurements (the first row also giving the measurement acronym). At the end of the measurements, the test result will also be printed (passed or failed).

The measurements list can be saved into a .csv file (after removing the extra "UART:" leading text)

Typical output as follows:

```
UART: Measure all shaves island turned on, then stopping one island
at a time
UART:
UART:
UART: VDDCV_I_MA, VDDCV_V_MV, VDDCC_I_MA, VDDCR_I_MA, VDDIO_I_MA
UART: 498.33, 889.00, 47.00, 1.35, 0.20
UART: 480.00, 889.00, 47.00, 1.35, 0.20
UART: 461.67, 889.00, 47.00, 1.35, 0.20
UART: 443.33, 890.00, 47.00, 1.35, 0.20
UART: 425.00, 890.00, 47.00, 1.35, 0.20
UART: 405.00, 890.00, 47.00, 1.35, 0.20
UART: 385.00, 891.00, 47.00, 1.35, 0.20
UART: 366.67, 891.00, 47.00, 1.35, 0.20
UART: 346.67, 891.00, 47.00, 1.35, 0.20
UART: 328.33, 891.00, 47.00, 1.35, 0.20
UART: 308.33, 892.00, 47.00, 1.35, 0.20
UART: 290.00, 892.00, 47.00, 1.35, 0.20
UART: 271.67, 892.00, 47.00, 1.35, 0.10
UART:
UART: moviUnitTest:PASSED
LOS: Application terminated successfully.
```

■ SPI_WiFi_test_TI_C33XX

6.17.1. Software description

This test evaluates the SPI connection between the Myriad2 and the CC3100 Wi-Fi module, both on the EoTDevBoardRev1.

This application uses only the Leon code. The application does the following:

1. The generic MA2100 setup stage is completed.
2. The integrity of the SPI connection between Myriad2 and CC3100 is tested.
3. Results at each stage are printed to the standard output.

6.17.2. Expected output

The result consists of a printf message of the test result for each stage of the SPI test.

```
UART: Spi Test Begin
UART: Setting SPI Clock speed to 0x40000000MHz
UART: Spi Open Passed
```

UART: Device Disable Passed
UART: Device Enable Passed
UART: Host IRQ Passed
UART: Spi Write Passed
UART: Spi Read Passed
UART: Spi Init read complete Passed
UART: Spi Test Completed
UART:

WiFi_AP_test_TI_CCXX

6.18.1. Software description

This test configures the CC3100 Wi-Fi module on the EoTDevBoardRev1 in access point mode, then pings a device that connects to it over Wi-Fi.

This application uses only the Leon code. The application does the following:

1. The generic MA2100 setup stage is completed.
2. The device is configured in access point mode.
3. The board awaits a Wi-Fi connection from a device.
4. When a connection is made the board pings the device and verifies correct connection.

6.18.2. Expected output

The result consists of printf messages showing the access point configuration and status, and indicates pinging of connected devices and successful connection.

```
UART: *****
Setting SPI clock speed to 0x40000000MHz.
UART: Device is configured in default state
Setting SPI clock speed to 0x40000000MHz.
UART: Setting SPI clock speed to 0x40000000MHz.
UART: Device started as Access Point
      Waiting for clients to connect...!
      Client connected to the device
      Pinging...!
      Device and the station are successfully connected
```

WiFi_Get_Time_test_TI_CCXX

6.19.1. Software description

This test connects the Wi-Fi module to a local Wi-Fi access point and acquires the current time from a time server over http.

The Wi-Fi access point SSID and password must be specified by the user (common/wifi_defaults.h).

This application uses only the Leon code. The application does the following:

1. The Clock Power reset module is initialized.
2. The DDR is also initialized.
3. The L2 Cache must be invalidate.
4. On this step, the setup is complete. The desired message can now be printed.
5. Time is acquired from remote time server and printed to the standard output.
6. Programme repeats.

