Design and Control of the BlueFoot Platform : A Multi-terrain Quadruped Robot

A Thesis
Submitted in Partial Fulfillment
of the Requirements for the
Degree of
Master of Science (Electrical Engineering)
at the Polytechnic University

by

Brian Cairl

May 2015

Master's Examination Committee:

Copy Number:

Approved by:

Advisor

Date

Department Head

Date

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Hardware and Design

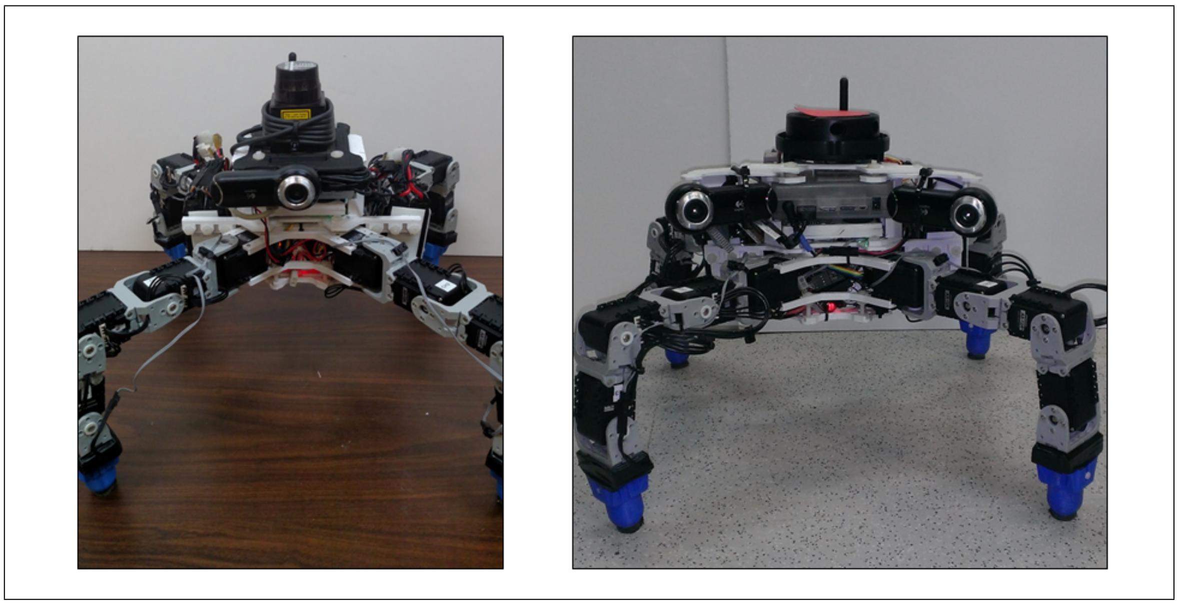

Figure 1: The BlueFoot Quadruped Robot: single-camera configuration *(left)*; stereo-camera configuration *(right)*

1.1 Overview and Design Goals

The BlueFoot quadruped robot is designed as a small-scale, general-purpose legged mobile platform with enough physical dexterity and on-board computational/sensory power to perform complex tasks in variable environments. BlueFoot's hardware configuration is aimed at performing of tasks as a standalone unit, i.e. without power tethering or off-board processing. BlueFoot's sensory, computational and power-source outfit make it fit to complete tasks in both settings fully and semi-autonomous modes.

The implementation of a legged robot which meets these general specification is inherently bottlenecked by several well-known shortcomings which plague legged robot design. These drawbacks can be summarized as follows: relatively low payload capacity, as leg joints are often subjected to substantial dynamic torque loading during gaiting; and higher power consumption due to a, typically, larger number of total actuators. Thus, a general-purpose, multi-legged system like the BlueFoot platform must ultimately achieve a balance between payload-carrying capacity (i.e. maximum joint-servo output torque); actual on-board payload; and on-board energy supply. It is desirable that the sensory and computational power; as well as overall mobility of a legged system are simultaneously maximized along with the aforementioned characteristics.

The design goals which have guided the implementation of the BlueFoot quadruped have been tailored to a yield an overall system design which is both feature rich, computationally powerful, and exploits the natural dexterity and terrain handling of legged robotic systems. Namely, the core design requirements which have guided BlueFoot's development can be summarized as follows:

- The use of legs with joint redundancy for improved dexterity
- The use of smart servos for extended joint feedback and control
- A distributed on-board and computing architecture for hierarchal task handling
- A vision sensor array including a camera and laser-ranging sensors
- 30+ minutes of total battery life

This chapter will outline how an implementation meeting these design goals is achieved, starting with the structural layout of the system. Next, major system payloads and the associated interfacing of major devices will be described. This section which will include details about BlueFoot's actuators, computational modules, and sensory mechanisms. Lastly, the system power routine, energy requirements, and runtime will be detailed.

1.2 Robot Structure

BlueFoot's body is designed in a modular fashion and is comprised of mostly custom designed, 3D printed parts. The use of 3D printing as a fabrication method allowed for rapid design iterations the early stages of system prototyping, and has kept the weight of the robot's overall structure relatively low. Parts were mainly printed from both PLA and SLA plastics. BlueFoot's overall weight (when fully outfitted) is 1.85 – 1.98 kg, depending on configuration.

The modularity of BlueFoot's overall structure arises from the inherent design requirements associated with 3D printing and general design practices aimed at keeping the system reconfigurable for the incorporation of updated sensory and computational hardware. Moreover, parts are designed to fit future replacements while conforming to the constraints imposed by the 3D printing fabrication method, i.e. particular part size and orientation requirements. Such constraints had to be met by each designed part to ensure print feasibility.

The BlueFoot platform has undergone several minor redesign phases since its inception. These redesigns were necessary to bring the BlueFoot platform to its final structural and hardware state and were performed to accommodate changes in sensory/computational hardware. The sections that follow will mainly focus BlueFoot's final hardware configurations.

1.2.1 Main Body (Trunk) Design

BlueFoot's trunk consists of the three main sections: a lower module which interfaces the legs with the main body; a center chassis, designed to hold computational and battery payloads; and a top platform, which interfaces the system's visions sensors to the trunk. These sections will be referred to as the *Root* module, the *Main* module, and the *Head*module, respectively. The full trunk (not including sensor dimensions) fits within a 21.6 by 21.6 by 15.3 cm bounding box.

The Root Module

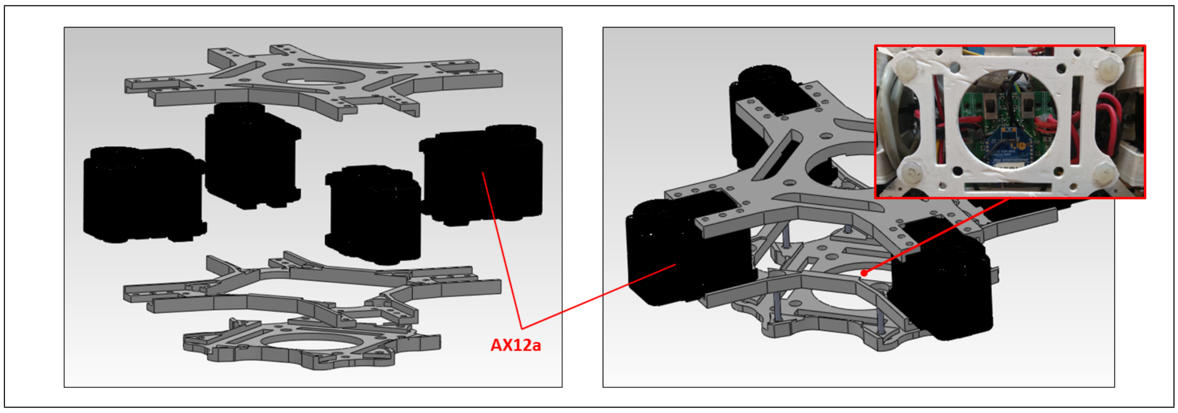

Figure 2: Root section of trunk. Call-out in top-right shows main-switch access through the bottom of the root module.

