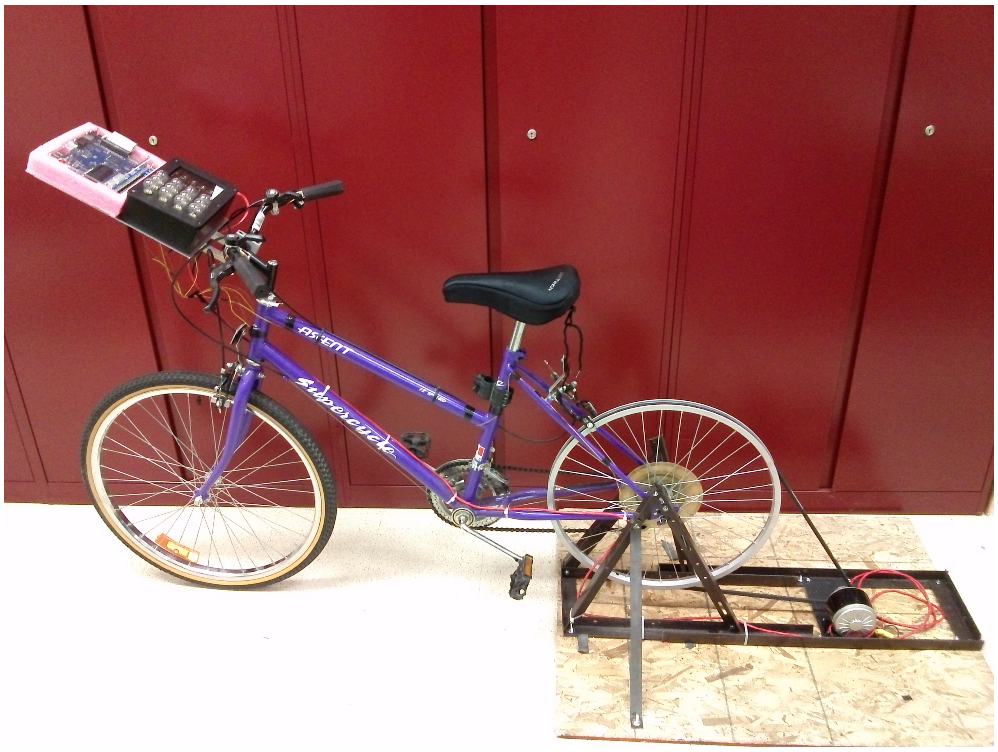
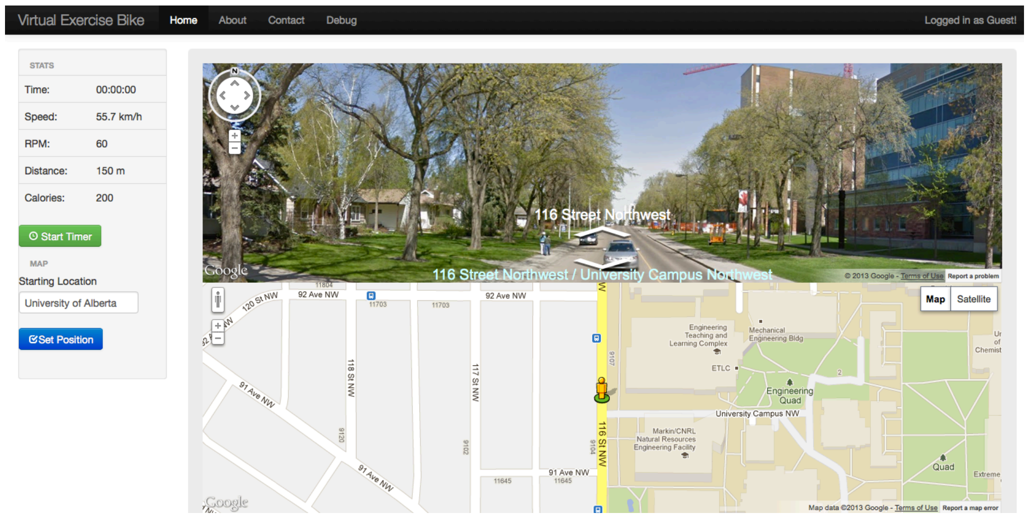
Virtual Exercise Bike

Integrating Google Maps With An Exercise Bike

Robert Miller • Adam Eliason

ECE 492 Final Report • University of Alberta • 12 April, 2013





STATS

Time:	00:00:00
Speed:	55.7 km/h
RPM:	60
Distance:	150 m
Calories:	200

MAP

Starting Location

University of Alberta

ABSTRACT

The Interactive Virtual Exercise Bike allows users to enjoy a simulated bicycle tour in Google Street View (GSV) while working out. The interface implements the Google JavaScript API V3 to display GSV images and maps to the user in a web browser in real time as if they were biking through the actual streets. The web interface allows users to choose their own starting location, provided that it is available in Google's library. The user can ride through the virtual streets in real time with the ability to turn left or right to change direction of the route via digital sensors mounted on the bicycle's front fork. The frequency of image updates is proportional to the speed of the bikers pedaling. Dynamic resistance control has been implemented by connecting the rear wheel of a standard adult bicycle to the rotor of an electric generator and then controlling the active resistance on the load seen by the generator. The resistance, or load, is controlled by switching through a circuit of 12V light bulbs (turning more bulbs on to increase resistance). The system incorporates a simple web server to provide the user interface, including workout statistics, to devices with network capabilities such as a laptop or smartphone.

1. FUNCTIONAL REQUIREMENTS

1.1 Dynamic Resistance Control

The resistance applied to the bicycle's wheel is controlled via the embedded system to simulate an increase or decrease of elevation in the route. Two push buttons are also provided to manually control (up or down) the current resistance setting. By increasing resistance in the generator's circuit, we can control the resulting magnetic field and therefore the torque required to turn the rotor.

1.2 Google Street View (GSV) Integration

By implementing the Google JavaScript API V3, the system has the ability to simulate a tour through a dynamic route using real time controls for navigation. The user is able to choose a starting destination in the simulation. Due to time constraints and the limitation of the API, the images will not be "stitched" together as they are in the traditional GSV.

1.3 Statistics

Using a digital Hall effect sensor to measure RPM of the rear bicycle wheel, the system derives metrics such as speed, calories burned, and distance traveled. The system has the ability to display these real time metrics on both the LCD screen and the web client interface.

1.4 Network Capabilities

The embedded system requires an ethernet connection to a router, or directly to the web client (computer) to display the user interface. The web client itself requires an internet connection in order to interface with Google's servers to retrieve the street view images and maps.

1.5 User Interface

A web browser is used to interface with the embedded system. The interface displays the workout statistics, Google Street View images and maps. The interface allows the user to choose any starting position available in the Google library and supports many of the same features as in a traditional Google Street View.

2. DESIGN AND DESCRIPTION OF OPERATION

2.1 Description of Operation

The Interactive Virtual Exercise Bike uses a standard adult bicycle connected to a 300W electric DC generator powering a circuit of light bulbs to provide physical resistance.

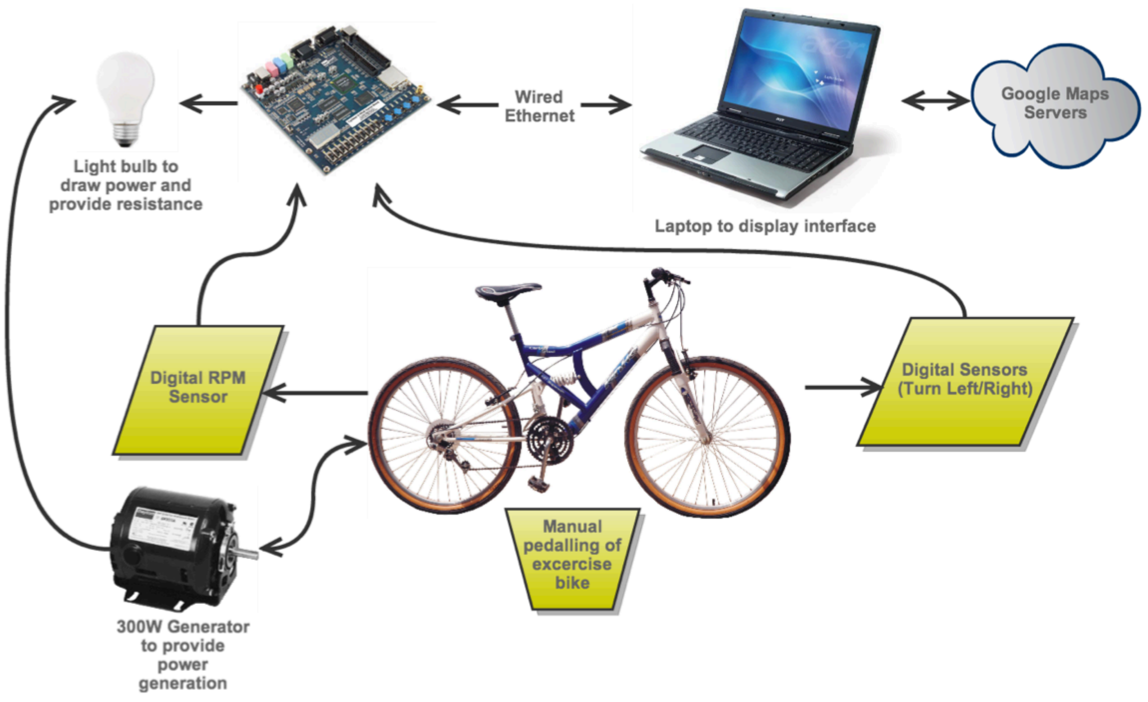
Dynamic resistance control is implemented by connecting the rear wheel of the bicycle to the rotor of an electric generator and then controlling the active resistance on the load seen by the generator. The resistance, or load, is implemented by digitally switching through a parallel connection of 27W light bulbs controlled by turning on or off NMOS transistors via the Altera DE2's GPIO pins. The voltage created by the generator is limited by parallel 12V voltage regulators (two regulators for each light bulb). Two push buttons mounted on the bicycle's handle bars are used to manually step up or down through the 12 resistance levels. The buttons are connected to the GPIO pins and initiate interrupt service request to the system when pressed.

The user initially chooses a starting location for their virtual tour by either dragging the map position in the interface, or typing in an address which is then geocoded and returns the corresponding GSV and maps if available.

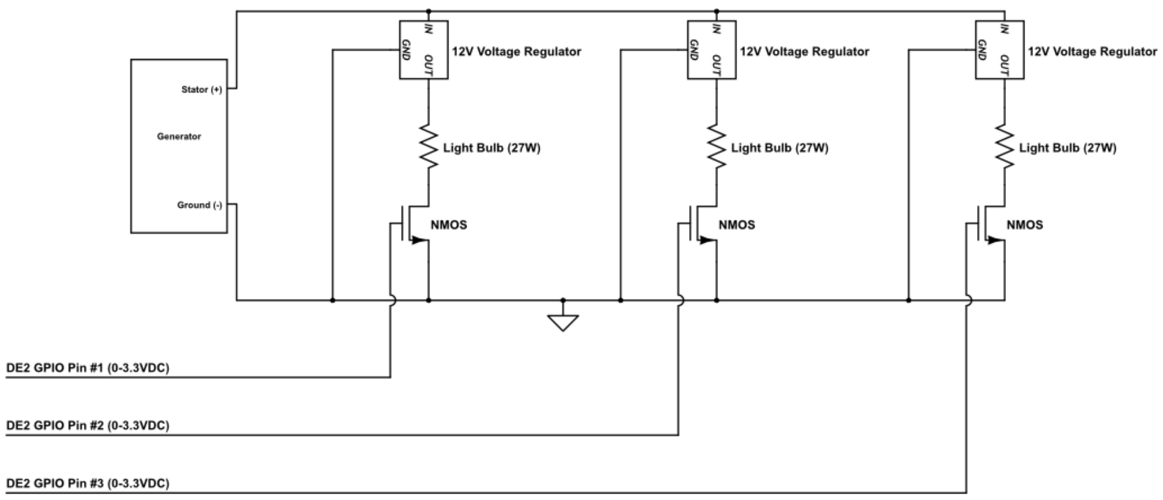
While the user pedals the bicycle, they will see GSV images and maps on a web client as if they were biking down the street in real life. The images are displayed in a flash frame statically (no motion blur effect) due to the limitations of the Google JavaScript API. The images are downloaded live from Google's servers on request of the web client.

