Experiment MOTORE++
D2.1 New software and firmware

MOTORE++: A new Rehabilitation Robot for the upper limb: refinement and experimental trials

---

Version 1
Submission date: 14.10.2015

Date	Name	Changes/Comments
14.10.2015		

1 Publishable Summary

The robot's main goal is to help the recovery of upper limb functionality in patients with neurological diseases and to assess their performance. To accomplish this, a set of exercises has been developed that the patient performs with the robot while looking at a PC screen. The PC screen displays a game-like exercise and the patient interacts with the robot as if it was a haptic I/O peripheral. The software has a user interface which allows to select among several exercises, which target several aspects and several types of movements, but also improve the attractiveness of the system.

Two new games are being developed in addition to the ones that were already present in the prototype. One has been made to target an important aspect of the usage of the robot, and the other to improve the offer on cognitive functions of the patient.

2 Specifications and limitations

The robot is a planar device, bound to move on a 2D surface. Therefore all the proposed exercises have to be bi-dimensional games.

The robot works in three possible modalities:

- it can move on the plane following the will of the patient (admittance control)
- it can allow the patient to move along one direction while preventing perpendicular movements by exerting a repulsive force (constrained admittance control)
- it can move autonomously, thus pushing the patient towards a given point on the workspace (position control)

One of the problems that have been noticed in the first prototype was that some patients had problems with dosing the force on the handle, so that with these patients most of the exercises were too difficult to perform. Being unable to remain "neutral" on the handle (i.e. avoid exerting force) means that the patient is constantly asking the robot to move around. Previous experiments have shown that a patient with such a condition will usually exert force in one specific direction, so that he/she will force the robot to always go in a certain direction. So, eventually this kind of patient will force the robot in a corner and will be unable to play the games.

This has led to the idea that the patients could benefit from a training exercise specifically devised to train them in remaining neutral on the handle and in exerting force gently on the robot.

Obviously, these mechanisms will not work during a muscular spasm. In such a case, the robot will comply to the spasm (the wheels of the robot slip on the workspace, so the robot cannot exert more than 30-35 N)

3 The mole game

Drawing inspiration from an old arcade game, a new game has been devised to help patients improve their interaction with the robot.

The game depicted on the computer screen consists of a hand sitting in the middle of the screen and holding a hammer. The game background is a grass field. At random intervals, a pest (a critter) will appear on the screen and the patient will have to move the hand so that it hovers above the critter. When the hand hovers above the critter, the hammer strikes on the critter and the critter escapes. The player is awarded points for the successful action.

On the other side, if left alone, the pest will remain on screen for a while and will, eventually leave.

The main difference between the mechanics of this game and the previous ones is that for this game the robot is used in a partially new modality: the robot tries to stay in a certain position but allows the patient to move it slightly. The feeling that the patient perceives is similar to trying to pull/push an object tied to a spring. When pulling it, the object will move slightly from its neutral position, and it will return to the neutral position when the patient lets it go.

At the beginning of the exercise, the therapist will help the patient to reach a point on the workspace where the patient feels comfortable, and then the robot will lock in place, in a modality called “hold”, which is basically a position control mode.

At this point, the hand will appear on the screen and the patient will have to move the hand around only by exerting force on the robot handle.

If the patient will exert force towards the right, the hand will move right. The greater the force exerted, the bigger the distance that the hand will reach from the center of the game screen.

In this way, the patients will have the chance to train their interaction with the robot, in an environment (working conditions) that is more restrictive than the other games, and where they will not be able to send the robot in places on the workspace that could cause pain to the shoulder or muscles.

Figure 1: the initial game window

Figure 2: a hole appears on the ground, indicating that a critter is about to show up

Figure 3: a critter appears

As said earlier, in this game the cursor doesn't show the position of the robot on its workspace, as in the other games. Instead, the cursor starts in the middle of the screen and moves from there according to the force that the patient is exerting on the robot. Meanwhile, the robot will try to remain in the spot where the therapist has placed it.

Ideally, if the user removes the arm from the robot, the cursor returns to the center of the screen, exactly as it does if the patient is capable of controlling the force exerted on the robot and decides to apply no force.

Moreover, the stronger the force that the user exerts, the more the cursor will move from the center of the screen.

So, when the critter appears, the user has to exert force on the robot in the correct direction. Even the amount of force exerted on the robot has to match a certain amount. If too much force is exerted, the cursor will go past the critter, towards the screen border.

