Cornell University Biorobotics and Locomotion Laboratory

End-of-Year Progress Report

Jehhal Liu (jl589@cornell.edu)
Advisor: Professor Andy Ruina
May 15, 2009

Abstract—Cornells Biorobotics and Locomotion Laboratory is developing a walking bipedal robot, with the intention of understanding the most versatile, robust, and efficient way of controlling it. A very important aspect of the control system is the electronics architecture, which needs to be robust and versatile enough to adapt to new mechanical robotic structures. Additionally, the data collection subsystem needs to be fast and reliable so that a substantial amount of information can be gathered to further the research goal of the lab. In constructing a walking bipedal robot, the lab hopes to expand the knowledge of the details of human motion.

1 Introduction

The purpose of the research in the biorobotics lab is to relate the controls of the robot to the mechanics of human locomotion, in hopes of expanding our knowledge in this still unfamiliar field. In knowing the nuances of an efficient control structure for a human-like robot, details about how the human body controls itself can be better understood. The experiments conducted in this lab can be used for various medical applications, such as the development of prosthetic body parts (particularly those directly related to walking).

Throughout the past year, the lab's electronics team was responsible for redesigning the hardware and software of the electronic control system of the Cornell Ranger. The purpose of the development was to create a more robust, modular system which could be implanted into different robots and quickly adapted to new mechanical structures. This electronics restructuring is an intermediate step with the long-term goal of introducing the system into the lab's new design for a two-legged bipedal robot (instead of the current four legged design).

As a member of the electronics team, I had two main tasks for the past two semesters. During the first semester, I designed the carrier board for the "Main-Brain," which is the ARM9 processor that controls the overarching structure of the software control architecture. During the second semester, after working briefly with programming with the User Interface board, I moved onto a testing phase of the bluetooth module, used in the robot for data retrieval.

2 Main-Brain Carrier Board

During the initial design process for the system's hardware architecture, the ARM9 processor used for the *Main Brain* was the Phytec LPC3180. This processor required a *Carrier Board* which allows the user to utilize the various input/output features of the processor, including the UART, SD, and USB interfaces. This carrier board is used for interfacing with the ARM9, connecting the processor either to the other boards on the robot or an outside computer for programming purposes.

Note: This report describes the carrier board for the LPC3180, however the system was redesigned to use the LPC3250 ARM9 Processor. The interfacing information is still relevant for both processor types.

2.1 UART/RS-232

UART (Universal Asynchronous Receiver/Transmitter) is a transmission standard used to send and receive data between two devices. The UART on this communication board could be used for several purposes, including communicating with other boards, or more importantly, communicating with an outside computer using the RS-232 standard. In order to use the UART port of the processor for RS-232 communication, an RS-232 transceiver is required to control

the respective voltage levels, ensuring that the PC serial port does not blow out the input to the processor.

Both the LPC3180 and 3250 have 7 UARTs. Three UARTs are high speed, meaning they are capable of baudrates of up to 921600. Of the 7 available, 2 of the UARTs have RS-232 transceivers and are therefore capable of connecting to computers. The connections for a female DB-9 cable, which is used to connect one of the RS-232 UARTs to a computer's serial port, is shown in figure 1 and table 1.

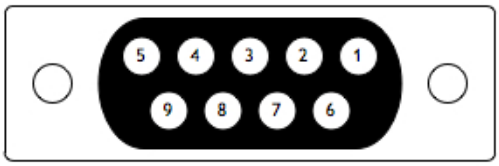

Fig. 1: Pins on a female DB9 connector (front view). Soldered connections are made in the back.

Pin	Signal Description
1,4,6,9	No Connection
2	Received Data
3	Transmitted Data
5	Common Ground
7	Request to Send
8	Clear to Send

TABLE 1: Pin-out for interfacing the UART with a female DB-9 connector.

The UARTs can be wired to the female DB-9 connectors through Tyco's standard four or six pin Micro-MaTch connectors on the carrier board. These connectors are standard multi-pin connectors that can be used for a variety of purposes. The corresponding pins described in table 1 should be wired to the correct pins on the Micro-MaTch connection, allowing the board to be connected to an outside device, such as a computer.

2.2 Micro SD Card

A micro Secure Digital (SD) card is a format for a small flash memory card, which is used in many portable devices for data storage or transfer purposes. The card itself is 15mm x 11mm, and can generally hold gigabytes of information.