6.19.2. Expected output

The result consists of a continuous stream of printf messages of the current date and time.

```
UART:
UART: Server 0.in.pool.ntp.org has responded with time information
UART:
UART: Thu Nov 19 2015 9:4:56
UART:
```

Motor_Control_GPIO/EN/PWN/I2C_test

6.20.1. Software description

This application performs tests on the GPIO and I2C pins used to control a 5V motor. An Arduino board simulates the presence of a motor and processes the data from the Myriad in the 5V interface and shows the results on a serial port.

At the end of the GPIO and I2C tests, the user will be informed if the tests passed or failed.

6.20.2. Hardware requirements

- Arduino Uno board and a serial terminal.

6.20.3. Arduino connections

Arduino pin	Motor header
A4	I2C_SDA
A5	I2C_SCL
8	DIR0
9	DIR1
10	BRAKE0
11	BRAKE1
12	PWM0
13	PWM1

6.20.4. Expected output from the debugger

The testing will be repeated indefinitely until the user exits the debugger.

```
UART: ----- Motor control test app -----
UART: Please connect the Arduino board via USB and open a serial
terminal
UART: with the following settings: 9600 baud, 8 bits, parity: none,
1 stop
UART:
UART: START: GPIO TEST
UART: Test status should be verified on the Arduino board
UART: Toggling pin status ...
```



```
UART:  Toggling pin status ...
UART:  Toggling pin status ...
UART:  END: GPIO TEST
UART:
UART:  START: I2C TEST
UART:  Writing on the Arduino board ...
UART:  DONE!
UART:  Reading from the Arduino board ...
UART:  DONE!
UART:  ***** PASSED *****
UART:  END: I2C TEST
UART:
```

6.20.5. Expected output on the Arduino

The Arduino will wait for any board to be connected and start the testing indefinitely.

```
----- Motor control test app -----
START: GPIO TEST
***** PASSED *****
END: GPIO TEST
START: I2C TEST
Sent OK to master
END: I2C TEST
```

■ InitClocksAndMemory

6.21.1. Software description

Test to configure clocks, DDR and SDRAM. The application initialises the required system clocks and memory, and provides progress output over JTAG.

6.21.2. Expected output

```
UART: Stage 1...
UART: Stage 2...
UART: Stage 3...
UART: Stage 4...
UART: Stage 5...
UART: Stage 6...
UART: initClocksAndMemory() completed successfully
LOS: Application terminated successfully.
```

■ SPI_flash_test

6.22.1. Software description

Test to flash an application (PM LEDs blink), run it, and then erase it. The application sets LED1 and resets LED2 and vice-versa by writing in the respective registers of the PMIC slave.

6.22.2. Expected output

6.23.2. Hardware required

An USB cable connection from EoT DevBoard to a PC host.

6.23.3. Expected output

The debug console should show the following messages:

UART:

UART: Configuring the SIPP pipeline

UART: Configuring imx208 dual cameras and datapath

UART: +UsbPumpRtems_UsbPumpInit_ServiceTask:

UART:

Compile and run USBCapture.exe to view the output in 260 x 260 format (the native format).

For compatibility with other USB video display applications, the streamed video is in the 640 x 480 pixels format. When the application starts, the PC host should identify the camera and any of the below applications can be used to view the camera output.

