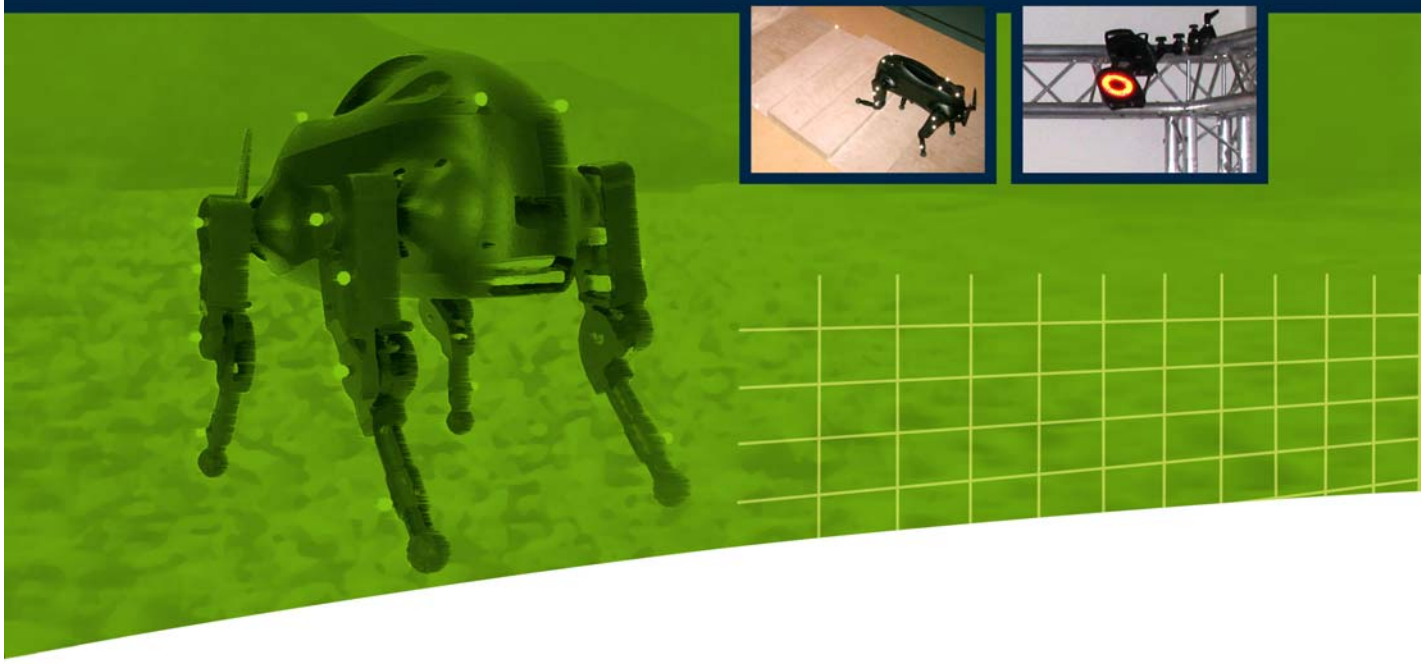
LittleDog™ Robot 1.0



User Guide

Boston Dynamics

LittleDog Robot 1.0 User Guide

Part No. LittleDog Robot 1.0.2 User Guide

Boston Dynamics
515 Massachusetts Avenue
Cambridge, MA 02139
Telephone: 617-868-5600
Fax: 617-868-5907
www.BostonDynamics.com

LittleDog Software License Agreement

This is a legally binding agreement between you ("LICENSEE") and Boston Dynamics, with its principal place of business at 515 Massachusetts Avenue, Cambridge, MA ("Boston Dynamics"). By breaking the seal on the package containing the LittleDog software, and/or by installing and/or using the software, regardless of its form or the method of the recording (the "SOFTWARE"), LICENSEE is agreeing to be bound by the terms and conditions of this agreement, including the software license and disclaimer of software warranty below. Please read this document carefully before opening the package and/or using the SOFTWARE. If LICENSEE does not agree with the terms and conditions of this agreement, LICENSEE should promptly return the unopened package and the SOFTWARE to the place where it was obtained, and LICENSEE will be given a full refund of any license fee that LICENSEE paid.

1. Grant of License: Subject to the terms and conditions of this Agreement, Boston Dynamics hereby grants to LICENSEE a non-transferable and non-exclusive right to use and execute the SOFTWARE on a single computer system at one time. LICENSEE agrees that it will not reverse engineer, decompile or disassemble any portion of the SOFTWARE, nor extract data of any kind from the SOFTWARE. If LICENSEE disposes of any media or apparatus containing SOFTWARE, LICENSEE will ensure that it has completely erased or otherwise destroyed any SOFTWARE contained on such media or stored in such apparatus. LICENSEE may not distribute, lease, transfer, export, loan or otherwise convey the SOFTWARE or any portion thereof to anyone.

2. Copying Restrictions: In order to effect LICENSEE's license rights hereunder, it may install the SOFTWARE by copying it onto the hard disk drive or into the CPU memory of a computer system for use thereon, and may make full or partial copies of the SOFTWARE, but only as necessary for backup or archival purposes. LICENSEE agrees that (i) LICENSEE's use and possession of such copies shall be solely under the terms and conditions of this Agreement, and (ii) LICENSEE shall place the same proprietary and copyright notices and legends on all such copies as included by Boston Dynamics on the media containing the authorized copy of the SOFTWARE originally provided by Boston Dynamics.

3. Ownership: LICENSEE agrees and acknowledges that Boston Dynamics transfers no ownership interest in the SOFTWARE, in the intellectual property in SOFTWARE, nor in any SOFTWARE copy to LICENSEE under this Agreement or otherwise, and that Boston Dynamics and its licensors reserve all rights not expressly granted hereunder. After LICENSEE pays any applicable license fees, LICENSEE will own the media on which the SOFTWARE was originally provided hereunder and on which LICENSEE subsequently copies the SOFTWARE, but Boston Dynamics and its licensors shall retain ownership of all SOFTWARE and copies of the SOFTWARE or portions thereof embodied in or on any media.

4. Transfer Restrictions: LICENSEE may not sublicense, transfer, or assign this Agreement or any rights or obligations under this Agreement, in whole or in part.

5. Export Restrictions: LICENSEE agrees not to export or re-export, directly or indirectly, from the United States through any country or to any country of ultimate destination defined by the United States Government or any agency thereof as an "Excluded Territory". LICENSEE will not export, directly or indirectly, the Licensed Programs or Documentation to any country from which the United States Government or any agency thereof requires an export license or other governmental approval at the time of export without first obtaining such license or approval. The U.S. Government's list of "Excluded Territories" is subject to change without notice and is therefore not included herein.

6. Confidentiality: By accepting this license, you acknowledge that the SOFTWARE and accompanying materials, and any proprietary information contained in the media, are proprietary in nature and contain valuable confidential information developed or acquired at great expense. You agree not to disclose to others or utilize such trade secrets or proprietary information except as provide herein.

7. Enforcement of Terms: If LICENSEE fails to fulfill any of its obligations under this Agreement, Boston Dynamics and/or its licensors may pursue all available legal remedies to enforce this Agreement, and Boston Dynamics may, at any time after LICENSEE's default of this Agreement, terminate this Agreement and all licenses and rights granted under this Agreement. LICENSEE agrees that Boston Dynamics's licensors referenced in the SOFTWARE are third-party beneficiaries of this Agreement, and may enforce this Agreement as it relates to their intellectual property. LICENSEE further agrees that, if Boston Dynamics terminates this Agreement for default, LICENSEE will, within ten (10) days after any such termination, deliver to Boston Dynamics or render unusable all SOFTWARE originally provided hereunder and any copies thereof embodied in any medium.

8. Governing Law: This Agreement shall be governed by and interpreted in accordance with the laws of the Commonwealth of Massachusetts, excluding its choice of law rules.

9. U.S. GOVERNMENT PURCHASES: If SOFTWARE is acquired for or on behalf of the U.S. Government, then:

a. It is recognized and agreed that SOFTWARE contains commercial computer software, as that term is used in FAR Parts 12 and 27.4 (and any applicable agency supplements thereto), and DFARS Parts 211.70, 227.4 (OCT 1988), 227.71 (JUN 1995) and 227.72 (JUN 1995).

b. If this purchase is subject to FAR Part 27, and FAR 52.227-19 has been incorporated into the terms of this purchase, the Government's rights in the SOFTWARE are no greater than RESTRICTED RIGHTS as specified in FAR 52.227-19.

c. If this purchase is subject to DFARS Part 227 (OCT 1988), the Government's rights in the SOFTWARE are no greater than RESTRICTED RIGHTS as specified in DFARS 252.227-7013(c)(1)(i).

d. The Government's rights in the SOFTWARE are no greater than the rights expressly stated in the body of this Standard Licensing Agreement, if this purchase is subject to (i) FAR Part 12; (ii) FAR Part 27, and FAR 52.227-19 has not been incorporated into the terms of this purchase; (iii) DFARS Part 211.70; (iv) DFARS Parts 227.71 and 227.72 (JUN 1995); or (v) any other regulation or clause permitting the contractor to deliver commercial computer software under the contractor's standard commercial license.

e. Any related technical data shall be delivered to the Government with no more than limited rights.

10. DISCLAIMER OF SOFTWARE WARRANTY: Boston Dynamics PROVIDES THE SOFTWARE TO LICENSEE “AS IS” AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR WITH RESPECT TO INTELLECTUAL PROPERTY OWNERSHIP, NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY ANY Boston Dynamics EMPLOYEE, REPRESENTATIVE OR DISTRIBUTOR WILL CREATE A WARRANTY FOR THE SOFTWARE, AND LICENSEE MAY NOT RELY ON ANY SUCH INFORMATION OR ADVICE.

11. Limited Warranty on Media: Boston Dynamics warrants the media on which SOFTWARE is recorded and provided to LICENSEE under this Agreement to be free from defects in materials and workmanship under normal use for a period of ninety (90) days after the date of the original delivery of SOFTWARE to LICENSEE. Such warranty is solely for LICENSEE's benefit and LICENSEE has no authority to assign, pass through or transfer this warranty to any other person or entity. If LICENSEE greater than the rights expressly stated in the body of th returns any defective media to Boston Dynamics or an authorized Boston Dynamics representative during the warranty period with proof of purchase, Boston Dynamics will, at its sole option, either replace such defective media or refund the purchase price for such media. This warranty will not apply to any media that has been damaged by abuse, accident or misuse. The foregoing warranty sets forth Boston Dynamics's entire liability and LICENSEE's exclusive remedy for any defects in any media and is in lieu of, and Boston Dynamics disclaims, all other warranties, express, implied, or otherwise, including without limitation any warranty of merchantability or fitness for a particular purpose.

12. LIMITATION OF LIABILITY: IN NO EVENT SHALL Boston Dynamics OR ITS LICENSORS BE LIABLE TO LICENSEE FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND (INCLUDING WITHOUT LIMITATION THE COST OF COVER, DAMAGES ARISING FROM LOSS OF DATA, USE, PROFITS OR GOODWILL, OR PROPERTY DAMAGE), WHETHER OR NOT BOSTON DYNAMICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY ARISING OUT OF THIS AGREEMENT. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY. BOSTON DYNAMICS's LIABILITY ARISING OUT OF THIS SOFTWARE LICENSE AGREEMENT AND/OR LICENSEE'S USE OR POSSESSION OF THE SOFTWARE, INCLUDING WITHOUT LIMITATION ANY AND ALL CLAIMS COMBINED, WILL NOT EXCEED THE AMOUNT OF THE LICENSE FEE LICENSEE PAID FOR THE SOFTWARE PROVIDED UNDER THIS AGREEMENT.

13. Laws Governing Warranties and Liability: The law(s) of a jurisdiction may define the scope of warranty to be provided for products or the manner in which a supplier's liability may be limited, and such law(s) shall govern this Agreement only to the extent a party protected by such law(s) cannot waive the protection thereof by contract. In the U.S. and other countries, some states, territories or other principalities do not allow the limitation or exclusion of liability for incidental or consequential damages, or allow the exclusion of implied warranties, so the limitation and exclusion above may not apply to LICENSEE, and LICENSEE may have other rights that vary from state, territory or principality to state, territory or principality.

Table of Contents

List of Figures

List of Tables

1 LittleDog System

LittleDog is a small quadruped robot for research applications. LittleDog was designed to explore the fundamental relationships among motor learning, dynamic control, perception of the environment, and rough terrain locomotion.

LittleDog has four legs, each powered by three electric motors. The legs have a large range of motion and workspace. The motors are strong enough for dynamic locomotion, including climbing. There is an onboard computer that performs sensing, actuator control and communications tasks. LittleDog's sensors measure joint angles, body orientation and foot/ground contact. Onboard lithium polymer batteries allow for approximately 30 minutes of continuous operation without recharging. Tethered power operation is available to extend operation time during development.

LittleDog comes with a set of software control utilities that are accessed through the LittleDog API. User developed control programs run on the Host computer and communicate with the robot over a wireless link to support remote operation and analysis.

A motion tracking system provides real-time measurement of the robot’s body and limb locations in the test environment. The motion tracking system is based on a high precision Vicon motion capture (MoCap) system. The MoCap system uses specialized cameras, special light sources and retroreflective markers on the robot to report accurate positions of the robot and terrain. This information is then processed into parameters such as joint angles, body posture and terrain location and made available to the user through the Robot API.

1.1 System Overview

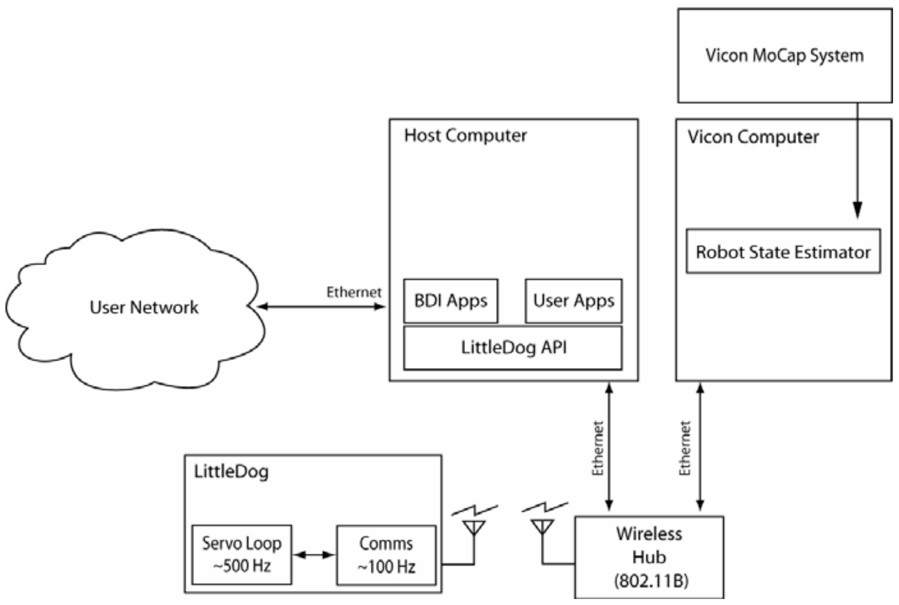


Figure 1: LittleDog System Schematic

The LittleDog System is comprised of a LittleDog Robot, a Host Computer, a MoCap Computer and a Vicon MoCap System. Figure 1 illustrates how these components are connected together. This manual describes how to set up the LittleDog Robot system and how to get started using it.

The LittleDog system comes pre-configured with regard to communications. The robot communicates with the Host computer over a private wireless Ethernet network. The Vicon motion capture system communicates with the Host computer via a wired connection. The Host computer has a second wired Ethernet port that can be used to connect the Host computer to the outside world.

Getting all that networking to perform correctly at high rates is a challenge. We encourage you to leave the network settings as pre-configured, with the exception of the Host computer's external network interface, which you can set as you please.

Robot control applications run on the Host computer and communicate with both the robot and motion capture system at 100Hz, collecting a current snapshot of the system state and issuing commands to the robot every 10ms. Joint level servo controllers are provided on the LittleDog robot and operate at a sampling frequency of 500Hz. A complete log of all robot state, motion capture data, and robot commands is collected on the Host computer whenever an experimental trial is performed with the LittleDog system.