Note that as the SD card specification has changed over the years, and as higher capacity cards were being released, compatibility issues have occurred relating to which cards could be read in a particular device. The SD card interface follows the SD Memory Card Specification version 1.01. This implies that SDHC cards are not compatible. These cards use a new format to allow for capacities above 4GB.

Additionally, standard MicroSD format cards with above 1GB storage employ a different identification syntax which cannot be interpreted by the current SD driver on the LPC3250. As a result, the interface can only communicate with cards that have a capacity less than or equal to 1GB.

The microcontroller can readily interface with microSD card slots by sending direct lines from the card reader to the MCU. The SD card employs a synchronous read/write format, and is clocked for data transmission, sending four bits of data during each clock cycle. The clock speed is dependent on the speed grade for the card, which can be one of three values as indicated in table 2. These speeds refer to minimum write speeds for when no memory unit on the card is occupied. Otherwise, data fragmentation could alter performance speeds.

The signal descriptions for each of the 8 pins of a microSD card are described in figure 2 and table 3. Signals sent through the command line are used to initiate certain operations from the controller to

Speed Grade	Transfer Rate
Class 2	2 MB/s
Class 4	4 MB/s
Class 6	6 MB/s

TABLE 2: Speed grades for microSD cards.

the reader. More information on SD cards in general and how to use them can be found in the SD Card Specification at www.sdcard.org.

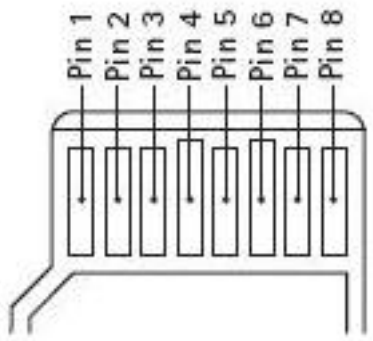

Fig. 2: Pins on a microSD card.

Pin	Signal Description
1	Data bit 2
2	Data bit 3
3	Command Line
4	Supply voltage
5	Clock
6	Common Ground
7	Data bit 0
8	Data bit 1

TABLE 3: MicroSD pins.

2.3 USB

The Universal Serial Bus (USB) is a connection standard for linking a peripheral to a host device. The microcontroller unit contains an on-board USB transceiver that is capable of allowing the MCU to act as a host to other USB-capable devices.

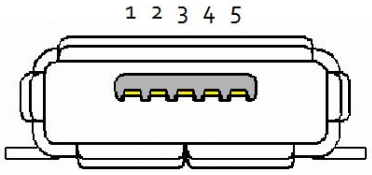

Fig. 3: Pins of MicroUSB connectors.

Micro USB connections require five pins for communication. The pins on the receptacles are shown in figure 3 and described in table 4.

Pin	Signal Description
1	Supply voltage
2	Data-
3	Data+
4	ID
5	Common ground

TABLE 4: MicroUSB pin descriptions.

Though the USB cable has two data transmission lines, USB is still a half-duplex transmission model. The two lines are inverted versions of the same signal. As such, the signals are differentiated at the receiving end, allowing the common noise to be wiped off from the signal. The ID pin is used to specify whether the device is on the A side or the B side of the communication line, which typically designate the host or device end of the USB connection.

USB devices are typically connected with a designated master/slave (or host/device). New USB2.0 devices which support USB On-The-Go (OTG) allow two devices to connect to one another without having a specified host. In an OTG connection, one of the two devices would act as the host, which is responsible for scheduling and configuring the data transmission between the two devices. The LPC3180 (and the LPC3250) is capable of USB OTG connections, and is also able to act as either a *host* or *device* in the

communication scheme.

2.4 Final Design for LPC3180

The final design for the LPC3180 (figure 4), which I completed last semester and built at the beginning of this semester, consisted of the communication standards described in the previous subsections as well as various other components for interfacing with the other capabilities of the MCU. Such components included more Micro-MaTch connectors for the MCU's on-board analog-to-digital converter, the MCU's SPI communication pins, and the power input from the 5 volt CAN bus line. A bluetooth module resides on the carrier board and communicates with the ARM9 via one of the UART lines. This wireless module is used for data collection purposes, and is described in more detail in section 3. Additionally, a battery is included as a backup power supply for the on-board MCU clock.

