Test Case	Scenario	Expected Results	Status
Transmit data from master MCU to slave MCU Using I2C protocol **Procedure Steps:** 1- **Input**: message from master MCU 2- **Input**: MCU slave address 3- Transmit message to slave MCU	1- **Message:** “Hello World”	“Hello World”	**Pass** Actual results are similar to expected results
	2- **Slave address:** 0x06	Message transmitted successfully	
Retrieve data from slave MCU to master MCU using I2C protocol **Procedure Steps:** 1- **Input**: MCU slave address 2- Retrieve message from slave MCU 3- Store message in storage	1- **Message:** “Hello World”	“Hello World”	**Pass** Actual results are similar to expected results
	2- **Slave address:** 0x06	Message retrieved successfully	
Master- Slave Communication Handling			
Master MCU constructs slave MCU and initialise preconfigured parameters **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input:** MCU slave address 3- **Input:** sleep option for slave MCU Idle mode = 1 Power-down mode = 2 4- **Input:** sleep hours for slave MCU 5- **Input: s**leep minutes for slave MCU 6- **Input:** sleep seconds for slave MCU 7- Initialise slave MCU status flags	1- **Slave MCU address:** 0x06 2- **Sleep configuration:** **Option:** 2 **Hours:** 0 **Minutes:** 30 **Seconds:** 0	Initialisation is successful and slave MCU object instance is activated	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU checks number of external sensors on slave MCU **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** 3- Master MCU retrieves response message from slave MCU and save it into storage - **Message length in bytes** - **Response command ID** - **Message status return code from slave MCU** - **Number of connected sensors**	**Correct request message parameters:** 1- **Message length:** 1 byte 2- **Request command ID:** 1	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 3 bytes **Response command ID:** 21 **Return code:** Message is OK, understood and executed 0x00 **Number of connected sensors on slave MCU:** 4 4- slave sensors Info checked, return **SUCCESS**	**Pass** Actual results are similar to expected results
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**)	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01 **Invalid sensors number:** Tested with invalid numbers **(**maximum number of external sensors on slave MCU **is restricted to 4 sensors** which can be adapted in further updates**)**	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results
Master MCU resets slave MCU in case of error issue **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** 3- Master MCU retrieves response message from slave MCU and save it into storage - **Message length in bytes** - **Response command ID** - **Message status return code**	**Correct request message parameters:** 1- **Message length:** 1 byte 2- **Request command ID:** 2	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 2 bytes **Response command ID:** 22 **Return code:** Message is OK, understood and executed 0x00 4- slave MCU reset, return **SUCCESS**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**)	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	Pass Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU sets slave MCU into ready mode to start measuring process **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** 3- Master MCU retrieves response message from slave MCU and store it - **Message length in bytes** - **Response command ID** - **Message status return code**	**Correct request message parameters:** 1- **Message length:** 1 byte 2- **Request command ID:** 3	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 2 bytes **Response command ID:** 23 **Return code:** Message is OK, understood and executed 0x00 4- slave MCU is ready to measure, return **SUCCESS**	**Pass** Actual results are similar to expected results
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**)	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	<u>**Malformed response message parameters:**</u> **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results
Master MCU sets slave MCU into sleep mode after measuring process is done <u>**Procedure Steps:**</u> 1- **Input**: pointer to the allocated slave MCU object instance 2- Master MCU transmits request message defined in **packet** to slave MCU - **Message length** - **Request command ID** - **Sleep mode** - **Sleep hours** - **Sleep minutes** - **Sleep seconds** 3- MCU master retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Message status return code**	<u>**Correct request message parameters:**</u> 1- **Message length:** 5 bytes 2- **Request command ID:** 9	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 2 bytes **Response command ID:** 29 **Return code:** Message is OK, understood and executed 0x00 4- Slave MCU slept, return **SUCCESS**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sleep mode** Tested with invalid values **Invalid sleep duration** **Values** > maximum size of duration data types(**255**)	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU constructs structure for external slave sensor and initialise related preconfigured parameters **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Initialise sensor parameters - **Status flags** - **Maximum measurement time** - **Type of measurement** - **Measurement data** - **Data size** - **Timestamp of measurement**	1- **Pointer to slave MCU** 2- **Sensor ID number** Tested with multiple IDs within the range of availability on slave MCU	Initialisation of selected sensor parameters on slave MCU is successful	**Pass** Actual results are similar to expected results