- Windows: amcap, vlc, skype
- Linux: cheese, guvcview, xawtv, skype

Results

Test number & status on MV0182	Test order & status on EoT Rev1	Test name:	Failed	Passed	Not needed	Requirement	Unknown expected result	Test set 0.01 (Date 02.04.2015)	Execution time (in minutes)	Required for EoT Rev1 board	Test Creator
			Available test on hardware type revision	Test set 0.05 (Date 12.10.2015)	Test set 0.04 (Date 27.08.2015)	Test set 0.03 (Date 22.07.2015)	Test set 0.02 (Date 03.04.2015)				
1	-	ts_ahb_drv_00	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
2	-	cd	R3/R4	No	No	Yes	Yes	Yes	-	N	Movidius MV182
3	9	PowerActiveBaseline	R4	Yes	No	No	No	No	6	Y	Movidius MV182
4	-	Cam208CvHdmi	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
5	-	Cam214CtrldDebayerHdmi	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
6	-	Cam214CVCtrldDebayer	R3/R4	Yes	Yes	Yes	Yes	Yes	4	N	Movidius MV182
7	-	Cam214OpipeDebayerHdmi	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
8	13	DisparityMap	R3/R4	Yes	Yes	Yes	Yes	Yes	6	Y	Movidius MV182
9	-	SimpleCamHDMI	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
10	-	Stereo208Hdmi	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
11	10	UsbVideo208	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
12	4	BMP180	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
13	5	BMX055	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
14	6	DipSwitchGpio	R3/R4	Yes	Yes	Yes	Yes	Yes	4	Y	Movidius MV182
15	-	EthernetR2	R3/R4	Yes	Yes	Yes	Yes	Yes	4	N	Movidius MV182
16	-	I2cTestR3	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
17	-	LcdNewHdmiDiffColorBars	R3/R4	Yes	Yes	Yes	Yes	Yes	3	N	Movidius MV182
18	1	PmicLedBlink	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
19	2	RemoteControl	R3/R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
20	7	I2CTestR4CAMBOnI2C0AndI2C1	R4	Yes	Yes	Yes	Yes	Yes	3	Y	Movidius MV182
21		ts_host_test501_bulkout	R3/R4	Yes	Yes	Yes	Yes	Yes	4	Y	Movidius MV182

22		spiCommunication	R3/R4	Yes	Yes	Yes	Yes	Yes	7	Y	Movidius MV182
23	3	RevisionWriterEEPROM	R4	Yes	Yes	Yes	Yes	Yes	4	Y	Movidius MV182
24	11	MV0198PowerMeasure	R3/R4	Yes	Yes	Yes	Yes	No	10	Y	Movidius MV182
25	12	SimpleRTEMS_sdCard	R3/R4	Yes	Yes	Yes	Yes	No	3	Y	Movidius MV182
26		simpleRTEMS_HTTPClient/	R3/R4	Yes	Yes	Yes	No	No	4	Y	Movidius MV182
27		ts_los_ddr_08	R3/R4	Yes	Yes	Yes	No	No	4	Y	Movidius MV182
28	8	powerMeasurementsShavesIslands	R4	Yes	Yes	No	No	No	3	Y	Movidius MV182
29	-	Check the HDMI output with the HDMI analyzer.	R3/R4	Yes	Yes	No	No	No	5	N	Movidius MV182
30	15	SPI_WiFi_test_TI_C33XX							1	Y	Alireza Dehghani
31	16	WiFi_AP_test_TI_CCXX							1	Y	Aubrey Dunne
32	17	WiFi_Get_Time_test_TI_CCXX							1	Y	Aubrey Dunne
33	-	SPI_WiFi_test_TI_Broadcom_WiCED							-	N	Alireza Dehghani
34		Motor_Control_GPIO/EN/PWN /I2C_test								Y	Dexmont Pena
43	14	InitClocksAndMemory								Y	Aubrey Dunne
44		SPI flash test								Y	Alireza Dehghani
45		NanEye0 External Interface Test (Mode B)								Y	Aubrey Dunne

Motor control tests failed due to missing tracks on board. I2C2 connections to level shifter for motor control were disconnected. This was corrected with a small cable soldered on the board.

7. INTEGRATION TESTS

■ Introduction

In this section, we describe the integration tests, i.e. non-unit tests that ensure that major modules can work together. The core software module in EoT is 'Control mode' (also known as 'Pulga'), which includes access to Wifi, camera, SD card and Flash memory. These capabilities have been developed by different partners. Some of the hardware components share a common hardware component, so the tests also allow to ensure that no conflict exists or, if it exists, it can be solved. As an example, WiFi and Flash did not work together in the MV0182 board-based prototype that was used at the start of the project, which only had a single SPI interface available. This was later corrected in Rev1 board (also known as 'DevBoard'). All of the integration tests required intervention from the tester.

