Manual Brain Cube

Simulink Support Package installation:

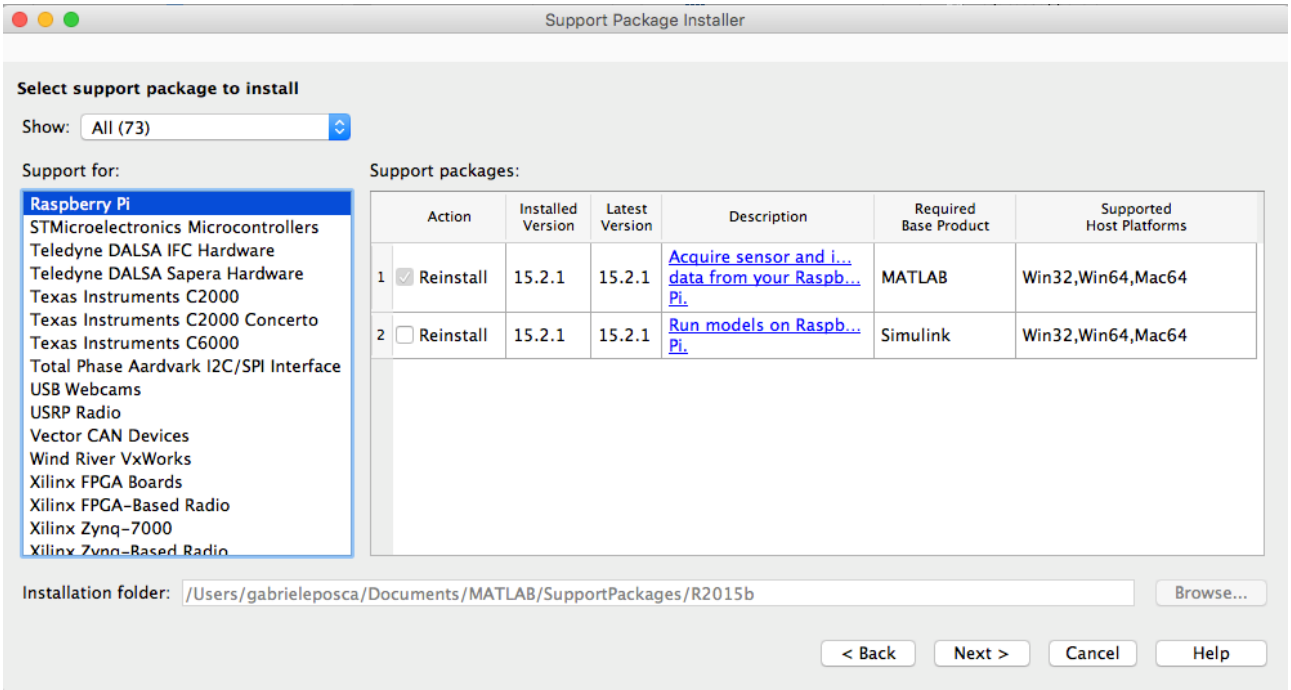
First of all you must install the Simulink Support Package for Raspberry Pi

Simulink® Support Package for Raspberry Pi™ lets you develop algorithms that run standalone on your Raspberry Pi. The support package extends Simulink with blocks to drive Raspberry Pi digital I/O and read and write data from them. After creating your Simulink model, you can simulate it and download the completed algorithm for standalone execution on the device. One particularly useful (and unique) capability offered by Simulink is the ability to tune parameters live from your Simulink model while the algorithm runs on the hardware.

- Run the following instruction in the Matlab Command Windows: ***supportPackageInstaller*** and select "Install From the Internet".

- Alternately you can download the Simulink Support Package from here.
  And execute the installer.

- Select Raspberry Pi in the section "Support for".
- Check all of the two support packages available (for Matlab and Simulink).
- After the installation on your PC of the Support Package, a wizard will start for the installation of the Raspbian Wheezy operating system, on a microSD connected to the PC.



Note: Review the following table to ensure you have the required Matlab and Simulink release.

Wi-Fi USB Dongle:

The Raspberry Pi 2 board does not have a wireless network chip integrated; it is therefore necessary to insert a Wi-Fi dongle for wireless connection.

Our choice: EDIMAX WIFI N150 it is directly compatible with Raspberry and does not require installation of additional drivers.