1.2 Platform Specifications

1.2.1 Mechanical Overview

Body Dimensions:	340 x 180 x 143 mm	
Ground Clearance:	120	mm
Leg Length:	180	mm
Total Weight:	3	kg

LittleDog has twelve actuated degrees of freedom, each powered by an electric motor with position sensing encoder. Each leg also has a single axis force sensor in the lower leg, capable of detecting touchdown events and measuring approximate lower leg axial forces.

Detailed parameters and kinematics information can be found in Section 2.3 .

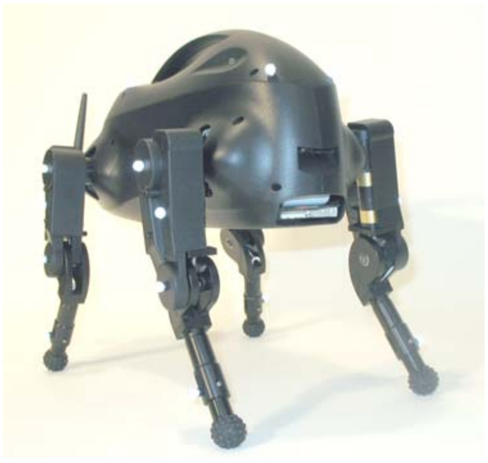


Figure 2: The LittleDog Platform

1.2.2 Electrical Overview

LittleDog's electronics includes the following:

- x86 architecture, 266 MHz CPU with 128MB RAM
- Mini-PCI bus, hosting onboard 802.11b wireless Ethernet card
- Platform I/O Support
  - Power
  - MEMS IMU (angles, rates, linear accelerations)
  - Proximity Sensor
- Power Source: 2.1 Ah @ 14.8V LiPol (4 cells in series)
  - Optional external tethered power

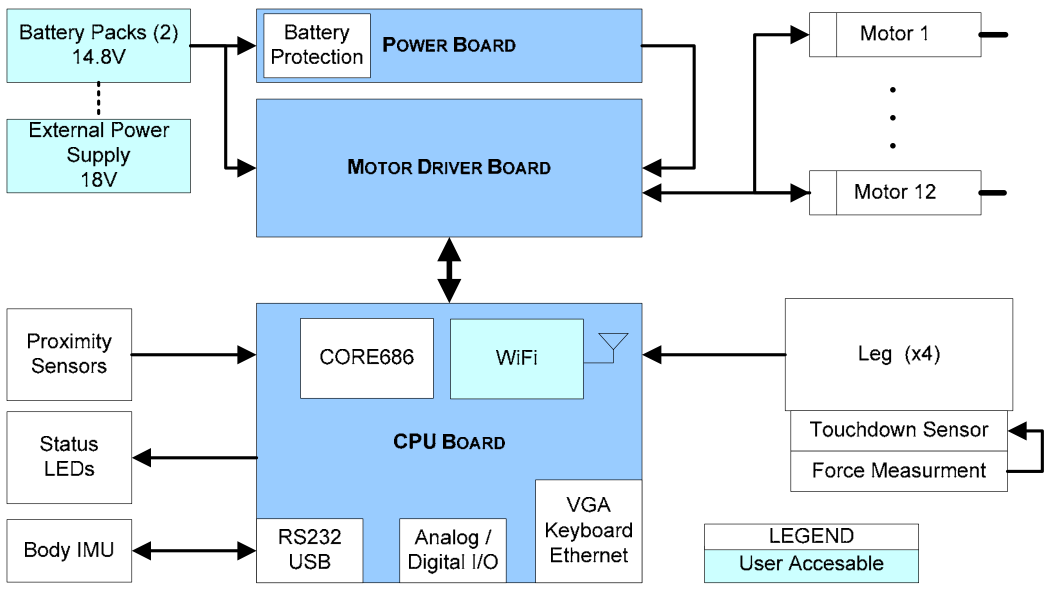


Figure 3: Electrical System Layout

1.3 Getting Started

1.3.1 Packing List:

- One HOST PC with Linux installed

- One MoCap PC with Windows XP installed, this box also includes:
    - One KVM (keyboard, video and mouse switch)
    - One Keyboard
    - One Mouse
    - Four color coded Ethernet cables
    - Linksys Wireless Bridge
    - Linksys Ethernet Switch

- One LCD Monitor

- One Vicon MoCap System, including:
    - Six Vicon MX40 4Mpixel Cameras, with strobes and lenses
    - Six 50m camera cables
    - One Vicon MXNet interface box
    - One calibration kit
    - One sample general purpose marker kit (not used in the LittleDog System)
    - Software (already installed on your computer), license dongles and Vicon manual

- One LittleDog Robot Shipping Case
    - One LittleDog robot
    - Two sets of battery chargers with power supplies
    - Six LiPol battery packs
    - One robot stand
    - One small spare kit: 2 MoCap markers, 1 thumb screw, 1 spare antenna
    - One User Manual

1.3.2 Setting Up the System

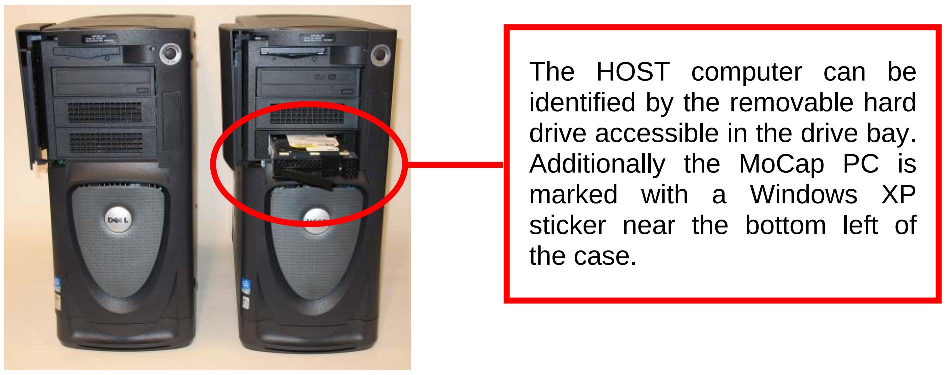
Please take the time to carefully read the instructions provided on how to unpack, assemble and run your LittleDog Robot System. It may take several hours to set up, and up to four people during parts of the MoCap frame assembly, as described later in Section 2.1.1.1.

There are a few items that are not provided that you will need to purchase separately prior to setting up your system:

1. One MoCap camera support frame, as specified by the LittleDog Government Project Team. Please purchase the correct frame so that your MoCap system closely matches the government test facility.
2. Six "BOGEN / Manfrotto Super Clamp with 3025 Head" mounts, readily available at most major online photo & video supply stores, the following part number is valid online at B&H Photo (http://www.bhphotovideo.com/):
    a. PN: BO2910 , BOGEN / Manfrotto Super Clamp with 3025 Head
3. Two surge protected power strips
4. Black cloth for masking items from the motion capture system

You will need a lab space that is about 12 ft by 15 ft with 8.5 ft ceilings and enough additional room to have a desk near by. To set up your system we recommend you perform steps in the following order:

1. The first step is to ensure you are in possession of all of the equipment listed in Packing List , Section 1.3.1, as well as the additional equipment listed above.
2. Set up your MoCap system. For instructions on setting up your MoCap system see Section 2.1 .
3. Unpack the HOST and MoCap computers as well as the Monitor, KVM switch, wireless bridge and Ethernet switch. Place both computers in a safe location near the intended MoCap tracking volume.


The HOST computer can be identified by the removable hard drive accessible in the drive bay. Additionally the MoCap PC is marked with a Windows XP sticker near the bottom left of the case.

4. The next step is to connect it all together. Place the two computers side by side, with the wireless bridge and Ethernet switch stacked on top, as shown in Figure 4. Using the short **BLUE** Ethernet patch cable connect the wireless bridge to the number (1) position on the Ethernet switch. Connect the **RED** three foot patch from the PCI Ethernet card in the MoCap PC to position (2) on the Ethernet switch. Connect the **YELLOW** three foot patch from the PCI Ethernet card in the Host PC to position (3) on the Ethernet switch. The built in Ethernet port on the MoCap PC goes to the Vicon MXnet box (the port is found on the back of the MXnet). The built in Ethernet port on the Host PC goes to the outside world.

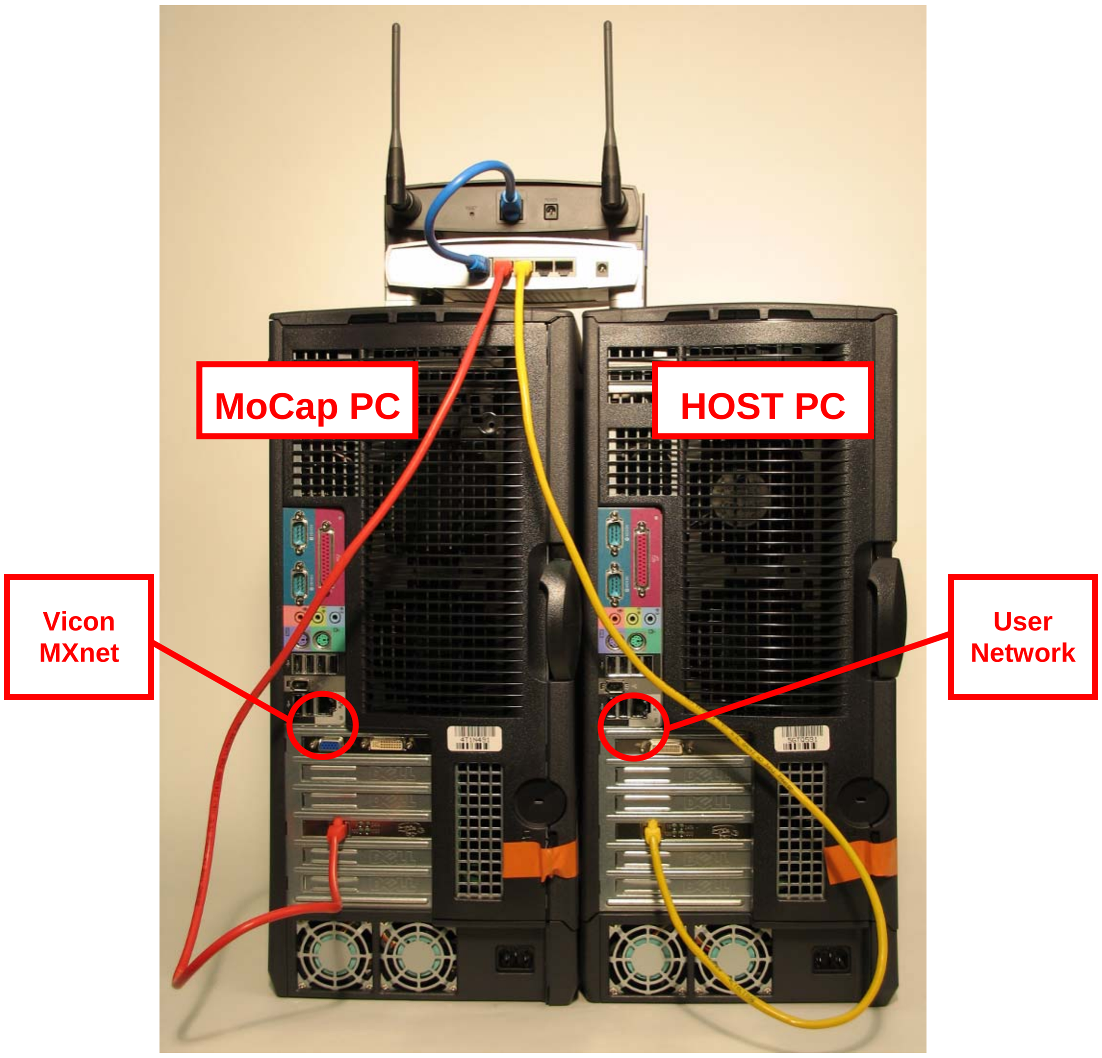


Figure 4: Setting up your system

Next plug the KVM switch to both machines. The KVM inputs go to the Monitor (via the provided DVI to VGA adapters), two USB ports, the sound and mic located on the back of each computer. The output of the KVM goes to the mouse, keyboard and monitor.

Hook up all the power cords.

5. Unpacking your Robot. Please note how we have packed the robot so you can pack it the same way in the unlikely event you need to ship the robot back to us.. Each kit should contain the items listed in the Packing List, Section 1.3.1.

6. Install a fresh set of batteries, or connect the external power tether. For information on how to charge and install your batteries, see Section 5. (OPTIONAL: setting up an external power supply, see Section 5.3.2 .)
7. You are ready to install the Robot API and power up your system. Please read Section 1.4 to get started using your system.

1.4 Running the Robot for the First Time

Prior to operating your LittleDog robot for the first time, there are administrative tasks you must perform to properly configure your robot system.

We suggest that you change the root password on your host computer system from the default value it was shipped with. After changing your root password you must contact both Boston Dynamics (at littledog@BostonDynamics.com) and the Learning Locomotion Government test group to inform them of your new root password. This is critical for testing that will be performed by the test group, and will allow Boston Dynamics remote access to your host computer to provide remote diagnosis and support if you encounter any problems with your system.

Next create a new user account (using `useradd`) on your Host computer (linux) that you will use for doing LittleDog software development. Finally you should log in to the host computer using this newly created account.

1.4.1 Installing LittleDog Software

Copy or download the LittleDog API software development kit to your home directory either from the CD provided with this manual or from the address given to you at the LittleDog training session. Uncompress and untar the kit in your home directory, this will create a directory named `LittleDogAPI_1_0_0` which contains all the software tools associated with operating and analyzing data from the robot. A subdirectory titled `win32` contains a zip file which includes a set of plotting and data analysis tools that you can install on a Windows PC (see the `README.txt` file included there for details).

To successfully build and run the example applications for LittleDog, you must set two environment variables.

```
LITTLEDOG = ~/LittleDogAPI_1_0_0
QTDIR = /usr/lib/qt-3.3
```

You can now change your working directory to `$LITTLEDOG` and finish configuring your Host control environment.

Optionally, at this point you can also complete configuration of your Vicon motion capture system, if it is currently set up. To do this you should refer to the `README.txt` file contained in `$LITTLEDOG/vicon_files` as well as the instructions contained in Section 2.1.2 of this manual. This directory contains the current Vicon models for the LittleDog robot and the currently deployed terrain boards, and they must be placed on the motion capture control PC as described in Section 2 of this manual in order for the motion capture system to function with the LittleDog control system.

Finally, you must edit your performer-specific configuration file. To do this, change directory to `$LITTLEDOG/config`, then copy `bdi_rt_DEFAULT.cfg` to `bdi_rt.cfg` and open `bdi_rt.cfg` in an editor. You must ensure that the IP address for the robot is set correctly in this file (it will be of the form 10.66.68.xxx, where xxx is the number indicated on the back of your robots CF disk), and that your performer id (as specified by the Government test group) is set correctly in the “performer” field near the bottom of this file.

1.4.2 Running the *BASIC* Example

If your motion capture system is available for use, we suggest that you perform the calibration and setup procedures associated with it prior to running this example. Performing the calibration and setup will give you experience with the entire LittleDog system through this example, and will generate data files that include motion capture information. Refer to Section 2.2 for instruction on setup and calibration of the motion capture system.

With your LittleDog developers kit installed as described above, you can now change directory to `$LITTLEDOG/examples/basic`. You can build the basic example by executing `qmake basic.pro` and then `make`. Confirm that both commands complete and that there are no errors reported by either. The entire program for this example is contained in `main.cpp`. This program will connect to the motion capture system, initiate communications with the robot, instruct the robot to calibrate itself, and finally it will sweep the joints through sinusoidal motions until the user interrupts the program by pressing control-C.

This program does not interact with the user at all after being started, so to begin with you must ensure that the robot is positioned safely on its stand. Begin by placing the robot on its stand with the front legs extended forward and hind legs extended backward. Be sure not to force motion of any of the joints (see Figure 5). Make sure that the robot is secure and that there is clearance around the robot for the legs to swing freely.