The user will be able to control their direction by turning the bicycle's handlebars left or right, which will trip one of two digital Hall effect sensors (one for left, one for right) connected to the Altera DE2 board, initiating a interrupt service request signaling that the user wants to turn.

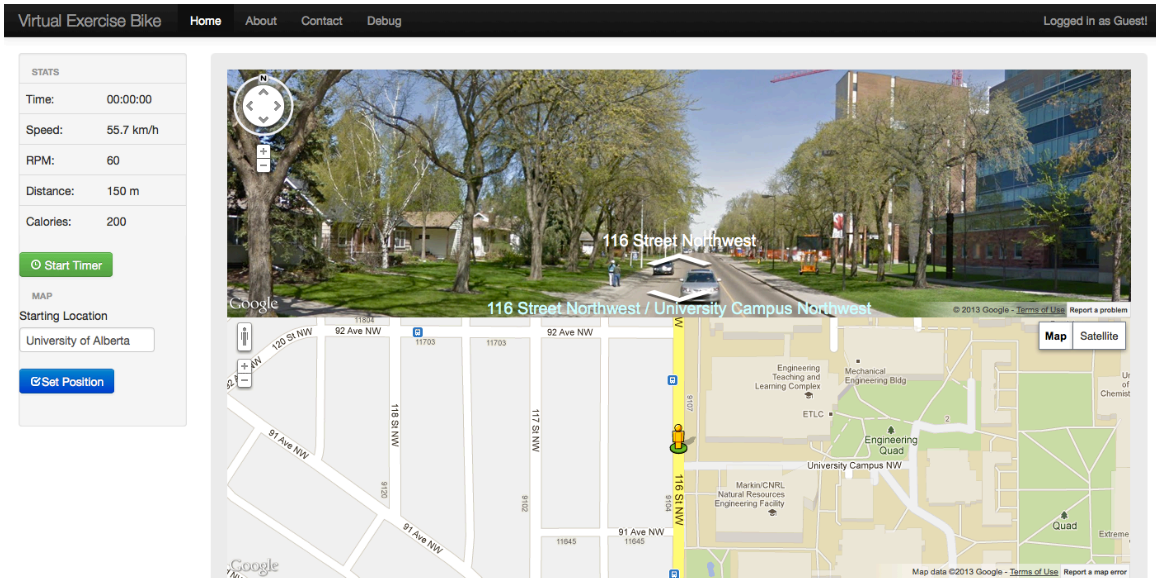
2.2 Design Flow Diagram


Light bulb to draw power and provide resistance
Wired Ethernet
Google Maps Servers
Laptop to display interface
Digital RPM Sensor
Digital Sensors (Turn Left/Right)
Manual pedalling of excercise bike
300W Generator to provide power generation

2.3 Generator Circuit Wiring Diagram


Stator (+)
Generator
Ground (-)
IN
GND
OUT
12V Voltage Regulator
Light Bulb (27W)
NMOS
DE2 GPIO Pin #1 (0-3.3VDC)
DE2 GPIO Pin #2 (0-3.3VDC)
DE2 GPIO Pin #3 (0-3.3VDC)

2.4 User Interface



Time:	00:00:00
Speed:	55.7 km/h
RPM:	60
Distance:	150 m
Calories:	200

Starting Location: University of Alberta

The user interface displays the current workout statistics in a table located in the upper left side of the web page. A button is provide to “Start Timer” which starts the timing of the workout as well as initiates the timed JavaScript function which starts requesting live data from the web server running on the Altera DE2 board.

Also located in the upper left side of the website is a text input box to enter an address or location of the desired starting position. Pressing the “Set Position” button runs a JavaScript function which uses the Google JavaScript API to geocode the supplied address, and then return the corresponding Street View and map images.

The main body of the webpage contains two Flash frames. The upper frame displays the Google Street View representation of the currently set position. The lower frame displays the corresponding top-down map view.

Both Flash frames support the standard features expected in a Google Street View or Map such as dragging location marker, pan and tilting of the view.

3. BILL OF MATERIALS

1. Item: Bicycle
   Part #: N/A
   Specifications: Standard Used Adult Mountain Bike
   Unit Cost: estimated $50
   Quantity: 1
   Total Cost: $50
   Supplier: Borrowed from EE Lab

2. Item: DC Generator
   Part #: N/A, borrowed from ECE stores
   Specifications: 300W, 0-40V, 20 Amps
   Unit Cost: $289
   Quantity: 1
   Total Cost: $289
   Supplier: Pedal Power Generators LLC. http://www.pedalpowergenerator.com/
   Data sheet: http://www.amazon.com/Bicycle-Generator-Power-Pulley-Dynamo/dp/B001WB3TZ4/ref=aag_m_pw_dp?ie=UTF8&m=AKJ98L6A7U0SB#productDetails

3. Item: 12VDC 27W Lights
   Part #: SKU: 8230831 - 2 pc 12V #1156 Automotive Bulbs
   Specifications: 12 V DC, 27 W
   Unit Cost: $1.49/pair
   Quantity: 15 (x2 = 30 bulbs, only used 12)
   Total Cost: $22.35
   Supplier: Princess Auto http://www.princessauto.com/
   Data sheet: http://www.princessauto.com/pal/product/8230831/Bulbs/2-pc-12V-%231156-Automotive-Bulbs

4. Item: 12V Light Sockets
   Part #: N/A
   Specifications: Single contact bayonet socket (BA15s)
   Unit Cost: $3.21
   Quantity: 13
   Total Cost: $41.73

Supplier: Light Bulbs Etc, Inc http://www.amazon.com/Single-Contact-Bayonet-Threaded-Socket/dp/B000XVK3RQ/ref=pd_bxgy_hi_img_y

5. Item: Digital Hall effect sensors for RPM detection and left/right turn detection
   Part #: Digi-Key Part Number 480-5197-ND manufacturer: SS445P
   Specifications:
   Unit Cost: $0.93
   Quantity: 10
   Total Cost: $9.30
   Supplier: Digi-Key http://www.digikey.ca/product-search/en?x=15&y=13&lang=en&site=ca&KeyWords=SS445P
   Data sheet: http://sensing.honeywell.com/index.php?ci_id=45612

6. Item: 10,800 gauss magnet for hall effect sensors
   Part #: Digi-Key Part Number 469-1003-ND
   Specifications: 0.250" Dia x 0.250" H (6.35mm x 6.35mm), 10,800 gauss.
   Unit Cost: $0.75
   Quantity: 2
   Total Cost: $1.50
   Supplier: Digikey http://www.digikey.ca/product-detail/en/1%2F4%22DIA%20X%201%2F4%22THICK/469-1003-ND/555327

7. Item: 12VDC Voltage regulators
   Part #: Digi-Key Part Number KA378R12CTUFS-ND
   Specifications: Voltage input up to 35V, 12V 3A output.
   Unit Cost: $1.16
   Quantity: 35 (used 24)
   Total Cost: $40.60
   Supplier: Digi-Key http://www.digikey.ca/product-detail/en/KA378R12CTU/KA378R12CTUFS-ND/1056184
   Data sheet: http://media.digikey.com/pdf/Data%20Sheets/Fairchild%20PDFs/KA378R12.pdf

8. Item: NMOS Transistors
   Part #: Digi-Key Part Number IRL520PBF-ND

Specifications: power - max 60W, current - continuous drain 9.2A

Unit Cost: $0.92

Quantity: 25 (used 12)

Total Cost: $22.74

Supplier: Digi-Key http://www.digikey.ca/product-detail/en/IRL520PBF/IRL520PBF-ND/811718

Data sheet: http://www.vishay.com/docs/91298/91298.pdf

9. Item: Ethernet Cable

Part #: N/A

Specifications: 3m length

Unit Cost: $5.33

Quantity: 1

Total Cost: $5.33

Supplier: Borrowed from ECE stores

10. Item: 40-pin ribbon cable

Part #: N/A

Specifications: 40 pin, 300V max

Unit Cost: $10

Quantity: 2

Total Cost: $20

Supplier: Borrowed from ECE stores

11. Item: Altera/Terasic DE2 development board

Part #:N/A

Specifications:

Unit Cost: $517.72

Quantity: 1

Total Cost: $517.72

Supplier: Terasic http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=30

Data sheet: ftp://ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2/DE2_Datasheets.zip

12. Item: Project Box
   Part #:1591-BSBK
   Specifications: 8.4"x4.6"x3.2"
   Unit Cost: $16.30
   Quantity: 1
   Total Cost: $16.30
   Supplier: http://www.allelectronics.com/make-a-store/item/1591-BSBK/4.4-X-2.6-X-1.2-BLACK-PLASTIC-CASE/1.html

13. Item: Heat Sinks
   Part #:HS106-ND
   Specifications: HEAT SINK TO-220 .375" COMPACT
   Unit Cost: $00.402
   Quantity: 36
   Total Cost: $14.47
   Supplier:Digi-Key http://www.digikey.ca/product-detail/en/577102B00000G/HS106-ND/108319
   Data sheet: http://www.aavid.com/sites/default/files/literature/Aavid-Board-Level-Heatsinks-Catalog.pdf#page=34

14. Item: Clinch Jones Terminal Block
   Part #: N/A borrowed from ECE stores
   Specifications: 3x2 screw connect
   Unit Cost: $1.91
   Quantity: 1
   Total Cost: $1.91
   Supplier: http://www.galco.com/buy/Cinch-Jones/3-142-P

15. Item:Perf board
   Part #:N/A
   Specifications:
   Unit Cost: $7.50
   Quantity: 1
   Total Cost: $7.50

Supplier:ECE lab supplies

Total Cost: $1,060.45

4. REUSABLE DESIGN UNITS

4.1 Web server application notes (Alex Newcomb - 2012)

We used the application notes and VHDL files provided by the 2012w 'webserver' group to implement a web server.

4.2 SD card application notes (Troy Davis and Caitlin Smart - 2013)

We used the application notes provided by 2013 group 8 to implement the SD card slot. This has been tested and confirmed working however was dropped from our final implementation due to time constraints.

4.3 GPIO inputs (2013 Lab 3)

We reused the components of Lab 3 where we read the output of a frequency generator via a GPIO pin on the board to read the sensors signaling left/right turns and RPM. This has already been tested and confirmed working.

4.4 SD Card and Ethernet IP

We used the VHDL and C files (IP) provided by the manufacturer for the Ethernet and SD Card on the DE2 board.

4.5 Nios II/f

We used the fast version of the Nios II processor in SOPC builder.

5. DATA SHEET

SIGNALS			
Top Level Signals			
Signal Name	**Type**	**Description**	**Location**
KEY[0]	Input (on DE2 board)	Reset DE2 board	PIN_G26
KEY[1]	Input (on DE2 board)	Reset bicycle statistics	PIN_N23
KEY[2]	Input (on DE2 board)	Increase resistance DE2 button	PIN_P23
KEY[3]	Input (on DE2 board)	Decrease resistance DE2 button	PIN_W26
IncreaseLoad	Digital Input (GPIO pin, 3.3VDC)	Increase resistance handlebar button	GPIO 1 Pin 3 (PIN_M23)
DecreaseLoad	Digital Input (GPIO pin, 3.3VDC)	Decrease resistance handlebar button	GPIO 1 Pin 4 (PIN_M19)
TurnLeft	Digital Input (GPIO pin, 3.3VDC)	Turn left Hall effect sensor	GPIO 1 Pin 1 (PIN_K26)
TurnRight	Digital Input (GPIO pin, 3.3VDC)	Turn right Hall effect sensor	GPIO 1 Pin 2 (PIN_KM22)
RPM	Digital Input (GPIO pin, 3.3VDC)	RPM Hall effect sensor	GPIO 1 Pin 0 (PIN_K25)
Ethernet	Input/Output (on DE2 board)	Ethernet signals to communicate with networked device	See Ethernet Signals

DE2 Power	Input	Power to DE2 board (9VDC, 1.3A Advertised, see below for measured)	N/A
CLK50	System Clock	50 MHz clock for system use	PIN_N2
Circuit Signals			
Signal Name	**Type**	**Description**	**Location**
Generator Stator	N/A	Positive output terminal from electric generator connected to light bulb circuit. No voltage/current advertised, measured 35VDC 5.6A at maximum bike pedalling	On generator
Generator Ground	N/A	Negative output terminal (ground) from electric generator connected to light bulb circuit	On generator
LoadController[0 - 11]	Output (GPIO pins, 3.3VDC)	12 output signals to turn light bulb circuit NMOS transistors on or off, turning the light bulbs on or off	GPIO 0 PIN_D25,J22,E26,E25,F24,F23,J21,J20,F25,F26,N18,P18