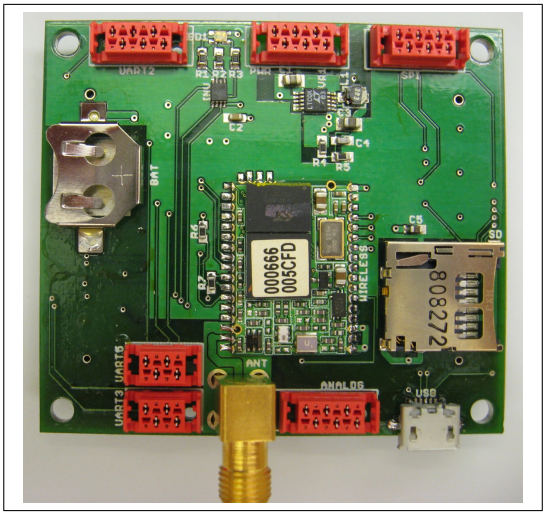

Fig. 4: The populated LPC3180 Carrier Board.

As a final note, as mentioned previously, the board pictured in this section is a carrier board for the LPC3180; however the board was later modified to be compatible with the LPC3250. Both products are ARM9 processors and have similar capabilities. The switch from one product to another came after a decision to alter the software architecture that would eventually be programmed into this module.

3 Bluetooth

Bluetooth is a wireless communication protocol that is used for relatively short distance (100m) connections. This standard is widely used for what is called "cable-replacement," allowing device peripherals (i.e. a computer mouse, or printer) to be connected without the wiring that was previously required.

The purpose of the bluetooth module in our robot electronics infrastructure is to remotely retrieve data that was collected during the robot's operation. This data can be used to analyze the movement and control of the robot and further our research initiative. Currently, the Ranger uses the Radiotronix WI232DTS wireless module that can send a wide range of variables back to the receiving computer. However, the throughput on the implemented system can be increased using a new module, allowing more data to be sent for analysis.

3.1 Module

The module that is currently being tested is the Roving Networks RN-21 bluetooth module. This module currently resides on the carrier board of the main brain, and can interface with the processor using one of the UARTs.

On the other end of the connection is the Roving Networks RN-24 module which acts as a stand-alone device. This module contains the RN-21 module onboard a custom board with status LEDs, an antenna

jack, and easily solderable general-purpose input/output pins for communication or interface tweaking.

The RN-21 module has two modes of communication through the UART: SPP and HCI. The claimed SPP data transfer rate when sending from the master device to the slave device is 300Kbps. The claimed HCI sustained data transfer rate is 1.5Mbps. The devices used in lab, as discovered after purchasing the devices, are only capable of SPP communication and thus cannot achieve the desired 1.5Mbps transfer rates. The RN-21 module has several different models, each of which has a different communication protocol for the different interfacing standards (SPP or HCI for UART or USB). The RN-21H model allows HCI communication through UART, which is what should be used for data transmission on the new robot, if this device is chosen for the final product.

3.1.1 Auto-Connect Feature

The module is capable of storing a particular address in memory and automatically connecting to that device upon startup. This mode can be configured in the command mode by altering a few settings (Information on how to enter command mode can be found in the RN-21 datasheet from Roving Networks' online website).

To configure a module to automatically connect to another device, the module must first be configured to run in **Auto-Master** mode. Next, the address of the secondary device should be stored in the master module's memory, using the appropriate command in command mode. Once the master module has been reset, the devices will connect if in range. *Note that the master module can never enter command mode if it has established a bluetooth connection.*

The address of a particular device can also be found easily in command mode by using the appropriate command sequence **GB**. More information on other get and set commands can be found in the datasheet on the Roving Networks website.

3.2 Throughput Testing Procedure

Two testing methods have been used to determine the maximum throughput between two of these devices. The first method is strictly done through the Windows' hyperterminal and involves no coding. The second involves transmitting data using matlab, and determining the difference between the time of arrival of the first byte sent and the last byte sent. The testing methods are described in the next two sections.

The RS-232 pins on the stand-alone RN-24 module were fed to a female DB-9 connector (which was described in section 2.1. This connection was subsequently fed into a serial-to-USB adapter which was then connected to one of the USB ports on the computer test bench. Both testing methods were performed using the same two RN-24 modules configured to a UART baudrate of 460800.

3.2.1 Hyperterminal Testing

Windows XP's Hyperterminal can display incoming traffic to a particular serial port, and can send outgoing traffic to the same port. This testing method involved connecting two RN-24 devices to the same computer, and using two hyperterminals to monitor incoming and outgoing traffic for both devices. The steps taken in this procedure are as follows:

1) Connect two RN-24 modules to the host computer using the serial-to-USB adapters. These modules should

be connected via bluetooth using the auto-connect feature described in section 3.1.1.