To configure the Wi-Fi connection you need:

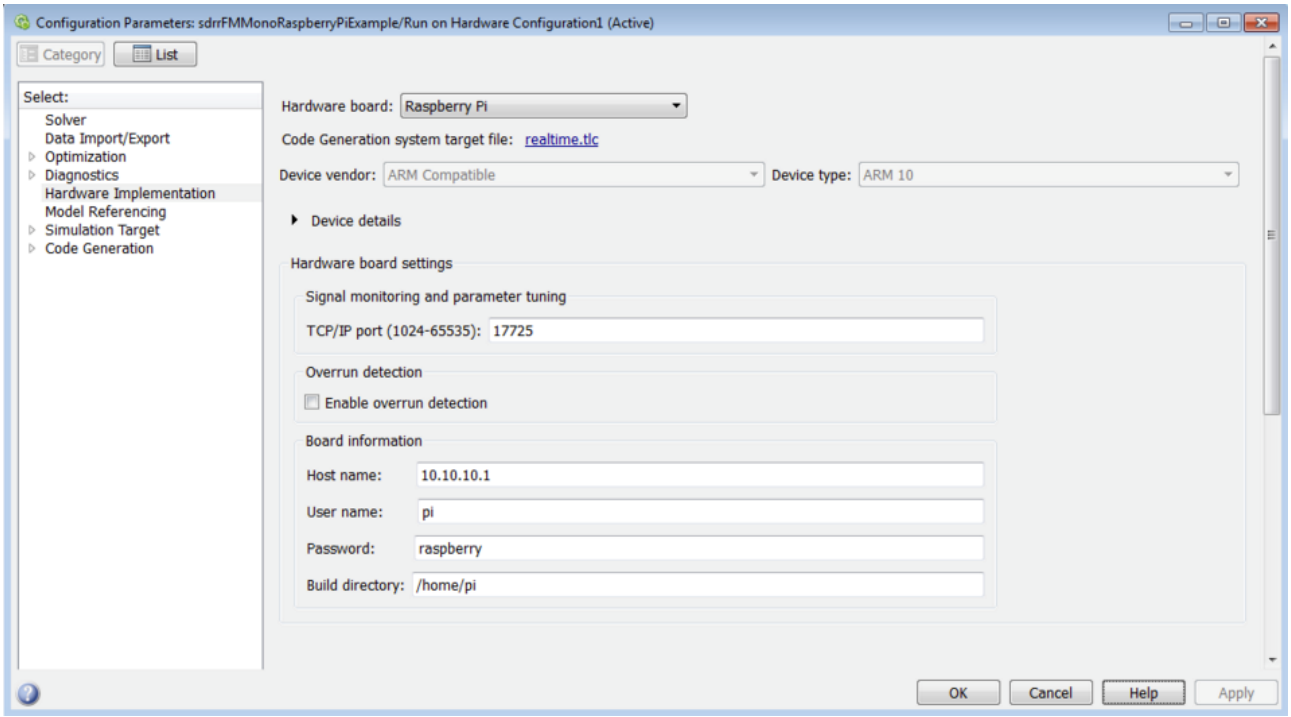
- Enter the Raspbian desktop:
  - Using a HDMI monitor and a USB keyboard
  - Or using any Remote Desktop application (in this case the board must be connected via Ethernet).
- Click Preference menu -> WifiConfiguration.

Run in external mode & deploy:

One of the most important aspects that the Simulink Support Package for Raspberry Pi Hardware provides is the possibility to perform the Simulink schemes directly on the target hardware.

Before running the simulation it is necessary that the Raspberry is plugged in and connected to the same local network as the PC:

1. In the Simulink toolbar, set "Simulation Stop Time" to "inf ".
2. Select "Tools" > "Run on Target Hardware". Then select "Prepare to Run" or "Options":



3. Select panel "Hardware Implementation"
4. Set as "Hardware board" the "Raspberry Pi".
5. In the "Signal monitoring and parameter tuning" leave the TCP/IP port 17725 by default.
6. In the Section "Board information" you need to set the IP of the Raspberry ,the username

and password for access (by default are user = raspberry and password = pi) and the build directory (default is / home / pi)

7. Finally in the Simulink toolbar, set the Simulation mode in External.

Click on the play button to start the simulation in **External mode.** The resulting executable runs in operating system kernel mode on the raspberry and exchanges parameter data with Simulink via a shared memory interface. In this way you can tune parameters and observe application output.