The Root module, consists of three plates, as shown in Figure 2. Each plate is designed with a central opening to allow for wired connections to pass to other trunk modules. Two such plates directly interface with four servos, which are mounted to four symmetric arms which extend from the center of each plate. These servos are the first joint (hip-joint) of each leg. Each servo mounts to the top a bottom plates via mounting holes located at the top and underside of each servo chassis. The assembly is mated with small steel bolts. A third, smaller plate is attached to the bottom of the module to provide more space for power components and associated wiring. This plate is attached to the bottom of the module via plastic standoffs. An opening in the middle of this plate provides access to the system's main power switches, as well as a removable XBEE wireless radio unit.

The Main Module

The Main module of BlueFoot's trunk includes compartments for an in-house designed AutoPilot unit and a main computer unit, an ODROID-XU. The Main module is designed such that the AutoPilot and ODROID-XU computer slide in and out of the body. The computer payloads are locked into position when the Head module is added to the assembly. The Main body section is designed to fit both computers when stacked upon one another, as depicted in Figure 3. The computer stack is positioned directly in the center of the module when inserted.

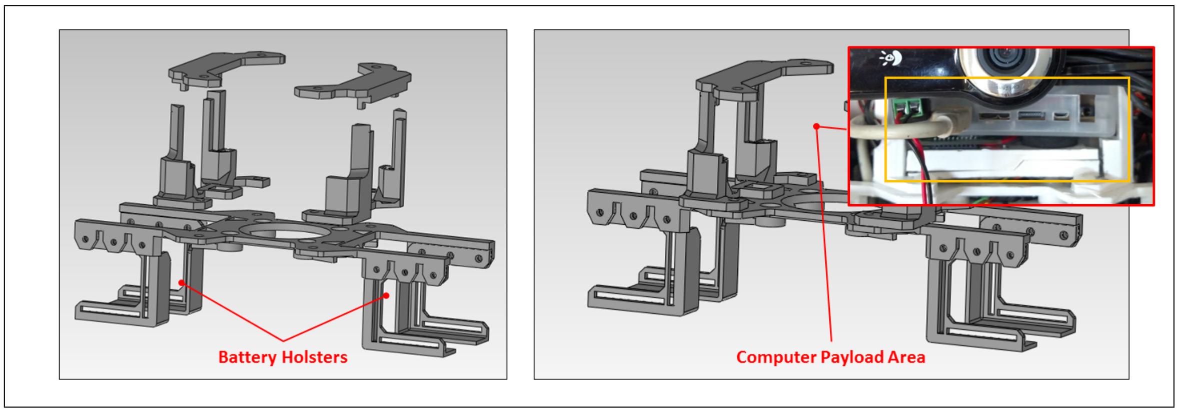

Figure 3: Main section of trunk. Call-out in the top-right shows how the ODROID-XU and AutoPilot computers fit within the module.

The Main module also includes two battery holsters, which hang over its left and right sides. The holsters align the battery packs with the center of Root module. This battery placement serves to lower the center of mass (COM) of the trunk. Doing so serves to lower the magnitude of dynamic torques imparted upon the leg servos during gaiting by decreasing the net moment due to gravity imparted upon the system when the body is oriented away from the direction of gravitational force. The entirety of the Main module is attached to the Root module through the battery holster sub-assembly by four plastic bolts.

The Head Module

Two separate Head modules have been designed for the BlueFoot system : one of which features a stereo camera pair and a Piccolo LIDAR sensor (PLDS); and a monocular design, which features a camera and a Hokuyo-URG LIDAR sensor, as shown in Figure 4. Each head module is attached to Main module via four plastic mounting screws. In the stereo-camera design, two adjustable wings are attached to either side of a top platform which hold cameras. These wings were designed to be adjustable to aid in stereo-camera configuration and calibration. The position of each camera on the trunk allows for a persistent field of view by each camera during mobilization. The PLDS unit is positioned such that the center of its rotating laser head is aligned to the center of the trunk.

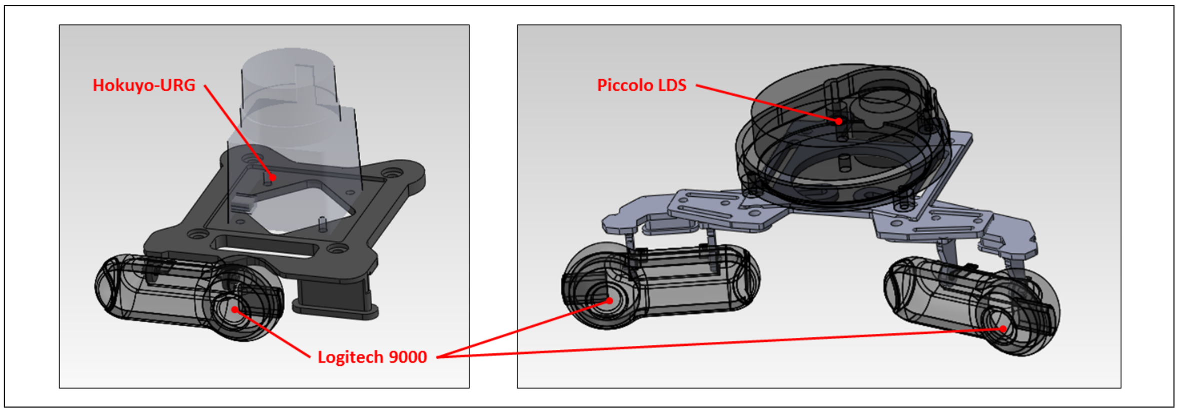

Figure 4: Head section of trunk. Monocular camera configuration with Hokuyo-URG *(left)*; and Stereo configuration with PLDS *(right)*.

In the monocular design, a single camera is mounted such that the lens of the camera is aligned to the sagittal plane of the trunk. This configuration is currently be used as BlueFoot's *primary* head configuration and is mainly being used for 3D point-cloud building and surface reconstruction via 2D LIDAR scans. This is because the

Hokuyo-URG laser scanner used in this configuration offers higher-resolution laser-scan outputs, which will be covered in more detail later in this chapter.

1.2.2 Leg Designs

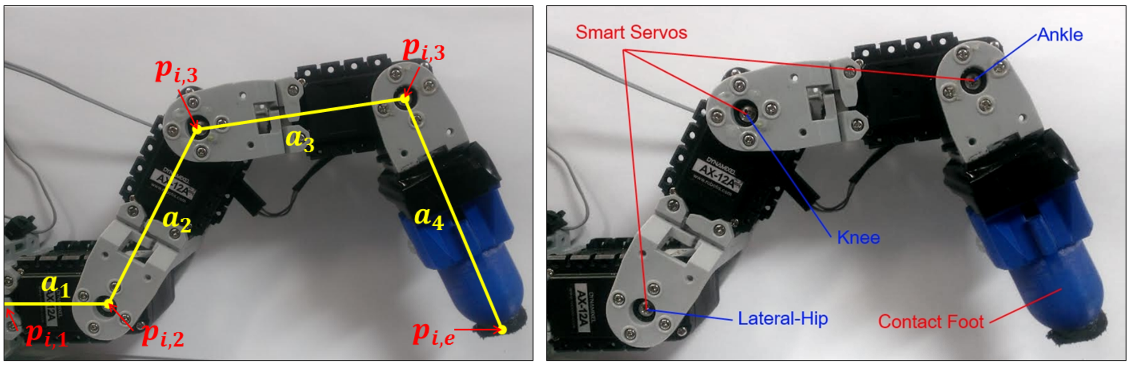

Figure 5: Closeup of BlueFoot's leg. *(left)* shows effective link lengths and the location of defined joint positions.

Each of BlueFoot's legs are identical and are comprised of four Dynamixel AX12a smart-servo actuators (see Figure 5). These actuators are connected via dedicated Dynamixel mounting brackets. Feet are attached to the ends of each leg which contain an embedded, two-state contact sensors. Each foot is designed with a spherical tip, which is rubberized to provide extra grip. The ankle joint of the platform has been added such that the platform can reconfigure its foot orientation while retaining a constant spatial position during gaiting. Additionally, this configuration allows for a considerable amount of independent body re-orientation and repositioning. This capability extends itself to the stabilization and gimbaling of vision sensors mounted on the upper body of the platform while the platform is in motion.

