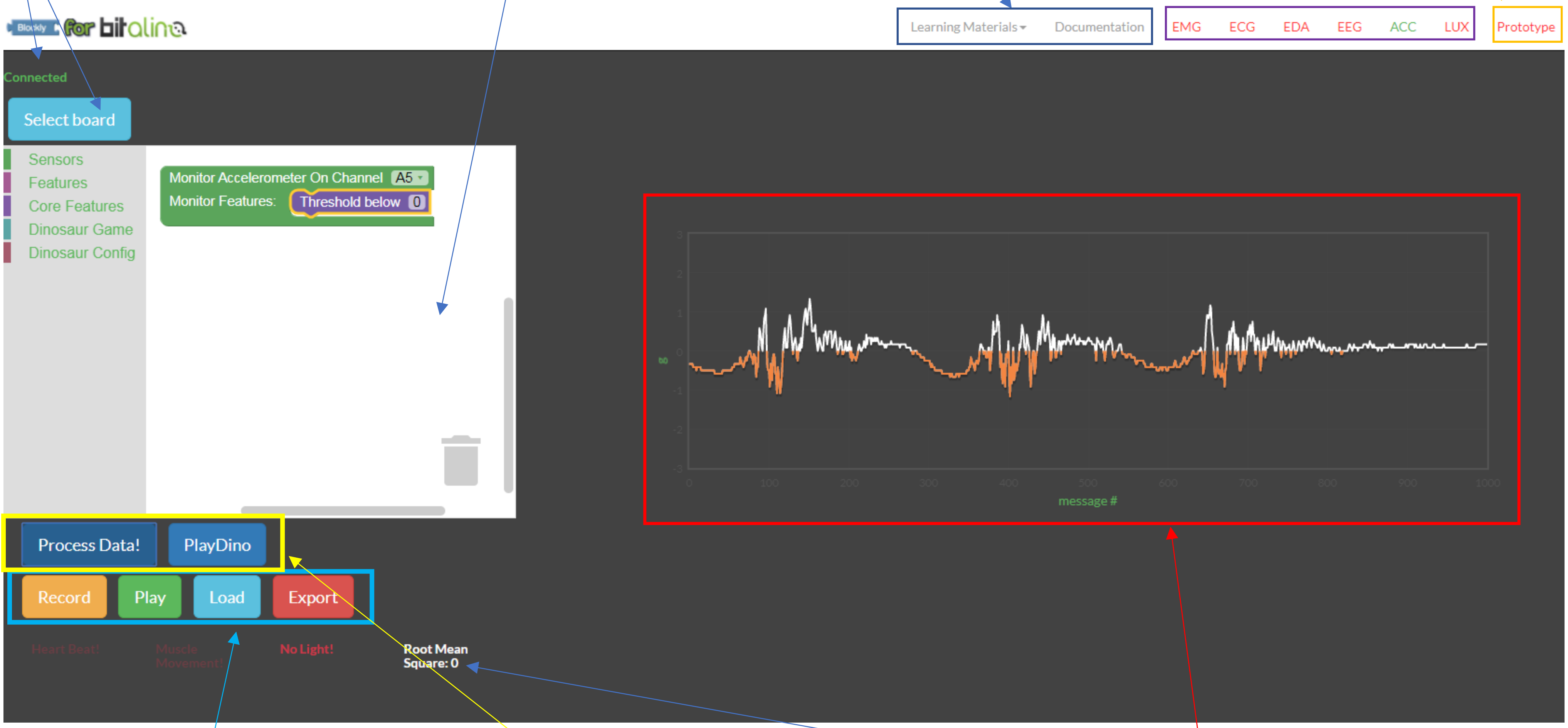
Determines which board the user has, and whether the ServerBIT program is connected or not **(A).**

Block workspace, enabling users to interact with the “drag and drop” blocks: the effects of which will be displayed in the game space (everything remains same from original design) **(E)**

Links to documentation and learning resources

Text shows current signals being monitored **(B)**

Shows whether a third-party sensor is being monitored **(B)**



Root mean square value is displayed here

Buttons for the user to interact with (details below) **(D)**

These should be clicked to visualise the data and/or play the game – if ‘PlayDino’ is clicked, the game space becomes available to see and interact with **(D)**

Sensor data is visualised in this graph. Game space is situated below the buttons **(C)**

Main Interface: Additional notes

The screen above represents the main interface for 'Blockly for BITalino' as it is now. It uses the same basic structural principles as the existing Blockly/Micro:bit interface.