Ethernet Signals			
Signal Name	**Type**	**Description**	**Location**
ENET_DATA[0-15]	Digital Input/ Output	Ethernet data signals	PIN_A17-20, B17-20, C17-20, D17-20, E17-20
ENET_CLK	Digital Output	Ethernet clock	PIN_B24
ENET_CMD	Digital Output	Command to send to Ethernet so it knows whether to send/ receive	PIN_A21
LCD Signals			
Signal Name	Type	Description	Location
LCD_RW	Digital Output	LCD read/write	PIN_K4
LCD_EN	Digital Output	LCD enable	PIN_K3
LCD_ON	Digital Output	LCD on (always 1 to use LCD)	PIN_L4
LCD_DATA[0-7]	Digital Output	LCD data lines (8 bits wide)	PIN_J1,J2,H1, H2, J4, J3, H4, H3
CIRCUIT DATA			
Parameter	**Value**	**Description**	
Minimum Circuit Voltage	3VDC	Minimum input voltage to circuit to see results (light bulb turn on)	

Maximum Circuit Voltage	35VDC	Maximum rated input voltage to circuit	
Maximum Power Produced by Generator	300W	Maximum power output of generator	
Operating Conditions	-20°C ~ 80°C	Based on voltage regulator maximum conditions which were the most stringent of all components	
DE2 Standby Consmption	9.12 VDC, 0.500A	Standby mode of DE2, voltage (in parallel) and current (in series) measured at same time with device lent to us by Nancy Minderman	

DE2 Full Operation Consumption	9.12VDC, 0.512A	Full operation mode of DE2, voltage (in parallel) and current (in series) measured at same time with device lent to us by Nancy Minderman. This was found by turning on all light bulbs, turning on all sensors, and requesting data via web server from the DE2	
DE2 FPGA Use			
Parameter	**Value**	**Description**	
Logic Elements Used	5634/33,216 (17%)	Gates used in FPGA programming	
Pins Used	230/475 (48%)	Total pins used on DE2 board to connect devices	
Memory Used	286,976/483840 (59%)	FPGA memory used	
Flash Memory Used	2.186MB/4MB (54%)	C code (1.6Mb) & Website Files (586KB)	

6. BACKGROUND READING

Aguirre, M.P. Using FPGA to control current source

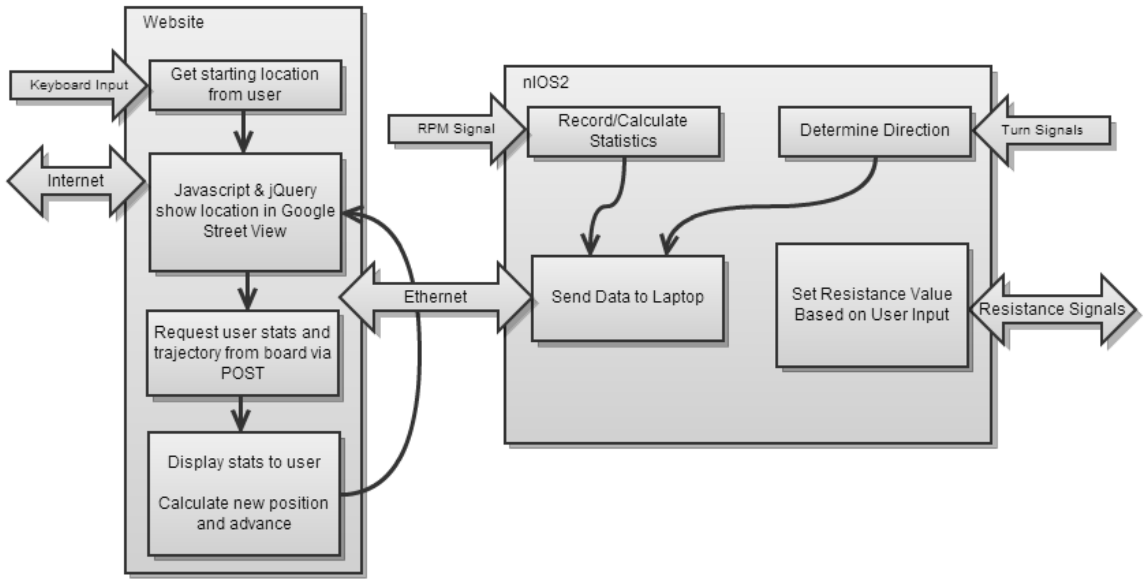
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6138305

Moradewicz, A.J. FPGA based control of series resonant converter for contact-less power supply

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4677292

7. SOFTWARE DESIGN

7.1 Software Flow Diagram


Website
Keyboard Input
Get starting location from user
Internet
Javascript & jQuery show location in Google Street View
Request user stats and trajectory from board via POST
Display stats to user
Calculate new position and advance
Ethernet
nIOS2
RPM Signal
Record/Calculate Statistics
Determine Direction
Turn Signals
Send Data to Laptop
Set Resistance Value Based on User Input
Resistance Signals

7.2 C Files

bike.c

Handles interrupts/tasks associated with the bicycle itself. Handles turning, rpm, speed, calorie calculations as well as sending those variables to the web server task when required.

webserver.c

Creates tasks and handles all interrupts, houses main function. Assigns IP address, initializes SD card and LCD.

http.c

Handles all commands associated with HTTP communication (POST, GET) with client such as laptop.

7.3 Libraries Used

Embedded Processor Core (Installed in Quartus V10.1 Package)

SD Card Interaction (http://www.alterawiki.com/wiki/MP3_Player?GSA_pos=1&WT.oss_r=1&WT.oss=mp3%20player)

Ethernet Controller (http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Webserver/)

Web Server (http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Webserver/)

jQuery (http://api.jquery.com/)

Twitter Bootstrap (http://twitter.github.io/bootstrap/)

Google JavaScript API (https://developers.google.com/maps/documentation/javascript/)

7.4 IO Rates

index.htm requests the data via a POST to the DE2 board every 1.0 seconds (system clock driven)

bike.c requests the current RPM count from the RPM_counter.vhd every 2.0 seconds (interrupt timer driven)

8. TEST PLAN

8.1 Software

Unit Tests

LCD

Can print stats to LCD (success)

Tasks

Activate tasks (success)

Interrupt tasks and resume (success)

Web Server

Respond to POST/GET requests (success)

Serve web site (success)

Integrated Tests

Website

Website pulls data from DE2 board successfully (success)

Website displays data accurately (success)

Tested multiple different locations (success)

Tested turning bike to activate left sensor (success)

Tested turning bike to activate right sensor (success)

Pedaled bike slowly to traverse Google Street View slowly (success)

Pedaled bike fast to traverse Google Street View quickly (success)

Reset stats on board, see stats reset on website (success)

Board

Gets RPM data from VHDL (success)

Reset button resets board (success)

Reset stats button resets stats (success)

Load decrease button decreases load (success)

Load increase button increases load (success)

8.2 Hardware

Unit tests

Sensors

Minimum distance from magnet required to activate sensor: 1.2cm. This was small enough to

Minimum time near magnet to drop to 0V: 3.2us (measured by oscilloscope). This was a short enough time to work with the high speed of the bicycle RPM.

Generator

Determined maximum voltage (no load attached) to be 35V which was within spec of our voltage regulators.

Determined maximum power output of user to be less than 150W, far below the generators maximum power rating of 300W

RPM Counter

Tested rollover after reaching maximum RPM (success)

Tested at frequency of 1MHz, far more than bicycle RPM will generate (success)

Light Bulbs

Can handle maximum of 15V, more than the voltage regulator output of 12V (success)

Voltage Regulators

Dropout voltage of near 0 (even below 12V)

Can handle voltages up to 35V (success)

NMOS Transistors

Are turned on by DE2 GPIO pins (success)

Low voltage drop across them

Integrated Tests

Single Circuit

Tested single light bulb, transistor, and voltage regulator. When transistor off, circuit would not conduct current. When transistor on, circuit could accept 35VDC 2.25A.

Entire Circuit

Tested whole circuit wired to generator. Switching through each circuit individually to ensure transistors turned on and each branch conducted as required (success)

Load decrease button decreases load (success)

Load increase button increases load (success)

Sensors

Turning bicycle left activates left sensors (success)

Turning bicycle right activates right sensor (success)

Spinning bicycle wheel activates RPM sensor at full speed (success)

9. RESULTS OF EXPERIMENTS AND CHARACTERIZATION

9.1 Transistors

Vthreshold (measured) = 1.7 VDC

Want 3.0V input at Vgate, GPIO pins output 3.3V @ 40mA. Place resistor (R) in between to drop voltage.

V=IR

0.3V = (40mA)R

R = 7.5Ohm

9.2 Light Bulbs

P = 27W (given)

V = 12VDC (given)

P=IV -> I = P/V -> I = 27W/12V = 2.25A

V=IR -> R=V/I = 12V/2.25A = 5.333Ohm

9.3 Voltage Regulators

Vdropoff (measured) = ~0V

Vout = 12V

Pdissipation (max) = 15W (given)

Pdissipated = (Vmax - V0) * I^2

15W = (Vmax - 12V) * 2.25A^2

Vmax = 14.96V

9.4 Sensors/Magnets (RPM and turning)

Distance from 10800g magnet to sensor to activate sensor (measured) = 1.2cm

Vin = Vout = 3.3V (sourced from GPIO)

9.5 Generator Output

Vmax (measured) = 35VDC

Pmax (given) = 300W

REFERENCES

[1] Available in Quartus 10.1 software package

[2] Google Street View Documentation: https://developers.google.com/maps/documentation/streetview/

[3] Google JavaScript API Documentation: https://developers.google.com/maps/documentation/javascript/

[4] Altera MP3 Player Example: http://www.alterawiki.com/wiki/MP3_Player?GSA_pos=1&WT.oss_r=1&WT.oss=mp3%20player

[5] 2012 Web Server Application Notes (Hardware): http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Webserver/

[6] 2012 Web Server Application Notes (Software): http://www.ece.ualberta.ca/~elliott/ece492/appnotes/2012w/Webserver/

[7] Twitter Bootstrap Library: http://twitter.github.io/bootstrap/

[8] jQuery Library: http://api.jquery.com/

[9] Health Status Calorie Counter: http://www.healthstatus.com/calculate/cbc

[10] Voltage Regulator Data Sheet: http://media.digikey.com/pdf/Data%20Sheets/Fairchild%20PDFs/KA378R12.pdf

[11] NMOS Transistor Data Sheet: http://www.vishay.com/docs/91298/91298.pdf

[12] Hall Effect Sensor Data Sheet: http://sensing.honeywell.com/index.php?ci_id=45612

APPENDIX

Appendix A: Quick start manual

Note: Please remember to disconnect generator from circuit when not in use. Pedaling the bike and generating power to the light bulb circuit may blow the voltage regulators if none of the NMOS are turned on!

1. How to flash DE2 with this project

Turn DE2 off

Flip switch into "prog" mode (switch is to the left of the LCD on the board)

Turn DE2 on

Open Quartus

Open "program device"

Ensure hardware setup is USB blaster (like always)

Mode: Active Serial Programming

File: niosII_microc_fp.pof

Device: EPCS16

Check the program/configure box by the file name

Click Start (should take 2-5 mins)

Once finished, turn board off, turn switch back to "run", turn board on

Open Eclipse for nios2 via SOPC builder

Regenerate BSP if needed (sopc builder or vhdl changes). If you regenerated the BSP, have to change #define OS_MAX_TASKS from 10 to 15 in system.h file

Go to NIOS2 menu at the top, click "Flash programmer"

May have to reconnect jtag if it cannot find a connection

File, new, choose sopcinfo file. Click OK

Click "Add", go "up" one directory and choose WebServer2.elf

Leave it as no offset in flash, click start, exit when done.

Flash the website files:

Go to NIOS2 menu at the top, click "Flash programmer"

File, new, choose sopcinfo file. Click OK

Click "add", choose ro_zipfs.zip

Add flash offset of 0x100000

PRESS ENTER

press start

When done, press exit

2. Hardware Setup

Use a 40 pin ribbon cable to connect the Altera DE2's GPIO_0 to the light bulb controller circuit's pin header. For orientation, pin 1 is at the top of the circuit.