Master MCU activates selected sensor on slave MCU **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** - **Sensor ID number** 4- Master MCU retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Message status return code**	**Correct request message parameters:** 1- **Message length:** 2 bytes 2- **Request command ID:** 4 3- **Sensor ID number** Tested with multiple IDs within the range of availability on slave MCU	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 3 bytes **Response command ID:** 24 **Return code:** Message is OK, understood and executed 0x00 4- Slave sensor activated, return **SUCCESS**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sensor ID number** Tested with undefined IDs	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** **er**ror 0x01 **Return code:** sensor not available 0x02	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU checks maximum measuring time for selected sensor on slave MCU **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** - **Sensor ID number** **4-** Master MCU retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Message status return code** - **Selected sensor ID number**	**Correct request message parameters:** 1- **Message length:** 2 bytes 2- **Request command ID:** 5 3- S**ensor ID number** Tested with multiple IDs within the range of availability on slave MCU	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 4 bytes **Response command ID:** 25 **Return code:** Message is OK, understood and executed 0x00 4- Maximum measuring time checked for selected slave sensor, return **SUCCESS**	**Pass** Actual results are similar to expected results
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sensor ID number** Tested with undefined IDs	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01 **Return code:** sensor not available 0x02	1- **Error**: Retrieval Communication - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results
	- Invalid **Measurement time** (more than) maximum measurement time size (255)	Invalid Maximum measuring time return **ERROR**	**Pass** Actual results are similar to expected results **Comment**: Data size can be upgraded from 8-bit to 16-bit if required
Master MCU triggers sensor on slave MCU **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** - **Sensor ID number** 4- Master MCU retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Message status return code** - **Selected sensor ID number**	**Correct request message parameters:** 1- **Message length:** 2 bytes 2- **Request command ID:** 6 3- **Sensor ID number** Tested with multiple IDs within the range of availability on slave MCU	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** **3** bytes **Response command ID:** 26 **Return code:** Message is OK, understood and executed 0x00 4- slave sensor is triggered, return **SUCCESS**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sensor ID number** Tested with undefined IDs	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01 **Return code:** sensor not available 0x02	1- **Error**: Retrieval Communication - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU checks if measurement data of slave sensor is ready to be retrieved after waiting for maximum measuring time to avoid any violation **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** - **Sensor ID number** 4- Master MCU retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Message status return code** - **Selected sensor ID number**	**Correct request message parameters:** 1- **Message length:** 2 bytes 2- **Request command ID:** 7 3- S**ensor ID number** Tested with multiple IDs within the range of availability on slave MCU	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 3 bytes **Response command ID:** 27 **Return code:** Message is OK, understood and executed 0x00 4- Measurement ready status is checked, return **SUCCESS**	**Pass** Actual results are similar to expected results