■ Integration Tests on the MV0182 board-based prototype

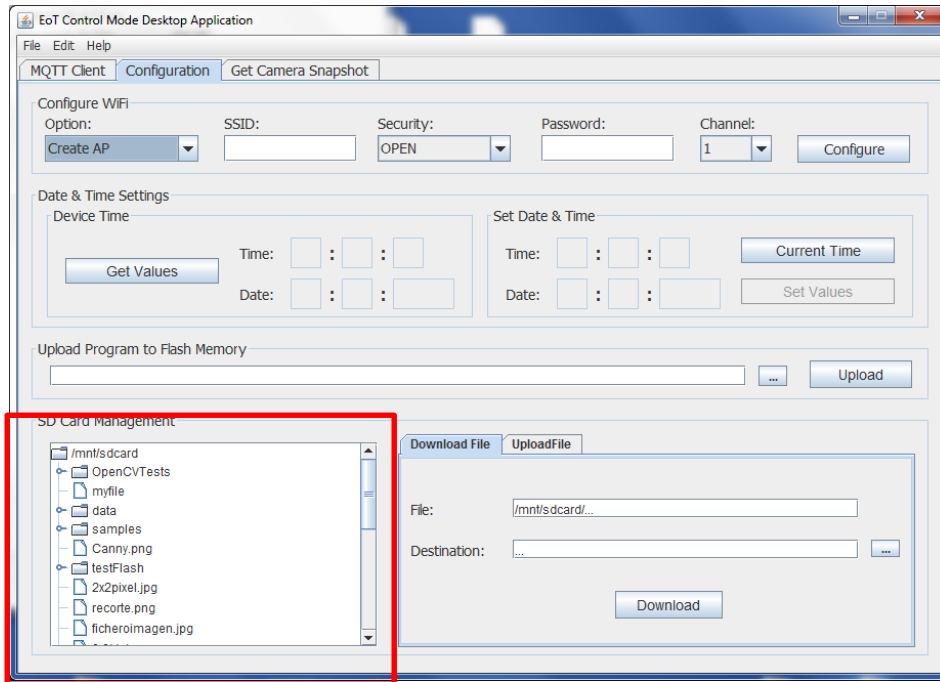
7.2.1. Control mode + SD

7.2.1.1. Software description

This test starts Pulga and allows the user to access the SD card filesystem through an MQTT client such as the EoT Android or desktop clients. The application mounts the SD card filesystem, lists directories, removes files and folders and uploads new files to the SD card.

7.2.1.2. Expected output

If the EoT Control Mode desktop application is used, the content of the SD card is shown in the Configuration tab as follows:



The output of the debug console is:

```
UART: Failed to get a profile
UART: No profile found on index 0
UART: Starting Pulga MQTT Broker: PulgaMqttBrokerControl
UART: Fri Jan 1 00:00:04 1988
UART: Waiting
UART: socket 0 was ready
UART: Handle message
UART: *****
UART: FRC value -1
UART: *****
UART:
UART: socket 16 was ready
UART: Accepting connection
UART: Got a connection on port 24735
UART: Fri Jan 1 00:00:41 1988
UART: Waiting
UART: socket 1 was ready
UART: Handle message
UART: *****
UART: FRC value 1
UART: *****
UART:
UART: Message received on socket 1
UART: Message Received (ID of client):
UART: -2146021026
UART:
UART: Connected clients socket:
UART: value: 1
UART: socket 17 was ready
UART: Handle message
UART: *****
```

```
UART: FRC value 8
UART: *****
UART:
UART: Subscribe received to topic EOTListFilesSD
UART: Subscribe received to topic (length topic) -2146021216
UART: requestedQoSs 0
UART: Count 1
UART: RC 1
UART: Initial pos of topic: -1
UART: Final pos of topic: 0
UART: value: 17
UART: Fri Jan 1 00:00:42 1988
UART: Waiting
UART: socket 1 was ready
UART: Handle message
UART: *****
UART: FRC value 3
UART: *****
UART:
UART: Message received on topic EOTListFilesSD/mnt/sdcard:
UART: /mnt/sdcard
UART: socket 17 was ready
UART: Handle message
UART: *****
UART: FRC value -1
UART: *****
UART:
UART: Fri Jan 1 00:00:43 1988
UART: Waiting
```

7.2.2. Control mode + Camera

As explained in deliverable D3.3, the Camera module operates using the I2C buses for the communication with sensors. This module needs to configure the I2C buses through the *BoardInitialise* function and the interrupt service routine (ISR) to periodically update the new buffer address where the image is stored.