If the patient waits too much time or is unable to reach the critter with the cursor, the critter will disappear and no points will be rewarded to the patient.

Figure 4: the critter disappears after some time

In this game the robot will not intervene to help the patient, if he/she is unable to accomplish the task. This is another important change, compared to the older games. But unfortunately, the task is such that the robot couldn't possibly intervene. There is no movement involved in the exercise, since the exercise is a task of 'balancing' and being able to dose the force.

If the patient, is able to reach the critter with the cursor, an animation will play and the hand will strike a hammer blow on the pest, as shown in the figure below.

Figure 5: the patient hits the critter

Figure 6: the critter was hit and disappears

The critter that has been hit will disappear just as the critter that hasn't been hit, but no points will be awarded to the patient.

As usual, the exercise can be interrupted by pressing the ESC key.

As described earlier, in this exercise the robot behaves like an object tied to a spring, so the dynamics parameters that regulate the robot behavior will still hold.

Therefore, this exercise allows the following parameters to be set (before starting):

- Length of the exercise (in minutes)
- Correction force (spring stiffness)
- Resistance (i.e. object weight)
- Viscosity

4 The dish washing game

Another aspect that the consortium has been working on is the adaptation of the robot (and its software) to be able to train cognitive disorders in addition to motor impairments.

For this reason, an exercise simulating an activity of daily life has been added, and this exercise has several aspects incorporated to train cognitive impairments.

The exercise appears on screen as a vertical wall with several objects placed in key positions. Compared to the other exercises, where the computer screen (which is vertical) displays a horizontal surface (a table or a road), in this exercise the computer screen displays a vertical surface (a wall). This facilitates the patients, as they are required to overcome (mentally) only one coordinate transformation (the robot moves on a horizontal surface) rather than two. In the other exercises, the robot still moves on a horizontal surface, but the exercise depicts a horizontal surface on a vertical screen.

Figure 7: this figure shows the workspace on screen. There is one dirty dish to clean, there is a sponge in the upper right corner and a cloth in the upper left corner. A yellow circle on the dirty dish indicates the patient that he/she is expected to drive the robot on the dirty dish.

Another important aspect of this exercise is the fact that it consists of two kinds of movements: reaching movements and functional movements. In reaching movements the patient is required to reach specific points on the workspace, in order to grab or release the game objects. In functional movements, the patient has to interact with objects and perform meaningful actions (clean a dish with round movements or open/close the tap with an arc movement). If needed, the game will give hints to the patient showing what is expected from him/her. These hints will disappear with time, when the patient learns the sequence of movements.

This is very useful when treating patients with motor apraxia, i.e. patients that don't have difficulties in performing movements, but have difficulties in planning a sequence of actions needed to achieve a goal.

The exercise size can be adjusted to the capabilities of the patient. Indeed, if the patient has to train the right arm, all the exercise objects will move slightly towards the right (and vice-versa if the patient has to train the left hand), and the therapist can choose how big must the movements that the patient has to perform.

Figure 8: it can be seen that the cursor (the hand avatar) is almost in the corner of the screen, but the robot's position is much closer to the patient. In this exercise the workspace on the screen is not an accurate representation of the table but instead there is a zoom factor which magnifies the patient movements. (The robot figure is in these picture only for information purposes. When doing the exercise, the patient doesn't see the picture of the robot)

The sequence of actions needed to wash a dish is obviously oversimplified compared to real life. The patient will have to:

- pick up a dish from a pile of dirty dishes
- place the dish in the sink
- pick up the sponge
- clean the dish with round movements of the sponge over the dish
- drop the sponge in its place
- open the tap to let the water rinse the plate
- close the tap
- pick up the drying cloth
- dry the dish with round movements
- drop the drying cloth
- pick up the clean dish
- place the dish on the pile of clean dishes

during these movements, the robot uses all three of its working modalities. When the patient has to pick up or drop an object, the patient is free to roam around and the

robot is in admittance control mode. The patient will not feel the full weight of the robot, but only the amount of weight that the therapist has decided to use for the exercise. During the round movements and the open/close tap movements the robot will be in constrained admittance mode. The patient will feel very little resistance in following a round trajectory over the dish, but will feel a repulsive force if he/she will try to get away from the dish. In both of these two types of movements, if the user is unable to finish the required action within a time limit, the robot will go in the so called auto-mode and will help the patient.