Though not kept in the system's final design, some experimentation was performed with the incorporation of series elastic joints, which were designed to relieve joint impact during gaiting. Series elastic actuation was achieved by replacing bracket interfacing the first and second hip joints of each leg with an elastic-compliant mounting bracket. This bracket includes spring loaded member which was mounted to the horn of the second hip servo on each leg, as shown in Figure 6 and allowed the leg to deflect a small amount at the lateral hip (second joint).

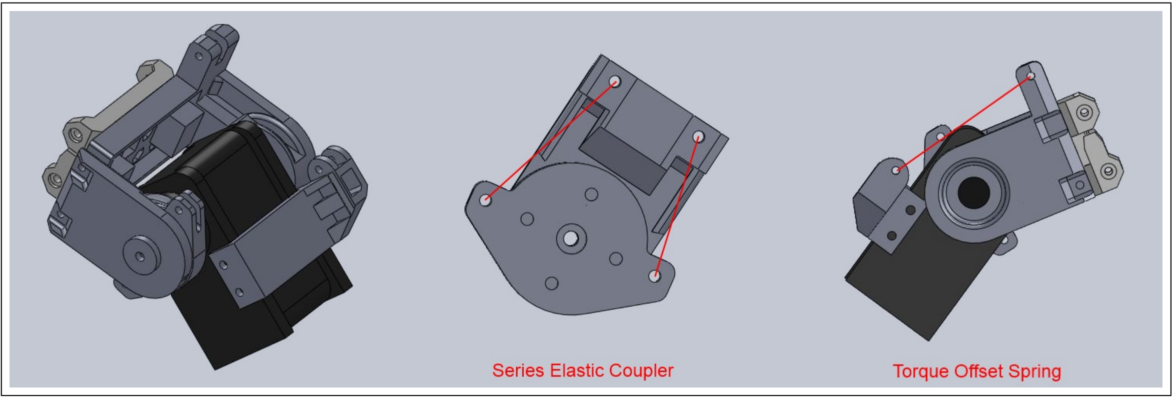

Figure 6: Series elastic brackets.

Link lengths, a_1, a_2, a_3 and a_4; and offset from the center for the Root module to the first joint of each leg, ν are defined in Table 1. These parameters are identical for each leg, and are corresponded with physical leg members labeled in Figure 5.

Link	**Length, m**
a_1	0.06500
a_2	0.06500
a_3	0.06500
a_4	0.06500
ν	0.09215

Table 1: Link and body-offset lengths for each leg.

1.3 Computational and Sensory Hardware

Major payloads on-board the BlueFoot robot are as follows:

- Dual processor AutoPilot unit with a 12-axis inertial measurement unit
- ODROID-XU Computer
- Logitech 9000 Web-cameras
- Hokuyo-URG / Piccolo LIDAR Units (configuration dependent)
- Two-state foot contact sensors (x4)

- Dynamixel AX12a Smart Serial Servos (x16)
- XBEE Wireless radio

Device selection has remained mostly consistent since the platform's inception and initial design, with the exception of computing its main computing units. The AutoPilot unit was updated from an older model, and the ODROID-XU computer replaced a Beaglebone computer for the sake of improving overall computing power.

1.3.1 Device Descriptions

AutoPilot

A dual processor AutoPilot unit performs BlueFoot's low-level gaiting and actuator control tasks, as well as handles communications with a computer running ground-station software. Given the set of low-level sensory and motor-handling tasks it performs, this module has been named the the "Lower Brain" (LB) of the system. The AutoPilot consists of two processing units: a TM4C and RM48 micro-controller (MCU), which operate at 80 MHz and 220 MHz, respectively. These processors communicate over a single UART line, which is used to transfer packeted data between the two processors using a unified inter-processor data transfer protocol, EXI. This protocol which will be described later in more detail. One UART of the RM48 MCU is also connected to an on-board computer, an ODROID-XU, through a USB-to-serial connection. The AutoPilot is powered via an external 12 V supply.

This AutoPilot unit includes a 12-axis inertial measurement unit (IMU) which consists of two, 3-axis accelerometers; one 3-axis rate gyro; and a 3-axis magnetometer unit. This sensor is used for acquiring angular rate data of BlueFoot's trunk and estimating of trunk orientation states using an Extended Kalman Filter (EKF).

ODROID-XU

An ODROID-XU performs many of system's high-level planning tasks, such as navigation, image processing and terrain reconstruction; and handles data data acquisition from both camera and LIDAR sensor units. Given that this unit performs mostly high-level planning tasks, it has been given the name "Upper Brain" (UB). This computer contains a 1.6 GHz, quad-core processor with 2 Gb of RAM. The ODROID-XU can be communicated with over WiFi via a USB WiFi antenna. Currently, SSH tunneling is used to start processes on the ODROID remotely and stream data. The ODROID-XU is powered via an external 5V connection.

Logitech 9000 Web Cameras

Logitech 9000 web cameras have been selected for creating a stereo camera pair, as well as for use in a single camera configuration. These cameras are high-definition web cameras and have a maximum frame rate of 30 fps and a max resolution of 1280 by 720. Cameras are currently read at a the max rate of 30 fps at a more conservative resolution of 640 by 480. These settings are adequate for image processing tasks and have been chosen to reduce nominal data throughput. These cameras are interfaced with the UB (OROID-XU) over a USB connection.

Laser Distance Sensors (PLDS and Hokuyo-URG)

The Piccolo Laser distance sensor (PLDS), which is used in BlueFoot's stereo-camera type configuration, is a 4 meter spinning-head laser range finder. The PLDS has a resolution of a point per degree and covers a range of 360 degrees. Ranging frames (which covers a full rotation) are acquired at a rate of 5 Hz, and are dispatched over a serial connection at 115200 baud. An FTDI break-out board is used to convert the sensor's raw serial output to USB protocol so that the sensor can be interfaced with the UB unit. The PLDS is powered via and external power 5 V source, which is regulated to a 3.3 V voltage level for powering the motor which spins the laser head, and 1.8 V for internal logic. Regulation is performed by an auxiliary power circuit.

The Hokuyo-URG Laser Distance sensor, which is used in BlueFoot's single-camera head configuration, has a a range of 5.6 metets and an angular resolution of 0.38 degrees per point (628 points per scan). This scanner covers a total angular range of 240 degrees. Ranging frames are acquired at a rate of approximately 10 Hz and dispatched directly over a USB connection at 115200 baud. The unit is powered directly over USB.

Foot Contact Sensors

Binary-state contact sensors are embedded in each foot. These contact sensors are essentially limit-switches which generate an active-low signal when the foot comes in contact with the ground. Each sensors is connected to ground and a GPIO pin on the TM4C MCU of the AutoPilot. A 500 Ω is added in series with the limit-switch for the purpose of pin protection.

Dynamixel AX12a Smart Serial Servos

BlueFoot uses 16 Dynamixel AX12a servo units (4 per leg). These servos are position-controlled and commanded over a daisy-chained, half-duplex serial bus (i.e. single wire) at a rate of 1 Mbps. These servos have a maximum holding torque of 1.618 N m and top speed of 306 degrees/s. The AX12s provide position, velocity and loading feedback, however velocity feedback is not used. Servo velocities are, instead, estimated in real time from position feedback because velocity readings provided by the AX12 is relatively noisy by comparison.

Commands are sent to the servos via an aggregate command packet which contains goal-position values for all servo units. Feedback is collected from each servo using individual data-request packets. Servos respond to each request with a response packet containing a corresponding feedback value. Given the number of servos in the network; communication overhead; and the one-wire communication configuration, servo updates are limited to a maximum update rate of 50 Hz over a half-duplex communication line. Gathering feedback over the half-duplex communication bus is particularly expensive because feedback requests require that the host processor wait after each dispatched for a response from each targeted servo. Moreover, each request/response cycle must finish to completion before a feedback request is made to another servo on the communication bus.

A dedicated circuit has been designed for use with these servos which converts a full-duplex serial line to a half-duplex AX12 bus. The circuit uses a two-state tri-state buffer which is switched via a general-purpose I/O line. This switching circuit is integrated into the system's main power switching and distribution board. Each servo is powered via an fused, software-switched 12 V supply line.