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sensor ID number** Tested with undefined IDs	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of retrieve buffer (**255**) **Return code:** error 0x01 **Return code:** sensor not available 0x02 **Return code:** measurement unavailable 0x03 **Return code:** Sensor busy ongoing measurement 0x04	1- **Error**: Retrieval Communication - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Master MCU retrieves measurement data content, size, type and timestamp of selected sensor on slave MCU **Procedure Steps:** 1- **Input**: pointer to the allocated slave MCU object instance 2- **Input**: selected sensor ID number 3- Master MCU transmits request message defined in **packet** to slave MCU - **Message length in bytes** - **Request command ID** - **Sensor ID number** 4- Slave MCU retrieves response message from slave MCU and save into storage - **Message length in bytes** - **Response command ID** - **Measurement data type** - **Measurement data size** - **Measurement timestamp in milliseconds** - **Measurement data** - **Selected sensor ID number**	**Correct request message parameters:** 1- **Message length:** 2 bytes 2- **Request command ID:** 8 3- S**ensor ID number** Tested with multiple IDs within the range of availability on slave MCU	1- Message is ok and transmission is complete 2- Message is ok and retrieval is complete 3- **Response message:** **Message length:** 12 bytes + sensor measurement data size **(restricted to max 12 bytes)** **Response command ID:** 28 (1 byte) **Measurement data type:** predefined type value (1 byte) **Measurement timestamp: In milliseconds** (8 bytes) **Measurement size:** up to 12 bytes (represented in 1 byte) **Measurement data:** Tested with random values generated from MATLAB (up to 12 bytes) **Return code:** Message is OK, understood and executed 0x00 (1 byte) 4- Measurement data is retrieved, return **SUCCESS**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed response message parameters: **Undefined response command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values **0** **Values** > maximum size of retrieve buffer (**255**) **Invalid measurement data:** > restricted maximum size **12** **Invalid measurement size:** > restricted maximum size **12 bytes** **Undefined measurement type:** Handled as generic type 0x00 **Return code:** error 0x01 **Return code:** sensor not available 0x02 **Return code:** measurement unavailable 0x03 **Return code:** Sensor busy ongoing measurement 0x04	1- **Error**: Retrieval - Invalid data packet - Insufficient buffer size - Invalid response command 2- return **ERROR** **Comment**: **measurement size can be adapted and increased to be more than the restricted maximum size of 12 bytes. However, it must be less than the retrieve buffer size**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
	Malformed request message parameters: **Undefined request command ID:** Tested with multiple undefined values **Invalid message length:** Tested with invalid values 0 **Values** > maximum size of transmit buffer (**255**) **Invalid sensor ID number** Tested with undefined IDs	1- **Error: transmission** - Invalid request command - Invalid data packet request 2- **Response message** **Message length:** 1 byte **Response command ID:** 0 **Return code:** Invalid Packet 0x05 Message Error 0x01 3- return **ERROR**	**Pass** Actual results are similar to expected results
Scheduler			
Task is added to scheduler that can be called from any context **Procedure Steps:** 1- **Input:** pointer to task descriptor to be added 2- Traverse through the task linked list 3- Add the task to the linked list	Task added to test functionality of scheduler >>>> Led blinking in a periodic cycle (every 5 seconds)	Led blinked every 5 seconds	**Pass** Actual results are similar to expected results
Task is removed from scheduler **Procedure Steps:** 1- **Input:** pointer to task descriptor to be removed 2- Traverse through the task linked list 3- Remove the task from the linked list	Task removed to test functionality of scheduler >>>> Turn on green led and remove task after 10 seconds	Green led is turned off after 10 seconds	**Pass** Actual results are similar to expected results
Initialise Scheduler with preconfigured hardware peripheral timer and update tasks **Procedure Steps:** 1- Register a callback function which updates scheduler every 1 millisecond 2- Traverse through the task list 3- Reset expiration time of periodic tasks	1- Multiple tasks added to be updated synchronously 2- Functionality and timing response observed	Scheduler is updated every 1 millisecond and executes tasks at specified time points	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Run Scheduler in super loop **Procedure Steps:** 1- Loop over the tasks and check if there is a task ready to be executed 2- Remove non-periodic tasks from the scheduler 3- Pass and store task parameters 4- Execute ready tasks		Super loop over executable tasks	**Pass** Actual results are similar to expected results
Get scheduler timestamp since program start in milliseconds		Current timestamp in milliseconds is updated	**Pass** Actual results are similar to expected results
Data Processing Ring Buffer			