Figure 9: in this figure, the patient has to make circles around the dish in order to clean it.

This exercise allows the following parameters to be set (before starting):

- Number of dishes to clean
- Correction force (spring stiffness)
- Resistance (i.e. object weight)
- Viscosity
- Speed at which the robot will move in case it will have to help the patient

A control scheme has been developed with the following structure.

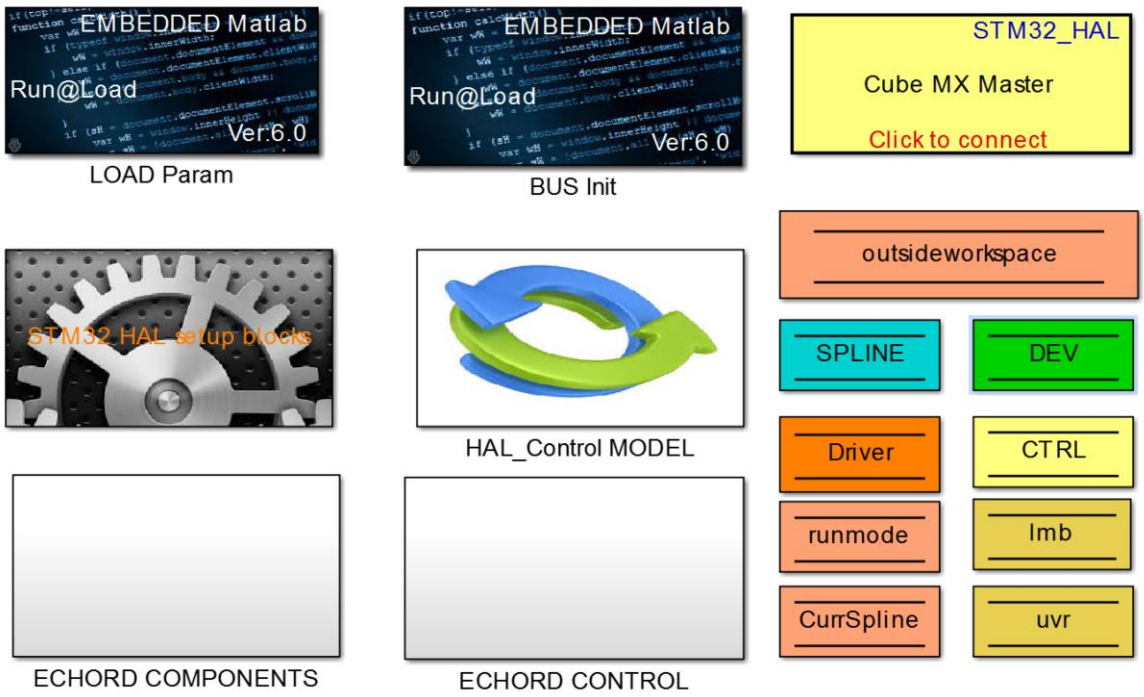

FIGURE 10: Top Level control structure

The control scheme exposes several new features with respect the pre-existing software. The huge amount of datastore memory employed for IO control have been completely replaced with structured data store that makes use of Simulink busses. This option allows not only having lesser datastore around the schematic but also allows Simulink to reduce the shared variable from 110+ to only four major variables. The benefit is that shared information is now stored into structures with homogeneous meanings (Drivers, CTRL, DEV and SPLINE). An overview of the structures is documented in the following three:

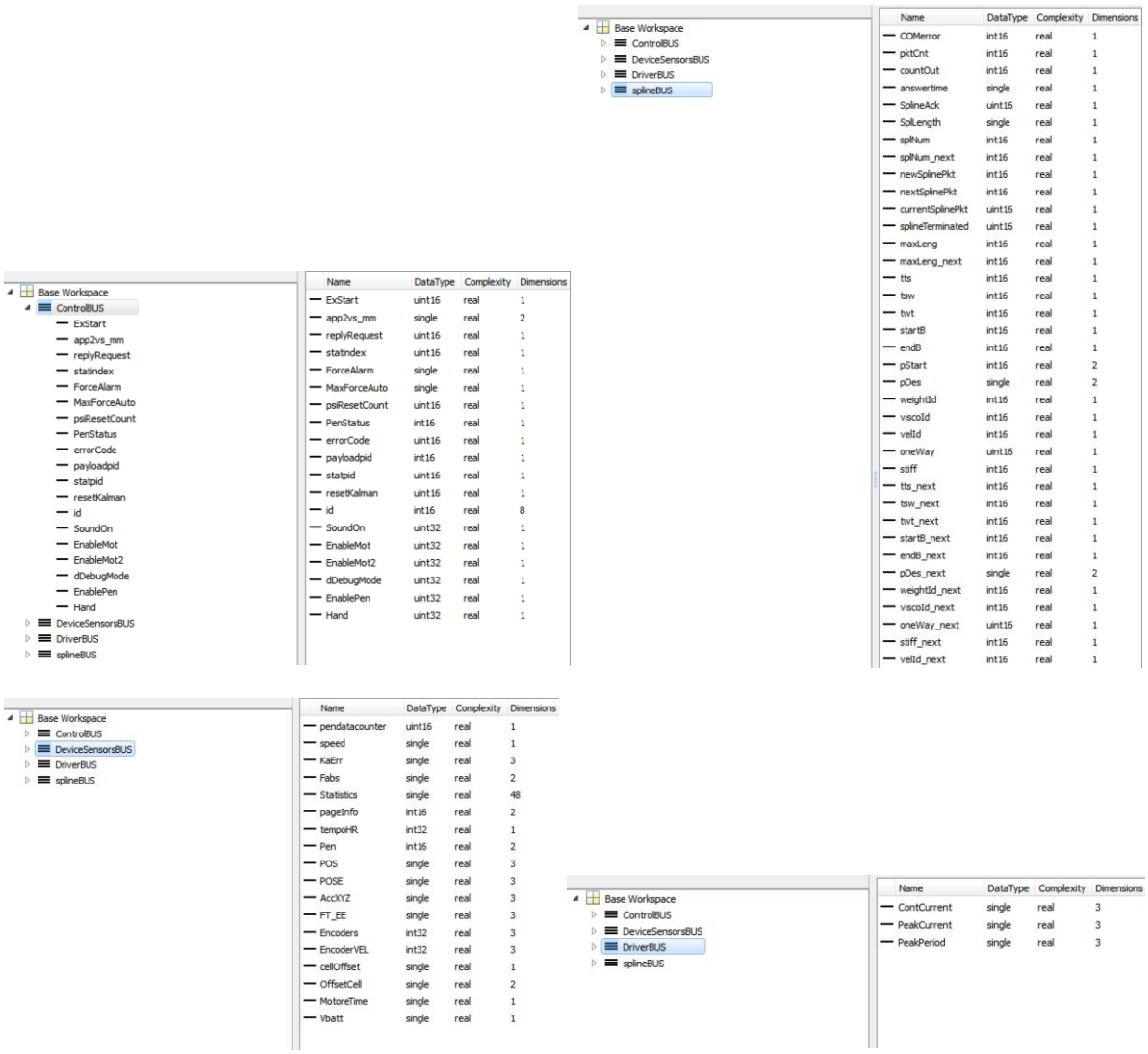

FIGURE 11: Busses architecture

Using structures we had the opportunity to name and document each Simulink entity thus minimizing program and debugging errors for most of the similar name signals. Within the scheme, new special interface has designed to handle the whole building process, from STMCUBEMX design to the target deploy.

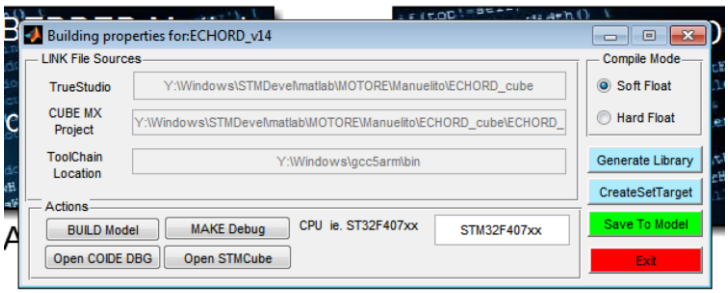

FIGURE 12: Integration Panel (Version 3)

Within the controller six special blocks have been designed to manage the ECHORD/MOTORE++ specific hardware:

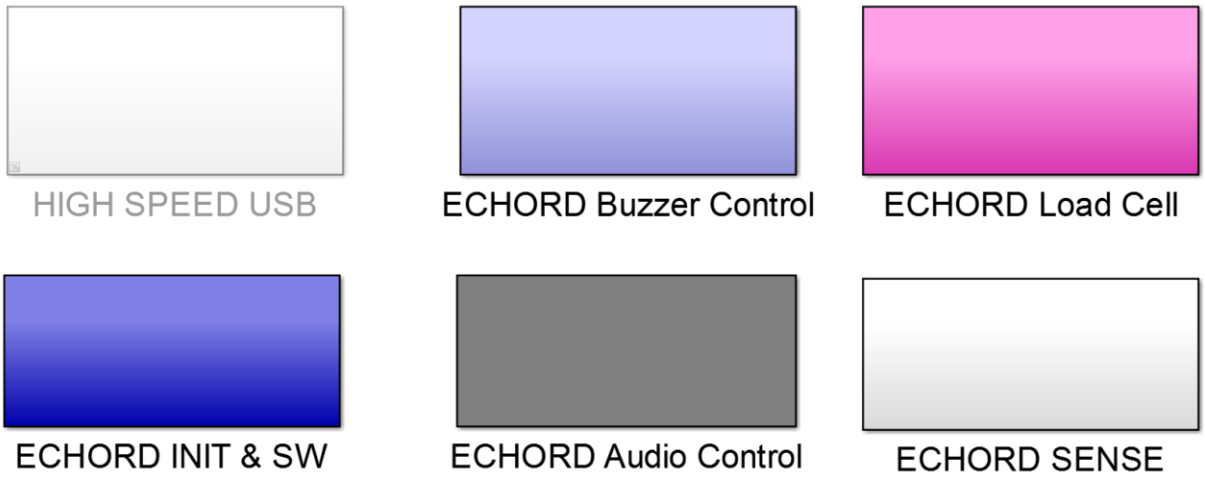

FIGURE 13: Subsystem of MOTORE++ specific drivers (ad hoc development)

Here it follows a detail on how the anoto optical sensor has been integrated in the new structure:

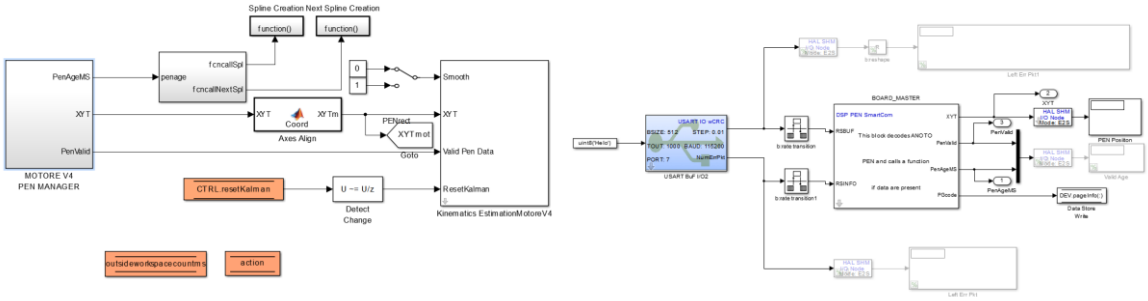

FIGURE 14: the new optical sensor driver design

Below this level of detail, specifically developed MEX function and Matlab-TLC blocks provide function inlining that support handling the host communication and the hardware management with the STM specific hardware. The design and the development of these blocks have been tested to properly work with the available board based on STM32F407.

Each block manages part of the IO in a way that is fully compliant with the STM_HAL library and such that all levels of code optimization can be implemented on the device.

In particular this allow a smart DMA management which delegates to basic HAL libraries and DMA architecture the burden to update memory coherently with data IO from/to any peripheral.

Some of the blocks inherit precedent result from research projects. For instance the Serial communication block does allow bidirectional data handling with any hosting PC and the same block can both be used as a data generator for the embedded target and a data consume from an hosting PC. This approach eases the debugging in the sense that the user can directly play a simulation to manage IO of data with a target application.

At the top level of interaction Simulink and STMCube integration is managed in a basic device control block.