Alternatively in the Simulink toolbar, click **deploy to Hardware.** The model is now running on the external hardware as standalone application.

qbMove set-up:

- Connect the qbMove (or qbMove robotic system) to the raspberry pi trough an USB cable.

It's important to remember that before working on the cube brain with Simulink you need to check and set the path of the USB where is connected the qbMove:
To do this you must install ***qbmoveadmin***. This operation should be performed only once, after the start of Raspberry.
The ***qbmoveadmin*** is a program that provides command line tools for the qbMoves.

The following are the ***qbmoveadmin*** installation steps on Raspberry:

1. Access to Raspberry
2. Download the ***qbAPI*** library and ***qbmoveadmin*** from the qbrobotics repository.

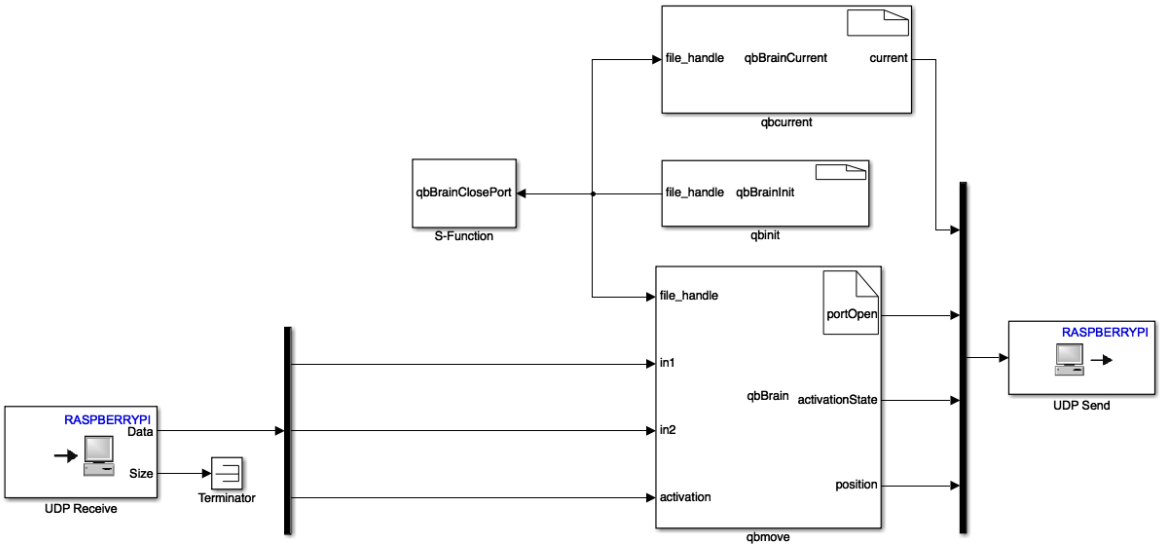
3. Extract the folders, remove "-master" suffix from their name and then place in a folder called qb.
4. Using the command line, enter in qb/qbAPI/src and run the command make.
5. Enter in qb/qbmoveadmin/src and run the command make.
6. Enter in qbmoveadmin/bin unix and start ./qbmove (administration interface cubes)
7. Using the command -t you set the port number on which you have to communicate
8. Using the command -p receive information about whatever it is connected to the port
9. Using the command: /qbmove [cube id] -a [1 / 0] you enable/disable the motors.
10. Now you can use simulink diagrams to control the qbMoves.

For a list of all the commands see the readme file.

Example / Template diagrams:

The following diagrams, are made to control a single qbMove but, properly adapted, are universally valid for the majority of robotic applications realized with qbMoves.
The example is composed by two different simulink diagrams: ***brainqb.slx*** e ***qbBrainPC*.slx**

The main diagram is ***brainqb.slx*** and run directly on the Raspberry pi in **External mode** or in **deploy**:



The first thing to configure are the blocks that handle data transmission via UDP protocol:

- UDP Send: Send UDP packets to another host on an Internet network. You need to set the "Remote IP address" parameter with the receiving host's IP address; also you have to set the "IP Remote port" parameter with the port number used by the host receiver.

- UDP Receive: Receives UDP packets from another UDP host on an Internet network. The sending host must send UDP packets on "IP Local port" specified in the block mask. You need to match the parameter "Data type" with the type of incoming data; also you have to match the "Date size parameter (elements)" with the number of items of input data. For example, if the type of the input data is uint8 data and dimensions are [3x1], set the parameter "Data type" in uint8 and set the parameter "Data size (elements)" 3.