Construct ring buffer and initialise configuration parameters **Procedure Steps:** 1- **Input:** user-defined memory used by the ring buffer to store the data (static allocation) 2- **Input:** number of items to be saved 3- **Input:** item size 4- **Input:** pointer to the buffer that holds data 5- Initialise internal ring buffer with the configured parameters	1- Maximum number of items (**must be power of 2**) **8 / 16 / 32 / 64** 2- Size of structure that holds the sensor data 3- Allocated array buffer to store structures of data	1- Internal Ring buffer attributes are initialised with the defined parameters 2- Head and tail are initialised to the beginning of buffer	**Pass** Actual results are similar to expected results
Enqueue data into ring buffer **Procedure Steps:** 1- **Input:** pointer to the data to be pushed into the ring buffer 2- Validate the buffer is not full 3- Calculate offset inside internal buffer to determine start index for each item 4- Data pushed from the caller is copied to the current location inside internal buffer 5- Increment head of buffer 6- Data is then saved on SD card	Pushing sensor dataset structure into the buffer - **Sensor ID** - **Data Type** - **Data Size** - **Timestamp** - **Measurement Data** Tested with multiple sensors data	Data is enqueued successfully inside ring buffer	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Dequeue data from ring buffer **Procedure Steps:** 1- **Input:** pointer to the data to be stored and popped out from the ring buffer 2- Validate the buffer is not empty 3- Calculate offset inside internal buffer to determine start index for each item 4- Item is copied outside the ring buffer back to data storage 5- Tail of buffer is incremented	Popping sensor dataset structure out from the buffer - **Sensor ID** - **Data Type** - **Data Size** - **Timestamp** - **Measurement Data** Tested with multiple sensors data	Data is dequeued successfully from ring buffer	**Pass** Actual results are similar to expected results
Event-driven Asynchronous Finite-State Machine			
Initialise state machine **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active slave MCU object instance 2- **Input:** initial state for the machine 3- Set initial state for the machine	MCU startup as initial state to the machine	State machine is initialised successfully	**Pass** Actual results are similar to expected results
Dispatch event signals to state machine **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active object instance 2- **Input:** pointer to event signal to be dispatched in the machine 3- Call EXIT action of last state in the machine 4- Call ENTRY action of new state in the machine	Possible **Signals**: - **ENTRY** - **EXIT** - **ERROR STATE TIMEOUT** - **MCU STARTUP COMPLETE** - **ERROR SENSORS UNCHECKED** - **MCU RESET COMPLETE** - **SENSORS CHECKED MCU READY** - **MCU START MEASURING** - **MCU MEASURING DONE SLEEP** - **MCU SLEEPING DONE READY**	Event signals are dispatched successfully	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
Construct state machine and link with active object slave MCU **Procedure Steps:** 1- **Input:** pointer to state machine linked with active object 2- **Input:** pointer to slave MCU		The configured slave MCU becomes active object in the state machine with same address location	**Pass** Actual results are similar to expected results
Configure scheduled tasks for state machine **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active slave MCU object 2- Configure scheduled tasks: - **Input:** time offsets in milliseconds after which they are called - **Input:** periodic time in milliseconds to repeat cycle - **Input:** pointer to task parameters to be executed - Register callback function to be executed when task is ready		Tasks are successfully configured	**Pass** Actual results are similar to expected results
Create array of sensor structures based on availability on slave MCU and initialise related parameters and status flags **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active object 2- Check number of available sensors on slave MCU by transmitting request and retrieving response messages 3- Loop over available sensors on slave MCU and initialise their parameters and status flags 4- State machine checks return status	**Number of sensors on slave MCU is valid**	Available sensors on slave MCU are constructed and initialised successfully, return **SUCCESS**	**Pass** Actual results are similar to expected results
	Number of sensors on slave MCU is invalid	Sensors are neither constructed nor initialised, return **ERROR**	**Pass** Actual results are similar to expected results
State machine resets active object slave MCU **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active object 2- Master MCU resets slave MCU by transmitting request and retrieving response messages 3- State machine checks return status	**Slave MCU reset successfully**	Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Slave MCU did not reset	Return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