XBEE Wireless Radio

An XBEE Wireless Radio, shown in Figure This could be the picture from before, is used for communication between the LB and an external computer ground-station. The radio is interfaced with the LB via 57600 baud serial connection. This radio has a range of outdoor range of 27 meters and a maximum one-way transfer-rate of 115200 bps. Transfer rates between the LB and ground station are currently being limited to 57600 bps to compensate for a lack of hardware flow-control, which is required for stable, two-way communication between two XBEE radios at maximum communication rates. However, the selected communication rate is more than adequate for transferring necessary control information to and from the system without the need for additional flow-control hardware.

This wireless endpoint is used currently used interchangeably with the ODROID-XU's wireless WiFi radio, but will soon be retired to simplify hardware design and increase the platform's data streaming capabilities by switching to a WiFi-based line of communication. Ground station software, as well as the system's internal command-routing and networking software, is designed in such a way to easily accommodate this change.

1.3.2 Device Networking

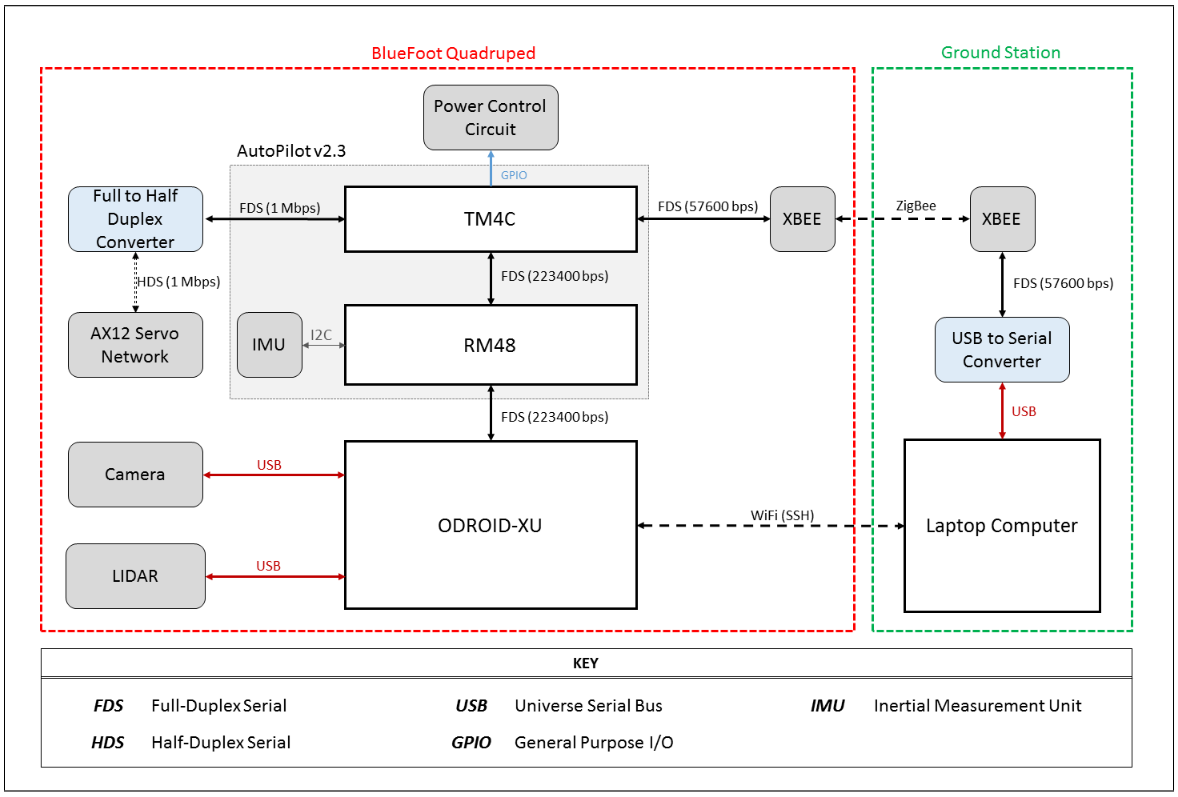

Figure 7: BlueFoot device networking diagram.

Figure 8 depicts how each major device is connected within the system and details the communication rates (f_{com}) between networked devices. Accompanying specifications are detailed in Table 2, which summarizes all device communication pairs and their corresponding baud rates.

Port$_A$	**Source$_A$**	**Port$_B$**	**Source$_B$**	f_{com}, **kbps**
UART0	TM4C	DIN/DOUT	XBEE*	55.7
UART2	TM4C	DIN+	AX12 Net.	1000
UART1	TM4C	LINSCI	RM48	223.4
SCI	RM48	USB (FTDI)	ODROID-XU	223.4
USB	ODROID-XU	DIN/DOUT	LIDAR	115.2
USB	ODROID-XU	USB	Cameras	No Spec.

Table 2: System communication port-pairs and corresponding data transfer rate*Refers to the on-board XBEE module which communicates with the ground station.

1.4 System Power

1.4.1 Power Routing

System power routing is handled via an integrated power switching and distribution board. This board includes physical, main power switches which connects external power to two main, internal 12 volt buses for computer power (Net-1) and motor power (Net-2), respectively. The board also regulates system input voltage to a 5 V bus for use with on-board ICs and 3.3 V bus for powering the XBEE radio. Regulated power and power the servo motors of each leg controlled via three, two-channel power-switching IC's, which are toggled using six digital I/O pins on the TM4C processor of the LB. These power-switching chips allow for software-controlled power configuration, and further, software controlled emergency power cutoff to the servo motors. System main power is supplied via four 12 V (3 cell), 2 Ah Lithium Polymer battery packs.

1.4.2 Energy Requirements and Runtime

The power consumptions of BlueFoot's component device's are summarized in Table 3, which provides the operating voltage, V_{op}, and nominal current draw, I_{nom}, of each active, on-board component. Table 4 details battery specifications (output voltage and amp-hour rating) and BlueFoot's estimated run-time under nominal operating conditions.

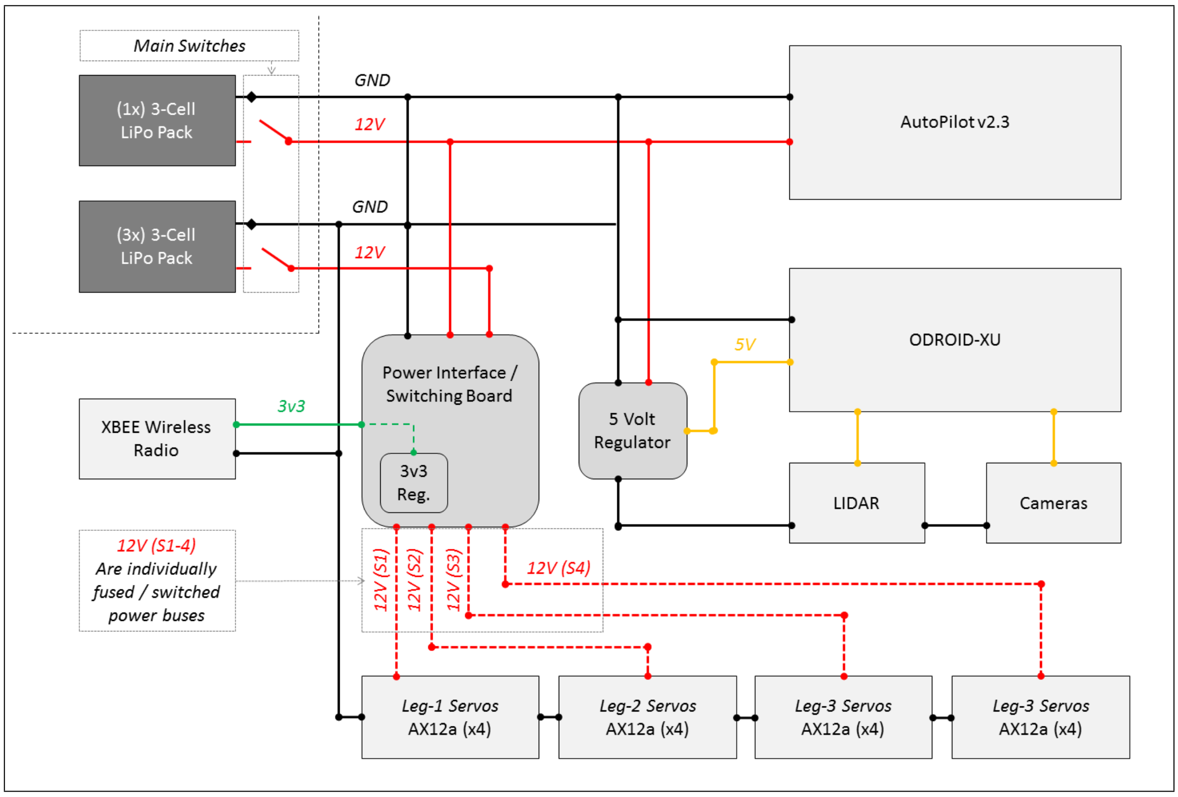

Figure 8: BlueFoot power routing.