These aspects should be taken into account when using this module together with Pulga. Pulga is based on the WifiFunctions library. This library uses the I2C buses to send the nHIB signal to the WiFi chip and, therefore, there are some conflicts while using WiFi and Camera together.

From the first integration tests we concluded that the Camera I2C buses initialization should be done before starting the WiFi configuration. The WifiFunctions I2C configuration does not affect the previous I2C configuration of the Camera module but using *BoardInitialise* after initialising the WiFi chip spoils WiFi communication.

On the other hand, with regard to the Camera ISR, the WiFi driver disables all interrupts during WiFi chip initialization. Therefore, initialising the WiFi while the camera is capturing frames can lead to blocking of the interrupt that updates the image buffer.

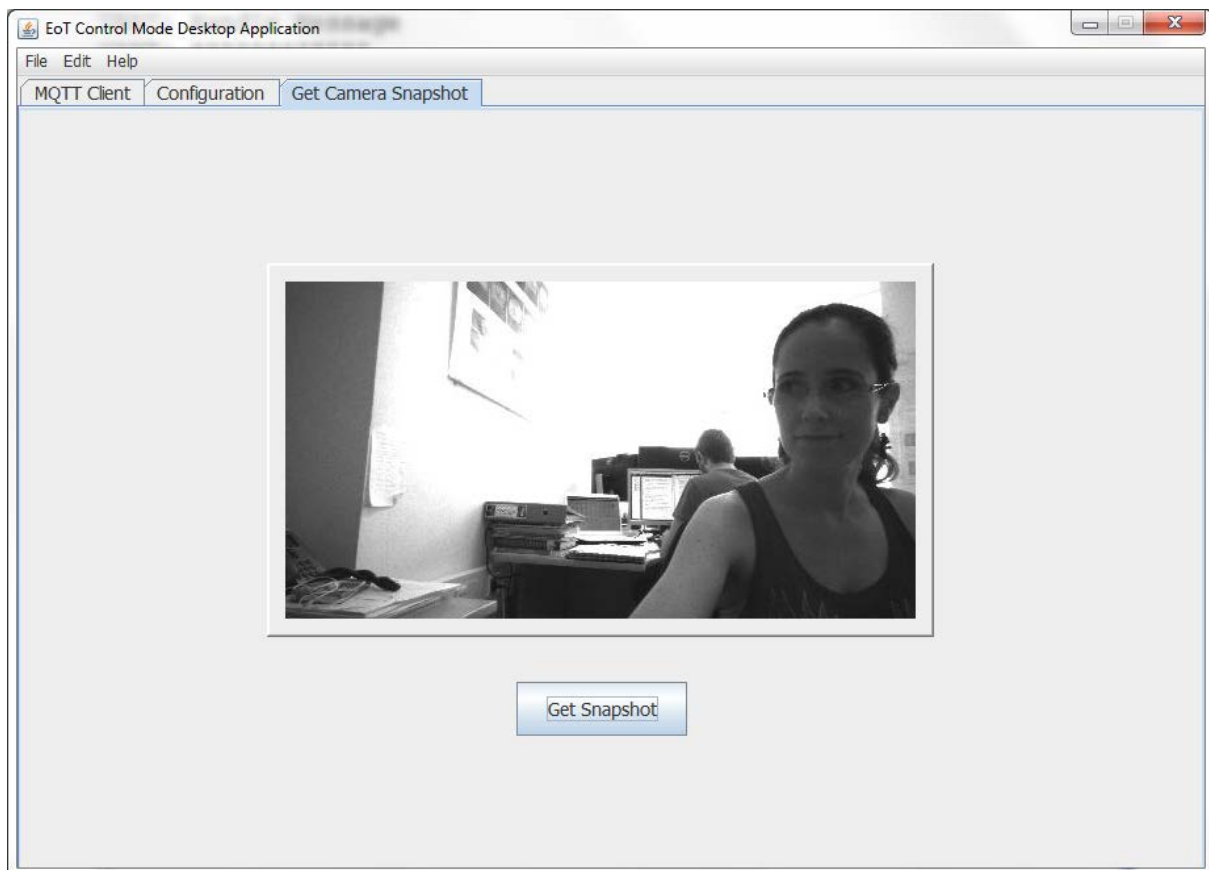
These problems were resolved by ensuring that the *init_camera* method is called after initialising the WiFi chip.