2) Open two instances of hyperterminal. Connect one instance to one of the RN-24 devices and the other instance to the other device.
3) On either instance of hyperterminal, click the **Send File** option in the **Transfer** menu.
4) Type the path of the file to transfer in the **Filename** box. The file sent during this experiment was *sendData.txt*, which simply contained 10,000 repetitions of the string "0123456789" (100,000 Bytes of data).
5) Select the **Zmodem with Crash Recovery** option in the protocol list and click **Send**.

After hitting **Send**, a new window appears which contains transmission information, including an output of the instantaneous throughput. A sample of this window is shown in figure 5. Notice in the figure that the throughput (boxed in red) is 240640bps, meaning 240640 bits are being sent per second. Each character (i.e. '0') is equal to one byte, or 8 bits.

The received bitstream is then stored in a separate text file by the receiving client. This received file was checked against the transmitted data file to ensure reliable transmission occurred (i.e. no corrupt/altered data).

The sustained throughput claimed for this device in SPP mode is 300Kbps when sending from the master device to the slave device, and 240Kbps when sending from slave to master. The results showed that the sustained throughput was roughly 240Kbps while sending from the master (figure 5), and roughly 200Kbps when sending from the slave (figure 6). These two numbers were obtained in two different experiments during which one-way transmission was performed.

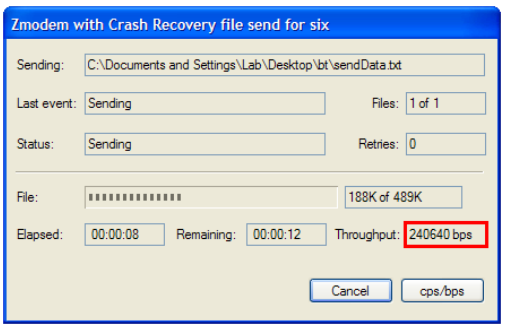

Fig. 5: The transmission window displayed when sending a file from the master to the slave device.

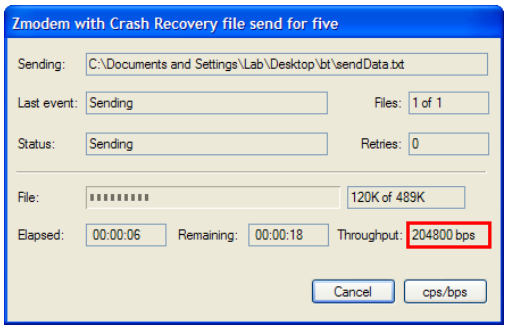

Fig. 6: The transmission window displayed when sending a file from the slave to the master device.

3.2.2 *Matlab Testing*

Matlab's serial port communication capabilities were used to test the reliability and throughput of this device. Matlab has a built in *serial* object which can be used to transmit or receive data from a particular serial port on the computer.

Two instances of Matlab were used during testing, since one instance of Matlab cannot run two processes simultaneously. The steps involved in throughput testing are as follows:

1) Connect two RN-24 modules to the host computer using the serial-to-

USB adapters. These modules should be connected via bluetooth using the auto-connect feature described in section 3.1.1.

2) Open two instances of Matlab.
3) In one instance of Matlab, create a serial object for the COM port that corresponds with one of the bluetooth devices. In the other instance of Matlab, create a serial object for the COM port of the other device.
4) Open both connections using the **fopen(*obj*)** command.
5) On the receiver side, flush the input buffer and run a script that constantly checks the size of its input buffer and the time of the check. This will monitor the time of arrival of the packets of bytes as they are transmitted via the wireless connection.
6) On the transmitter side, use the **fprintf(*)** command to send a stream of known characters to the serial object.
7) When the receiver buffer size reaches the size of the transmitted data, check the timestamps to determine the time it took for the receiver buffer size to reach the size of the transmitted data, starting from the time of reception of the very first packet.

A copy of the code that runs this script for both the transmitter side and the receiver side is in the appendix. During the experiment, 50,000 bytes were sent, consisting of 5000 repetitions of the string ”0123456789” in step 6.

Upon knowing the number of bytes transmitted and the time it took for them to enter the receiver input buffer, the throughput of the communication can be easily calculated. The steps above were repeated 40 times. The mean transmission rate for the 40 iterations was 157036bps with a standard deviation of 2883bps.