- Remember to configure the size of the mux and demux blocks connected to them.

The second thing to configure are the in / out signals and parameters of each s-function (except qbBrainClosePort that does not need any settings):

qbBrainInit
Description: This s-function open the RS485 port, by calling the appropriate C API command: *openRS485*; the s-function send, as an output signal, an integer value called "file_handle" characterizing the structure "comm_settings".

  - Parameters:
    - path_comm: string indicating the path of the USB connected to the cubes
    - baud_rate: the USB port transmission speed

- Signals input: nothing
- Signals output:
  - File_handle: integer number characterizing the structure comm_settings, essential for the use of qbAPI

Qbmove

Description: This s-function read the input reference signals and send the position commands to the motors. It also print the port status, motor activation status and qbMoves actual positions.

- Parameters:
  - Qbot_id: vector of the cubes ID's.
  - Comminucation_direction: shows the type of communication:
    - 1=RX
    - 2=TX
    - 3=BOTH
    - 4=NONE
  - input type: shows the type of command to send qbmoves:
    - 1=Prime Movers Position
    - 2=Equilibrium Position and Stifness Preset
    - 3=Equilibrium Position and Stifness Preset Percentage
  - unity: shows the reference measuring unit:
    - 1=Degrees
    - 2=Radiants
    - 3=TICK
- Signals input:
  - File_handle: integer value from the s-function block qbBrainInit
  - in1: input for the first motor of the qbmoves
  - in2: input for the second motor of the qbmoves
  - activation: activation state of the qbMove
- Signals output:
  - portOpen: port state:
    - 0= port close
    - 1= port open
    - -1= failure to open port
  - activationState: activation motor state:
    - 0= disabled motors
    - 3= enable motors
    - -1=error activation

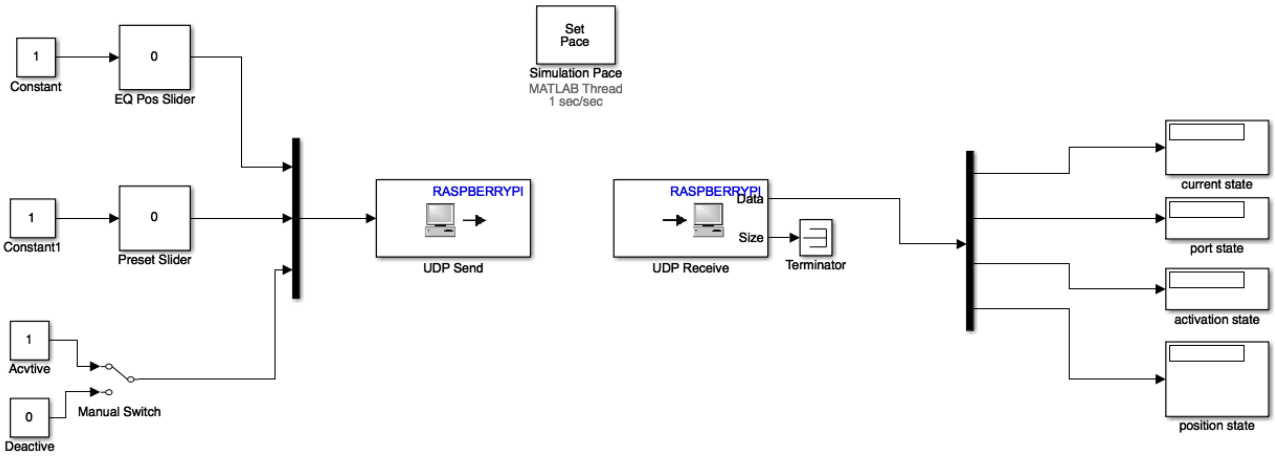
qbCurrent

Description: This s-function read qbmove absorbed current.

- Parameters:
  - Qbot_id: vector of the cubes ID's.
  - enable: enabling the reading of the current
- Signals input:
  - file handle: integer number characterizing the structure comm_settings, essential for the use of qbAPI
- Signals output:
  - current: read currents

The ***qbBrainPC*.slx** diagram run on the computer in normal mode.

It transmits the raspberry references position, stiffness and the activation command for each qbMoves; it also receives from Raspberry the port state, activation state, actual position of the motors and absorbed currents.



Remember to set the IP addresses and the data size of the two UDP blocks and the mux / demux connected to them.