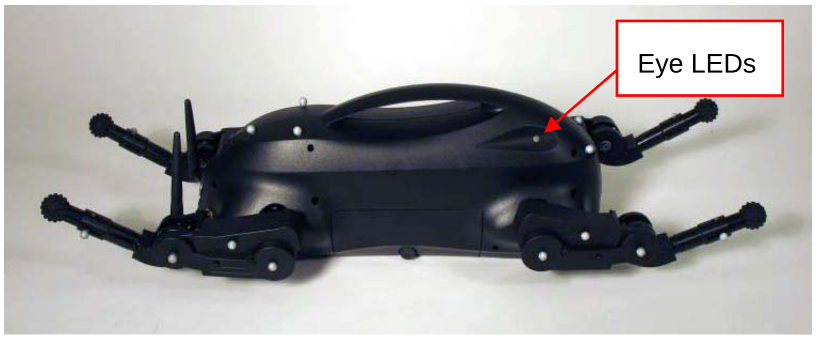


Figure 5: Calibration posture.

CAUTION: Although the motions produced by this example will execute safely when the robot is on the stand, most motions will not be safely executed on the stand. In general it is advised that you hold the robot by its handle when initiating motion.

Power your LittleDog on so that the red LED in the power switch illuminates. After approximately 45 seconds the eyes of the robot will turn red. This indicates that the embedded software on the robot has started and that it is waiting for a connection from the host computer. The LED eyes on the robot indicate status of the onboard software (DARK: robot off or trial running, RED: waiting for connection, ORANGE: connected to host but not calibrated, GREEN: calibrating or calibrated – see Table 6).

At this point you can execute `./basic` on the host computer to cause LittleDog to run through the basic motion sequence.

The sequence of execution for this program is as follows:

- Host connects to the Vicon motion capture system
- Host connects to LittleDog (eyes turn yellow/orange)
- Host calibrates the joints of the robot (eyes turn green)
    1. knees fold up
    2. hips (rx) rotate down
    3. hips (ry) rotate up
    4. the robot unfolds itself to its zero configuration with the legs extended straight down below the robot with both the upper and lower legs perpendicular to the ground (see Figure 6)
- The host commands sinusoidal motion to all the joints on LittleDog (eyes go dark)

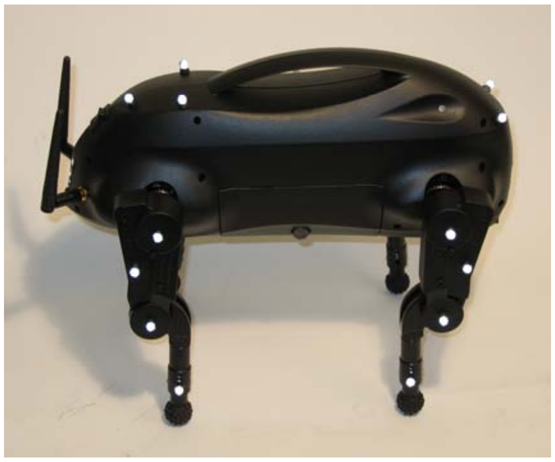


Figure 6: Robot Zero Configuration Position

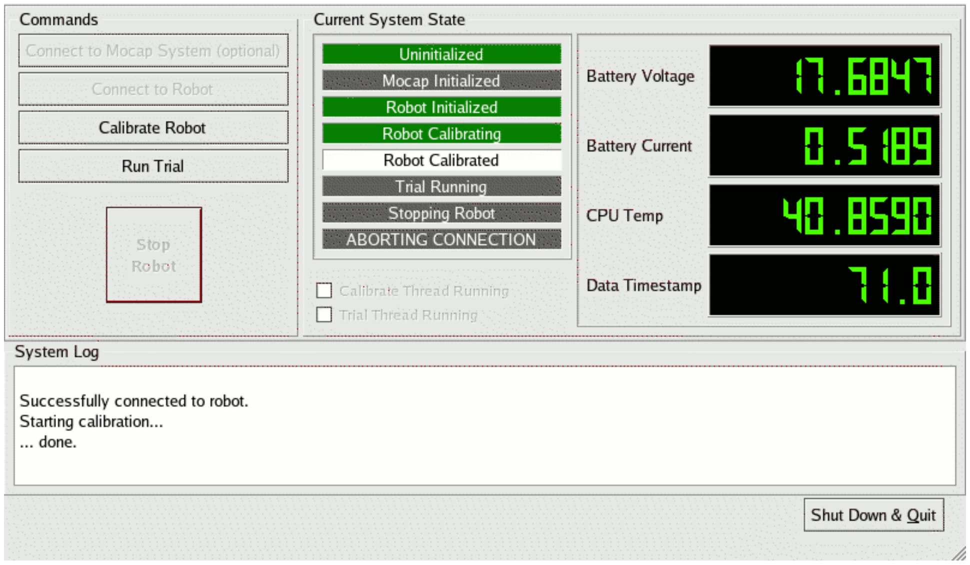
This program can be terminated at any time by pressing control-C on the host computer (eyes turn red again, indicating that the robot is waiting for a new connection).

If the basic example encounters any errors it will display an error message, and if the error prevents further operation, the program will exit. Typical errors include incorrect IP address settings in your `bdi_rt.cfg` file, or problems with calibration due to the robot being unable to complete motions. It is important that the legs be free of any obstacles during the calibration process as the robot detects motor stall in order to align the encoders, and any external obstacles may cause this process to fail. The preferred method of calibration is for the user to hold the robot by its handle when executing the calibration procedure, this will ensure that the legs to not inadvertently make contact with either the stand or any other objects during the calibration process.

After termination of the basic program you will find a dataset file (.data) containing the full record of robot state and commands that were sent to the robot during this trial. There are three ways you can access the information contained in this file; you can use BDIPlot as described in Section 4 of this manual, you can use the `bduDataSetFile` utility classes provided with the development kit and documented in the electronic API reference manual included with your kit (see `$LITTLEDOG/doc/html/index.html`), or you can load the file directly into MatLab using the tools provided in the `win32` directory of your development kit (see `$LITTLEDOG/win32`).

1.4.3 Running the *WALK* Example

While the basic example provides a convenient method to verify basic functionality of LittleDog, it does not afford much interaction or provide any operator feedback. The walk example included with the development kit includes a graphical user interface (GUI) and provides a more complete example of how LittleDog can be operated. This example makes use of the LittleDog reference UI – this user interface system must be used with all application programs submitted for evaluation by the government test team as part of the Learning locomotion program to facilitate testing. This example allows a user to control the initialization and execution of robot control software through the interface in order to execute a number of experimental walking trials with the robot.



Battery Voltage 17.6847
Battery Current 0.5189
CPU Temp 40.8590
Data Timestamp 71.0

```
Successfully connected to robot.
Starting calibration...
... done.
```

Figure 7: LittleDog walk example GUI

First we will build the walk example. Change directory to `$LITTLEDOG/examples/walk`. Use `qmake walk.pro; make` to build the example application, again check for any errors and resolve them before proceeding. You can now start the interface by executing `./walk`. The program will display a GUI that allows you to manually sequence through the process of connecting to both the motion capture system and the robot, calibrating the robot, and controlling a trial. The text display area at the bottom of the window reports status and error messages from the robot and robot interface software (users can use the utility class `bduLog` to report messages to this text box).

To operate the example, begin by pressing the button labeled "Connect to Mocap System." This is an optional step, but will enable you to collect information from the motion capture system during your trial, presuming that the motion capture system is properly set up and running. Next you can press the "Connect to Robot" button – this will initiate communications between the robot and the host computer. Note both how the central display indicators change to indicate the current state of the robot system, and that after connecting to the robot the status displays on the right side of the window actively report the health of the robot (as shown in Figure 7). At this point the robots eyes should have changed from Red to Orange, indicating that the host is successfully connected to and communicating with the robot.

Much like what happened automatically in the previous example we can now go through the calibration process. Calibrate the robot by pressing the "Calibrate Robot" button. This will present you with a small dialog box asking you to confirm the current posture of the robot – adjust the robot to match the posture shown in the picture (front legs stretched forward, rear legs stretched backwards). If you are using the stand, be sure that the robot is positioned squarely on its stand, and press "OK" to initiate joint calibration of the robot. During the calibration process the robots eyes change from Orange to Green to indicate that calibration is in process. At the end of the calibration process the eyes will stay Green to indicate successful calibration, or return to orange to indicate that calibration failed.

NOTE: If you have previously calibrated the robot since powering it on, calibration is optional as the robot will remember the prior joint calibration parameters.

At this point you should note the state of the system, as reported by the middle section of the UI, indicates that the robot is calibrated. You should also visually confirm that the robot is in a "standing" zero configuration posture (all four legs extended perpendicular to the ground), if this is not the case, you should reposition the legs to be outstretched and execute the calibration procedure again.

You can now lift the robot by the handle, and press the button labeled "Run Trial" to initiate a trial. Be aware that there is no active control of the robot joints prior to pressing "Run Trial," so the robot will be limp when you pick it up off of the stand. As soon as the trial begins execution, the robot will begin performing a kinematic walk behavior.

At any time while LittleDog is moving, either during a trial, or during calibration, the large red "Stop Robot" button is active, and can be pressed to stop motion of LittleDog. Using the "Stop" button during a trial results in the LittleDog software finalizing the data sets associated with the most recent run, and returns the software system to the "Robot Calibrated" state. From this state you can safely begin another trial or exit from the example by using the "Shutdown and Quit" button located at the lower right corner of the window. Note that the eyes will be dark during execution of a trial to avoid interference with the motion capture system.

1.4.4 Notes about LittleDog operation

The status displays located on the upper right side of the UI window provide a real-time assessment of robot status. These displays include battery voltage, battery current, CPU temperature, and a data timestamp. Battery voltage and CPU Temperature will both change color to warn the operator of critical situations (over temperature for the CPU, or low battery voltage). The battery current allows the user to assess overall performance – typical hotel load from the electronics is approximately 0.5A, typical operation will draw 3-4A, the absolute maximum draw for the platform is 12.5A and should not be seen in operation. The timestamp provides a crude indication of communications status between the robot and host. When the two are communicating the timestamp will increase regularly, if communications are dropped the timestamp may freeze and jump forward as communications are recovered. Note that all of these values are recorded in the data set associated with a trial whenever a trial is running, so users can assess performance after a trial is completed rather than watching these displayed values in real-time.

Be aware that if the host loses communications with the robot at any point, the majority of buttons will become inactive, and the status display will indicate that the host is "ABORTING CONNECTION." This is a fatal error, and the user should recover by exiting the UI via the "Shut Down & Quit" button and restart the application. This is an indication that the host and robot were unable to communicate for an extended period of time and that the connection between them has been terminated to force the robot into a safe state (the robot eyes will turn Red).

2 MoCap System

The high resolution motion capture system (MoCap) is based on the Vicon MX system. Each system consists of 6 camera-like devices that are mounted around the terrain board. Each camera has a strobed light source and an array sensor. The robot will be instrumented with retroreflectors mounted on its body, legs and feet. Data from all six cameras are integrated by a tracking control unit that correlates and transforms the data, and makes them available to the control system of the LittleDog robots.

This section will discuss the MoCap system as it applies to the LittleDog System. For further information about specific components of the Vicon system, including hardware and software, please see the Vicon User Manual included with your system.

2.1 Setting Up

Setting up the MoCap system involves assembling a large support structure for the cameras, mounting and aiming the cameras, connecting the cameras to the Vicon MX Net, and finally setting up the camera hardware and software. Once that is complete the system is ready for calibration.

2.1.1 Mounting the Cameras

2.1.1.1 Building the Frame

The MoCap frame, Figure 9(C), comes in several large pieces and needs to be assembled. To do so you will need a few extra sets of hands. Figure 8 shows a simplified view of the components that will come with the truss structures.

Assembly of the truss is relatively simple. The four corner posts are bolted to the floor base as shown in Figure 9(A). The upper structure, forming a large rectangle as shown in Figure 9(C), is pre assembled on the ground and lifted up onto the four corner posts. The only tricky part to the assembly order is to ensure that you install one of the long support beams with two end pieces in place as the last step in assembling the top structure, as shown in Figure 9(B).

Once the upper structure is assembled you are ready to lift it into place. The lift will require a minimum of five people, four to support the top structure and one to place the corner legs (more people makes the job easier, tall colleagues are the best).

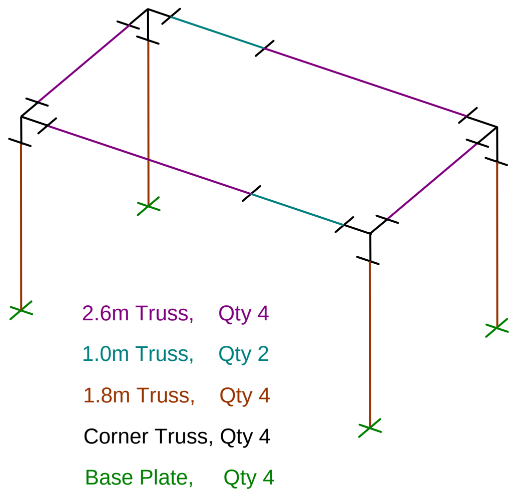
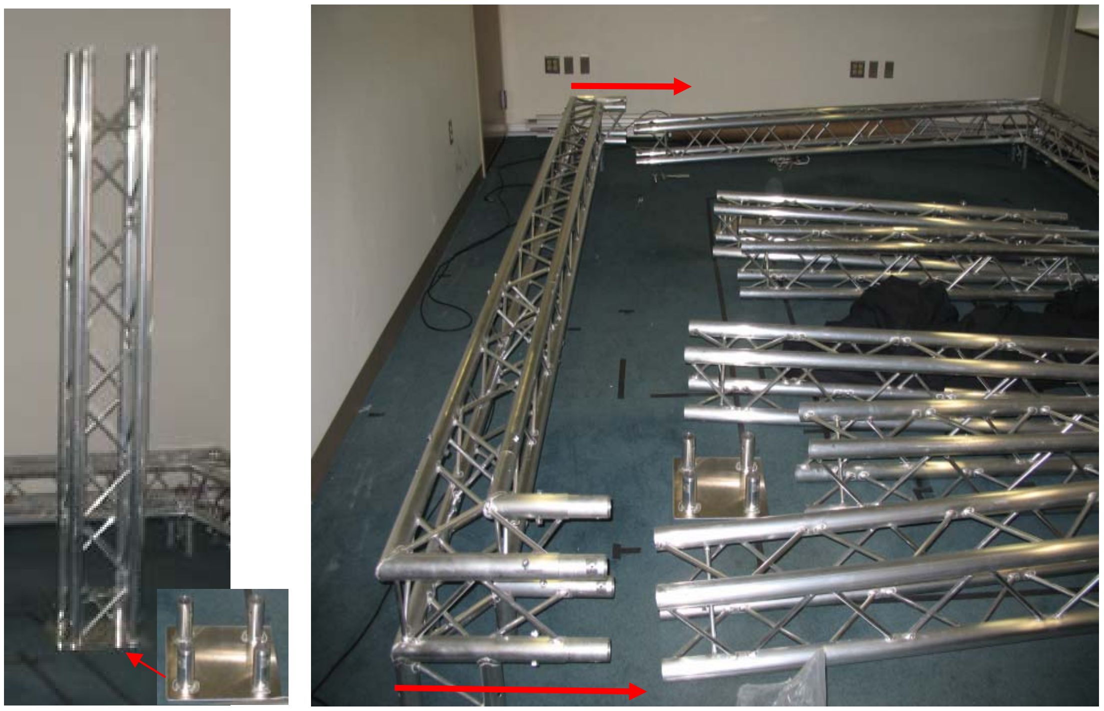
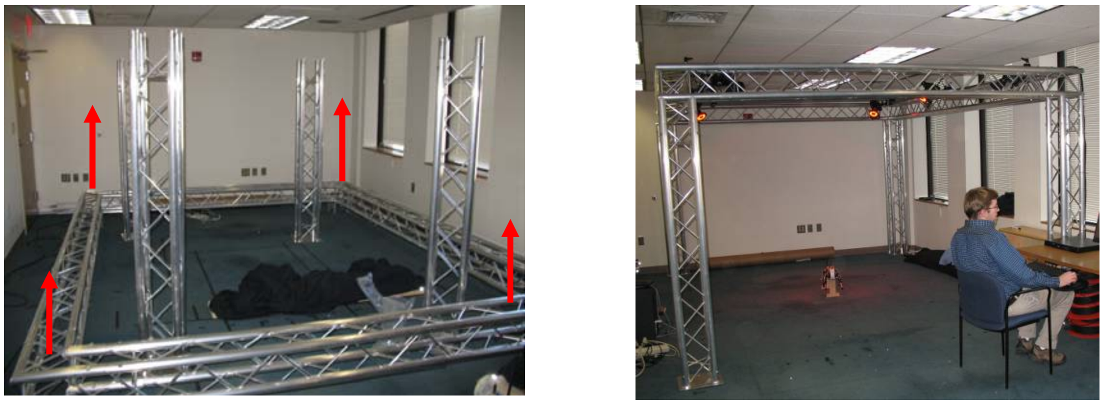


Figure 8: MoCap Frame Components



(A) (B)



(C) (D)

Figure 9: MoCap Frame Assembly

2.1.2 Vicon System Configuration

2.1.2.1 Realtime Engine Parameters

The Vicon realtime engine has a number of parameters which can be tuned to change marker reconstruction, marker labeling, and kinematic fitting of the LittleDog model. A realtime engine parameter settings file (.rtp) has been included with your LittleDog system. Load the settings file by first going into setup mode by pressing the *Setup* tab on the upper left hand side. Select *RealTime Engine*, *RealTime Engine Parameters*, *Load RealTime Options…*, and select the parameters file.