The cause of the significantly worse results in the Matlab throughput testing is unknown. It is possible that the **fprintf(*)** command produces enough overhead that the throughput is compromised. However, it was proven via the hyperterminal test that the bluetooth is capable of at least 200Kbps transmission rates. Testing the throughput for when the device is actually implemented into the robot can be completed once programming of the mainbrain (or a similar MCU) has begun.

3.3 Test Results

The results of both the Matlab testing and the Hyperterminal testing both show that transmission speeds could potentially surpass the transmission speed of the data retrieval system currently implemented on the Ranger; however the reason for choosing this particular device was to achieve the 1.5Mbps sustained transmission rate that was claimed in the datasheet.

It should be noted that the Zmodem protocol employed by the hyperterminal transmits files in 1024-byte data packets, however the packets from the robot for data transmission will be smaller, resulting in more overhead which will affect the throughput of the device. This testing simply shows the capabilities of the bluetooth module, but may not reveal typical data transmission speeds for when the communication is implemented into the software control architecture.

The decision must now be made whether to purchase the HCI-compatible RN-21/24 modules for further testing, or to purchase different wireless modules which are capable of faster (greater than 240Kbps) transmission. Seeing as the hyperterminal testing showed throughput at roughly 80% of the claim, it could be a safe assumption to say that the

HCI-compatible devices would perform at roughly 80% of its maximum throughput rating. Additionally, it must be taken into consideration that the maximum UART baudrate of the RN-21 module is 921600, which could limit the maximum data transmission and reception rate by a considerable amount.

4 Conclusion

While the new LPC3250 carrier board is complete, it cannot be truly tested until the coding has begun and the communication between various boards is established. The new electronics system is currently being implanted into the Ranger to ensure that the system can function correctly and efficiently. Once the new robot is designed and constructed, the electronics system can then be ported to the new design.

As for the bluetooth module for data extraction, it has been established that the device is capable of transmitting at high speeds. However, the absence of the HCI-mode compatibility is a drawback to the use of the module. A module that is, at the very least, capable of near 1Mbps transmission would be desirable. In the long run, a higher throughput device is necessary especially with the current state of the work in the lab. At this transition point between a new electronics design and a new mechanical design, the data collection portion of the experimentation is crucial for providing the information that can help further the knowledge of robot and human motion.

5 Appendix

5.1 Online References

1) Roving Networks RN-21: http://rovingnetworks.com/bluetooth-modules.php
2) Roving Networks RN-24: http://rovingnetworks.com/bluetooth-super-modules.php
3) MicroUSB Connector: http://www.hirose.co.jp/cataloge_hp/e24200011.pdf
4) MicroSD Card Receptacle: http://www.molex.com/molex/products/datasheet.jsp?part=active/5025700893_MEMORY_CARD_SOCKET.xml&channel=Products&Lang=en-US

5.2 Matlab Code

5.2.1 Transmitter

```
% Create a serial port object.
tx = serial('COM3','BaudRate',115200,'Flow

% Create transmitted data
sendData = '01234567890123456789012345678

% Connect to serial.
fopen(tx);

% Flush the data in the input buffer.
flushinput(tx);

dataSize = 5e4;

for i=1:11
    sendData=[sendData sendData];
end

fwrite(tx,[sendData(1:dataSize) 10]);

% Disconnect from serial.
fclose(tx);

% Clean up all objects.
delete(tx);
clear tx;
```

5.2.2 Receiver

```
% Create a serial port object.
rx = serial('COM4','BaudRate',115200,'FlowControl','hardware','InputBufferSi

% Connect to serial, tx.
fopen(rx);

% Flush the data in the input buffer.
flushinput(rx);

% Communicate with serial, tx.
dataLog = [];
dataSize=5e4;
buffer=0;

tic;
while buffer<dataSize
    buffer=rx.BytesAvailable;
    dataLog = [dataLog;toc,buffer];
end

data=fread(rx,rx.BytesAvailable);
if ~any(find((uint8(sendData(1:dataSize))'==data(1:end-1))==0))
    disp('Data is correct');
end

time = dataLog(find(dataLog(:,2)>=dataSize,1,'first'),1) - ...
   dataLog(find(dataLog(:,2)==0,1,'last'),1);
disp(sprintf('Time to transmit %d Bytes = %d', dataSize, time));
disp(sprintf('Transmission speed = %dbps', dataSize*8/time));

tlog = [tlog;dataSize*8/time];

data=fread(rx,rx.BytesAvailable);

% Disconnect from instrument object, obj1.
fclose(rx);

% Clean up all objects.
delete(rx);
clear rx;
```