The base device control is instead based on the following sub-diagram:

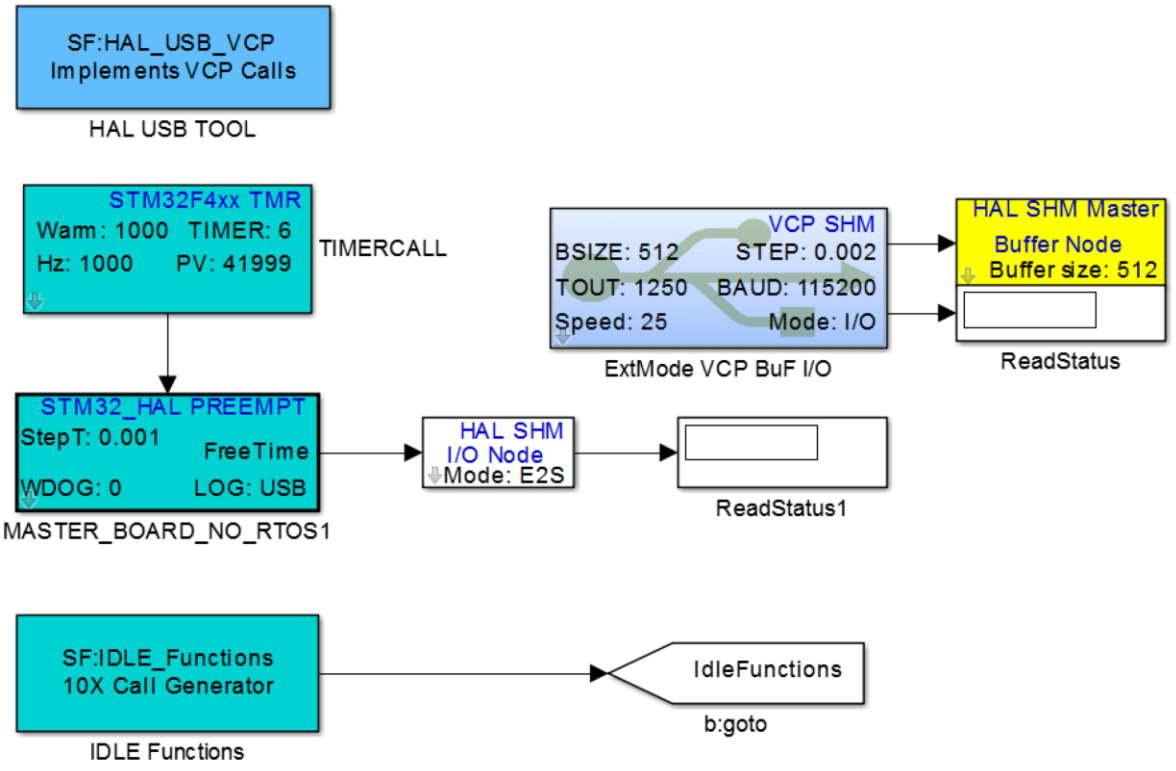

FIGURE 15: Scheduler subsystem

Considering the above picture, three main components can be listed:

(1) The cyan components deal with the process scheduling. One device timer can be programmed to generate regular interrupts which consequently trigger the simulation of the overall schematic as an interrupt driven procedure call. Even being prioritized in the ARM architecture, each interrupt is non-pre-emptable. Hence we designed a specific stack manipulation scheduler (names HAL

PREEMPT) that allows pre-emptable interrupts. This is particularly relevant when user has the need to run multithreaded task at different sampling times.

In addition during the idle time, the main loop has been modified in order to execute sequentially up to 10 function call procedures.

(2) The blue components handle the onboard USB device existing on the micro-controller. Using this device it is possible to print debug information, or as in most cases to executes diagnostic with processor in the loop.
(3) The diagnostic with the processor in the loop is handled by the yellow master block, which is one of several blocks connected thorough the schematic. The SE block collect in a shared memory all data which are required to exchange with an external Simulink schematic. Here the user has the following options: when the scheme is built these block are inserted in the target platform, alternatively, if the blocks are played, the USB interface queries the remote target to know the value stored in the block and exhibit them as an output of the simulation. In such a way, playing a schematic has the effect of inspecting values running in the remote target.

6 Deviations & Reasons

The release of the new software and firmware has been delayed due the late release of the prototype as mentioned in D2.2. However much of the related work has been anticipated using the existing electronics. Overall activities for firmware and software development are on-time.

The definition and selection of the wearable sensors (inertial sensors, EMG device, Heart rate sensor, Galvanic Skin response sensor) will be ended by month 12 as described in the task description (Task 2)