Use a 40 pin ribbon cable to connect the Altera DE2's GPIO_1 to the pin header on the sensor daughter board located on the bottom side of the mounting plate. For orientation, pin 1 is on the left hand side when looking at the daughter board.

The webserver code currently hard codes the Altera DE2's IP address to 192.168.1.69 which can be changed in the webserver.h file to match your network setup. Look for the #define IPADDR0 - IPADDR3. You can choose to enable DCHP as mentioned in the webserver application notes however this adds extra time to the boot up sequence.

The simplest way to get started is to connect an ethernet cable directly from the DE2 to a laptop. Configure your laptop's NIC to a static ip address that won't conflict with the DE2 (i.e use 192.168.1.10) and set the subnet to 255.255.255.0.

The laptop will require a internet connection via WIFI to reach the Google servers if you wish to have the street views and maps enabled.

Once you power on the Altera DE2, some status messages including the IP address will be displayed on the LCD. Once you see the values for RPM and Speed show up, it is ready to use.

Key_0 on the Altera DE2 is the reset button for the webserver.

Key_1 on the Altera DE2 is the reset button for the current workout stats.

Key_2 on the Altera DE2 is redundant to the right button on the bicycle and will increase the load.

Key_3 on the Altera DE2 is redundant to the right button on the bicycle and will decrease the load.

Connect to the webserver from any web browser by inputing the webserver's IP address in the URL box of the browser. The interface will not start working (posting for stats, advancing or turning) until you press the "Start Timer" button on the screen.

Appendix B: Future Work

B.1 Power Generation

To be self sustaining, the system will recover power generated by the bicycle's wheel during operation. Enough power will be generated to meet the systems own power demands as well as a 5V supply to charge peripherals such as a cell phone. A separate 12VDC to 110VAC voltage inverter can be used to power a laptop or other devices.

B.2 Database Features

By utilizing a database (on board or in a separate server), the system will be able to use create, read, update and delete (CRUD) capabilities to maintain multiple user records. Stored procedures will be used to calculate competitive statistics and leader board generation for the UI. Import and export of user records will be available.

B.3 Competition Modes

An extension to the UI would allow for 'ghost' competition against other user's saved tours or the current user's saved times. Further extensions could allow exchange of data between a central data server in which anyone with an internet connection could compete on the server.

B.4 Social Network Integration

By utilizing API's from Facebook and Twitter, the system could post updates to the users social networks.

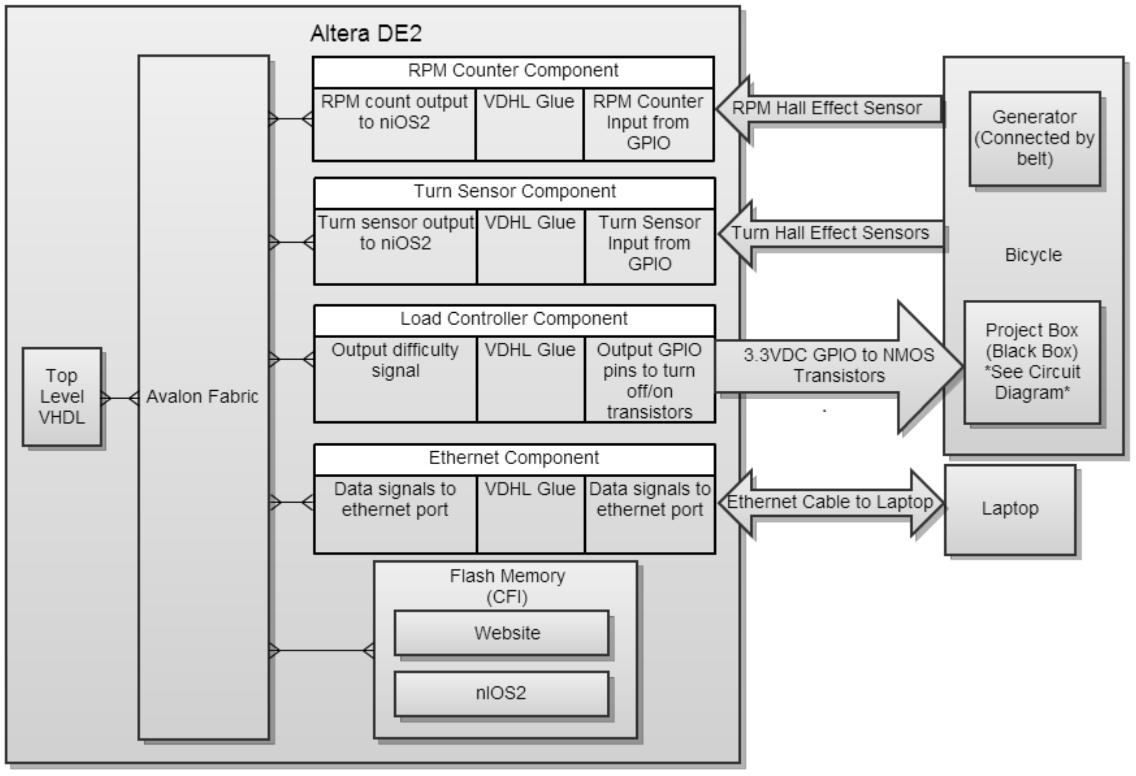
B.5 Calculate Resistance from Elevation

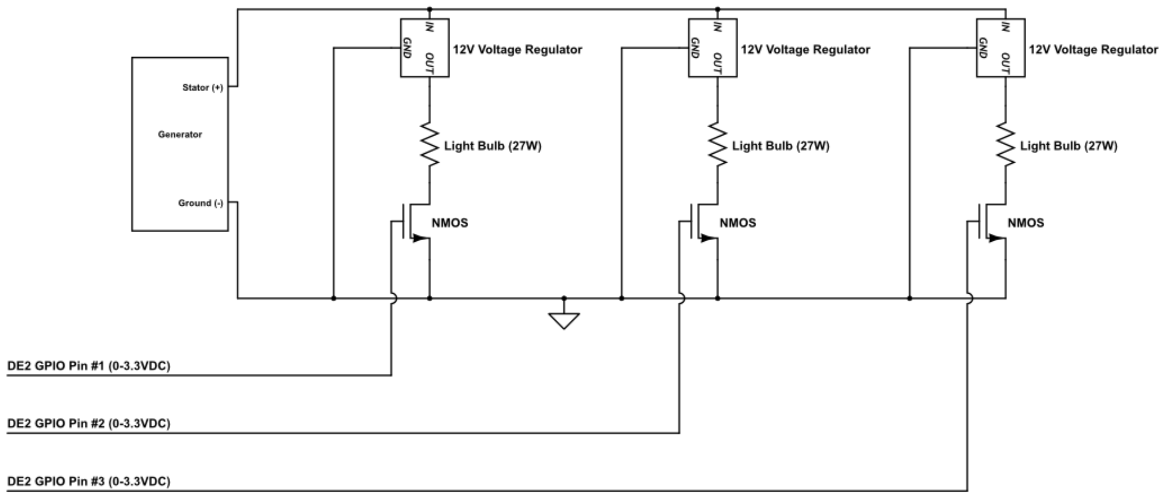
By using the elevation data available through the Google JavaScript API, the system can automatically calculate changes in route elevation and adjust the resistance to the rear wheel accordingly.

B.6 Native Application for Mobile Devices

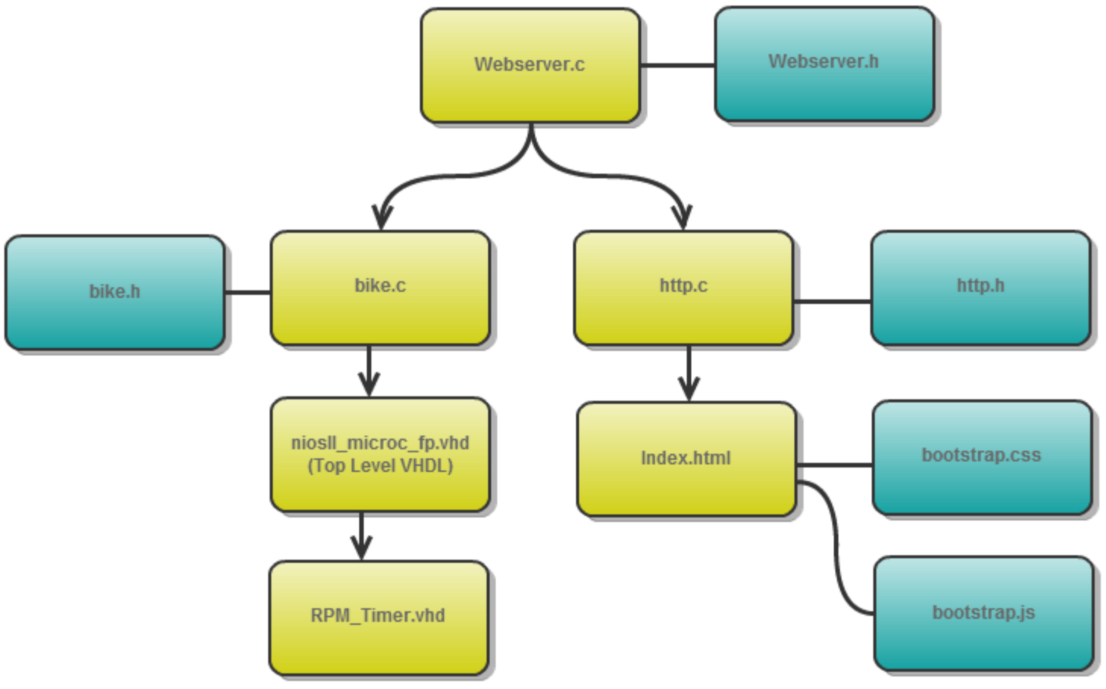
The system currently uses the Twitter Bootstrap CSS to accommodate (responsive webpage) mobile devices. An additional XML or JSON feed could be made available to serve the required data that would make it possible to build a native application for Android or iOS devices.

Appendix C: Hardware documentation


Altera DE2
Top Level VHDL
Avalon Fabric
RPM Counter Component
RPM count output to niOS2
VDHL Glue
RPM Counter Input from GPIO
Turn Sensor Component
Turn sensor output to niOS2
VDHL Glue
Turn Sensor Input from GPIO
Load Controller Component
Output difficulty signal
VDHL Glue
Output GPIO pins to turn off/on transistors
Ethernet Component
Data signals to ethernet port
VDHL Glue
Data signals to ethernet port
Flash Memory (CFI)
Website
nIOS2
RPM Hall Effect Sensor
Turn Hall Effect Sensors
3.3VDC GPIO to NMOS Transistors
Ethernet Cable to Laptop
Generator (Connected by belt)
Bicycle
Project Box (Black Box) *See Circuit Diagram*
Laptop


Stator (+)
Generator
Ground (-)
IN
GND
OUT
12V Voltage Regulator
Light Bulb (27W)
NMOS
DE2 GPIO Pin #1 (0-3.3VDC)
DE2 GPIO Pin #2 (0-3.3VDC)
DE2 GPIO Pin #3 (0-3.3VDC)

Appendix D: Source Code



D.1 Index.html

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="utf-8">
        <title>G2 Virtual Exercise Bike</title>
        <meta name="viewport" content="width=device-width, inital-scale=1.0">
        <meta name="description" content="">
        <meta name="author" content="">
        <!-- Bootstrap -->
        <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet" media="screen">
        <!--<link href="http://twitter.github.com/bootstrap/assets/css/bootstrap.css"
rel="stylesheet">-->
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></
script>
```

```
        <!-- Derp, Styles, Derp -->
        <style type="text/css">
                body {
                        padding-top: 60px;
                padding-bottom: 40px;
                        }
                .sidebar-nav {
                        padding: 9px 0;
                }
                @media (max-width: 980px) {
        /* Enable use of floated navbar text */
                .navbar-text.pull-right {
                        float: none;
                        padding-left: 5px;
                        padding-right: 5px;
                }
                }
        html { height: 100% }
    <!-- // #map-canvas { height: 100% } -->
        #map-canvas {
                                width: auto;
                                height: 317px;
                                margin-right: 1px;
                                overflow: hidden;
                        }
                        #pano img {
  border: none !important;
  max-width: none !important;
}
#map-canvas img {
 max-width: none;
}
```

```
#map-canvas label {
 width: auto; display:inline;
}
   </style>
   <!-- Start Google Maps API V3 -->
   <script type="text/javascript"
            src="https://maps.googleapis.com/maps/api/js?
key=AIzaSyBHmmDUfEkeokGweMQuvU2N7CY5v8Azv_c&sensor=false">
   </script>
   <script type="text/javascript">

    var lat = 53.527031;

 var lng = -113.530666;

 //for address to lat long resolution

 var geocoderService;

 var map;

 var panorama;

 var elevationService;



$(document).ready(function () {

       // put all your jQuery goodness in here.