7.2.2.1. Software description

This test starts Pulga and allows the user to access and retrieve a camera frame through EoT Android or Desktop clients. The application makes all the initializations, accesses the camera buffers using the Camera module and sends the image encapsulated into several MQTT messages.

7.2.2.2. Expected output

The output in the EoT Control Mode desktop application should be an image like the following one:



The output of the debug console is:

```
UART: Subscribe received to topic EOTSnapshot
UART: Subscribe received to topic (length topic) -2146021216
UART: requestedQoSs 0
UART: Count 1
UART: RC 1
UART: Initial pos of topic: 1
UART: Final pos of topic: 1
```

```
UART: value: 1
UART: socket 17 was ready
UART: Handle message
UART: *****
UART: FRC value 3
UART: *****
UART:
UART: Message received on topic EOTSnapshotEOT - MQTT Client:
UART: EOT - MQTT Client
UART: Thread 1 created
UART: number_of_packets: 14
UART: Snapshot sent
UART: Fri Jan 1 00:01:23 1988
UART: Waiting
```

7.2.3. Control mode + Flash

As mentioned above, WiFi and Flash did not work together in the MV0182 board-based prototype due to the existence of only one SPI interface. Therefore, in the early MV0182-based prototype it is not possible to run applications that use WiFi and Flash at the same time.

7.2.4. Bootloader + Control mode

Since the bootloader's task is to either boot into control mode or to load and run a user application, one of the integration tests should consist of flashing the Bootloader together the Control mode (i.e. Pulga) application. To this end, a new *flasher* tool was developed by DFKI, since Movidius' flashing tool only allowed to flash a single binary.

The choice what to boot is made through the use of a DIP switch on the physical board. The bootloader is always stored at the beginning of the flash memory and the Control mode executable is stored at a fixed location after the bootloader (see D3.3 document for more information). This integration test was completed successfully by DFKI.

Integration Tests on Rev1 board

The Rev1 board introduced some minor hardware changes. In particular, two SPI interfaces were devoted to WiFi and Flash (this allowed us to test Pulga+Flash). On the other hand, a number of GPIOs changed. After taking this into account (and changing the configuration of the GPIOs that were used by the WiFi driver), all the above integration tests were run on the revision 1 board once it was available.

7.3.1. Control mode + SD

The output, in this case, is the same as the output obtained when running the test on the MV0182 board.

7.3.2. Control mode + Camera

The new board does not introduce changes in the way the camera is accessed. Therefore, the output of this test using the revision 1 board is the same as the output using the MV0182 board.

7.3.3. Control mode + Flash

This test performs some reads and writes in the flash memory while Pulga is running to ensure that WiFi and flash can be accessed simultaneously. Since the revision 1 board has separate SPI interfaces for WiFi and Flash, this test can now be run.

If this test is executed and a MQTT client is connected to Pulga, the expected output of this test is:

Board initialization:

```
UART: GPIO initialised
UART: GPIO initialised
UART: Setting SPI clock speed to 0x0 MHz.
UART: Setting SPI clock speed to 0x0 MHz.
UART: Setting SPI clock speed to 0x0 MHz.
```

WiFi and Camera initialization:

```
UART: Failed to get a profile
UART: No profile found on index 0
UART: Setting SPI clock speed to 0x0 MHz.
UART: Configuring camera and datapath
```