2.1.2.2 Vicon Skeleton Files

For MoCap system operation you must have the Vicon skeleton (*.vsk) files located in the session directory you are currently operating in. To check which session you're currently in, press the *Data Management* tab at the top left hand side of the window. The Vicon system has the capability to record and post-process motion, but this is not required for real-time motion capture which will be the primary use of the system. If you change capture sessions, just copy the Vicon skeleton files over to the new session directory located within *C:\MOCAP_DATA*. Three Vicon skeleton files have been included with your LittleDog system: *littledog.vsk*, *terrainA.vsk*, and *terrainB.vsk*. Copy these files over to your current session directory when setting up your system.

In the directory $LITTLEDOG/vicon_files on the host computer there are four files that must be copied to the Vicon motion capture PC. The files and where they should be copied to are:

- `littledog.vsk`
- `terrainA.vsk`
- `terrainB.vsk`
- `RTEparams_001.rtp`

2.2 Calibration

A detailed description of how to calibrate your MoCap system can be found in the Vicon documentation located in *C:\Program Files\Vicon\ Documentation\Books\Foundation Guides\Vicon Manual\ preparation_v1_2.pdf*

Groups should calibrate typically at least once every day they are using the motion capture system. Additionally, if the motion capture frame is jarred or hit by persons or objects, the cameras may move slightly and your system performance will suffer if the system is not recalibrated. The motion capture workspace is small and calibrating takes less than 5 minutes; so groups should calibrate whenever necessary to maintain peak system performance.

When should you recalibrate the system:

1. At the beginning of every test day.
2. If markers oscillate in the Vicon 3D workspace window.
3. If a particular camera has a large percentage of unassigned rays (as shown by the white ray in the 3D workspace window)
4. If there are multiple reconstructions of a single marker appearing in the 3D workspace window.

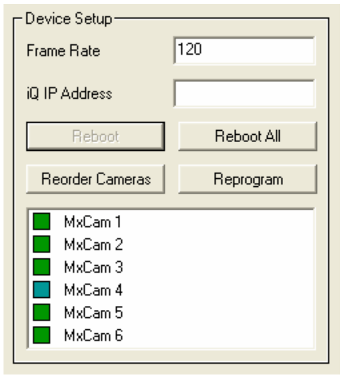
2.2.1 Quick and Dirty System Startup & Calibration

System calibration is a quick and painless procedure that all members of your group should be able to do with ease. With very little practice you will be able to start and calibrate your system in only a few minutes by following the steps outlined below.

1. **Starting the System**
   a. Turn on the MoCap computer and the Vicon MX Net system (the switch is on the back.)
   b. On the MoCap computer, start the Vicon iQ 2.0 Software. During program startup the red LED rings surrounding the camera lenses should turn off and on again after about a second.
   c. Change into setup mode by clicking the

button on the upper left hand side of the Vicon iQ window.
   d. Click the hardware configurations tab

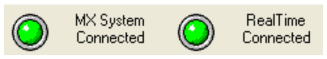
on the right hand side and make sure that cameras 1-6 are listed on the right hand side as shown below. If they are not, check to make sure the system is connected properly and restart the Vicon software.



Frame Rate 120
iQ IP Address

🟩 MxCam 1
🟩 MxCam 2
🟩 MxCam 3
🟦 MxCam 4
🟩 MxCam 5
🟩 MxCam 6

e. Connect to the Vicon system by clicking the icon. If the system connects properly the green and blue boxes next to the camera names will turn a lighter color and the system status bar on the bottom of the Vicon window should show you have now connected. If there was a problem connecting to any of the cameras, the color of the boxes next to their name will not change. If this happens try restarting the Vicon iQ software and cycling the power to the Vicon MX Net system.



2. Calibrating the System

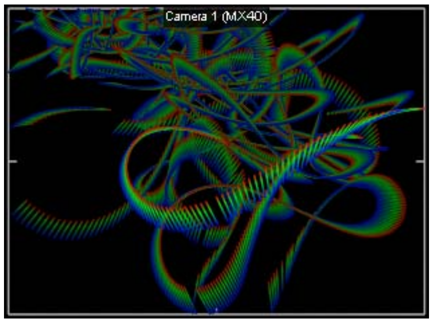
a. Change into calibration mode my by clicking the Calibrate button on the upper left hand side of the Vicon window.

b. On the right hand side of the Vicon window, make sure the *120_mm_Wand* and the *Ergo_9mm_LFrame* are selected for the Wand and LFrame respectively.

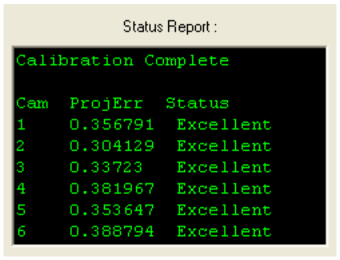
c. Click the Camera tab at the top of the Vicon window, and select cameras 1-6 while pressing the *Ctrl* key. You should see the output of each of the six cameras in the center of the Vicon window.

d. Make sure there are no reflective objects in the motion capture workspace. Remove the robot and cover any terrain board markers with a dark piece of fabric.

e. Press the Start Wand Wave button on the upper right hand side and begin the wand wave. It is important to cover the entire motion capture space a few times with wand. Make an effort to keep out of the view of the cameras when waving the wand. While waving the wand, a rainbow pattern signifying the wand will appear in the each of the six camera views. Make sure to get good coverage in each of the six cameras as shown below.



f. Press the Stop Wand Wave button to start the camera position fitting iterations; this should take about a minute and you will see the status on the bottom of Vicon window. When calibration is finished, the quality of calibration is displayed in the status window as shown below. We have consistently been able to get mostly excellent results; if you have less than this you may want to try calibration again.


Status Report :

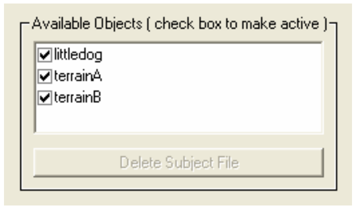
```
Calibration Complete

Cam  ProjErr   Status
1    0.356791  Excellent
2    0.304129  Excellent
3    0.33723   Excellent
4    0.381967  Excellent
5    0.353647  Excellent
6    0.388794  Excellent
```

g. Place the L-Frame from your calibration toolkit into the workspace. This will define your world origin, so it may make some sense to align it with the orientation of one of the terrain boards.

h. Click the Track L-Frame and then the Set Origin buttons. If the world is inverted for some reason in the 3D workspace window, repeat this step.

i. Calibration is now complete. Change into capture mode by pressing the Capture button at the top of the Vicon window.

j. Press the Active Objects tab on the right hand side and make sure that the LittleDog model and any terrain board models you have in the workspace are check-marked as shown below. If the models are not fitting the reconstructed markers well, try unloading and reloading the models by toggling the check-mark.


Available Objects (check box to make active)

☑ littledog
☑ terrainA
☑ terrainB

2.3 Kinematics: Definitions and Coordinate Conventions

2.3.1 Body Coordinates

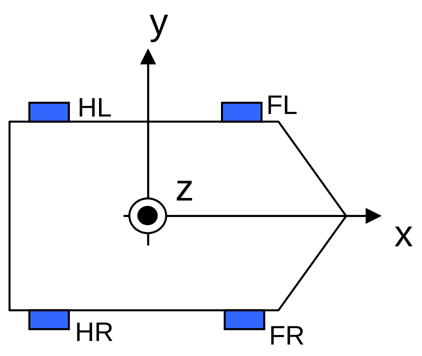
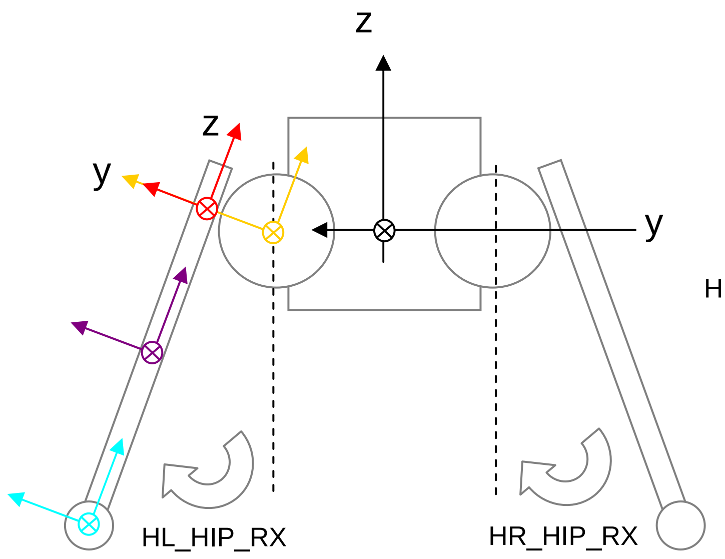


Figure 10: Body Coordinates

2.3.2 Hip and Leg Coordinate Frames

`



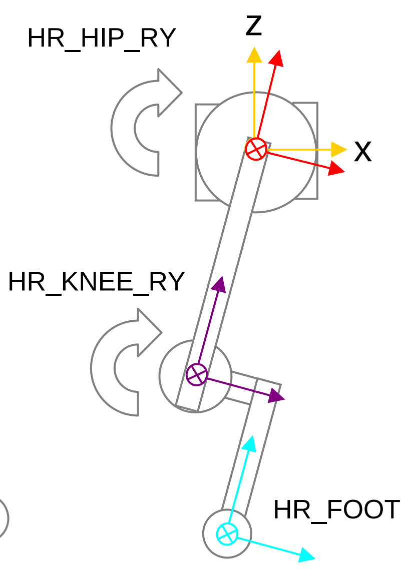


Figure 11: Hip Wing Coordinates

Figure 12: Leg Crank Coordinates

Table 1: Joint Offsets

Frame Name	Color Code	X [m]	Y [m]	Z [m]	Description
FL_HIP_RX	ORANGE	0.101	0.0365	0.0	Position with respect to Body Frame
FL_HIP_RY	RED	0.0	0.0302	0.0	Position with respect to Hip X Frame
FL_KNEE_RY	PURPLE	0.0	0.0	-0.0751	Position with respect to Hip Y Frame
FL_FOOT	BLUE	-0.021	0.0	-0.1027	Position with respect to Knee Frame
FR_HIP_RX	ORANGE	0.101	-0.0365	0.0	Position with respect to Body Frame
FR_HIP_RY	RED	0.0	-0.0302	0.0	Position with respect to Hip X Frame
FR_KNEE_RY	PURPLE	0.0	0.0	-0.0751	Position with respect to Hip Y Frame
FR_FOOT	BLUE	-0.021	0.0	-0.1027	Position with respect to Knee Frame
HL_HIP_RX	ORANGE	-0.101	0.0365	0.0	Position with respect to Body Frame
HL_HIP_RY	RED	0.0	0.0302	0.0	Position with respect to Hip X Frame
HL_KNEE_RY	PURPLE	0.0	0.0	-0.0751	Position with respect to Hip Y Frame
HL_FOOT	BLUE	0.021	0.0	-0.1027	Position with respect to Knee Frame
HR_HIP_RX	ORANGE	-0.101	-0.0365	0.0	Position with respect to Body Frame
HR_HIP_RY	RED	0.0	-0.0302	0.0	Position with respect to Hip X Frame
HR_KNEE_RY	PURPLE	0.0	0.0	-0.0751	Position with respect to Hip Y Frame
HR_FOOT	BLUE	0.021	0.0	-0.1027	Position with respect to Knee Frame

Table 2: Joint Limits

Property	Min	Max	Unit	Description
FL_KNEE_RY	-3.1	1.0	Rad	Knee Limit
FL_HIP_RX	-0.6	0.6	Rad	Hip RX Limit
FL_HIP_RY	-3.5	2.4	Rad	Hip RY Limit
FR_KNEE_RY	-3.1	1.0	Rad	Knee Limit
FR_HIP_RX	-0.6	0.6	Rad	Hip RX Limit
FR_HIP_RY	-3.5	2.4	Rad	Hip RY Limit
HL_KNEE_RY	-1.0	3.1	Rad	Hip RY Limit
HL_HIP_RX	-0.6	0.6	Rad	Hip RY Limit
HL_HIP_RY	-2.4	3.5	Rad	Hip RY Limit
HR_KNEE_RY	-1.0	3.1	Rad	Knee Limit
HR_HIP_RX	-0.6	0.6	Rad	Hip RX Limit
HR_HIP_RY	-2.4	3.5	Rad	Hip RY Limit

Table 3: Body Parameters

Property	Value	Unit	Description
Mass	2240	g	Body mass w/ Battery
Length	304	mm	Total body length
Width	116 [180]	mm	Total body width [body plus hips]
Height	143	mm	Total body height
Clearance	120	mm	Total Ground Clearance (minimum)
L_{HIP}	203	mm	Hip separation, length (symmetrical about center)
W_{HIP}	73	mm	Hip separation, width (symmetrical about center)
Xz_{CG}	12	mm	Approximate Value (aligned with Y and X)

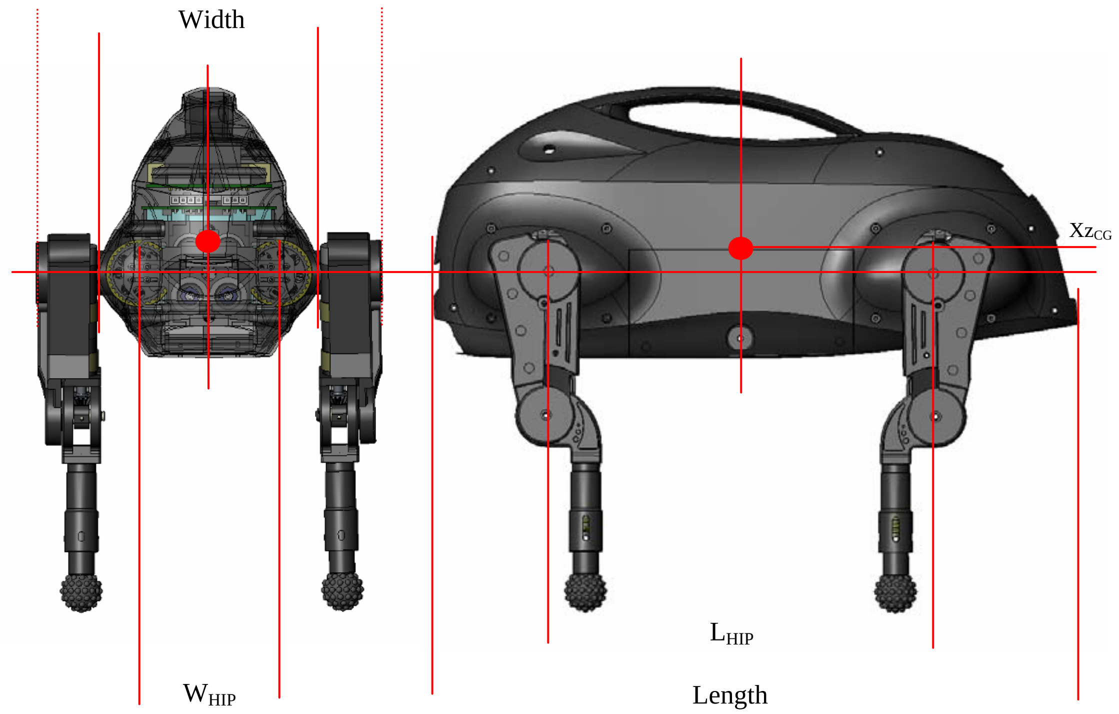


Figure 13: Body Parameters

Table 4: Leg Parameters

Property	**Value**	**Unit**	**Description**
Mass	190	g	Leg Mass
L	177.8	mm	Total leg length
L_{UP}	75.1	mm	Upper Leg Length
X_{OFFSET}	21	mm	Lower leg length
L_{LOW}	102.7	mm	Lower Leg Length
$FS_{DEFLECTION}$	2	mm	Max Force Sensor Deflection