       $("#startTimerBtn").click(function () {

          //alert("timer button clicked");

          startClock();

          updateStats();
});

$("#press").click(function () {

  //alert("button1 clicked");

  var newloc = new google.maps.LatLng(53.529031, -113.530666);

  panorama.setPosition(newloc);

});


$("#press2").click(function () {

//alert("geo button clicked");

  codeAddress();

});


$("#move").click(function () {

  //          alert("move button clicked");

  moveMap();

});


$("#turnRight").click(function () {

  //          alert("turn Right button clicked");

  turnRight();

});
```

```
$("#turnLeft").click(function () {
  //          alert("turn Left button clicked");
  turnLeft();
});


//To prevent Enter from submitting forms
$('form input').keydown(function (event) {
  if (event.keyCode == 13) {
    event.preventDefault();
    return false;
  }
});


$("#testStats").click(function () {
//alert('test stats clicked');
                    /* Send the data using post */
                    var posting = $.post("/distance");
                    /* Put the results in a div */
                    posting.done(function(xml) {
                            var distance = $(xml).find('distance').first().text();
                            //alert(distance);
                            $( "#distance" ).empty().append( distance );
});

                var speed = $(xml).find('speed').first().text();

                $( "#speed" ).empty().append( speed );

                var rpm = $(xml).find('rpm').first().text();

                $( "#rpm" ).empty().append( rpm );

                var calories = $(xml).find('calories').first().text();

                $( "#calories" ).empty().append( calories );

                var direction = $(xml).find('direction').first().text();

                // 0 nothing, 1 go forward

                if (direction == 1) {

                        moveMap();

                }

                var heading = $(xml).find('heading').first().text();

                //0 nothing, 1 left, 2 right

                if (heading == 1) {

                        turnLeft();

                }

                else if (heading == 2) {

                        turnRight();

                }


        });

});
```

```
        function updateStats() {
//alert('test stats clicked');
                /* Send the data using post */
                var posting = $.post("/distance");
                /* Put the results in a div */
                posting.done(function(xml) {
                        var distance = $(xml).find('distance').first().text();
                        //alert(distance);
                        $( "#distance" ).empty().append( distance );
                        var speed = $(xml).find('speed').first().text();
                        $( "#speed" ).empty().append( speed );
                        var rpm = $(xml).find('rpm').first().text();
                        $( "#rpm" ).empty().append( rpm );
                        var calories = $(xml).find('calories').first().text();
                        $( "#calories" ).empty().append( calories );
                        var direction = $(xml).find('direction').first().text();
                        // 0 nothing, 1 go forward
                        if (direction == 1) {
                                moveMap();
                        }
                        var heading = $(xml).find('heading').first().text();
```

```
                                        //0 nothing, 1 left, 2 right
                                        if (heading == 1) {
                                                turnLeft();
                                        }
                                        else if (heading == 2) {
                                                turnRight();
                                        }


                                });
        setTimeout(updateStats, 1000);
      }


   });
function initialize() {
   geocoderService = new google.maps.Geocoder();
   elevationService = new google.maps.ElevationService();
   var uofa = new google.maps.LatLng(lat, lng);
   var mapOptions = {
      center: uofa,
      zoom: 17,
      mapTypeId: google.maps.MapTypeId.ROADMAP
   };
```

```
    map = new google.maps.Map(document.getElementById("map-canvas"), mapOptions);

    var panoramaOptions = {

        addressControl: false,

        position: uofa,

        pov: {

            heading: 0,

            pitch: 5

        }

    };

    panorama = new google.maps.StreetViewPanorama(document.getElementById('pano'),
panoramaOptions);

    map.setStreetView(panorama);



}

google.maps.event.addDomListener(window, 'load', initialize);



/**

 * Set Starting class

 */

function setStart() {

    //alert("set start");

    var startLoc = document.getElementById("startPos").value;
```

```
  //var to = document.getElementById("to").value;

  //directions.load("from: " + from + " to: " + to, { preserveViewport: true, getSteps: true });

  var newloc = new google.maps.LatLng(53.529031, -113.530666);

  panorama.setPosition(newloc);

}


function codeAddress() {

//alert("called codeaddress");

  var address = document.getElementById("startPos").value;

  geocoderService.geocode({

    'address': address

  }, function (results, status) {

    if (status == google.maps.GeocoderStatus.OK) {

      map.setCenter(results[0].geometry.location);

      var marker = new google.maps.Marker({

        map: map,

        position: results[0].geometry.location

      });

      //alert("setting pano");

      panorama.setPosition(results[0].geometry.location);

    } else {

      alert("Geocode was not successful for the following reason: " + status);
      }

    });

}


function moveMap() {

  //  alert("moving map");

    var currentLoc = panorama.getPosition();

    var currentPov = panorama.getPov();

    var newLoc = google.maps.geometry.spherical.computeOffset(currentLoc, 10,
currentPov.heading);

   // alert(currentLoc);

   // alert(newLoc);

    panorama.setPosition(newLoc);

}


function turnRight() {

    //alert("turning Right");

    var currentPov = panorama.getPov();

    //alert("current heading: " + currentPov.heading);

    var newPov =  currentPov;

    newPov.heading = newPov.heading + 45;

    //alert("New heading: " + newPov.heading);
    panorama.setPov(newPov);

}


function turnLeft() {

    //alert("turning Left");

    var currentPov = panorama.getPov();

    //alert("current heading: " + currentPov.heading);

    var newPov =  currentPov;

    newPov.heading = newPov.heading - 45;

    //alert("New heading: " + newPov.heading);

    panorama.setPov(newPov);

}


//For timer

var startTime;

function startClock()

{

$("#startTimerBtn").empty().append('<i class="icon-time icon-white"></i> Reset Timer');

startTime = new Date();

updateClock();


}
```

```
function updateClock() {
  var timeField = $("#clock");
  var currentTime = new Date();
  var difference =   currentTime - startTime;
  timeField.html(MillisecondsToDuration(difference));
  setTimeout(updateClock, 1000);
}
function MillisecondsToDuration(n) {
  var hms = "";

  var dtm = new Date();
  dtm.setTime(n);
  var h = "000" + Math.floor(n / 3600000);

  var m = "0" + dtm.getMinutes();
  var s = "0" + dtm.getSeconds();


 hms = h.substr(h.length-2) + ":" + m.substr(m.length-2) + ":";
 hms += s.substr(s.length-2);
  return hms;
}
```

```
  </script>

  <!-- END Google Maps API V3 -->

 </head>

 <body onload="initialize()">

         <div class="navbar navbar-inverse navbar-fixed-top">

    <div class="navbar-inner">

     <div class="container-fluid">

      <button type="button" class="btn btn-navbar" data-toggle="collapse" data-
target=".nav-collapse">

       <span class="icon-bar"></span>

       <span class="icon-bar"></span>

       <span class="icon-bar"></span>

      </button>

      <a class="brand" href="#">Virtual Exercise Bike</a>

      <div class="nav-collapse collapse">

       <p class="navbar-text pull-right">

        Logged in as <a href="#" class="navbar-link">Guest!</a>

       </p>

       <ul class="nav">

        <li class="active"><a href="index.html">Home</a></li>
```

```
          <li><a href="about.html">About</a></li>

          <li><a href="contact.html">Contact</a></li>

          <li><a href="debug.html">Debug</a><li>

        </ul>

      </div><!--/.nav-collapse -->

    </div>

  </div>

</div>


<div class="container-fluid">

  <div class="row-fluid">

    <div class="span2">

      <div class="well sidebar-nav">

        <ul class="nav nav-list">

          <li class="nav-header">Stats</li>

        </ul>

          <table class="table table-striped">

            <tbody>

                    <tr>

                            <td>Time:</td>

                            <td id="clock">00:00:00</td>

              </tr>
            <tr>
                  <td>Speed:</td>
                  <td id="speed">55.7 km/h</td>
            </tr>
            <tr>
                  <td>RPM:</td>
                  <td id="rpm">60</td>
            </tr>
            <tr>
                  <td>Distance:</td>
                  <td id="distance">150 m</td>
            </tr>
            <tr>
                  <td>Calories:</td>
                  <td id="calories">200</td>
            </tr>
      </tbody>
     </table>
     <button type="button" id="startTimerBtn" class="btn btn-success"><i class="icon-
time icon-white"></i> Start Timer</button>
     <button type="button" id="testStats" class="btn btn-success">Test Stats</button>
     <div>
```

```
<button type="button" id="move" class="btn btn-success">move</button>

</div>

<div>

  <button type="button" id="turnRight" class="btn btn-info">turn Right</button>

</div>

<div>

  <button type="button" id="turnLeft" class="btn btn-info">turn Left</button>

</div>



                        <p></p>

                        <ul class="nav nav-list">

<li class="nav-header">Map</li>

</ul>

<form class="form-vertical" id="registerHere" method='post' action='/distance'>

                        <fieldset>

        <div class="control-group">

                                       <label class="control-label">Starting
Location</label>

                <div class="controls">

                        <!--<div class="input"><input id="from" size="30"
value="University of Alberta"/></div>-->

                        <!--<input type="text" class="input-large" id="startPos1"
value="University of Alberta"/>-->
```

```
                                <input type="text" class="input-medium" id="startPos"
value="University of Alberta"/>

                        </div>

                        </div>

                        <div class="control-group">

                                <label class="control-label"></label>

                                        <div class="controls">

                                                <a class="btn btn-primary"
id="press2"><i class="icon-check icon-white"></i>Set Position</a>

                                        </div>

                                </div>

                </fieldset>

                        </form>



     </div><!--/.well -->

    </div><!--/span-->

    <div class="span10">

     <div class="hero-unit" style="padding: 20px;">

      <div class="row-fluid">

      </div>

                        <div id="Container1">

        <div id="pano" style="width: auto; height: 300px;"></div>

        <div id="map-canvas"></div>
```

```
 </div>

</div>

<div class="row-fluid">

 <div class="span4">

  <h2>All Time Stats</h2>

  <table class="table table-striped">

   <thead>

           <tr>

                   <th>Stat</th>

                   <th>Value</th>

           </tr>

   </thead>

   <tbody>

           <tr>

                   <td>Fastest Speed</td>

                   <td>55.7 km/h</td>

           </tr>

           <tr>

                   <td>Calories</td>

                   <td>7000</td>

           </tr>

           <tr>
```

```
                <td>Total Distance</td>

                <td>10,000 km</td>

            </tr>

            <tr>

                <td>Total Time</td>

                <td>22:46:32</td>

            </tr>

  </tbody>

  </table>

</div><!--/span-->

<div class="span8">

 <h2>Leader Boards</h2>

 <table class="table table-striped">

  <thead>

            <tr>

                <th>Stat</th>

                <th>Value</th>

                <th>User</th>

            </tr>

  </thead>

  <tbody>

            <tr>
```

```
                <td>Fastest Speed</td>

                <td>55.7 km/h</td>

                <td>YOU!</td>

            </tr>

            <tr>

                <td>Calories</td>

                <td>7000</td>

                <td>YOU!</td>

            </tr>

            <tr>

                <td>Total Distance</td>

                <td>15,000 km</td>

                <td>Jimmy69</td>

            </tr>

            <tr>

                <td>Total Time</td>

                <td>33:26:19</td>

                <td>Rosie_Cheeks</td>

            </tr>

        </tbody>

        </table>

      </div><!--/span-->
```

```
        </div><!--/row-->
       </div><!--/span-->
     </div><!--/row-->
     <hr>
     <footer>
       <p>&copy; ECE492 G2 @ 2013</p>
     </footer>
   </div><!--/.fluid-container-->
   <!-- Le javascript
   ================================================== -->
   <!-- Placed at the end of the document so the pages load faster -->
                       <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></
script>
                       <!--<script src="http://code.jquery.com/jquery.js"></script>-->
                       <script src="bootstrap/js/bootstrap.min.js"></script>
                       <!--<script src="bootstrap/js/jquery.js"></script>-->
 </body>
</html>
```

D.2 Bike.C