Flash memory test:

```
UART: size saved: 250
UART: size saved: 10
```

Pulga continues working and accepting connections:

```
UART: Starting Pulga MQTT Broker: PulgaMqttBrokerControl
UART: Fri Jan 1 00:00:04 1988
UART: Waiting
UART: socket 0 was ready
UART: Handle message
UART: *****
UART: FRC value -1
UART: *****
UART:
UART: socket 16 was ready
UART: Accepting connection
UART: Got a connection on port 24735
UART: Fri Jan 1 00:00:41 1988
UART: Waiting
UART: socket 1 was ready
```

```
UART: Handle message
UART: *****
UART:  FRC value 1
UART: *****
UART:
UART: Message received on socket 1
UART: Message Received (ID of client):
UART:  -2146021026
UART:
UART: Connected clients socket:
UART: value: 1
UART: socket 17 was ready
UART: Handle message
UART: *****
UART:  FRC value 8
UART: *****
```

7.3.4. Bootloader + Control mode + Flash

In this test the bootloader and the control mode app (Pulga) were flashed on the EEPROM of the EoT device using the EoT flashing tool. Then, DIP switch 0 was set to boot the control mode app, which creates a WiFi access point. Using the EoT Android or desktop clients to connect to the access point, the ELF file of another application was transferred wirelessly to the EoT board. After switching the DIP switch to boot into the user application, which was installed just before, the EoT board is rebooted. In this setup the EoT device is not connected physically with the PC. As a consequence, it is not possible to print messages to the PC's console. In order to be able to verify that the transfer and booting of the user app was performed correctly, the LED control test app was chosen as the user application. It allows the user to control the LEDs on the EoT device through the push buttons. Following this procedure, it could be verified, that the LEDs could be controlled through the push buttons as expected. In total this verifies, that the bootloader correctly switches between control mode and user application according to the state of the first DIP switch. Furthermore, it shows that the control mode and a user app function correctly, when loaded from the EEPROM by the EoT bootloader.

8. CONCLUSIONS

This deliverable describes all software tests performed in EoT so far. The tests are divided into: 1) Firmware tests, i.e. unit tests for the different modules developed as part of EoT's firmware, 2) Prototype tests, i.e. test sets applied to the hardware prototypes and 3) Integration tests, i.e. non-unit tests that check that major modules can work together. The results of these tests have been satisfactory so far.

9. GLOSSARY

AON	Always ON
AP	Access Point
API	Application Programming Interface
BRISK	Binary Robust Invariant Scalable Keypoints
BSD	Berkeley Software Distribution
CISC	Complex Instruction Set Computing
CNN	Convolutional Neural Network
CSS	CPU Sub-System
DDR	Double Data Rate
DIP	Dual in-line Package
DMA	Direct Memory Access
DSS	DDR Sub-System
DoW	Description of Work
GPIO	General Purpose Input/Output
GPL	General Public License
HD	Hard Disk
HOG	Histogram of Oriented Gradients
HTTP	Hypertext Transfer Protocol
HW	Hardware
I2C	Inter-Integrated Circuit
I2S	Integrated Interchip Sound
IO	Input/Output
ISR	Interrupt Service Routine
JPEG	Joint Photographic Experts Group
LED	Light-Emitting Diode
LK	Lucas-Kanade
MAC	Media Access Control
MDK	Movidius Development Kit
MIPI	Mobile Industry Processor Interface
MQTT	Message Query Telemetry Transport
MSS	Media Sub-System
MvCv	Movidius Computer Vision
OS	Operating System
PC	Personal Computer
PDF	Portable Document Format
PMB	Processor Memory Block
PNG	Portable Network Graphics
POSIX	Portable Operating System Interface
QR	Quick Response
REST	Representational State Transfer
RISC	Reduced Instruction Set Computing
RTEMS	Real-Time Executive for Multiprocessor Systems
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
SD	Secure Digital
SIFT	Scale-Invariant Feature Transform

SoC	System on Chip
SPI	Serial Peripheral Interface
SVM	Support Vector Machines
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TI	Texas Instruments
VPU	Vision Processing Unit
WAV	Waveform Audio File
WEP	Wired Equivalent Privacy
WPA	WiFi Protected Access

- End of document -