(A) At the top left, the interface determines whether the "ServerBIT" python program has been started (and therefore connected) or not. If connected, it shows 'connected' in green, as seen above, but if not, 'Not connected' is shown in red instead. The interface requires a connection to serverBIT and the BITalino to be switched on. Details on selecting which board users have is contained in a separate interface below (A.1).

(B) Users can see text displaying which biosignals are currently being monitored. If the acronym is in green, it is being monitored and if the acronym is red, then it is not currently being monitored. Third party sensors are also supported. In the screenshot above, the accelerometer is being monitored.

(C) Underneath the acronyms we can then see the graph that is visualising the biosignal data. This will update in real time according to the user actions and record the data simultaneously.

Also simultaneously below the graph is the game space: this is where the respective games will be running (the 'PlayDino' button on the UI will have to be clicked to allow this to be visible, see C.1)

(D) There are some buttons the user will need to interact with. The most important of these is the 'Process Data button. Once the user has finished programming with the blocks (see below), they will then need to click this button to visualise the game and graph.

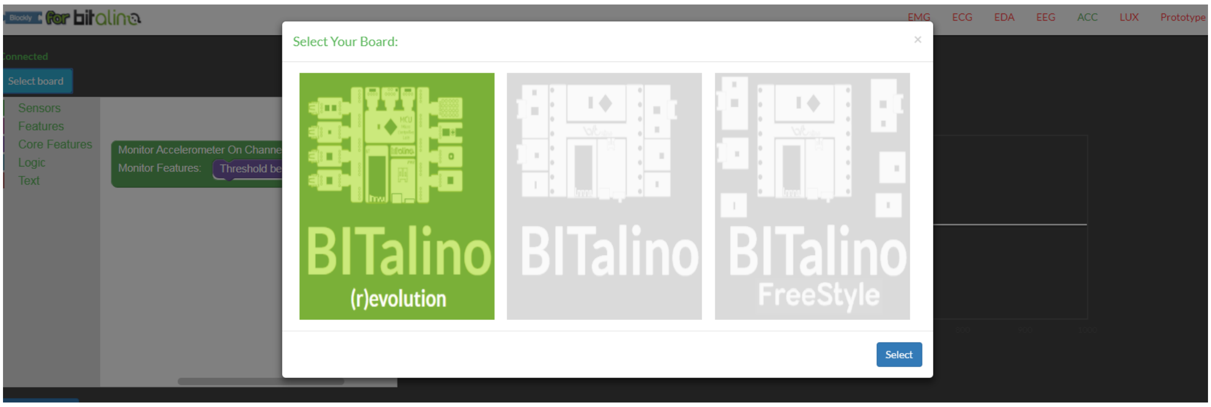
There are also options to load data from a previous session into the current workspace, and to export the data from the graph for later analysis outside of the application (Notepad etc.). The 'record' button records the data and 'Play' visualises it.

(E) The final part of the interface (on the left, under 'connected') is perhaps the most important part. This is where the users interact with the Blockly interface component, to drag a block into the white "workspace" (the box where you can put the blocks) where their functionality is executed.

Also shown is the bin, where unwanted blocks can be dragged and deleted. The different versions of the BITalino board have also been accommodated (see E.1)

Extra information:

(A.1)