State machine puts active object slave MCU into ready mode **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- Master MCU puts slave MCU into ready mode by transmitting request and retrieving response messages 3- State machine checks return status	Slave MCU is ready	Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Slave MCU is not ready	Return **ERROR**	**Pass** Actual results are similar to expected results
State machine activates active object slave sensor **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput: selected sensor ID** **3-** Master MCU activates sensor on slave MCU by transmitting request and retrieving response messages 4- State machine checks return status	Sensor is activated	Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Sensor is not activated	Return **ERROR**	**Pass** Actual results are similar to expected results
State machine checks maximum measuring time for active object slave sensor **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput: selected sensor ID** **3-** Master MCU checks maximum measuring time for selected sensor on slave MCU by transmitting request and retrieving response messages 4- State machine checks return status	Maximum measuring time checked	Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Maximum measuring time is unchecked	Return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
State machine starts processing for selected sensor on slave MCU **Procedure Steps:** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput: selected sensor ID** 3- State machine checks if selected sensor is within range of available sensors 4- Master MCU triggers selected sensor on slave MCU by transmitting request and retrieving response messages 5- State machine checks triggering status 6- If triggering is successful, state machine creates some delay based on maximum measuring time for the sensor 7- After delay period is finished, state machine start checking measurement status and retrieving process	**Valid selected sensor IDs**	1- Trigger process is successful 2- Delay period related to the sensor maximum measuring time is similar and accurate Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Invalid selected sensor IDs	Return **ERROR**	**Pass** Actual results are similar to expected results
State machine checks measurement status and retrieves data content **Procedure Steps:** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput: selected sensor ID** 3- Master MCU checks if measurement is ready on slave MCU by transmitting request and retrieving response messages 4- If measurement is ready, Master MCU retrieves measurement data from slave MCU by transmitting request and retrieving response messages 5- Data content is then retrieved and enqueued into the defined ring buffer 6- If enqueuing is successful, measuring counter for the selected sensor is incremented 7- When the counter reaches its preconfigured maximum value, the selected sensor stops measuring and waits for other sensors to stop so slave MCU can go into sleep mode.	**Valid selected sensor IDs**	1- Measurement is ready to be retrieved 2- Retrieve process is successful 3- Data content is successfully enqueued into the ring buffer 4- Measuring counter for the selected sensor is incremented 5- Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Invalid selected sensor IDs	Return **ERROR**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
State machine puts slave MCU into sleep **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- Master MCU puts slave MCU into sleep mode by transmitting request and retrieving response messages 3- State machine checks return status	Slave MCU is in sleep mode	Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Slave MCU is not in sleep mode	Return **ERROR**	**Pass** Actual results are similar to expected results
Check available sensors state in periodic cycle **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- Check if number of selected sensors in configuration file on master MCU is **less than/ equals to/more than** the total sensors available on slave MCU 3- A check state counter is checked periodically and compared based on previous conditions to determine if all available sensors are done with measuring so slave MCU goes into sleep mode	Selected number of sensors in configuration file on Master MCU is less than or equals to total available sensors on slave MCU	1- Sensor selection process is completed successfully, 2- Slave MCU goes into sleep mode after preconfigured measuring cycles for each sensor is done 3- Return **SUCCESS**	**Pass** Actual results are similar to expected results
	Selected number of sensors in configuration file on Master MCU is more than total available sensors on slave MCU		**Pass** Actual results are similar to expected results