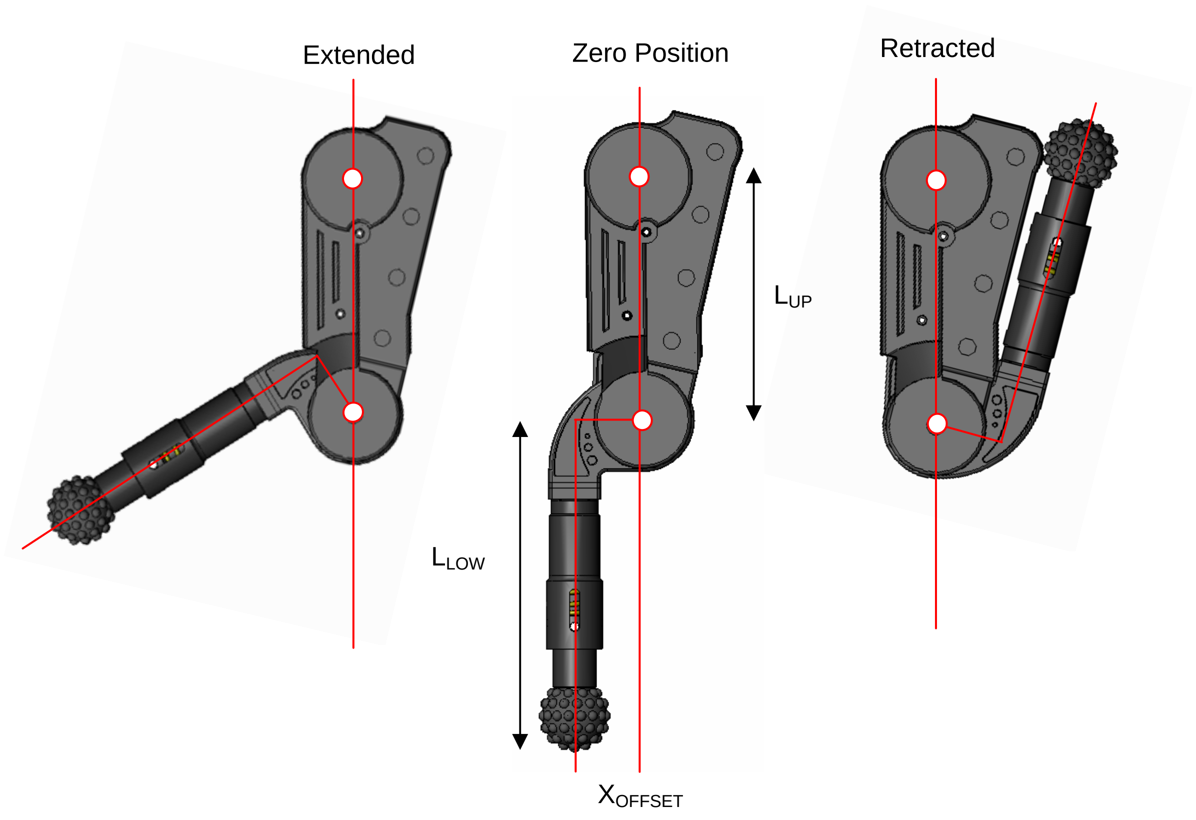


Figure 14: Leg Parameters

3 Software

Control of LittleDog is divided between an embedded computer in the robot and software running on the Host computer.

The LittleDog onboard computer performs the following functions:

- reads, calibrates and filters the robot's sensors
- writes to actuators
- servos individual joints
- communicates with the host computers

The onboard software is divided into two layers. The servo layer samples and filters data for each sensor, and performs joint level servo functions every 2 millisecond (500 hz). The communication layer supports communications between LittleDog and the Host computer, reporting sensor values and passing servo commands to the servo layer at a rate of 100Hz over an 802.11b wireless Ethernet link.

A control/interface layer running on the host computer presents the LittleDog robot to the users for behavioral control and learning. It formats data from both LittleDog's internal sensors and the motion capture system, and presents a coherent interface for use on the host computer. The API provided on the host computer allows users to perform a variety of control tasks on the robot in either robot joint coordinates or Cartesian coordinates relative to the robot's body, and provides reasonably synchronous access to data from both the robot and the motion capture system. Furthermore, during execution of a trial (when a users control software has control of the robot) all data received from, and sent to the robot is automatically collected into a data set that can be analyzed with a number of tools.

3.1 Working with the API

The user's control software runs in an environment that is controlled by the LittleDog API, and communicates with the LittleDog robot system via that same LittleDog C++ API. At the highest level, the API has functions to:

- initialize the robot and motion capture system
- read robot sensors
- write torques to robot actuators
- select servo function
- specify gains to the joint servos
- specify setpoints to the joint servos
- perform kinematic transformations
- read motion capture data
- collect data from both LittleDog and the motion capture system and save it to a local file

The primary interface between a user and the LittleDog API is through the `LittleDog` class. A control program must be subclassed from `LittleDog` and must implement an `updateControl()` method. Once an object of a `LittleDog` subclass is created, there is a specific progression of states the system must proceed through to run trials. Developers should refer to the electronic reference manual included with the software distribution for a detailed description of the methods provided by this base class (`$LITTLEDOG/doc/html/index.html`).

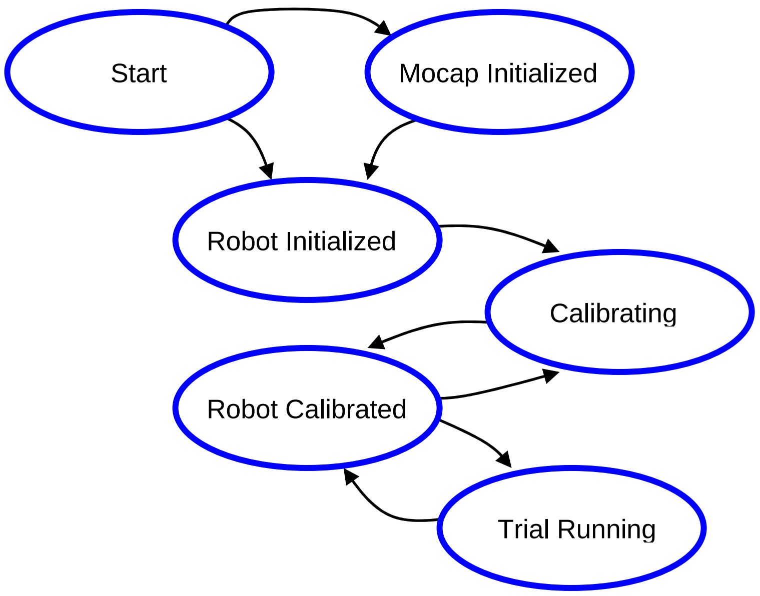


Figure 15: LittleDog API state diagram

A typical session would proceed as follows:

- user creates `MyLittleDog` object
- user (optionally) calls `initializeMocap()` to setup host/vicon communications
- user calls `initializeRobot()` to set up host/robot communications
- user calls `calibrate()`
- user calls `runTrial()`
    - LittleDog calls virtual `initControl()` once
    - LittleDog makes multiple calls to virtual `updateControl()`
        - user calls `stopRobot()` from within `updateControl()` or in response to button click in UI thread
    - LittleDog calls virtual `uninitControl()` once
- user destroys `MyLittleDog` object

The call to `initializeMocap()` is optional and only needs to be called if motion capture system will be used during the trial. If a trial has been started without initializing the motion capture system, the trial must be terminated and the host/robot communications must be closed prior to calling `initializeMocap()`. This progression of states is shown in Figure 15.

Users should refer to the electronic reference manual provided with the LittleDog software development kit for the specifics of using the LittleDog API. The information provided in this section only provides an overview of the basic functions along with a brief explanation of the basic example program discussed in Section.1.4.2 .

3.2 Basic API functions

The following methods of the LittleDog class allow a user program to control the overall operation of the robot:

`State getState (void)`

Queries the system for current state of the API. .

`const char * getPerformerName (void)`

Queries the system for the performer name, as derived from bdi_rt.cfg.

`const char * getErrorCodeString (LD_ERROR_CODE error_code)`

Convert an `LD_ERROR_CODE` to a string.

`LD_ERROR_CODE abortConnection (void)`

Abort all communication with the robot. Calling this function stops the robot and terminates all communication with both the robot and the motion capture system. The system is returned to the `UNINITIALIZED` state

`LD_ERROR_CODE initializeMocap (void)`

Initiate communication with the mocap system. Once initialized the motion capture system will be used with all future trials.

`LD_ERROR_CODE initializeRobot (void)`

Initiate the communication with the robot.

`LD_ERROR_CODE calibrate (bool force=false)`

Initiate the robot calibration procedure. Calibrate will cause the robot to move, and will not return until the calibration procedure has succeeded or detected a failure. If the process was successful it will be possible to now call `runTrial()`, however, if the calibration process failed it must be attempted again. Unless the optional `force` argument is set to true, `calibrate` will return immediately if the robot has already been calibrated. If `force` is true, then calibrate will force a recalibration of the robot.

`LD_ERROR_CODE runTrial (void)`

This function begins a robot trial run during which user control functions will be called and data will be collected. The function will block until the trial is either stopped by calling `stopRobot(),` or aborted by calling `abortConnection()`, as such it is recommended that `runTrial()` be called from a thread separate from any UI thread, so that the trial can be stopped or aborted if necessary. Trials can be stopped asynchronously by calling `stopRobot()` or `abortConnection()`. Whenever `runTrial()` returns a new data file will be produced in the current directory that contains all the information collected during the previous trial the file includes a date stamp and performer identification in the name to uniquely indicate when the trial was performed.

Multiple trials can be run during a session, that is to say `runTrial()` can be called multiple times before the LittleDog object is destroyed.

`LD_ERROR_CODE stopRobot (void)`

Stop current operation being performed on robot. This is valid either when the robot is in the process of calibrating or if a trial is currently being run on the robot. Calling `stopRobot()` either from within a users `updateControl()` method, or from some other thread is the preferred way to stop robot operation.

There are three virtual methods defined in the LittleDog base class. A derived class that intends to control the robot must implement the `updateControl()` method, and may optionally implement both `initControl()` and `uninitControl()`. When a trial is executing (in response to `runTrial()` having been called) a user's `updateControl()` method will be invoked approximately every 10 msec in response to the host computer receiving state information from the robot. The users implementation of `updateControl()` has complete access to this information as well as information from the motion capture system through the variety of get/set methods provided by the LittleDog base class, and may make use of all of those functions to implement behavior on the robot. Programmers are advised to be sensitive to the amount of time it takes for them to return from a call to `updateControl()`, as if it takes longer than 10 msec to complete they will miss data from the robot (although the LittleDog system will log the data, and record that the data was missed by the program).

Programmers should consult the electronic LittleDog reference manual, located in `$LITTLEDOG/doc/html/index.html`, for a complete list of methods provided by the base class. In addition to the functions provided by the LittleDog class, there are a number of utility classes included with the development kit. Of particular note are the functions provided in `bduLog.h` for interacting with the Boston Dynamics message logging system (used to send messages to the text display at the bottom of the window used in the example GUI applications), and the `LittleDogDataSetFileReader/Writer` which provide access to the data sets produced by the LittleDog system and allow programmers to log internal information from their software in the same format as the LittleDog system uses for the automatically generated data sets..

3.3 The *BASIC* Example

To better understand how these functions interact with one another and the LittleDog robot we revisit the basic example that was used to first operate the robot in Section 1.4.2. The `main()` defined by this program constructs a `MyLittleDog` object that is derived from the LittleDog class provided with the development kit. It then parses the one argument it takes (-f causes it to force recalibration), and proceeds to initialize the motion capture system, initialize the robot, calibrate the robot, and initiate a trial. The listing for this procedure follows:

```
static MyLittleDog * dog = NULL;
int main( int argc, char * argv[] )
{
   signal( SIGINT, signal_handler );
   MyLittleDog l;
   dog = &l;

   bool forced = false;
   // pass in "-f" if you want to force calibration always
   if ( (argc == 2) && (!strcmp(argv[1],"-f")) )
      forced = true;

   LD_ERROR_CODE result;

   printf( "Connecting to Mocap ...\n" );
   result = l.initializeMocap();
   handle_result("initializeMocap", result, false);

   printf( "Connecting to robot ...\n" );
   result = l.initializeRobot();
   handle_result("initializeRobot", result, true);

   printf( "%s calibration of robot ...\n", forced ? "Forced " : "Non-forced" );
   result = l.calibrate(forced);
   handle_result("calibrate", result, true);

   printf( "Starting user control ...\n" );
   result = l.runTrial();
   handle_result("runTrial", result, true);

   printf( "Exiting main.\n" );
   fflush( stdout );
   return 0;
}
```

The heart of the control for this example is contained in the `updateControl()` method of the derived `MyLittleDog` object.

```
class MyLittleDog : public LittleDog
{
public:
   bool first_time;
   virtual bool initControl( void )
   {
      printf( "initControl(): called\n");
      first_time = true;
      return true; // tells the caller that we are ok
   }
   virtual void updateControl( void )
   {
      LD_ERROR_CODE result;
      // one-time only
      if ( first_time ) {
         first_time = false;
         // Initialize the PD servos the first_time time through
         printf( "updateControl(): Initializing all joints to use PD servos\n" );
         float k[] = { 30.0, 30.0, 30.0 };
         float b[] = { 0.6, 0.6, 0.6 };
         for ( int l=0; l<NUM_LEGS; ++l )
            for ( int j=0; j<NUM_JOINTS; ++j ) {
               result = setJointServoType( (LegIndex)l, (JointIndex)j, SERVO_PD );
               handle_result("setJointServoType", result, true);
               result = setJointPDServoGains((LegIndex)l,(JointIndex)j, k[j], b[j]);
               handle_result("setJointPDServoGains", result, true);
            }
      }
      // Do control for this particular timestamp
      float t;
      getDataTimestamp( &t );

      float q_d1  = 0.2 * sin( 3.14 * t ) + 0.2;
      float q_d2 =  0.2 * cos( 4.14 * t ) + 0.2;
      float q_d3  = 0.2 * sin( 5.14 * t ) + 0.2;

      for ( int l=0; l<NUM_LEGS; ++l ) {
         float x_d = 2 * q_d1;
         if ( (l == 1) || (l == 3) ) x_d *= -1;
         result = setJointPDServoSetPoints( (LegIndex)l, HIP_RX, x_d );
         handle_result("setJointPDServoPoints", result, false);
         setJointPDServoSetPoints( (LegIndex)l, HIP_RY, -4 * q_d2 + (1.57/2.0) );
         handle_result("setJointPDServoPoints", result, false);
         setJointPDServoSetPoints( (LegIndex)l, KNEE_RY, 2 * q_d3 );
         handle_result("setJointPDServoPoints", result, false);
      }
   }
   virtual void uninitControl( void ) {
      printf( "uninitControl(): Cleaning up ...\n" );
   }
};
```

This derived class makes use of the `initControl()` method to initialize its internal state (to indicate when the first call to `updateControl()` is made). It is important to note that the `initControl()` method is not allowed to control settings or parameters on the robot, thus it is the responsibility of the `updateControl()` method to make these settings the first time it is called. As you can see `updateControl()` deals with these "first time" issues, then goes on to issue commands to the robot. In this example all the joints are set to perform PD control with moderate gain settings, and the PD servos associated with each joint are sent sinusoidal position commands as time progresses.