```
/*
* File: bike.c                                        *
*                                                    *

* Functions/variables for everything to do with bicycle

* Adam Eliason and Robert Miller

* Some parts modified from Web-server demo

*/



/* MicroC/OS-II definitions */

#include "includes.h"

#include <stdio.h>

#include <errno.h>

#include <ctype.h>

#include "system.h"

#include "altera_avalon_pio_regs.h"

#include <unistd.h>



/* Web Server definitions */

#include "alt_error_handler.h"

#include "web_server.h"



/* Nichestack definitions */

#include "ipport.h"

#include "libport.h"
```

```
#include "osport.h"

#include "tcpport.h"

#include "net.h"


/* Various other includes */

#include "sd_controller.h" // Added SD controller

#include <string.h> // For string manipulation

#include "io.h" // For IORD IOWR (register reading/writing)

#include "sys/alt_irq.h" // For interrupts/irqs

#include "altera_avalon_timer_regs.h" // For ALTERA_AVALON_TIMER_STATUS_REG

#include "bike.h"


/* Global variables */

extern int distance;

extern float calories;

extern int speed;

extern int rpm3;

extern int heading; // Which way to "point" in Google Street view, 0 to go straight, 1 to go left,
2 to go right

extern int direction; // 0 to stay still, all else move forward


/* Function Declarations */
```

```
int   CalcRPM(int RPM);

int   CalcDistance(int RPM, int Circumference);

float CalcSpeed(int RPM, int circumference);

float CalcCalories(int distance);

void  Init_Load(void);

void  Init_SD(void);

void  IncreaseLoad(void);

void  DecreaseLoad(void);

void  PrintLoad(void);

void  resetHeading(void);

void  resetDirection(void);



/* Task Declarations */

void  RPM_task(void *pdata);

void  LOADINCREASE_task(void *pdata);

void  LOADDECREASE_task(void *pdata);

void  TURNLEFT_task(void *pdata);

void  TURNRIGHT_task(void *pdata);

void  RESET_task(void *pdata);



/* Task Stack Initialization */

OS_STK    RPMTaskStk[TASK_STACKSIZE];
OS_STK   LoadIncreaseTaskStk[TASK_STACKSIZE];

OS_STK   LoadDecreaseTaskStk[TASK_STACKSIZE];

OS_STK   TurnLeftTaskStk[TASK_STACKSIZE];

OS_STK   TurnRightTaskStk[TASK_STACKSIZE];

OS_STK   ResetTaskStk[TASK_STACKSIZE];


/* Returns Distance */

int getDistance(void){

	return(distance);

}


/* Returns Calories */

float getCalories(void){

	return(calories);

}


/* Returns RPM */

int getRPM(void){

	return(rpm3);

}


/* Returns Speed */
```

```
int getSpeed(void){
        return(speed);
}


/* Returns Direction */
char getDirection(void){
        return(direction);
}


/* Returns Heading */
char getHeading(void){
                return(heading);
}


/* Resets heading */
void resetHeading(void){
        heading = 0;
}


/* Resets direction */
void resetDirection(void){
        direction = 0;
```

```
}


/* Initializes load to minimum value */

void Init_Load(void)

{

		IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, 1); // Reset load (LED
tests)

		IOWR_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE, 1); // Reset
load (load controller tests)

}


/* Polls and calculates RPM and Speed data */

void RPM_task(void* pdata)

{

		// Variable Declaration

		FILE* fp; // LCD

		distance = 0, speed = 0, rpm3 = 0, calories = 0; // Reset stats

   int OldRPM = 0,Difference = 0, OldDistance = 0; // Note in this case RPM means
"Revolutions per 2 second interval"

   float circumference = WHEEL_DIAMETER * PI;

   short NewRPM = 0;

   char* ClearLCD = "\x1B[2J"; // "Magic Number" code to clear DE2 LCD provided by
Altera
```

```
   IOWR_ALTERA_AVALON_PIO_DATA(RPM_COUNTER_0_BASE, 0); // zero out rpms


   printf("\nStarting RPM Task");
   OSTimeDlyHMSM(0,0,0,50);// Delay to stop RPM from reading bad value


   while (1)
   {
           INT8U err;
           OSQPend(commQ, 0, &err);// Read from queue (after 2 seconds)


           NewRPM = IORD_ALTERA_AVALON_PIO_DATA(RPM_COUNTER_0_BASE);
           Difference = NewRPM  - OldRPM; // Find difference from current RPM and last
recorded RPM
           OldRPM = NewRPM ; // Record current RPM for next iteration of while loop


           // Calculate Distance & Direction
           distance += CalcDistance(Difference, circumference);


                   if(distance == 0){
                           OldDistance = 0;
                   }
```

```
	if((distance - OldDistance)>5){ // If user has travelled 5 meters, increase Google Street
View frame

		direction = 1; // 1 means move forward

		OldDistance = distance;

	}




	// Calculate Calories

	calories = CalcCalories(distance);


	// Calculate Speed

	speed = CalcSpeed(Difference,circumference);


	// Calculate RPM

	rpm3 = CalcRPM(Difference);


	printf("\nDistance = %dm RPM = %d Calories = %f Speed = %d Direction = %d
Heading = %d",distance, rpm3, calories, speed, direction, heading);


	// Print stats to LCD screen

	fp = fopen (LCD_DISPLAY_NAME, "w"); // Open file stream to LCD

	if (fp!=NULL){// If successful; print to LCD

		fprintf(fp, "%s", ClearLCD);// Clear LCD
```

```
                        fprintf(fp, "RP2S:%d RPM:%d", Difference, rpm3); // Print RPM (Rounds per
second)
                        fprintf(fp, "\nSpeed: %3.2fkm/h",speed);// Print speed
                fclose (fp); // Close file stream to LCD
            }
    }
}



/*Calculates speed from RPM and circumference
 * Speed = RPM * 2 * Circumference * (1km/1000m) * (3600seconds/1hr)
 * Where RPM = (Revolutions/2 seconds) and Circumference = (Wheel diameter * pi)
 * Note RPM is multiplied by 2 to convert from rounds/2seconds to rounds/second */
float CalcSpeed(int RP2S, int circumference)
{
            return((RP2S/2)*circumference*3.6);
}

float CalcCalories(int distance)
{
            // fix to work with int
            return(distance*CALORIES_PER_METER);
```

```
}


/* Calculates actual RPM from measured "RP2S" (Revolutions/2seconds)

 * RPM = RP2S * 30 seconds */

int CalcRPM(int RPM)

{

		return(RPM*30);

}


/* Calculates distance and adds to total */

int CalcDistance(int RPM, int Circumference)

{// fix to work with dist

		return(RPM * Circumference);

}


/* Task that is triggered when left turn interrupt is fired */

void TURNLEFT_task(void* pdata)

{

		printf("\nStarting Left Turn Task");

  while (1)

  {

		INT8U err;
```

```
            heading = 1;// Data to be sent to server to turn left
            OSQPend(leftQ, 0, &err);// Read from queue (interrupt driven)
            printf("\nTurn Left!");
    }
}

/*  */
void TURNRIGHT_task(void *pdata)
{
            printf("\nStarting Right Turn Task");
    while (1)
    {
            INT8U err;
            heading = 2;// Data to be sent to server to turn right
            OSQPend(rightQ, 0, &err);// Read from queue (interrupt driven)
            printf("\nTurn Right!");
    }
}

/*  */
void LOADINCREASE_task(void *pdata)
{
            printf("\nStarting Load Increase Task");
    while (1)
    {
            INT8U err;
            OSQPend(increaseQ, 0, &err);// Read from queue (interrupt driven)
            OSTimeDlyHMSM(0,0,0,50);
            if((IORD_ALTERA_AVALON_PIO_DATA(LOADINCREASE_BASE)==0) | |
(IORD_ALTERA_AVALON_PIO_DATA(LOCAL_LOADINCREASE_BASE)==1)){
            printf("\nIncrease Load!");
            //PrintLoad();
            IncreaseLoad();
            PrintLoad();
            }


            /*
            int test = IORD_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE);
            printf("\nOld Load = %d", test);
            test = test << 1;
            test = test + 1;
            IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE, test);
            test = IORD_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE);
            printf("\nNew Load = %d", test);*/
    }
}


/*  */
void LOADDECREASE_task(void *pdata)
{


            printf("\nStarting Decrease Load Task");
    while (1)
    {
            INT8U err;
            OSQPend(decreaseQ, 0, &err);// Read from queue (interrupt driven)
            OSTimeDlyHMSM(0,0,0,50);
            if((IORD_ALTERA_AVALON_PIO_DATA(LOADDECREASE_BASE)==0) ||
(IORD_ALTERA_AVALON_PIO_DATA(LOCAL_LOADDECREASE_BASE)==1)){
            printf("\nDecrease Load!");
            //PrintLoad(); // This line can be removed for production code
            DecreaseLoad();
            PrintLoad();
            }
    }
```

```
}


/* Reset global variables  */
void RESET_task(void *pdata)
{
        printf("\nStarting Reset Task");
  while (1)
  {
        INT8U err;
        distance = 0;
        rpm3 = 0;
        calories = 0;
        direction = 0;
        heading = 0;
        speed = 0;
        OSQPend(resetQ, 0, &err);// Read from queue (interrupt driven)
        printf("\nResetting Variables!");
  }
}


// Prints current bicycle load (number of light bulbs turned on) to the console
void PrintLoad(void){
```

```
//int load = IORD_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE);

int load = IORD_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE);

int adjustedload = 0;


switch(load){

case 1:

            adjustedload = 1;

            break;

case 3:

            adjustedload = 2;

            break;

case 7:

            adjustedload = 3;

            break;

case 15:

            adjustedload = 4;

            break;

case 31:

            adjustedload = 5;

            break;

case 63:

            adjustedload = 6;
```

```
			break;

	case 127:

			adjustedload = 7;

			break;

	case 255:

			adjustedload = 8;

			break;

	case 511:

			adjustedload = 9;

			break;

	case 1023:

			adjustedload = 10;

			break;

	case 2047:

			adjustedload = 11;

			break;

	case 4095:

			adjustedload = 12;

			break;

	}

	printf("\nCurrent Load = %d/12", adjustedload);// Change to 7 once minimum is set
to 5 bulbs
```

```
}


// Decreases current bicycle load (number of light bulbs turned on). Range of load: 5-12
void DecreaseLoad(void){
	//int load = IORD_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE);
	int load = IORD_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE);
	if(load > MIN_LOAD){
	load = load >> 1;
	IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, load);
	IOWR_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE, load);
	}
}


// Increases current bicycle load (number of light bulbs turned on). Range of load: 5-12
void IncreaseLoad(void){
	//int load = IORD_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE);
	int load = IORD_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE);
	if(load < MAX_LOAD){
	load = load << 1;
	load += 1;
	IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, load);
	IOWR_ALTERA_AVALON_PIO_DATA(LOADCONTROLLER_BASE, load);
```

```
        }

}
```

D.3 http.C

```
**********************************************************************
***********
* Copyright (c) 2006 Altera Corporation, San Jose, California, USA.        *
* All rights reserved. All use of this software and documentation is       *
* subject to the License Agreement located at the end of this file below.    *
**********************************************************************
************
*                                          *
* File: http.c                                  *
*                                          *
* A rough imlementation of HTTP. This is not intended to be a complete      *
* implementation, just enough for a demo simple web server. This example     *
* application is more complex than the telnet serer example in that it uses   *
* non-blocking IO & multiplexing to allow for multiple simultaneous HTTP      *
* sessions.                                     *
*                                          *
* This example uses the sockets interface. A good introduction to sockets    *
* programming is the book Unix Network Programming by Richard Stevens        *
*                                          *

* Please refer to file ReadMe.txt for notes on this software example.        *

**********************************************************************
************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/param.h>
#include <sys/fcntl.h>
#include "sys/alt_alarm.h"
#include "alt_types.h"
#include "http.h"
#include "web_server.h"
#include "ipport.h"
#include "libport.h"
#include "osport.h"
#include "tcpport.h"
```

```
#ifdef DEBUG
  #include alt_debug.h
#else
  #define ALT_DEBUG_ASSERT(a)
#endif /* DEBUG */

/*
 * TX & RX buffers.
 *
 * These are declared globally to prevent the MicroC/OS-II thread from
 * consuming too much OS-stack space
 */
alt_u8 http_rx_buffer[HTTP_NUM_CONNECTIONS][HTTP_RX_BUF_SIZE];
alt_u8 http_tx_buffer[HTTP_NUM_CONNECTIONS][HTTP_TX_BUF_SIZE];