“Startup slave MCU object” state in the finite machine **<u>Procedure Steps:</u>** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput:** pointer to the event signal 3- Add scheduled tasks to callback dispatched events that drive state machine to next state	Task is added using the scheduler acting as timeout delay until slave MCU startup is completed and modules on master MCU are initialised successfully **(30 seconds)**	1- Slave MCU startup is completed 2- **“MCU STARTUP COMPLETED”**event is dispatched in the machine as callback function to the timeout delay 3- Return **TRANSITION** into next state **(check and create sensors)**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
“Check sensors info on slave MCU object” state in the finite machine **Procedure Steps:** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput:** pointer to the event signal 3- State machine checks if sensors are created 4- If creation is successful, state machine loops over available sensors on slave MCU, activates every sensor and gets its max measuring time 5- Add scheduled tasks to callback dispatched events that drive state machine to next state based on the check condition of sensors	Sensors are created and activated successfully	1- Sensors info on slave MCU is checked 2- “**SENSORS CHECKED MCU READY**” event is dispatched in the machine 3- Return **TRANSITION** into next state **(MCU Ready)**	**Pass** Actual results are similar to expected results
	Sensors are neither created nor activated	1- Sensors info is unchecked 2- “ERROR **SENSORS UNCHECKED**” event is dispatched in the machine 3- Return **TRANSITION** into next state **(Error)**	**Pass** Actual results are similar to expected results
	Unresponsive task (The program is stuck inside the state for a long time without response)	1- State timeout is completed 2- **“ERROR STATE TIMEOUT”** event is dispatched in the machine as callback function to the timeout delay 3- Return **TRANSITION** into next state **(Error)**	**Pass** Actual results are similar to expected results
“Error” state in the finite machine **Procedure Steps:** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput:** pointer to the event signal 3- State machine resets slave MCU	Slave MCU reset is completed successfully	1- **“MCU RESET COMPLETED”** event is dispatched in the machine 2- Return **TRANSITION** into next state **(MCU Startup)**	**Pass** Actual results are similar to expected results
“Slave MCU object ready” state in the finite machine **Procedure Steps:** 1- I**nput:** pointer to the state machine linked with the active object 2- I**nput:** pointer to the event signal 3- State machine puts slave MCU into ready mode 4- Add scheduled tasks to callback dispatched events that drive state machine to next state	Slave MCU is ready to start measuring process	1- Slave MCU is ready to measure 2- “**MCU START MEASURING**” event is dispatched in the machine 3- Return **TRANSITION** into next state **(MCU Measuring)**	**Pass** Actual results are similar to expected results
	Slave MCU is not ready to start measuring process **Unresponsive task**	1- State timeout is completed 2- **“ERROR STATE TIMEOUT”** event is dispatched in the machine as callback function to the timeout delay 3- Return **TRANSITION** into next state **(Error)**	**Pass** Actual results are similar to expected results

Test Case	Scenario	Expected Results	Status
“Slave MCU measuring” state in the finite machine **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active object 2- **Input:** pointer to the event signal 3- State machine checks selected sensors IDs from configuration file on master MCU 4- If selected sensors IDs are valid and within the availability of total sensors on slave MCU, scheduled task is added for every sensor to enable its measuring process in a periodic cycle until the task is removed and the measuring counter for every sensor is fulfilled with the preconfigured maximum value before MCU goes to sleep	1- Tested with different combinations of selected sensors number in the configuration file on master MCU and the total number of available sensors on slave MCU **2-** Tested with different combinations of sensor measuring periodic cycles in the configuration file on master MCU **Combinations Restriction up to 4 sensors**	1- Periodic measuring task is added in the scheduler for every sensor that fulfils the requirements criteria of selection and availability on slave MCU 2- “**MCU MEASURING DONE SLEEP**” event is dispatched in the machine after measuring counter for every sensor reaches the preconfigured maximum value 3- Return **TRANSITION** into next state **(MCU Sleep)**	**Pass** Actual results are similar to expected results **Comment:** Number of sensors can be adapted inside the program