Finally, the example program includes a signal handler and error handler that are shown below. The signal handler is used to catch `SIGINT` generated by a user pressing control-C and to call the `stopRobot()` method of the `MyLittleDog` object which causes motion to stop and `runTrial()` to return. The error handler simply prints the appropriate text error message associated with a returned error code and exits or returns.

```
static void signal_handler( int )
{
   printf( "Stopping control process\n" );
   fflush( stdout );

   // this will stop initializeRobot(), calibrate(), or runTrial()
   // in their tracks (not immediately, but quickly).  this won't
   // affect initializeMocap(), but abortConnection() would.
   // stopRobot() and abortConnection() can be called nearly
   // anywhere and return immediately.  Do not delete 'dog' right
   // here.  One could delete it after initializeRobot(), calibrate(),
   // or runTrial() returns, however.

   LD_ERROR_CODE result;
   result = dog->stopRobot();
   handle_result("stopRobot", result, true);
}

static void handle_result( const char* func, LD_ERROR_CODE result, bool
exit_on_error)
{
   // if an error occurs, print it out and (optionally) exit

   if ( result == LD_OKAY )
      return;

   printf( "An error occurred calling '%s()'.\n\tError was: '%s'\n",
      func, dog->getErrorCodeString(result));

   if ( exit_on_error )
      exit(EXIT_FAILURE);
}
```

3.4 Building Your First LittleDog Behavior

A skeleton application for controlling LittleDog has been provided as part of the development kit. Under `$LITTLEDOG/examples` there is a directory named `minimum_ui_example`. This directory contains a system that uses the same interface as the walk example discussed in Section 1.4.3, but issues no commands to the robot. To facilitate testing and system control by the government testing group, all performers in the Learning Locomotion program will be required to use an interface that includes this basic functionality, so we strongly suggest that you begin developing all applications using this skeleton. Simple behaviors can quickly be developed by copying this entire directory, and editing the `MyLittleDog.cpp` file to issue the desired commands to the robot. Note that included with the example is a `README.txt` file which explains the key components of the system.

3.5 Contents of a LittleDog Data Set

Whenever a trial is terminated (typically by a call to `stopTrial()`), the LittleDog system will save a data set file (`.data`) containing a full record of all the information collected from the robot and motion capture system as well as commands sent to the robot during the previous trial. This data is logged at a rate of approximately 100Hz, and every data entry is time stamped both by the robot when the data is acquired and by the host when it is received from the robot.

Variable	**Description**
`host.timestamp`	Timestamp when data was received on Host
`host.status.skipped_control`	Count of control cycles missed
`robot.timestamp`	Timestamp when data was collected on robot
`robot.is_calibrated`	Flag indicating the robot is calibrated
`robot.is_frozen`	Flag indicating that low-level robot software has frozen robot motion
`robot.is_undervoltage`	Flag indicating the battery voltage is low
`robot.killswitch`	Flag indicating the kill switch is active (not present in the Learning Locomotion project)
`robot.watchdog`	Flag indicating the robot watchdog has timed out
`battery_voltage`	Battery voltage in Volts.
`battery_current`	Battery current in Amps
`imu.orientation.(rz,rx,ry)`	IMU orientation in Yaw-Roll-Pitch Euler coordinates
`imu.rates.(x,y,z)`	IMU angular rates

`imu.accel.(x,y,z)`	IMU acceleration vector in body coordinates
`cpu_temperature`	Robot CPU temperature
`prox_sensor`	Proximity reading in meters
`(fl,hl,fr,fl).force`	Leg force measurements in Newtons
leg.joint`.q`	Current joint angle (rad)
leg.joint`.q_d`	PD joint position command (rad)
leg.joint`.qd`	Current joint velocity (rad/sec)
leg.joint`.qd_d`	PD joint velocity command (rad/sec)
leg.joint`.tau_d`	Current commanded joint torque (N m)
leg.joint`.k`	Joint proportional gain (N m/rad)
leg.joint`.b`	Joint differential gain (N m sec/rad)
leg.joint`.ff_d`	Commanded feed-forward joint torque (n m)
leg.joint`.tau_saturated`	Flag indicating the joint is torque saturated
leg.joint`.at_limit`	Flag indicating the joint is at its limit
`mocap.num_markers`	Number of markers logged
`mocap.frame_number`	Current motion capture frame number
`mocap.body[`*i*`].position.(x,y,z)`	Reported location of body i
`mocap.body[`*i*`].orientation.(rz,rx,ry)`	Reported orientation of body i in Euler coordinates
`mocap.body[`*i*`].primary_angle`	Reported primary angle
`mocap.body[`*i*`].secondary_angle`	Reported secondary angle
`mocap.body[`*i*`].age`	Age (in frames) of body data
`mocap.marker[`*j*`].position.(x,y,z)`	Reported position of marker j
`mocap.marker[`*j*`].age`	Age (in frames) of marker data
`terrain[`*k*`].id`	ID for terrain board k
`terrain[`*k*`].position.(x,y,z)`	Position of terrain board k
`terrain[`*k*`].orientation.(rz,rx,ry)`	Orientation of terrain board k
`terrain[`*k*`].age`	Age (in frames) of terrain data

4 Working with BDIPlot

`BDIPlot` is a tool for viewing and analyzing numerical data sets generated by the Boston Dynamics' software delivered with LittleDog. `BDIPlot` is based on Matlab, but can run stand-alone. If more advanced mathematical analysis or plotting routines are needed than `BDIPlot` provides, LittleDog data can be imported directly into Matlab for customized analysis using the Boston Dynamics Matlab toolbox.

In `BDIPlot,` data can be displayed in two ways:

- as graphs of variables over time, and
- as graphs of one variable plotted against another (phase plots).

`BDIPlot` can plot variables from multiple datafiles at the same time so that data from multiple trials can be compared side-by-side.

`BDIPlot` supports data manipulation functions including: multiplication, division, integration, differentiation, scaling, and shifting functions.

4.1 Installing BDIPlot

To install `BDIPlot`, unzip the file `LittleDog_win32_tools.zip` (located in the `win32` directory of the LittleDog software distribution) into an installation directory. The rest of this document will assume that the installation directory is `C:\LittleDog`. If `BDIPlot` is installed in a different directory, substitute the true installation directory for `C:\LittleDog` in the rest of this document.

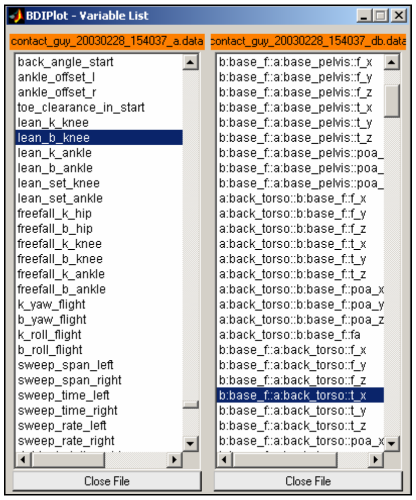
The directory `bdiplot` will contain the `BDIPlot` program.

The directory `matlab` will contain the Boston Dynamics Matlab toolbox, which is described in section 5.3.

4.2 Running BDIPlot

To start `BDIPlot`, go to the `C:\LittleDog\bdiplot` directory and run the file `BDIPlot_Launch.cmd`. This command script makes a number of settings to the environment before executing `bdiplot.exe`.

`BDIPlot` has three windows: a Variable List, a Data window, and a Command window. Note that the Variable List window does not appear until a data file is opened.

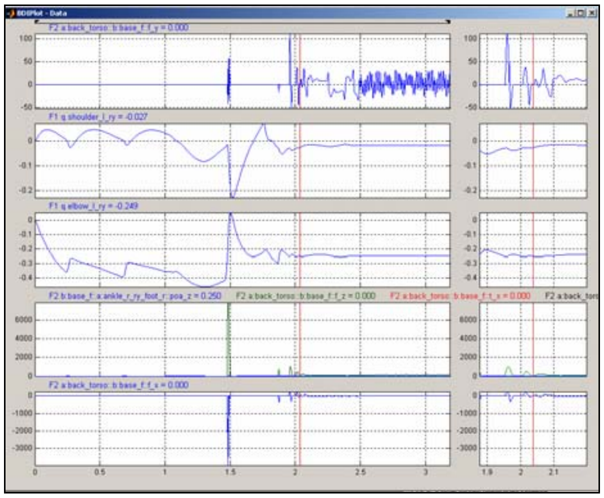


contact_guy_20030228_154037_a.data

```
back_angle_start
ankle_offset_l
ankle_offset_r
toe_clearance_in_start
lean_k_knee
lean_b_knee
lean_k_ankle
lean_b_ankle
lean_set_knee
lean_set_ankle
freefall_k_hip
freefall_b_hip
freefall_k_knee
freefall_b_knee
freefall_k_ankle
freefall_b_ankle
k_yaw_flight
b_yaw_flight
k_roll_flight
b_roll_flight
sweep_span_left
sweep_span_right
sweep_time_left
sweep_time_right
sweep_rate_left
sweep_rate_right
```

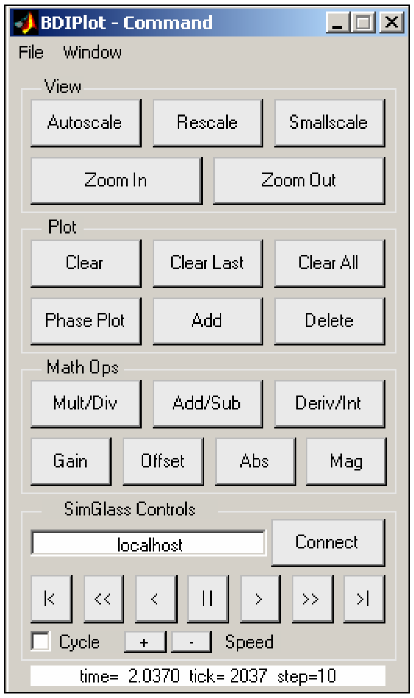
contact_guy_20030228_154037_db.data

```
b:base_f::a:base_pelvis::f_x
b:base_f::a:base_pelvis::f_y
b:base_f::a:base_pelvis::f_z
b:base_f::a:base_pelvis::t_x
b:base_f::a:base_pelvis::t_y
b:base_f::a:base_pelvis::t_z
b:base_f::a:base_pelvis::poa_
b:base_f::a:base_pelvis::poa_
b:base_f::a:base_pelvis::poa_
a:back_torso::b:base_f::f_x
a:back_torso::b:base_f::f_y
a:back_torso::b:base_f::f_z
a:back_torso::b:base_f::t_x
a:back_torso::b:base_f::t_y
a:back_torso::b:base_f::t_z
a:back_torso::b:base_f::poa_x
a:back_torso::b:base_f::poa_y
a:back_torso::b:base_f::poa_z
a:back_torso::b:base_f::fa
b:base_f::a:back_torso::f_x
b:base_f::a:back_torso::f_y
b:base_f::a:back_torso::f_z
b:base_f::a:back_torso::t_x
b:base_f::a:back_torso::t_y
b:base_f::a:back_torso::t_z
b:base_f::a:back_torso::poa_x
```


BDIPlot - Data
F2 a:back_torso::b:base_f::f_y = 0.000
F1 q shoulder_l_ry = -0.027
F1 q elbow_l_ry = -0.249
F2 b:base_f::a:ankle_r_ry_foot_r::poa_z = 0.250
F2 a:back_torso::b:base_f::f_z = 0.000
F2 a:back_torso::b:base_f::f_x = 0.000

Variable List Window

Data Window



Command Window

Figure 16: BDIPlot Windows

4.2.1 Variable List Window

The Variable List window contains the list of recorded variables for each data file that is loaded. This window does not appear until a data file is opened. A data file can be opened by using the Command window's `File|Open` menu item (see the Command Window section below).

The orange bar at the top of the Variable List window displays the name(s) of the open data file(s).

Along the bottom of the Variable List window are three buttons:

Button	**Function**
`Close File`	Closes the data file associated with the Variable List.
`Load Plot`	Loads a saved list of plots.
`Save Plot`	Saves the current plots, including scales and number of plots. (This can save plots from only a single data file at a time.).

To plot a variable in the Data window:

1. Click a variable in the Variable List.
2. Click a plot row in the Data window.

There is currently no way to delete variables.

4.2.2 Data Window

The Data window displays plots of variables against time. Time is on the horizontal axis and variable values are on the vertical axis. By default, there are five plots in the Data window. The left column of plots shows data for the entire simulation. The right column shows close ups of portions of the same data. Plots can be vertically scaled, depending on the level of detail required. The left and right plots in a single row are always shown with the same scale. Several variables may be plotted concurrently in the same row.

The vertical red line shown in all the plots, referred to as the Data Cursor, indicates the current location in the data. The current value of all traces at the cursor position is always displayed on the top left of each plot.

Note on plotting variables from multiple files of different lengths: If more than one data file at a time is open and one of the data files is shorter than the other, the last value of every variable in the shorter file is used for any comparisons with later data in the longer file.

4.2.3 Command Window

The Command Window displays controls that are used to alter the view of data plotted in the Data window. A menu at the top of the Command Window provides access to file and window functions.

Functions are grouped into three categories: Plot, View, and Math. Several plot features can be controlled by using the buttons in the View and Plot sections in the Command Window. This includes adding and removing plots, clearing plots, scaling plots, and zooming in and out. Time and tick (index number) corresponding to the cursor position are shown below the control buttons.

The menu contains the following options:

`File Menu Options`

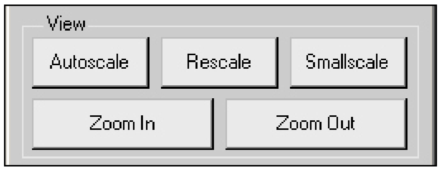
Option	Hotkey	Function
`Open`	Ctrl-O	Displays a Select Data File dialog.
`Connect`	Ctrl-C	(Not currently supported.)
`Print`	Ctrl-P	Prints the plots in the data window. The `Options|Print to Image` menu item (see below) determines whether printing happens to a printer or to a file.
`Exit`	Ctrl-Q	Exits `BDIPlot`.

`Window Menu Options`

Option	Hotkey	Function
`Grid`	Ctrl-G	Toggles grid on and off.
`Print to Image`	Ctrl-I	Toggles whether the `File|Print` command sends output to a printer or to a JPEG file.
`Show Video Window`	Ctrl-V	(Not currently supported.)

4.2.3.1 Using View Controls

The view controls modify the appearance of plots in the data window.



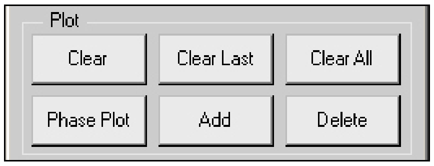
Click Autoscale, Rescale, and Smallscale to modify the vertical axis scaling of a selected plot. Click Zoom In and Zoom Out to modify the horizontal axis scale in the right hand plot on each row.

View Controls

Control	**Function**
`Autoscale`	Click `Autoscale` and click a plot to set vertical axis scaling to include maximum and minimum points of all data traces.
`Rescale`	Click `Rescale` and click a plot to manually set the vertical axis scale of a plot. Enter desired maximum and minimum vertical axis values in the dialog.
`Smallscale`	Click `Smallscale` and click a plot to set vertical axis scaling to include the maximum and minimum points of all data traces plotted in the right column plot.
`Zoom In`	Click `Zoom In` to decrease the timespan displayed in the right hand column of the Data window.
`Zoom Out`	Click `Zoom Out` to increase the timespan in the right hand column of the Data window.

4.2.3.2 Using Plot Controls

Plot controls perform general purpose plotting functions.



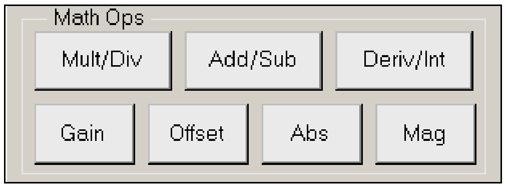
`Clear`, `Clear Last`, and `Clear All` remove data traces from plots. `Phase Plot` plots two variables against one another. `Add` and `Delete` increases or decreases the number of plots in the Data window.

Plot Controls

Control	Function
`Clear`	Click `Clear` and click a plot to remove all variables.
`Clear Last`	Click `Clear Last` and click a plot from which to remove the most-recently-added plot variable. If only one variable is being displayed, Clear Last clears the plot.
`Clear All`	Removes all variables from all plots.
`Phase Plot`	Click `Phase Plot` and then click two variables. A figure is displayed that plots the variables with respect to each other. The first variable is plotted along the horizontal axis and the second is plotted along the vertical axis. To set the start and end points of both axes, slide the control at the top of the Data window.
`Add` `Delete`	Click `Add` or `Delete` to increase or decrease the number of plots displayed. Select whether you want the plot to be added to (or removed from) the top or the bottom of the Data window.