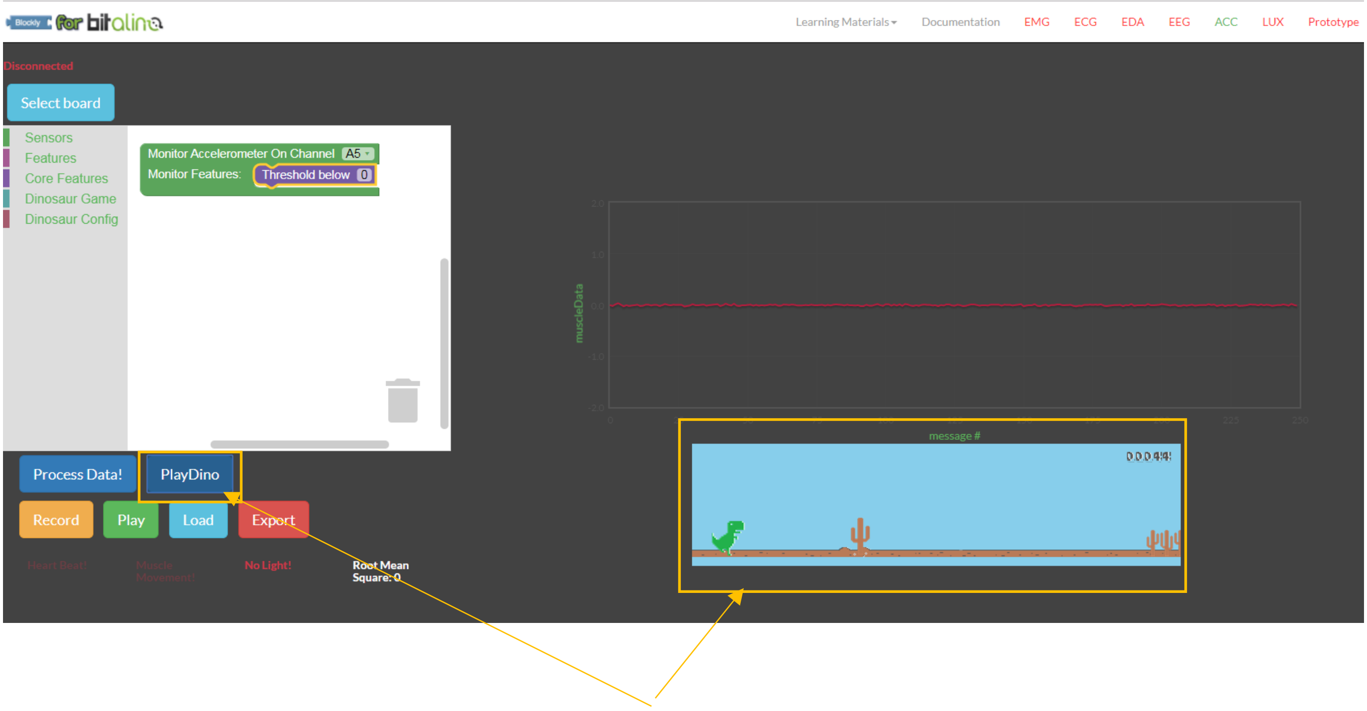


Users of the interface can select the board that they are using, via the ‘select board’ button. This brings up an alert as shown here.

When the image of the board is selected, only the sensors and channels used in that configuration are available for use (e.g. if EMG is not supported with an older version, that block will not appear in the left hand pane, if the older board is selected).

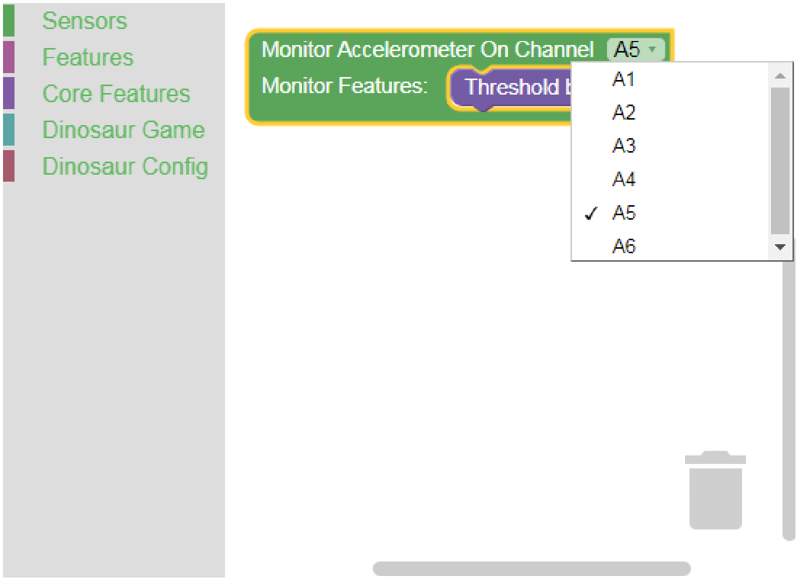
If freestyle is chosen, all blocks and configurations will be open for use as described in the main interface notes.

(C.1)



This illustrates the dinosaur gamespace being visible, once the PlayDino button has been clicked.

(E.1)



Different versions of BITalino have different versions, particularly with the channels used to monitor a certain sensor (e.g. EMG). On one device it could be set to one channel, but on a newer version, this could be monitoring an entirely different sensor.

For this reason there is a drop down box on each of the blocks which allows the user to configure the block (and associated channel) themselves for the freestyle option. It would be assumed that users were aware of the configuration of their hardware device beforehand, so would know which channel to choose. This feature would only be displayed for the freestyle board, as the other boards will have default channel values.