“Slave MCU sleep” state in the finite machine **Procedure Steps:** 1- **Input:** pointer to the state machine linked with the active object 2- **Input:** pointer to the event signal 3- State machine puts slave MCU into sleep mode 4- Add scheduled tasks to callback dispatched events that drive state machine to next state	Slave MCU sleep is successful	1- Delay is added with the preconfigured sleep duration for slave MCU 2- “**MCU SLEEPING DONE READY**” event is dispatched in the machine after the delay 3- Return **TRANSITION** into next state **(MCU Measuring)**	**Pass** Actual results are similar to expected results
	Slave MCU sleep is unsuccessful **Unresponsive task**	1- State timeout is completed 2- **“ERROR STATE TIMEOUT”** event is dispatched in the machine as callback function to the timeout delay 3- Return **TRANSITION** into next state **(Error)**	**Pass** Actual results are similar to expected results **Comment:** Sleep duration delay must be less than state timeout
Master MCU saves sensor data on SD card **Procedure Steps:** 1- **Input:** pointer to size percentage of ring buffer before dequeuing data to SD card - **ONE PERCENT** - **TWENTY FIVE PERCENT** - **FIFTY PERCENT** - **SEVENTY FIVE PERCENT** 2- Open data file 3- Dequeue recorded sensors data to SD card based on the ring buffer size percentage specified 4- Close file	Tested with different size percentages of ring buffer	1- Sensor data is dequeued successfully based on the choice of size percentage 2- Sensor data is saved on SD card **successfully**	**Pass** Actual results are similar to expected results
	SD card directory full	**ERROR**, Sensor data is not saved on SD card	**Fail** **Comment:** Backup storage option to satellite network can be implemented to fix this issue

Test Case	Scenario	Expected Results	Status
Review and optimise on-board sensor modules on Master MCU **(GPS. IMU, Temperature Sensor)** **Procedure Steps:** 1- Review and optimise implemented drivers from existing project 2- Transform super loop blocking system into scheduling system which manages multiple sensors asynchronously 3- Transform static array allocation for data processing into first-in-first-out ring buffer which stores structures of sensors datasets	1- Tasks added to update measuring cycle for each sensor according to the requirements specification 2- Ring buffer is defined for processing of sensor datasets	1- Measurement readings for each sensor are periodically stored on SD card according to the preconfigured duration 2- Sensor data processed successfully through the ring buffer to the SD card	**Pass** Actual results are similar to expected results
Hardware wiring issue between master MCU and slave MCU in middle of the process	1- I2C wiring between master MCU and slave MCU is disconnected 2- Data stream transmission is checked 3- Wiring is reconnected	1- Program does not process sensor data when wiring is disconnected, return **ERROR** 2- Program proceeds the process once the wiring is connected again	**Pass** Actual results are similar to expected results **Comment:** MCU software/ hardware reset can be added to troubleshoot this issue
Power source disconnection issue in middle of the process	1- Power source is disconnected from master MCU 2- Data stream transmission is checked 3- Power source is reconnected	1- Program does not process sensor data when power source is off, return **ERROR** 2- Program does not proceed once the power is back on	**Pass** Actual results are similar to expected results **Comment:** MCU software/ hardware reset can be added to troubleshoot this issue
User extension interface: SD card reads customer configuration file on master MCU 1- Save customer requirements and parameters on SD card 2- SD card is inserted inside master MCU 3- Master MCU reads SD card and executes program according to customer configurations			**Comment:** Future works
Backup sensor data to Satellite network to be retrieved by the customer In case of SD card failures (e.g. **full directory**) or specific selected sensor data backup.			**Comment:** Future works

Test Case	Scenario	Expected Results	Status
Complete System Test 1- Scheduling system 2- Event-driven asynchronous state machine for active objects 3- Communication Handling between master MCU and slave MCU based on I2C protocol 4- Data processing from ring buffer to SD Card 5- Optimised on-board sensor modules (GPS - IMU - Temperature Sensor)	System is fully tested and sensors data readings are observed for several days with visualisation of data streaming on both master and slave MCUs	1- Scheduling system works **successfully** 2- Sensors data readings are processed **successfully** **Data Readings:** - Active Sensor ID - Measurement Type - Measurement Size - Measurement Timestamp - Measurement Data 3- Communication is handled **successfully** between master and slave MCUs 4- Asynchronous event-driven state machine works **successfully** along with the scheduler system and data processing methodology 5- On-board sensor modules work successfully with the scheduling system after modification	**Pass** Actual results are similar to expected results