Net	**Device**	V_{op}, V	I_{nom}, A
1	AutoPilot (RM48, TM4C)	12.0	0.40
1	XBEE Radio	3.3	0.25
1	ODROID-XU	5.0	2.0
1	Logitech-9000	5.0	0.1
1	Hokuyo-URG	5.0	0.5
2	AX12a Servos (x16)	12.0	9.6 (0.6 ea.)

Table 3: Power consumption summary by device (for single-camera with Hokuyo-URG).

Net	**Battery Pack**	V_{out}, V	**Rating,** $A.hr$
1	3S LiPo Pack (x1)	12.0	2.0
2	3S LiPo Pack (x3)	12.0	6.0
Total Estimated Runtime		35-40 minutes	

Table 4: Battery power supply and estimated runtime summary, assuming low regulator losses.

CHAPTER II

Software

2.1 System Software Architecture

BlueFoot is controlled using a multi-processor software architecture which incorporates several independent core programs. Each of these programs handles portions of system control in a cooperative fashion. Moreover, each of these processors handles specific subsets of operations essential to the macro-system. This distribution of system tasks across several core operating units allows for low-level tasks, such as actuator command and feedback handling, battery monitoring, etc., to be decoupled from more computationally heavy tasks, such as high-level planning and navigation. With task-decoupling in mind, BlueFoot's software architecture was designed such that core programs could be readily offloaded to physically separate computing modules. Each of these control modules handles thier own set of assigned tasks in independent control loops. Information is forwarded from each independent processor to update the overall BlueFoot software macro-system in an asynchronous fashion. System control tasks essential to BlueFoot's overall operation are divided into four main categories, which can be summarized as follows:

- *Low-Level Control* : power monitoring/switching, actuator command handling, communications routing, sensor data acquisition, script parsing and evaluation
- *Locomotion Control* : gait planning, gait adaptation, trunk pose adaptation
- *High-Level Control* : perception, motion planning, surface reconstruction, navigation, localization
- *Human-Operator Control* : joystick/keyboard commands, scripting commands

Low-level and locomotion control tasks are handled, exclusively, by the *Lower Brain* (LB), which designates the software collective spanning over the RM48 and TM4C processors on-board the AutoPilot. High-level control tasks are handled by the Upper

Brain (UB), which is a collection of software which runs on the ODROID-XU (ODROID) module. Lastly, a human operator can interface with the system wirelessly from a personal computer running ground-station software. The ground station communicates with the system through communication lines which enter the TM4C processor and the ODROID computer. The ground-station also interfaces with the UB over an SSH connection. This secondary wireless connection is used, mainly, for on-board data-logging configurations.

Since this software architecture is distributed over several separate computational units, an integral part of this control architecture is an efficient, reconfigurable inter-processor communication protocol. Namely, BlueFoot utilizes data packets transfered over serial lines to update system states between processors. These data packets are formatted using an in-house designed, binary-XML protocol, called EXI. This protocol facilitates a highly customizable packeting structure for asynchronous inter-module communication and utilizes robust packet-error checking routines. This sections will detail the specifics of BlueFoot's interprocessor communication protocol, namely the composition of packets transferred between processor.

This section will also detail the specific software-level tasks handled by each of BlueFoot's processor; the speed at which each core software element is run (update frequency); and what data must be communicated between software elements for operation. Additionally, this section will describe the ground-station software and corresponding user-interface used to control the BlueFoot Quadruped and administer high-level commands.

2.1.1 System Task Allocation

Figure 9 depicts how core software and associated control elements are related within the BlueFoot software macro-system. This section will detail a general description of the tasks carried out by each major software module implemented on the BlueFoot quadruped.

TM4C (Lower Brain)

As previously mentioned, the TM4C processor on-board the AutoPilot module is responsible for *Low-Level* tasks and can be viewed a safety/communications routing co-processor within the overall system. Within its main program loop, the TM4C polls the system's main battery voltage via ADC interface routines; handles transmit and receive

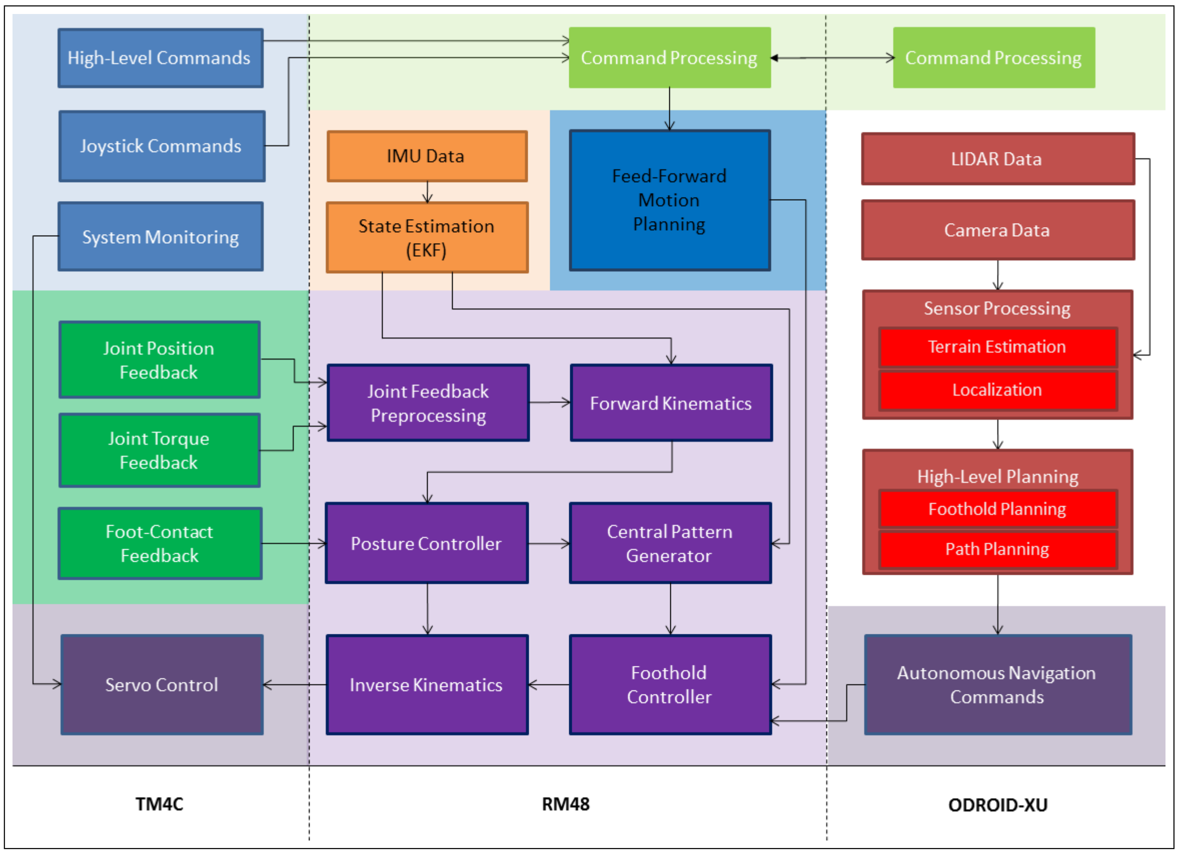

Figure 9: BlueFoot's on-board processes/signals and their relationships.