/* Declare upload buffer structure globally. */
struct upload_buf_struct
{
  alt_u8* wr_pos;
  alt_u8* rd_pos;
  alt_u8 buffer[UPLOAD_BUF_SIZE];
} upload_buf;

/* Declare a structure to hold flash programming information. */
struct flash_inf_struct
{
  alt_u8* start;
  int size;
  alt_u8 device[40];
}flash_inf;

/* Function prototypes and external functions.  */

#ifdef RECONFIG_REQUEST_PIO_NAME
extern void trigger_reset();
#endif
extern int ParseSRECBuf(struct flash_inf_struct *flash_info);
int http_find_file();



/*
 * This canned HTTP reply will serve as a "404 - Not Found" Web page. HTTP
 * headers and HTML embedded into the single string.
 */
static const alt_8 canned_http_response[] = {"\
HTTP/1.0 404 Not Found\r\n\
Content-Type: text/html\r\n\
Content-Length: 272\r\n\r\n\
<HTML><HEAD><TITLE>Nios II Web Server Demonstration</TITLE></HEAD>\
<title>NicheStack on Nios II</title><BODY><h1>HTTP Error 404</h1>\
<center><h2>Nios II Web Server Demonstration</h2>\
Can't find the requested file file. \
Have you programmed the flash filing system into flash?</html>\
"};

static const alt_8 canned_response2[] = {"\
HTTP/1.0 200 OK\r\n\
Content-Type: text/html\r\n\
Content-Length: 2000\r\n\r\n\
<HTML><HEAD><TITLE>Nios II Web Server Demonstration</TITLE></HEAD>\
<title>NicheStack on Nios II</title><BODY>\
<center><h2>Nios II Web Server Hardware Report</h2>\
</center>\
"};

static const alt_8 json_http_response[] = {"\
HTTP/1.0 json page\r\n\
Content-Type: text/html\r\n\
Content-Length: 272\r\n\r\n\
<HTML><HEAD><TITLE>Json Demonstration</TITLE></HEAD>\
<title>NicheStack on Nios XXXXXXX</title><BODY><h1>HTTP NOT BAD!</h1>\
<center><h2>Nios II Web Server Demonstration</h2>\
Hi Rob!. \
</html>\
"};


static const alt_8 distance_http_response[] = {"\
HTTP/1.0 XXX Not Found\r\n\
Content-Type: text/html\r\n\
Content-Length: 272\r\n\r\n\
<HTML><HEAD><TITLE>Nios II Web Server Dxxxxxxxxation</TITLE></HEAD>\
<title>NicheStack on Nios II</title><BODY><h1>HTTP Error 404</h1>\
<center><h2>Nios II Web Server Demonstration</h2>\
Can't find the requested file file. \
Have you programmed the flash filing system into flash?</html>\
"};
/*
 * Mapping between pages to post to and functions to call: This allows us to
 * have the HTTP server respond to a POST requset saying "print" by calling
 * a "print" routine (below).
 */
typedef struct funcs
{
  alt_u8*  name;
  void (*func)();
}post_funcs;


struct http_form_data board_funcs;


/*
 * print()
 *
 * This routine is called to demonstrate doing something server-side when an
 * HTTP "POST" command is received.
 */
void print()
```

```
{
 printf("HTTP POST received.\n");
}


/* To sweep or not to sweep?  The sweep form will tell this function what to do.
 */


/* Sends bicycle data to connection via XML */
void sendDistance(http_conn* conn)
{
  printf("\nRequest for data received.");


   int iDistance = getDistance();
   int iRPM = getRPM();
   int iSpeed = getSpeed();
   int iHeading = getHeading();
   int iDirection = getDirection();
   float iCalories = getCalories();



   char cDistance[sizeof(iDistance)], cRPM[sizeof(iRPM)], cCalories[sizeof(iCalories)],
cSpeed[sizeof(iSpeed)];
   char cDirection[sizeof(iDirection)], cHeading[sizeof(iHeading)];
   //char cDirection = getDirection();
   //char cHeading = getHeading();


   printf("Size of dist: %04d  rpm: %04d calories:  %04d speed:  %04d ", sizeof(iDistance),
sizeof(iRPM), sizeof(iCalories), sizeof(iSpeed));
```

```
   sprintf(cDistance, "%04d\0", iDistance);
   sprintf(cRPM, "%04d\0", iRPM);
   sprintf(cCalories, "%04f\0", iCalories);
   sprintf(cSpeed, "%04d\0", iSpeed);
   sprintf(cDirection, "%04d\0", iDirection);
   sprintf(cHeading, "%04d\0", iHeading);

   alt_u8* tx_wr_pos = conn->tx_buffer;
   tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_OK_STRING);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_LENGTH);
         //tx_wr_pos += sprintf(tx_wr_pos, "300\r\n");
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_XML_LENGTH);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_XML);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS_STATS);

   send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);
   alt_8 send_stats1[] ={"<?xml version=\"1.0\" encoding=\"UTF-8\"?
><stats><distance>"};
   alt_8 send_stats2[]    ={"</distance><direction>"};
   alt_8 send_stats3[]    ={"</direction><rpm>"};
   alt_8 send_stats4[]    ={"</rpm><speed>"};
   alt_8 send_stats5[]    ={"</speed><calories>"};
   alt_8 send_stats6[]    ={"</calories><heading>"};
   alt_8 send_stats7[]    ={"</heading></stats>"};
```

```
    send(conn->fd,(void*)send_stats1,strlen(send_stats1),0);
    send(conn->fd,(void*)cDistance,sizeof(cDistance),0);
    send(conn->fd,(void*)send_stats2,strlen(send_stats2),0);
    send(conn->fd,(void*)cDirection,sizeof(cDirection),0);
    send(conn->fd,(void*)send_stats3,strlen(send_stats3),0);
    send(conn->fd,(void*)cRPM,sizeof(cRPM),0);
    send(conn->fd,(void*)send_stats4,strlen(send_stats4),0);
    send(conn->fd,(void*)cSpeed,sizeof(cSpeed),0);
    send(conn->fd,(void*)send_stats5,strlen(send_stats5),0);
    send(conn->fd,(void*)cCalories,sizeof(cCalories),0);
    send(conn->fd,(void*)send_stats6,strlen(send_stats6),0);
    send(conn->fd,(void*)cHeading,sizeof(cHeading),0);
    send(conn->fd,(void*)send_stats7,strlen(send_stats7),0);

    resetHeading(); // Reset heading
    resetDirection(); // Reset direction

}



void sweep(http_conn* conn)
{
  alt_u8* delimiter_token;

  /* Set board_funcs to be off, by default. */

  board_funcs.LED_ON = 0;
```

```
 board_funcs.SSD_ON = 0;


 delimiter_token = strtok(conn->rx_rd_pos, "&");


 while(delimiter_token != NULL)
 {
  if (strstr(delimiter_token, "LED"))
  {
   board_funcs.LED_ON = 1;
  }
  else if (strstr(delimiter_token, "seven"))
  {
   board_funcs.SSD_ON = 1;
  }


  delimiter_token = strtok( NULL, "&" );
 }


 OSMboxPost(board_control_mbox, (void*)&board_funcs);


}


void http_cleanup_lcd_text()
{
 int index;


 /* Step through the board_funcs.LCD_TEXT array looking for chars to replace.
```

```
  * For now, just replace '+'s with space characters.
  */

 for (index = 0; index < 20; ++index)
 {
   if (board_funcs.LCD_TEXT[index] == '+')
   {
     board_funcs.LCD_TEXT[index] = ' ';
   }
 }
}

void lcd_output(http_conn* conn)
{
 alt_u8* delimiter_token;
 alt_u8* temp_pos;

 delimiter_token = strtok(conn->rx_rd_pos, "&");

 while( delimiter_token != NULL )
 {
   if (strstr(delimiter_token, "lcd_text"))
   {
     temp_pos = strstr(delimiter_token, "=");
     ++temp_pos;
     /* LCD_TEXT is limited to 20 characters...limiting string to that size. */
     strncpy( board_funcs.LCD_TEXT, temp_pos, 20 );
```

```
    }
    delimiter_token = strtok( NULL, "&" );
  }

  /* Clean up the HTTP-formatted text string.  For now this just replaces '+'s with spaces. */

  http_cleanup_lcd_text();

  /* Post the updated LCD_TEXT to the board_control_mbox. */
  OSMboxPost(board_control_mbox, (void*)&board_funcs);

}



int http_parse_multipart_header( http_conn* conn )
{
  /* Most of the information is on the first line following the boundary. */
  alt_u8* cr_pos;
  alt_u8* temp_pos;

  /*
   * For now, make the assumption that no multipart headers are split
   * across packets.  This is a reasonable assumption, but not a surety.
   *
   */
  while( (temp_pos = strstr( conn->rx_rd_pos, conn->boundary )) )
  {
```

```
if( strstr( conn->rx_rd_pos, "upload_image" ) )
{
 /* Terminate the received data by going back 5
  * from temp_pos and setting it to NULL.*/
 *(temp_pos-5) = '\0';
 conn->file_upload = 0;
 break;
}
/* Find the end of the content disposition line. */
conn->rx_rd_pos = strstr( conn->rx_rd_pos, "Content-Disposition" );
if( conn->rx_rd_pos == 0 ) return(-1);
cr_pos = strchr( conn->rx_rd_pos, '\r' );
if( cr_pos == 0 ) return(-1);
/* Insert a NULL byte over the second quotation mark. */
*(cr_pos - 1) = '\0';
/* Move rx_rd_pos to end of the line, just beyond the newly
 * inserted NULL.
 */
/* Look for "=" delimiter. */
temp_pos = strchr( conn->rx_rd_pos, '=' );
if( temp_pos == 0 ) return(-1);
/* If second "=" delimiter exists, then parse for conn->filename. */
if( (temp_pos = strchr( (temp_pos+1), '=' )) )
{
                        if( strlen(temp_pos+2) > 256 )
                        {
                                return(-1);
                      }
 strcpy( conn->filename, (temp_pos+2) );
 /*
  * Place rx_rd_pos at the start of the next pertinent line.
  * In this case, skip two lines ahead.
  */
 cr_pos = strchr( (cr_pos+1), '\r');
 if( cr_pos == 0 ) return(-1);
 cr_pos = strchr( (cr_pos+1), '\r');
 if( cr_pos == 0 ) return(-1);
 conn->rx_rd_pos = cr_pos+2;
}
else
{
 /*
  * If no second delimiter, then skip ahead to start of 2nd. line.
  * That will be the start of the flash device name.
  *
  */
 temp_pos = strchr( (cr_pos+1), '\r' );
 conn->rx_rd_pos = temp_pos+2;
 cr_pos = strchr( conn->rx_rd_pos, '\r' );
 *cr_pos = '\0';
 /* Ok, now copy the flash_device string. */
 strcpy( conn->flash_device, conn->rx_rd_pos );
 /* Place rx_rd_pos at the start of the next line. */
 conn->rx_rd_pos = cr_pos+2;
 }
}
return(0);
}

void file_upload(http_conn* conn)
{
int buf_len;
int data_used;
struct upload_buf_struct *upload_buffer = &upload_buf;
struct flash_inf_struct *flash_info = &flash_inf;
/* Look for boundary, parse multipart form "mini" header information if found. */
if( strstr( conn->rx_rd_pos, conn->boundary ) )
{
  if( http_parse_multipart_header( conn ) )
  {
    printf( "multipart-form:  header parse failure...resetting connection!" );
    conn->state = RESET;
  }
}
/* Exception for IE.  It sometimes sends _really_ small initial packets! */
if( strchr( conn->rx_rd_pos, ':' ) )
{
  conn->state = READY;
  return;
}
/* Calculate the string size... */
```