4.2.3.3 Using Math Controls

To perform data analysis data sometimes needs to be manipulated mathematically. The Math controls provide this functionality.



Mathematical operations can be divided into two groups: operations between two variables and operations on a single variable. `Mult/Div` and `Add/Sub` both take two variables as input. These functions multiply, divide, add, and subtract one variable from another. `Deriv/Int`, `Gain`, `Offset`, `Avg`, and `Abs`, on the other hand, require only a single input variable.

NOTE: When new variables are created, double-check variable name before pressing Enter. There is currently no way to delete a variable once it has been created.

Math Controls

Control	**Function**
`Mult/Div`	Click `Mult/Div` and then click two variables. Select multiply or divide in the dialog. The function is performed and a new variable is appended to the end of the list. The default variable name is created by concatenating the two variable names separated by * or /.
`Add/Sub`	Click `Add/Sub` and then click two variables. Select add or subtract in the dialog. The function is performed and a new variable is appended to the end of the list. The default variable name is created by concatenating the two variable names separated by a + or -.
`Deriv/Int`	Click `Deriv/Int` and click a variable. Select differentiate or integrate in the dialog. If Integrate is selected, minimum and maximum times are asked for over which to integrate the variable. By default these bounds cover the entire duration of the data. Once time bounds have been entered a new variable containing the integral of the chosen variable is added to the end of the variable list.
`Gain`	Click `Gain` and click a variable. Enter a scale factor. A new variable is created.
`Offset`	Click `Offset` and click a variable to shift a variable up or down. Enter an offset. A new variable is appended to the end of the variable list
`Avg`	Click Avg and click a variable. A new variable containing the average of all values of the clicked variable is appended to the end of the variable list.
`Abs`	Click `Abs` and click a variable to calculate the absolute value of a variable. A new variable is appended to the end of the variable list.
`Mag`	Click `Mag` and click either two (2D) or three (3D) variables to calculate the magnitude of the variables. The magnitude is the square root of the sum of the squares of the variables.

4.3 The Boston Dynamics Matlab Toolbox

If there are advanced data analysis or presentation needs that go beyond the capabilities of `BDIPlot`, there is a Boston Dynamics toolbox for use in Matlab. This toolbox allows access to data from Boston Dynamics data files. Note that Matlab and a Matlab license are required to use this toolbox.

4.3.1 Installing and Running the Matlab Toolbox

Install the LittleDog win32 tools as described under section 5.1. This will also install the Boston Dynamics Matlab toolbox in the `C:\LittleDog\matlab` directory.

(If LittleDog is installed in a different directory, substitute the directory for `C:\LittleDog`.)

To access the toolbox from Matlab, add the `C:\LittleDog\matlab` directory to the Matlab path. This may be done from the main Matlab window by clicking `File | Set Path` or at the Matlab command prompt by entering:

```
path(path, ‘C:\LittleDog\matlab’)
```

It should now be possible to test the Boston Dynamics Matlab toolbox by typing:

```
sample
```

at the Matlab prompt. This will run a small program that will ask for a data file and plot a random variable from the file. Open the following file:

```
C:\LittleDog\matlab\example_data\BostonDynamics_LittleDog.data
```

The sample script (listed in Section 4.3.4 below) serves as a good example of how to use the toolbox.

As with all Matlab functions, type help scriptname to get detailed instructions on how to use the toolboxes' functions. All Boston Dynamics functions are prefixed by the acronym BDI. A good starting point is to look at the toolboxes' two key functions BDI_read_file and BDI_get_var.

4.3.2 Manipulating Data in Matlab

When `BDIPlot` opens a data file, it creates a global Matlab data structure for that data file. Key fields in this data structure are described below.

Boston Dynamics Matlab Data Structure Elements

Field	**Description**
`BDI.fname`	Name of open data file
`BDI.model`	Name of model associated with the data
`BDI.t`	Vector of time values synchronized with data readings
`BDI.data`	Data array containing runtime data. The size of the array is `BDI.rows` x `BDI.cols`
`BDI.rows`	Number of readings for each variable
`BDI.cols`	Number of variables in data file
`BDI.vars`	Names and units of all variables stored in the `BDI.data` array.

4.3.3 Accessing Variable Names And Values

Variable names and units are stored in the BDI.data array.

```
BDI.vars = [varname1 varname2 … … varnameN]
```

The name of the *n*th variable can be obtained by entering:

```
BDI.vars(n).name
```

Data corresponding to this variable can be assessed by entering:

```
BDI.data(:, n)
```

4.3.4 Sample Matlab Script

Below is a sample Matlab script (see sample.m) that asks the user for a valid data file, prints key file statistics, and plots a random variable from the file. This example illustrates usage of the Boston Dynamics Matlab toolbox.

```
[ file, pathname] = uigetfile('*.data','Select Data File');
if ( file == 0),
  return
end

%% Read the file
S = BDI_read_file( strcat( pathname, file ));

%% Print statistics
disp( sprintf( '\nFile statistics' ) );
disp( sprintf( '         name: %s', file ) );
disp( sprintf( '    modelname: %s', S.model ) );
disp( sprintf( '    # of vars: %d', S.n_vars ) );
disp( sprintf( ' # of samples: %d', S.rows ) );
disp( sprintf( '   average dt: %d\n', S.dt ) );

%% Pick a random variable
inx  = min( max( 1, floor( rand * S.n_vars ) ), S.n_vars );
name = horzcat( S.vars(inx).name );

%% If we don't know the index, we can search by name
[ i, data ] = BDI_get_var( S, name );

%% Otherwise, we can just grab the data by index
plot( S.t, S.data(:,inx) );

%% Add some labels
xlabel( 't' );
ylabel( name );
title( sprintf( 't vs. %s', name ) );
```

5 Maintaining Your LittleDog

The key to keeping your LittleDog happy is to pay attention to wear and tear. The sooner you catch problems or wearing components the more likely it can be repaired with minimum down time. The following section covers some of the major maintenance items. Recognizing when your LittleDog's behavior changes is the best way to catch problems. Section 5.2 covers the basic things to watch for, such as sloppy or stiff joints, loose fasteners etc.

5.1 How to change batteries

There are two battery compartments in each LittleDog robot. They are located on the right and left side of the body as shown in Figure 17. The door on the battery compartment can be removed by loosening the thumb screw and tilting the door up, starting at the bottom of the robot. The following page describes the sequence used to exchange the batteries.

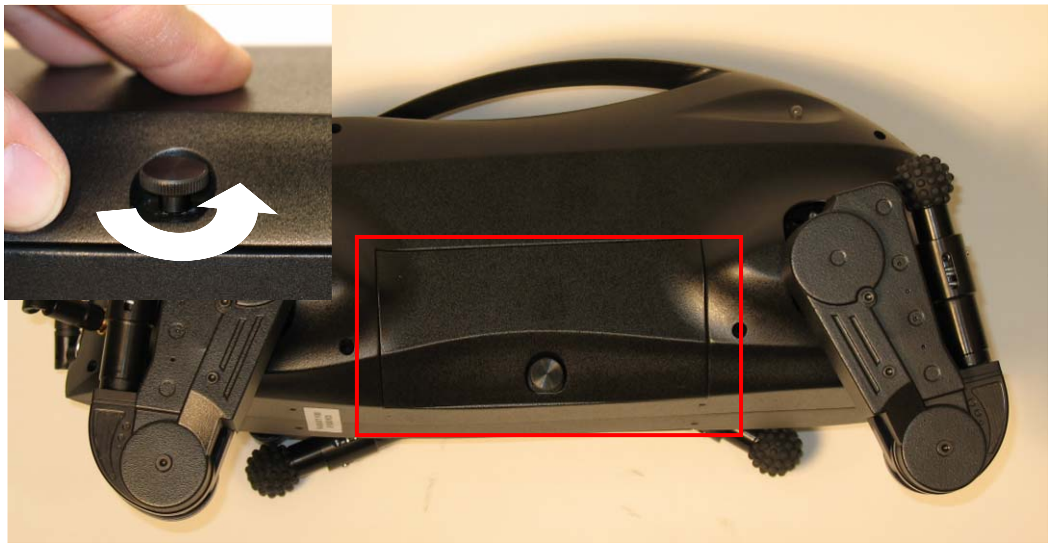


Figure 17: Battery Compartments

<u>CAUTION:</u> Do not over tighten the thumb screw as this may damage the insert in the plastic shell and prevent installing the battery door in the future.

Removing the Battery:

1. Once the battery compartment is open you can lift the battery out of the compartment.
2. To disconnect press on the connectors release tab and the white connectors will separate, as shown in Figure 18 (A).

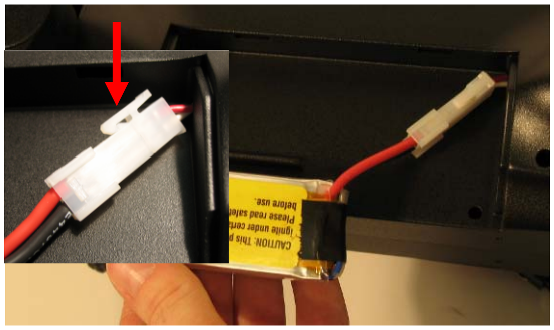
WARNING: Do not pull on the connectors without releasing the latch.

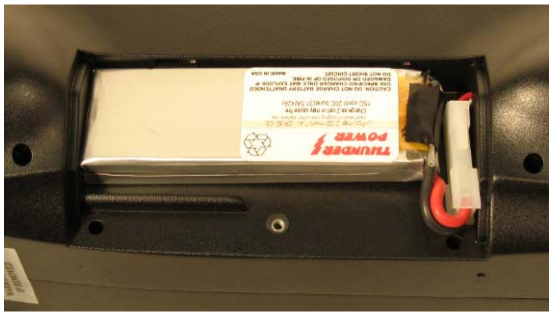
Installing the Battery:

1 Connect the two mating connectors before installing battery

2 Place the battery in the compartment, insuring the leads coming from the battery pack are pointing down, as shown in Figure 18 (B).

3 Insert the battery door, ensure the tabs engage at the top of the compartment. Tilt the battery door down as shown in Figure 18 (C), insuring the red and black battery wires do not get pinched under the edge of the door.


(A)


(B)

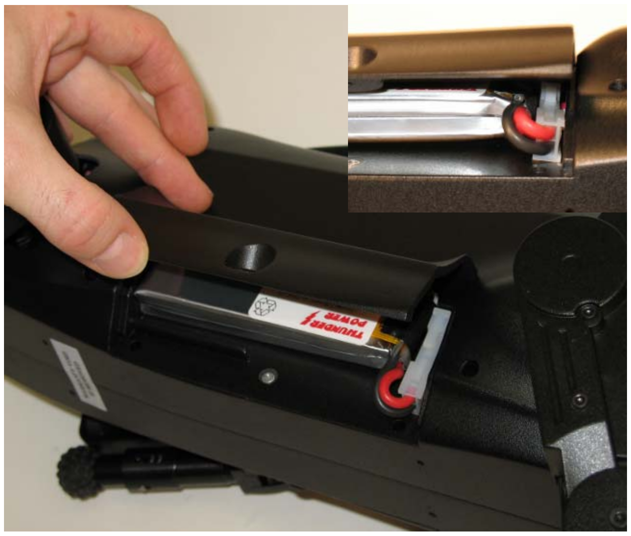

(C)

Figure 18: Battery Installation

5.2 How to inspect your robot

There are several items that should be checked on a regular basis to ensure the healthy operation of your LittleDog:

- Check to ensure the CF disk is properly seated.
- Check the MoCap markers for damage; see Section 5.4 for more information.
- Check the battery connectors and wires on both the robot and battery packs for wear.
- Check the condition of the tether to ensure there is no wear that could cause a short and damage the system.
- Check the measurement range of the leg force sensors.
- Check the wear of the rubber feet. These can be replaced as required by simply cutting off the old ones and gluing new ones on. More feet can be obtained from Boston Dynamics.
- Inspect the antennas for wear. They can get caught in legs or be damaged during a fall.

5.3 Power Source

The LittleDog platform can be powered either by a pair of internal batteries or an external power supply through a 16 foot tether. The following two sections describe how to use the batteries or external power supply.

5.3.1 Internal Batteries

The internal batteries are 2100mAh LiPol batteries that can be charged with the chargers provided (or any correctly configured LiPol capable battery charger). Access to the batteries is in the midsection of the platform on the left and right sides, as shown in Section 5.1 .

<u>WARNING:</u> Read charging instructions provided with TP-425 charger, and on the following two pages, carefully prior to charging the batteries.

Please set up your charger to charge 2 cells. This is done by pressing the [CELL] button until an LED is illuminated beside the [-2CL] mark on the charger. You also need to set the current limit during recharge. This is done by pressing the [CURRENT] button until an LED is illuminated beside the [2.0A-] mark on the charger. ***This 2.0A charge rate is the highest recommended for these batteries***.

It is very important to monitor the health of your battery packs. From time to time, prior to charging, measure the voltage of each battery pack. If the value is less than 6.25V under load, you may have problems charging the pack. ***You should always charge and discharge your batteries in pairs as marked.***

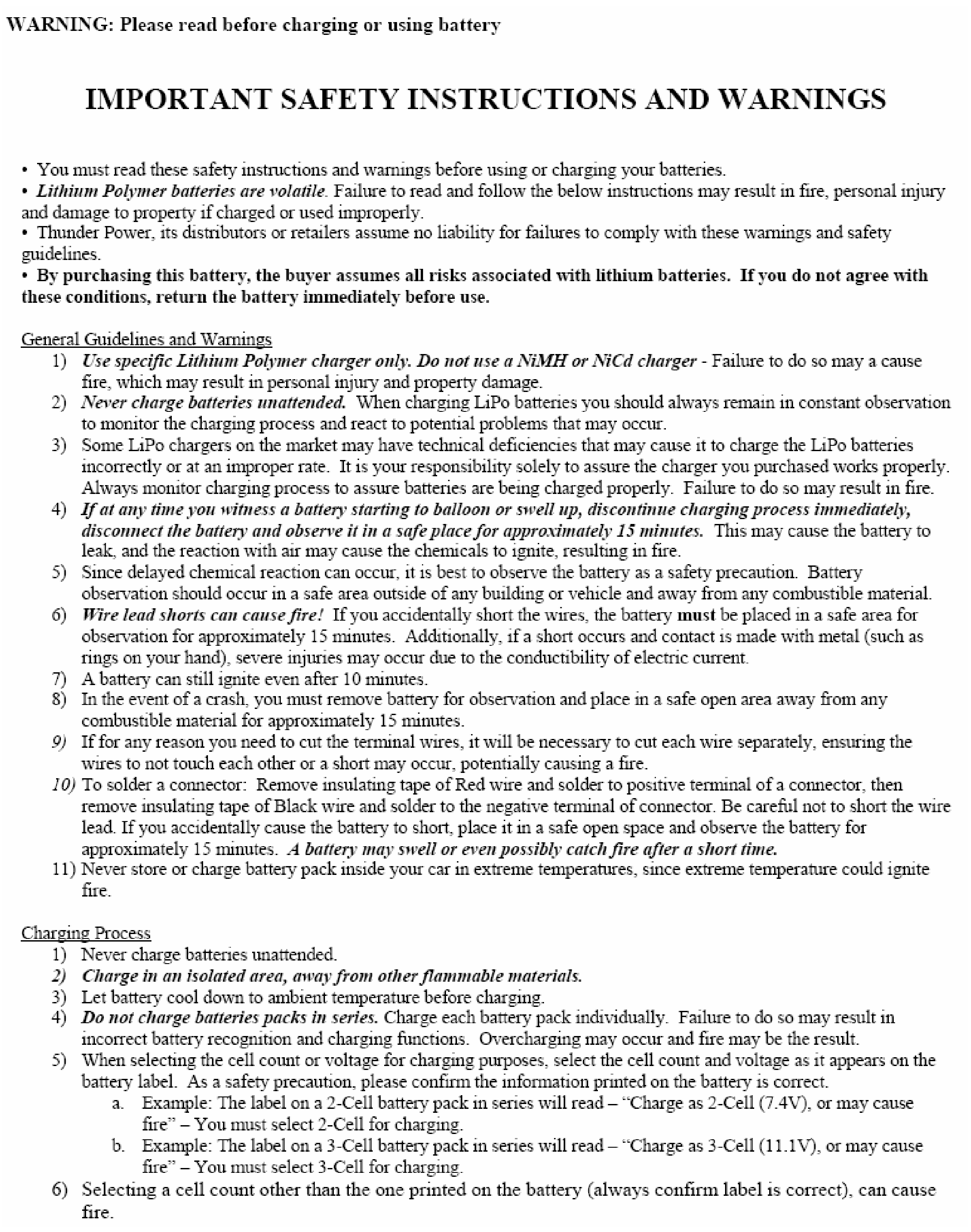
CAUTION: When one of these LiPol batteries fails it will swell up. If a battery pack feels or looks swollen DO NOT recharge it. Recharging a swollen battery pack can catch fire. Contact manufacturer for proper disposal instructions:

Thunder Power Batteries
4720 West University Ave., Las Vegas, NV 89103
Phone: (702) 228-8883
Fax: (702) 228-8885
www.thunderpower-batteries.com

5.3.1.1 Battery Safety Instructions and Warnings

The following two pages contain a reference copy of the original safety and warning sheet authored and provided by Thunder Power Batteries. It is identical to the one provided with your original shipment. This warning sheet can also be found at:

http://www.thunderpower-batteries.com/images/THPSafetyWarnings.pdf



WARNING: Please read before charging or using battery

IMPORTANT SAFETY INSTRUCTIONS AND WARNINGS

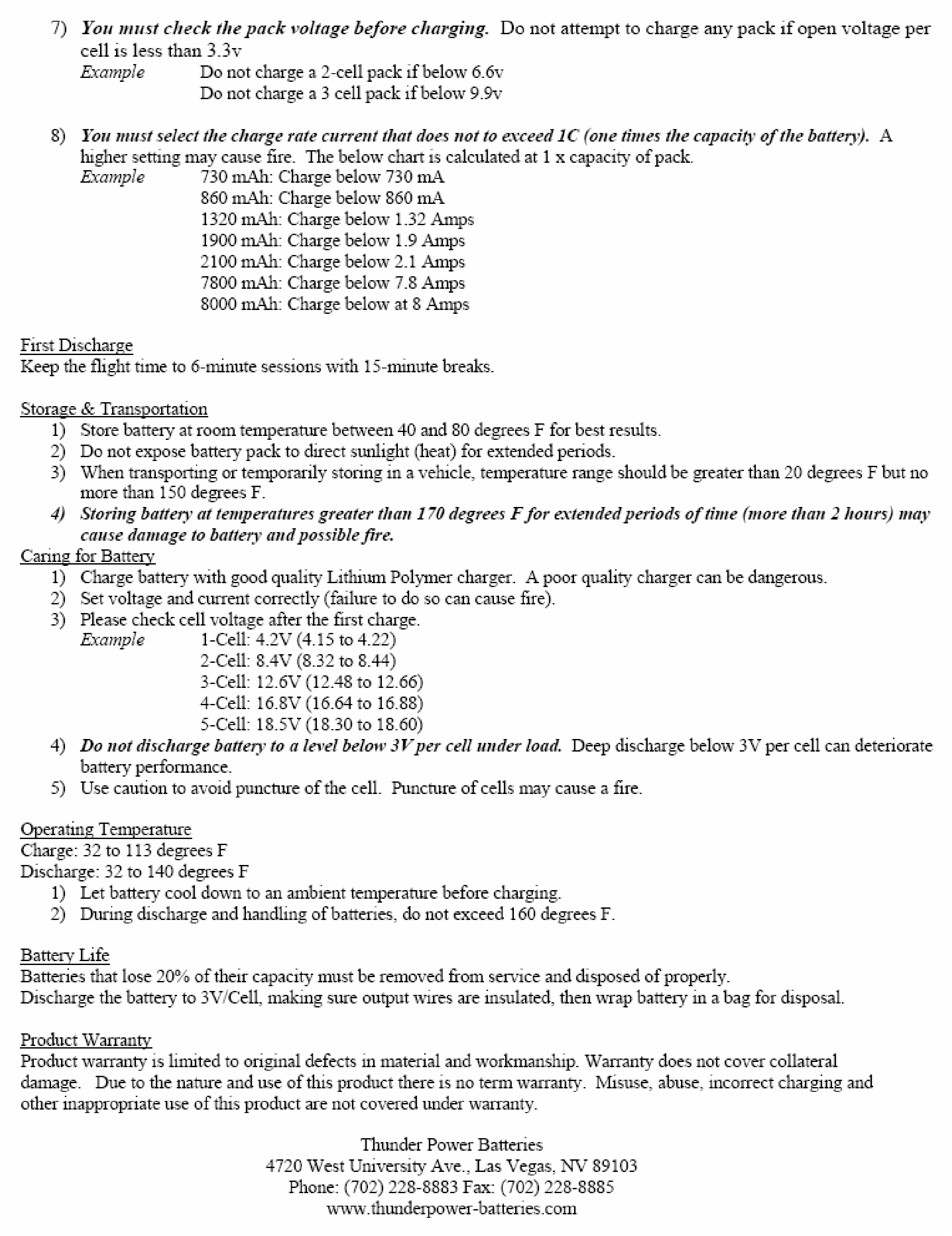
• You must read these safety instructions and warnings before using or charging your batteries.
• ***Lithium Polymer batteries are volatile.*** Failure to read and follow the below instructions may result in fire, personal injury and damage to property if charged or used improperly.
• Thunder Power, its distributors or retailers assume no liability for failures to comply with these warnings and safety guidelines.
• **By purchasing this battery, the buyer assumes all risks associated with lithium batteries. If you do not agree with these conditions, return the battery immediately before use.**

General Guidelines and Warnings

1) ***Use specific Lithium Polymer charger only. Do not use a NiMH or NiCd charger*** - Failure to do so may a cause fire, which may result in personal injury and property damage.
2) ***Never charge batteries unattended.*** When charging LiPo batteries you should always remain in constant observation to monitor the charging process and react to potential problems that may occur.
3) Some LiPo chargers on the market may have technical deficiencies that may cause it to charge the LiPo batteries incorrectly or at an improper rate. It is your responsibility solely to assure the charger you purchased works properly. Always monitor charging process to assure batteries are being charged properly. Failure to do so may result in fire.
4) ***If at any time you witness a battery starting to balloon or swell up, discontinue charging process immediately, disconnect the battery and observe it in a safe place for approximately 15 minutes.*** This may cause the battery to leak, and the reaction with air may cause the chemicals to ignite, resulting in fire.
5) Since delayed chemical reaction can occur, it is best to observe the battery as a safety precaution. Battery observation should occur in a safe area outside of any building or vehicle and away from any combustible material.
6) ***Wire lead shorts can cause fire!*** If you accidentally short the wires, the battery **must** be placed in a safe area for observation for approximately 15 minutes. Additionally, if a short occurs and contact is made with metal (such as rings on your hand), severe injuries may occur due to the conductibility of electric current.
7) A battery can still ignite even after 10 minutes.
8) In the event of a crash, you must remove battery for observation and place in a safe open area away from any combustible material for approximately 15 minutes.
9) If for any reason you need to cut the terminal wires, it will be necessary to cut each wire separately, ensuring the wires to not touch each other or a short may occur, potentially causing a fire.
10) To solder a connector: Remove insulating tape of Red wire and solder to positive terminal of a connector, then remove insulating tape of Black wire and solder to the negative terminal of connector. Be careful not to short the wire lead. If you accidentally cause the battery to short, place it in a safe open space and observe the battery for approximately 15 minutes. ***A battery may swell or even possibly catch fire after a short time.***
11) Never store or charge battery pack inside your car in extreme temperatures, since extreme temperature could ignite fire.

Charging Process

1) Never charge batteries unattended.
2) ***Charge in an isolated area, away from other flammable materials.***
3) Let battery cool down to ambient temperature before charging.
4) ***Do not charge batteries packs in series.*** Charge each battery pack individually. Failure to do so may result in incorrect battery recognition and charging functions. Overcharging may occur and fire may be the result.
5) When selecting the cell count or voltage for charging purposes, select the cell count and voltage as it appears on the battery label. As a safety precaution, please confirm the information printed on the battery is correct.
    a. Example: The label on a 2-Cell battery pack in series will read – "Charge as 2-Cell (7.4V), or may cause fire" – You must select 2-Cell for charging.
    b. Example: The label on a 3-Cell battery pack in series will read – "Charge as 3-Cell (11.1V), or may cause fire" – You must select 3-Cell for charging.
6) Selecting a cell count other than the one printed on the battery (always confirm label is correct), can cause fire.



7) ***You must check the pack voltage before charging.*** Do not attempt to charge any pack if open voltage per cell is less than 3.3v
   Example Do not charge a 2-cell pack if below 6.6v
   Do not charge a 3 cell pack if below 9.9v

8) ***You must select the charge rate current that does not to exceed 1C (one times the capacity of the battery).*** A higher setting may cause fire. The below chart is calculated at 1 x capacity of pack.
   Example 730 mAh: Charge below 730 mA
   860 mAh: Charge below 860 mA
   1320 mAh: Charge below 1.32 Amps
   1900 mAh: Charge below 1.9 Amps
   2100 mAh: Charge below 2.1 Amps
   7800 mAh: Charge below 7.8 Amps
   8000 mAh: Charge below at 8 Amps

First Discharge

Keep the flight time to 6-minute sessions with 15-minute breaks.

Storage & Transportation

1) Store battery at room temperature between 40 and 80 degrees F for best results.
2) Do not expose battery pack to direct sunlight (heat) for extended periods.
3) When transporting or temporarily storing in a vehicle, temperature range should be greater than 20 degrees F but no more than 150 degrees F.
4) ***Storing battery at temperatures greater than 170 degrees F for extended periods of time (more than 2 hours) may cause damage to battery and possible fire.***

Caring for Battery

1) Charge battery with good quality Lithium Polymer charger. A poor quality charger can be dangerous.
2) Set voltage and current correctly (failure to do so can cause fire).
3) Please check cell voltage after the first charge.
   Example 1-Cell: 4.2V (4.15 to 4.22)
   2-Cell: 8.4V (8.32 to 8.44)
   3-Cell: 12.6V (12.48 to 12.66)
   4-Cell: 16.8V (16.64 to 16.88)
   5-Cell: 18.5V (18.30 to 18.60)
4) ***Do not discharge battery to a level below 3V per cell under load.*** Deep discharge below 3V per cell can deteriorate battery performance.
5) Use caution to avoid puncture of the cell. Puncture of cells may cause a fire.

Operating Temperature

Charge: 32 to 113 degrees F
Discharge: 32 to 140 degrees F

1) Let battery cool down to an ambient temperature before charging.
2) During discharge and handling of batteries, do not exceed 160 degrees F.

Battery Life

Batteries that lose 20% of their capacity must be removed from service and disposed of properly.
Discharge the battery to 3V/Cell, making sure output wires are insulated, then wrap battery in a bag for disposal.

Product Warranty

Product warranty is limited to original defects in material and workmanship. Warranty does not cover collateral damage. Due to the nature and use of this product there is no term warranty. Misuse, abuse, incorrect charging and other inappropriate use of this product are not covered under warranty.

Thunder Power Batteries
4720 West University Ave., Las Vegas, NV 89103
Phone: (702) 228-8883 Fax: (702) 228-8885
www.thunderpower-batteries.com

5.3.2 External Power Supply

LittleDog can be powered from an external power supply through the 16 foot power tether provided. The tether can often be a time saving feature during development as it removes the time associated with charging batteries. The tether can be connected or disconnected with internal battery packs installed. That means if you have a fresh set of batteries in the robot and you want to do a tether-less test you can simply unplug the tether without shutting down the robot.

The external power supply must be able to provide a nominal voltage of 18 VDC, at 6 A continuous to exercise the full capabilities of the platform. The robot will operate with an external supply range of 13 to 20VDC. Note that if the supply voltage is below 18VDC batteries in the robot will be discharged. It is worth noting that desktop development of robot behaviors (robot in the air on a stand) can be done with an 18 VDC supply at 2 A assuming no significant external load is applied to the legs.

5.4 MoCap Marker Maintanence

The retroreflective markers on the body of the robot are prone to damage. It is required that they protrude from the body to improve visibility to the MoCap system. This means that they are often the first things to make contact with the ground during a fall, resulting in damage to the marker.

In order to keep your MoCap system working you MUST ensure that damaged markers are replaced or repaired. The markers can be repaired using small pieces if retroreflective tape if the damage is small. If the marker is damaged beyond repair it will need to be replaced. Extra markers were included in a small kit with your initial delivery. The markers are glued on with a soft glue that can be removed, simply “un screw” the marker from its post. If you are unsure about what type of glue to use please contact Boston Dynamics. When the retroreflective tape is scraped off any portion of the sphere the marker is considered damaged.

5.5 The Do’s and Don’ts

- Do make LittleDog walk over really difficult obstacles!
- Do inspect your robot frequently to catch minor problems before they become major ones.
- Do not hesitate to contact Boston Dynamics if you suspect something is wrong with your LittleDog system, email littledog@BostonDynamics.com.
- Do be careful with your robot. When it breaks you may lose valuable research time.
- Do write all code that is to be delivered to the Government Test Team compatible with the provided UI.
- Don’t drop your LittleDog or handle it roughly; the joint stops are designed to handle forces generated by the motors.
- Don’t force joints that feel jammed. The human body is much stronger than the small motors and gears and you can break things with enough force. Attempt to wiggle the joint back and forth until it moves smoothly again.

- Don’t run LittleDog on its stand as the stand may inhibit the range of motion of the legs and the robot may damage itself.

6 Reference

Table 5: Motor Parameters

Property	Value	Unit	Description
Hip Torque Const	1.703	Nm/A	Transmission Scaled Motor Constant
Hip Motor Current Limit	1.0	A	Maximum Motor Current
Knee Torque Const	1.022	Nm/A	Transmission Scaled Motor Constant
Knee Motor Current Limit	0.6	A	Maximum Motor Current

7 Trouble Shooting

7.1 Error Messages

7.1.1 Text Messages

The text error messages that are returned by the API are fairly descriptive. Additional information may be found in the electronic reference manual provided with the LittleDog software, located in `$LITTLEDOG/doc/html/index.html`.

7.1.2 LED Codes

The LittleDog platform has two status LEDs located near the top of the robot at the front end (appear as “eyes”), as shown in. The following combinations of status LEDs, can provide useful information about your platforms status.

Table 6: Status LED Codes

RIGHT Eye Color	LEFT Eye Color	STATUS
OFF	**OFF**	Booting {OR} Trial running with no errors
RED	**RED**	Robot software is running, not connected to HOST
ORANGE	**ORANGE**	Robot is connected to HOST computer
GREEN	**GREEN**	Calibrating {OR} Calibration Completed Properly

7.2 Frequently Asked Questions

At times, machines do not operate the way we would like. If the LittleDog System is not performing properly review possible problems and their resolutions as described below. If these do not solve the problem, contact Boston Dynamics through the contact information provided with your LittleDog system.

- “Can I paint or modify my robot?”
  - No.
  - You may not have the same robot throughout the entire program. Please insure any markings (such as university stickers) are non permanent.

- “I read the whole manual and I still need help!”
  - The first place you should go is the official project website where you will find useful posting and contact instructions for reaching the correct person at Boston Dynamics.

- “LittleDog will not turn on…”
  - Check to see if battery is charged / power tether is attached
  - Cycle the power switch

- “When I cycle the power on my LittleDog, sometimes it does not turn on…”
  - Every now and then the power board will latch off, just try turning the switch to the off position and back again.

- “I own more than one LittleDog robot – can I operate them simultaneously?”
  - No, your HOST machines have been configured to talk to one robot at a time.

- “LittleDog’s onboard IMU accurately tracks body position and orientation at the beginning of an experimental run, but after awhile, the values drift – why?”
  - The IMU contains accelerometers which integrate acceleration to compute body position and orientation. Over time, small errors in the detected acceleration will accumulate and yield an inaccurate position reading. The IMU data is only valid for short periods of time.

Boston Dynamics

515 Massachusetts Avenue, Cambridge MA 02139 USA   617-868-5600   www.BostonDynamics.com