(packet decoding) routines between the ground-station and the RM48 system nodes; and handles command dispatching and feedback polling with the system's 16 servo actuators. The TM4C is directly interfaced with two dual-channel power switching IC's and is used to control power supply to each leg by toggling general purpose IO pins in software. The state of these pins is administered as part of a periodic packet command/update packet sent from the ground-station. Since the TM4C has this control over the system's actuators (which consume most of the system's power) and battery monitoring capabilities, it runs a safety routine which is responsible for halting motor activity and/or cutting system power on low-battery or power-fault conditions, as well as during unexpected breaks in communication with the ground-station.

As previously mentioned, the TM4C handles communication routing between system processing modules; as well as with controllers on-board each smart-servo. Administering servo commands and collection servo feedback it the TM4C's highest priority task. This process, which involves both commanding and requesting feedback from each servo, is relatively expensive and limits the TM4C's loop frequency to roughly 50 Hz.

Thus, it is particularly important that this task is offloaded to this processor, as its other safety and communication-related tasks are much less expensive, by comparison, and allow the servo actuators to be updated quickly as possible without encumbering other system control operations.

RM48 (Lower Brain)

The RM48 is responsible for several *Low-Level* tasks, including IMU polling and handling communication with the TM4C and ODROID-XU. Each collected IMU sample is passed along to an extended Kalman Filter routine, which generates a trunk orientation estimate, $\hat{\theta}_b$ in the world frame, O_0.

The RM48's primary function is to carry out motion control and gait-planning tasks. To achieve this, the RM48 handles a state machine which switches between planned motion execution and trajectory control; and gait control via a Central Pattern Generator (CPG) based gaiting controller, which will be discussed in more detail in Chapter **??**. Additional functions for body and posture (position and orientation) control, including trunk leveling procedures, and gait-stabilization are run in tandem with the aforementioned gait-control task.

Motion and gait controls, which are performed in the robot task-space, are converted into joint-space reference angles, q^r, via an inverse kinematics (IK) routine. The IK routine is exectued at all times when the legs are engaged for the purpose of issuing servo position commands, given desited task-space configurations for body and feet positions, which will be more formally defined in Chapter **??**. The RM48 also maintains BlueFoot's forward kinematic model (specifically, foot position relative to the trunk), which relies on an EKF-generated trunk orientation estimate, $\hat{\theta}_b$, and joint position feedback, q. BlueFoot's inverse and forward kinematics models will be detailed in Chapter **??**. The RM48 runs its full control loop at approximatively 100 Hz (twice the speed of the TM4C control loop) to facilitate higher integration stability when updating gait related controller dynamics, dynamic motion controls, task-space reference trajectories.

Lastly, the RM48 handles an on-board scripting engine (based on the MIT Squirrel Scripting), which interprets lexical commands. This scripting engine is capable of handling a large number of high-level commands and is complex enough to handle function and class definitions in real time [37]. The scripting engine currently being used to evaluate BlueFoot's core user command set, ranging from simple state toggling and parameter modification, to the prescription of user-specified way-points for navigation, among other high-level command itams. Scripting commands are passed from the ground station (via terminal) and routed through the TM4C to the RM48, where they are finally evaluated.

ODROID-XU (Upper Brain)

The ODROID-XU runs software upon a Debian (Linux) operating system distribution "Jessie." The use of an Linux operating system extends itself to a number of programmatic conveniences, such as to ability to run several tasks in parallel threads. Inbuilt USB drivers are used in functions which are used to acquire data from USB-interfaced vision sensors. Namely, the ORDOID runs sensor handling elements used for acquiring and buffering camera images and controlling camera frame-rate control; as well as LIDAR scan frames. The ORIOID uses these sensor inputs, in conjunction with orientation estimates and inertial data passed from the RM48 to perform several navigation-related tasks (*e.g.*, potential fields, mapping, localization, terrain-reconstruction). These tasks will be described in more detail in Chapter **??**.

The ODROID utilizes 2D-LIDAR scans (frames) and trunk-pose estimates to form organized 3D point clouds. These point-clouds are further processed to reconstruct 3D terrain surfaces and height-maps, which are then used for step-planning. LIDAR frames are utilized in potential fields-based navigation tasks. These navigation modes also incorporate camera data for the purpose of goal-targeting (where the goal is typically an object of particular shape or color). Image processing and image feature detection is run as a separate process on the ODROID, which is incorporated with the aforementioned processed sensor data to produce a set of forward and turning velocity commands, v^r and ω^r, respectively; as well as foot-hold positions generated from step-planning algorithms. In particular, the open-source libraries OpenCV (Open Computer Vision Library), OpenPCL (Open Point-Cloud Library), and Boost are heavily used in the software developed to carry out the aforementioned tasks [38–40]. Software written for this platform was generated using a mixture of C++ and Python.

As previously mentioned, the ODROID can handle a limited set user-command on its own, which are administered directly to the ODROID from an SSH terminal on the ground-station computer. These commands include core-program start-ups and data-logging configurators. Essentially, the ODROID's software core is designed as a completely independent software module which replaces the roll a human director, as it handles the bulk of the systems high-level planning and navigation tasks. Moreover, if the ODROID is removed from the BlueFoot system, the system can still be operated via remote-control heading commands provided from human operator (*i.e.*, ground-station joystick control).

2.1.2 Inter-processor Communication

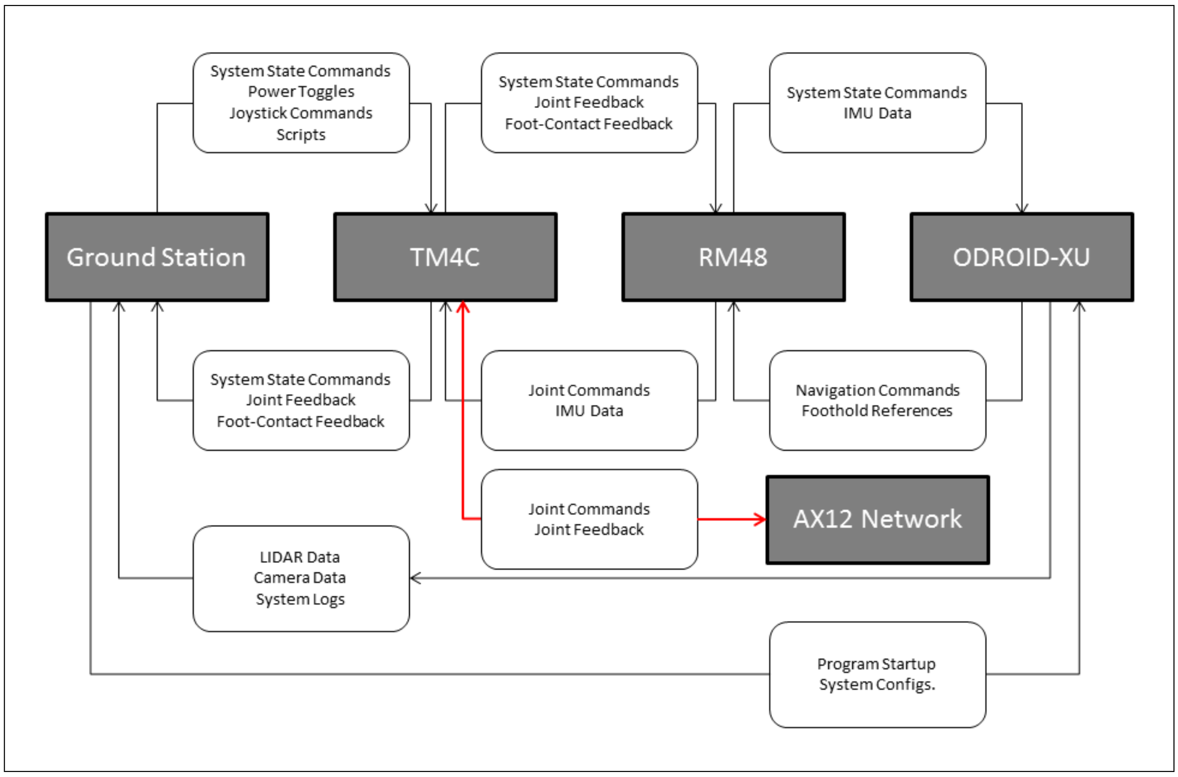

Figure 10: Communication flow between processors on the BlueFoot Platform.

This section will detail the contents of the data packets transfered between processors, which is summarized in Figure 10. System directives, generated by a human operator who interacts with the robot via a graphical user interface and/or joystick controllers, are generated from a ground-station computer. Packets (without padding) sent from the ground station to the BlueFoot robot (TM4C) are composed as shown in Table 5:

Every packet issued using the EXI protocol has a 4-byte (32-bit) header. As part of the internal system protocol, every packet sent between system nodes contains a "Master Status Vector", which is comprised of a fixed length, 7-byte sequence of essential system information/control items. The "Master Toggles" section (first 8-bits after header) enumerate major systems states, including *On-line, Standby, Off-line* and *Suspended*

32-bits	*8-bits*	*8-bits*	*16-bits*	*16-bits*
HEADER	Master-Tog.	Power-Tog.	Unused	Network Info
Variable				
Scripts				

Table 5: Structure of the packets sent from Ground-Station to TM4C.

system state designations. The next 8-bits are used to toggle on-board power, namely the power supplied to each leg. The remaining 32-bits are used for specifying battery voltage (8-bits), power-fault states (8-bits), generic binary feedback toggles (8-bits), and system networking information (16-bits). The last section of this packet contains scripting commands, which can be of varying lengths. For example, foward velocity and turning rate commands (gathered form a joy-stick controller) are administered in the form of scripted commands.

Packets sent from the TM4C to the Ground-station contain status items generated on board the robot, and appear as shown in Table 6: The *Joint Pos. FB* (joint position

32-bits	*8-bits*	*8-bits*	*8-bits*	*16-bits*
HEADER	Master-Tog.	Unused	Foot-Contacts	Network Info
2048-bits				
Joint Pos. FB				

Table 6: Structure of the packets sent from the TM4C to the Ground-Station.

feedback) element is composed of 16 2-byte sequences corresponding to the joint positions of read-back by each actuator. To avoid redundancy, the structure of the packets communicated from the TM4C to the RM48 and vice-versa will not be depicted explicitly. Like all system packets, these packets contain a 7-byte master status vector with only the *Master-Toggle* and *Network Info* fields populated. The TM4C sends the same joint feedback information to the RM48 as it does the Ground Station. For packets sent from the TM4C to the RM48, this field is replaced with corresponding joint-position commands for each of the 16 servo actuators. This field is also 2048 bits in length.

Packets sent from the RM48 to the ODROID contain additional dynamical-state fields for use in planning on the ODROID. State information is sent in the form a vectors with 32-bit, single precision floating-point elements. This set of information includes trunk a orientation estimation, angular rate, and global position (generated from open-loop command integration), each of which are represented as 3-element vector; and foot-position estimates (four, 3-element vectors.) generated. These packets have a structure which is depicted in Table 7. Packets sent form the ODROID to the RM48 contain

32-bits	*8-bits*	*8-bits*	*8-bits*	*16-bits*
HEADER	Master-Tog.	Unused	Foot-Contacts	Network Info
96-bits	*96-bits*	*328-bits*	*328-bits*	
Orientation	Angular Rate	Trunk Pos.	Foot Positions	

Table 7: Structure of the packets sent from the RM48 to the ODROID.

command items, such as forward velocity, turning rate and trunk-pose commands, as well as foothold-references and corrected global trunk position estimates. These packets are constructed as shown in Table 8 :

32-bits	*8-bits*	*8-bits*	*8-bits*	*16-bits*
HEADER	Master-Tog.	Unused	Unused	Network Info
32-bits	*32-bits*	*96-bits*	*328-bits*	*96-bits*
Velocity	Turning Rate	Trunk Pose	Footholds.	Trunk Pos.

Table 8: Structure of the packets sent from the ODROID to the RM48.

2.2 Ground Station

Ground-station software used for controlling the BlueFoot platform is generated in C++ using an open-source graphical user interface design library called *wxWidgets* [41]. *wxWidgets* is used, primarily, for the ground-station's front-end design. Namely, the ground-station code is designed such that that its UI design is reconfigurable via an XML-based design specification file. Furthermote, this allows for UI reconfigurations without the need to change internal, back-end processes. In terms of the GUI backend, *wxWidgets* handles general UI signal processing, and allows for easy registration of user callbacks on events such as button presses. Additionally, *wxWidgets* provides an interface for collective joystick inputs, which are interpretted and sent to the BlueFoot as remote-control commands. *Boost*, a C++ utility library, is utilized to provide socket and serial-port IO handling. This library is particularly important for use with the ground-station's wireless radio, which transmits information to the robot from data sent to the radio module over USB.

The ground station is composed of several main UI sections which allow for system master-state, opterating-state and power toggling; administering system commands and script. Basic system state controls are used to periodically populate a fixed-structure, robot-command packet, shown in Table 5. This packet is used to refresh the robot operating state and manually-set navigation commands, and is sent to the platform rate of 25 Hz using the an XBEE radio module. BlueFoot replies to each ground-station update with a packet of internal configurations, as shown in Table 6, which is then used to ensure that system updates and command are being adminstered properly and that the system is live.

Namely, BlueFoot sends the following particles of system information back to he ground-station: battery voltage levels; power fault coniditons; foot contact feedback; and joint position feedback. This data is used to update several key portions of the UI. Firstly, battery levels are used to update a battery meter, which indicates the current system voltage level. Additionally, a dynamic text box displays the system's power-fault state to the user. Joint position commands are used to update a visualization of the robot in real time. The foot-color of the vizualized BlueFoot robot changes from blue to red to represent foot-contact conditions. Joint positions, foot-contact states and battery levels are time-stamped and automatically logged during each ground-station session.

[1] R.B. McGhee and A.A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3(0):331 – 351, 1968.

[2] J.K. Hodgins and M. Raibert. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation*, 7(3):289–298, Jun 1991.

[3] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, Jr. Brown, H.B., D. McMordie, U. Saranli, R. Full, and D.E. Koditschek. RHex: A Biologically Inspired Hexapod Runner. *Autonomous Robots*, 11(3):207–213, 2001.

[4] J.Z. Kolter, M.P. Rodgers, and A.Y. Ng. A control architecture for quadruped locomotion over rough terrain. In *IEEE International Conference on Robotics and Automation*, pages 811–818, May 2008.

[5] Tedrake R. Kuindersma S. Wieber, P.-B. Modeling and control of legged robots. In Siciliano and Khatib, editors, *Springer Handbook of Robotics, 2nd Ed*, Lecture Notes in Control and Information Sciences, chapter 48. Springer Berlin Heidelberg, 2015.

[6] Y. Fukuoka, H. Kimura, Y. Hada, and K. Takase. Adaptive dynamic walking of a quadruped robot 'Tekken' on irregular terrain using a neural system model. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2037–2042 vol.2, Sept 2003.

[7] Joaquin Estremera and Kenneth J. Waldron. Leg Thrust Control for Stabilization of Dynamic Gaits in a Quadruped Robot. In Teresa Zieliska and Cezary Zieliski, editors, *Romansy 16*, volume 487 of *CISM Courses and Lectures*, pages 213–220. Springer Vienna, 2006.

[8] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. BigDog, the Rough-Terrain Quaduped Robot, 2008.

[9] C. Semini. *HyQ Design and Development of a Hydraulically Actuated Quadruped Robot*. PhD thesis, Apr 2010.

[10] J.R. Rebula, P.D. Neuhaus, B.V. Bonnlander, M.J. Johnson, and J.E. Pratt. A Controller for the Littledog quadruped walking on Rough Terrain. In *IEEE International Conference on Robotics and Automation*, pages 1467–1473, April 2007.

[11] M. Ajallooeian, S. Pouya, A Sproewitz, and A.J. Ijspeert. Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion. In *IEEE International Conference on Robotics and Automation*, pages 3321–3328, May 2013.

[12] Simon Rutishauser, A Sprowitz, L. Righetti, and A.J. Ijspeert. Passive compliant quadruped robot using Central Pattern Generators for locomotion control. pages 710–715, Oct 2008.

[13] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural networks*, 21(4), 2008.

[14] P. Arena. The central pattern generator: a paradigm for artificial locomotion. *Soft Computing*, 4(4):251–266, 2000.

[15] Paolo Arena. A mechatronic lamprey controlled by analog circuits. In *MED'01 9th Mediterranean Conference on Control and Automation*. IEEE, June 2001.

[16] Bernhard Klaassen, Ralf Linnemann, Dirk Spenneberg, and Frank Kirchner. Biomimetic walking robot scorpion: Control and modeling. *Robotics and Autonomous Systems*, 41(23):69 – 76, 2002. Ninth International Symposium on Intelligent Robotic Systems.

[17] P. Arena, L. Fortuna, M. Frasca, and G. Sicurella. An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(4):1823–1837, Aug 2004.

[18] Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. {CPG} model for autonomous decentralized multi-legged robot systemgeneration and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(34):171 – 179, 2003. Best papers presented at IAS-7.

[19] Shinkichi Inagaki, Hideo Yuasa, Takanori Suzuki, and Tamio Arai. Wave {CPG} model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118 – 126, 2006. Intelligent Autonomous Systems 8th Conference on Intelligent Autonomous Systems (IAS-8).

[20] Giorgio Brambilla, Jonas Buchli, and AukeJan Ijspeert. Adaptive four legged locomotion control based on nonlinear dynamical systems. In Stefano Nolfi, Gianluca Baldassarre, Raffaele Calabretta, JohnC.T. Hallam, Davide Marocco, Jean-Arcady Meyer, Orazio Miglino, and Domenico Parisi, editors, *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 138–149. Springer Berlin Heidelberg, 2006.

[21] J. Buchli, F. Iida, and A.J. Ijspeert. Finding resonance: Adaptive frequency oscillators for dynamic legged locomotion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3903–3909, Oct 2006.

[22] K. Tsujita, K. Tsuchiya, and A. Onat. Adaptive gait pattern control of a quadruped locomotion robot. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2318–2325 vol.4, 2001.

[23] K. Tsujita, H. Toui, and K. Tsuchiya. Dynamic turning control of a quadruped robot using oscillator network. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2613–2618 Vol.3, April 2004.

[24] G. Endo, J. Morimoto, J. Nakanishi, and G. Cheng. An empirical exploration of a neural oscillator for biped locomotion control. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE Int. Conf. on*, volume 3, pages 3036–3042 Vol.3, April 2004.

[25] Miomir K. Vukobratovi. Contribution to the study of anthropomorphic systems. *Kybernetika*, 08(5):(404)–418, 1972.

[26] J.-I. Yamaguchi, A. Takanishi, and I. Kato. Development of a biped walking robot compensating for three-axis moment by trunk motion. In *Intelligent Robots and Systems '93, IROS '93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 1, pages 561–566 vol.1, Jul 1993.

[27] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *Trans. Sys. Man Cyber. Part A*, 34(5):630–637, September 2004.

[28] A. Goswami. Foot rotation indicator (fri) point: a new gait planning tool to evaluate postural stability of biped robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 47–52 vol.1, 1999.

[29] Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in NARX recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7(6):1329–1338, Nov 1996.

[30] Grant P. M. Chen S., Billings S. A. Non-linear system identification using neural networks. In *Int. Journal of Control*, volume 51 of *Lecture Notes in Control and Information Sciences*, pages 1191–1214. Taylor and Francis, 1990.

[31] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies, 1996.

[32] S. A Billings. *Nonlinear system identification : NARMAX methods in the time, frequency, and spatio-temporal domains.* John Wiley and Sons Ltd, 2013.

[33] Oliver Nelles. Neural networks with internal dynamics. In *Nonlinear System Identification*, pages 645–651. Springer Berlin Heidelberg, 2001.

[34] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[35] David E. Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[36] Raúl Rojas. The backpropagation algorithm. In *Neural Networks: A Systematic Introduction*, chapter 7. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[37] Alberto Demichelis. Squirrel, the programming language, 2011.

[38] G. Bradski. *Dr. Dobb's Journal of Software Tools.*

[39] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[40] Beman Dawes. Boost C++ Libraries, 2005. [Online; http://www.boost.org/].

[41] wxWidgets, Cross-Platform GUI Library, 2015. [Online; https://www.wxwidgets.org/].

[42] P.-B. Wieber. Holonomy and nonholonomy in the dynamics of articulated motion. In Moritz Diehl and Katja Mombaur, editors, *Fast Motions in Biomechanics and Robotics*, volume 340 of *Lecture Notes in Control and Information Sciences*, pages 411–425. Springer Berlin Heidelberg, 2006.

[43] Russel Smith. Open Dynamics Engine, 2008. [Online; http://www.ode.org/].

[44] Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52(6):367–376, 1985.

[45] J.J. Collins and Ian Stewart. Hexapodal gaits and coupled nonlinear oscillator models. *Biological Cybernetics*, 68(4):287–298, 1993.

[46] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. pages 1585–1590, May 2006.

[47] Vitor Matos and Cristina P. Santos. Omnidirectional locomotion in a quadruped robot: A CPG-based approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3392–3397, Oct 2010.

[48] Meng Yee (Michael) Chuah Park, Hae-Won and Sangbae Kim. Quadruped bounding control with variable duty cycle via vertical impulse scaling. 2014.

[49] Y. Fukuoka, H. Yasushi, and F. Takahiro. A simple rule for quadrupedal gait generation determined by leg loading feedback: a modeling study. *Sci. Rep.*, 5, 2015.

[50] Luiz Castro, Cristina P. Santos, Miguel Oliveira, and Auke Ijspeert. Postural Control on a Quadruped Robot Using Lateral Tilt: A Dynamical System Approach. In Herman Bruyninckx, Libor Peuil, and Miroslav Kulich, editors, *European Robotics Symposium 2008*, volume 44 of *Springer Tracts in Advanced Robotics*, pages 205–214. Springer Berlin Heidelberg, 2008.

[51] Jia-wang Li, Chao Wu, and Tong Ge. Central pattern generator based gait control for planar quadruped robots. *Journal of Shanghai Jiaotong University*, 19(1):1–10, 2014.

[52] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1620–1626 vol.2, Sept 2003.

[53] Katie Byl, Alec Shkolnik, Sam Prentice, Nick Roy, and Russ Tedrake. Reliable Dynamic Motions for a Stiff Quadruped. *International Symposium on Experimental Robotics*, pages 319–328, 2009.

[54] A. Takanishi, M. Tochizawa, T. Takeya, H. Karaki, and I. Kato. Realization of dynamic biped walking stabilized with trunk motion under known external force. In KennethJ. Waldron, editor, *Advanced Robotics: 1989*, pages 299–310. Springer Berlin Heidelberg, 1989.

[55] R. Kurazume, T. Hasegawa, and K. Yoneda. The sway compensation trajectory for a biped robot. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 925–931 vol.1, Sept 2003.

[56] Roberto Battiti. First and second-order methods for learning: between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.

[57] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Comput.*, 11(7):1769–1796, oct 1999.

[58] F. Arambula Coso and M.A. Padilla Castaeda. Autonomous robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*, 40(910):1141 – 1156, 2004.

[59] G. Bradski and A. Kaehler. *Learning OpenCV*, chapter Image Morphology. OReilly Media, 2008.

[60] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[61] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 322–328, New York, NY, USA, 2003. ACM.

[62] Edward Castillo, Jian Liang, and Hongkai Zhao. Point cloud segmentation and denoising via constrained nonlinear least squares normal estimates. In Michael Breu, Alfred Bruckstein, and Petros Maragos, editors, *Innovations for Shape Analysis*, Mathematics and Visualization, pages 283–299. Springer Berlin Heidelberg, 2013.

[63] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.