```
buf_len = strlen(conn->rx_rd_pos);
conn->content_received = conn->content_received + buf_len;
/* Copy all the received data into the upload buffer.  */
if ( memcpy( (void*) upload_buffer->wr_pos,
        (void*) conn->rx_rd_pos,
        buf_len ) == NULL )
{
  printf( "ERROR:  memcpy to file upload buffer failed!" );
}
/* Increment the wr_pos pointer to just after the received data.  */
upload_buffer->wr_pos = upload_buffer->wr_pos + buf_len;
conn->rx_rd_pos = conn->rx_rd_pos + buf_len;
/* Reset the buffers after copying the data into the big intermediate
 * buffer.*/
data_used = conn->rx_rd_pos - conn->rx_buffer;
memmove(conn->rx_buffer,conn->rx_rd_pos,conn->rx_wr_pos-conn->rx_rd_pos);
conn->rx_rd_pos = conn->rx_buffer;
conn->rx_wr_pos -= data_used;
memset(conn->rx_wr_pos, 0, data_used);
if ( conn->file_upload == 0 )
{
  printf( "Received a total of %d bytes.\n", conn->content_received );
  /* Insert a NULL character (temporarily).  */
  *upload_buffer->wr_pos = '\0';
  /* Populate flash_info struct... print the buffer size.  */
  flash_info->size = (int) strlen(upload_buffer->buffer);
  printf( "Upload Buffer size = %d.\n", flash_info->size);
```

```
    strcpy( flash_info->device, conn->flash_device );
    flash_info->start = upload_buffer->rd_pos;
    /* Populate the flash_inf struct. */
    //printf( "Here's the Buffer:\n\n%s", upload_buffer->buffer);
    http_find_file(conn);
    conn->close = 1;
  }
  else
  {
    conn->state = READY;
  }
}

/* ProgFlashStub()
 *
 * A thin wrapper around the ProgSRECBuf() function in srec_flash.c.
 *
 */
void ProgFlashStub(http_conn* conn)
{
        struct flash_inf_struct *flash_info = &flash_inf;
        /* Call ParseSRECBuf, with the flash_info argument. */
        ParseSRECBuf( flash_info );
        /* Go find and send the reset_system.html file. */
        http_find_file( conn );
        /* Close the connection. */
        conn->close = 1;
  return;
}



/*
 * The mapping (using our struct defined above) between HTTP POST commands
 * that we will service and the subroutine called to service that POST
 * command.
 */
post_funcs mapping =
{
  "/PRINT",
  print
};

post_funcs sweep_field =
{ "/SWEEP",
  sweep
};

post_funcs lcd_field =
{ "/LCD",
  lcd_output
};

post_funcs upload_field =
{ "/program_flash.html",
 file_upload
};

post_funcs flash_field =
{ "/reset_system.html",
  ProgFlashStub
};

// Added code
post_funcs distance_field =
{ "/distance",
  sendDistance
};

#ifdef RECONFIG_REQUEST_PIO_NAME
post_funcs reset_field =
{
 "/RESET_SYSTEM",
 trigger_reset
};
#endif

/*
 * http_reset_connection()
 *
 * This routine will clear our HTTP connection structure & prepare it to handle
 * a new HTTP connection.
 */
void http_reset_connection(http_conn* conn, int http_instance)
{
  memset(conn, 0, sizeof(http_conn));

  conn->fd = -1;
  conn->state = READY;
  conn->keep_alive_count = HTTP_KEEP_ALIVE_COUNT;

  conn->rx_buffer = (alt_u8 *) &http_rx_buffer[http_instance][0];
  conn->tx_buffer = (alt_u8 *) &http_tx_buffer[http_instance][0];
  conn->rx_wr_pos = (alt_u8 *) &http_rx_buffer[http_instance][0];
  conn->rx_rd_pos = (alt_u8 *) &http_rx_buffer[http_instance][0];
}

/*
 * http_manage_connection()
 *
 * This routine performs house-keeping duties for a specific HTTP connection
 * structure. It is called from various points in the HTTP server code to
 * ensure that connections are reset properly on error, completion, and
 * to ensure that "zombie" connections are dealt with.
 */
void http_manage_connection(http_conn* conn, int http_instance)
{
  alt_u32 current_time = 0;
```

```
/*
 * Keep track of whether an open connection has timed out. This will be
 * determined by comparing the current time with that of the most recent
 * activity.
 */
if(conn->state == READY || conn->state == PROCESS || conn->state == DATA)
{
  current_time = alt_nticks();

  if( ((current_time - conn->activity_time) >= HTTP_KEEP_ALIVE_TIME) && conn-
>file_upload != 1 )
  {
    conn->state = RESET;
  }
}

/*
 * The reply has been sent. Is is time to drop this connection, or
 * should we persist? We'll keep track of these here and mark our
 * state machine as ready for additional connections... or not.
 *  - Only send so many files per connection.
 *  - Stop when we reach a timeout.
 *  - If someone (like the client) asked to close the connection, do so.
 */
if(conn->state == COMPLETE)
{
  if(conn->file_handle != NULL)
  {
    fclose(conn->file_handle);
  }

  conn->keep_alive_count--;
  conn->data_sent = 0;

  if(conn->keep_alive_count == 0)
  {
    conn->close = 1;
  }

  conn->state = conn->close ? CLOSE : READY;
}

/*
 * Some error occured. http_reset_connection() will take care of most
 * things, but the RX buffer still needs to be cleared, and any open
 * files need to be closed. We do this in a separate state to maintain
 * efficiency between successive (error-free) connections.
 */
if(conn->state == RESET)
{
  if(conn->file_handle != NULL)
  {
    fclose(conn->file_handle);
  }
```

```
    memset(conn->rx_buffer, 0, HTTP_RX_BUF_SIZE);
    conn->state = CLOSE;
  }

  /* Close the TCP connection */
  if(conn->state == CLOSE)
  {
    close(conn->fd);
    http_reset_connection(conn, http_instance);
  }
}

/*
 * http_handle_accept()
 *
 * A listening socket has detected someone trying to connect to us. If we have
 * any open connection slots we will accept the connection (this creates a
 * new socket for the data transfer), but if all available connections are in
 * use we'll ignore the client's incoming connection request.
 */
int http_handle_accept(int listen_socket, http_conn* conn)
{
  int ret_code = 0, i, socket, len;
  struct sockaddr_in  rem;

  len = sizeof(rem);
```

```
/*
 * Loop through available connection slots to determine the first available
 * connection.
 */
for(i=0; i<HTTP_NUM_CONNECTIONS; i++)
{
  if((conn+i)->fd == -1)
  {
    break;
  }
}

/*
 * There are no more connection slots available. Ignore the connection
 * request for now.
 */
if(i == HTTP_NUM_CONNECTIONS)
  return -1;

if((socket = accept(listen_socket,(struct sockaddr*)&rem,&len)) < 0)
{
  fprintf(stderr, "[http_handle_accept] accept failed (%d)\n", socket);
  return socket;
}

(conn+i)->fd = socket;
```

```
  (conn+i)->activity_time = alt_nticks();


  return ret_code;
}


/*
 * http_read_line()
 *
 * This routine will scan the RX data buffer for a newline, allowing us to
 * parse an in-coming HTTP request line-by-line.
 */
int http_read_line(http_conn* conn)
{
  alt_u8* lf_addr;
  int ret_code = 0;


  /* Find the Carriage return which marks the end of the header */
  lf_addr = strchr(conn->rx_rd_pos, '\n');


  if (lf_addr == NULL)
  {
    ret_code = -1;
  }
  else
  {
    /*
     * Check that the line feed has a matching CR, if so zero that
     * else zero the LF so we can use the string searching functions.
     */
    if ((lf_addr > conn->rx_buffer) && (*(lf_addr-1) == '\r'))
    {
      *(lf_addr-1) = 0;
    }

    *lf_addr = 0;
    conn->rx_rd_pos = lf_addr+1;
  }

  return ret_code;
}

/* http_process_multipart()
 *
 * This function parses and parses relevant "header-like" information
 * from HTTP multipart forms.
 *   - Content-Type, Content-Disposition, boundary, etc.
 */
int http_parse_type_boundary( http_conn* conn,
                              char* start,
                              int len )
{
  char* delimiter;
  char* boundary_start;
  char line[HTTP_MAX_LINE_SIZE];
  /* Copy the Content-Type/Boundary line. */
  if( len > HTTP_MAX_LINE_SIZE )
  {
    printf( "process headers:  overflow content-type/boundary parsing.\n" );
    return(-1);
  }
  strncpy( line, start, len );
  /* Add a null byte to the end of it. */
  *(line + len) = '\0';
  /* Get the Content-Type value. */
  if( (delimiter = strchr( line, ';' )) )
  {
    /* Need to parse both a boundary and Content-Type. */
    boundary_start = strchr( line, '=' ) + 2;
    strcpy( conn->boundary, boundary_start);
    /* Insert a null space in place of the delimiter. */
    *delimiter = '\0';
    /* First part of the line is the Content-Type. */
    strcpy( conn->content_type, line);
  }
  else
  {
    strcpy( conn->content_type, line );
  }
  return 0;
}
```

```
/*
 * http_process_headers()
 *
 * This routine looks for HTTP header commands, specified by a ":" character.
 * We will look for "Connection: Close" and "Content-length: <len>" strings.
 * A more advanced server would parse far more header information.
 *
 * This routine should be modified in the future not to use strtok() as its
 * a bit invasive and is not thread-safe!
 *
 */
int http_process_headers(http_conn* conn)
{
 alt_u8* option;
 alt_u8* cr_pos;
 alt_u8* ct_start;
 alt_u8* orig_read_pos = conn->rx_rd_pos;
 alt_u8* delimiter_token;
 alt_u8 temp_null;
 alt_u8* boundary_start;
 int ct_len;
 int opt_len;


 /*
  * A boundary was found.  This is a multi-part form
  * and header processing stops here!
  *
  */
 if( (conn->boundary[0] == '-') && (conn->content_length > 0) )

 {
  boundary_start = strstr( conn->rx_rd_pos, conn->boundary );
  //conn->rx_rd_pos = boundary_start + strlen(conn->boundary);
  return -1;
 }
 /* Skip the next section we'll chop with strtok(). Perl for Nios, anyone? */
 else if( (delimiter_token = strchr(conn->rx_rd_pos, ':')) )
 {
  conn->rx_rd_pos = delimiter_token + 1;
  conn->content_received = conn->rx_rd_pos - conn->rx_buffer;
 }
 else
 {
  return -1;
 }

 option = strtok(orig_read_pos, ":");

 if(stricmp(option,"Connection") == 0)
 {
  temp_null = *(option + 17);
  *(option + 17) = 0;
  if(stricmp((option+12), "close") == 0)
  {
    conn->close = 1;
  }
  *(option + 17) = temp_null;
 }
 else if (stricmp(option, "Content-Length") == 0)
 {
  conn->content_length = atoi(option+16);
  //printf( "Content Length = %d.\n", conn->content_length );
 }
 /* When getting the Content-Type, get the whole line and throw it
  * to another function.  This will be done several times.
  */
 else if (stricmp(option, "Content-Type" ) == 0)
 {
  /* Determine the end of line for "Content-Type" line. */
  cr_pos = strchr( conn->rx_rd_pos, '\r' );
  /* Find the length of the string. */
  opt_len = strlen(option);
  ct_len = cr_pos - (option + opt_len + 2);
  /* Calculate the start of the string. */
  ct_start = cr_pos - ct_len;
  /* Pass the start of the string and the size of the string to
   * a function.
   */
```

```
    if( (http_parse_type_boundary( conn, ct_start, ct_len ) < 0) )
    {
      /* Something failed...return a negative value. */
      return -1;
    }
  }
  return 0;
}

/*
 * http_process_request()
 *
 * This routine parses the beginnings of an HTTP request to extract the
 * command, version, and URI. Unsupported commands/versions/etc. will cause
 * us to error out drop the connection.
 */
int http_process_request(http_conn* conn)
{
  alt_u8* uri = 0;
  alt_u8* version = 0;
  alt_u8* temp = 0;
  if( (temp = strstr(conn->rx_rd_pos, "GET")) )
  {
    conn->action = GET;
    conn->rx_rd_pos = temp;
  }
  else if( (temp = strstr(conn->rx_rd_pos, "POST")) )
```

```
{
  conn->action = POST;
  conn->rx_rd_pos = temp;
}
else
{
  fprintf(stderr, "Unsupported (for now) request\n");
  conn->action = UNKNOWN;
  return -1;
}

/* First space char separates action from URI */
if( (conn->rx_rd_pos = strchr(conn->rx_rd_pos, ' ')) )
{
  conn->rx_rd_pos++;
  uri = conn->rx_rd_pos;
}
else
{
  return -1;
}

/* Second space char separates URI from HTTP version. */
if( (conn->rx_rd_pos = strchr(conn->rx_rd_pos, ' ')) )
{
  *conn->rx_rd_pos = 0;
  conn->rx_rd_pos++;
```

```
    version = conn->rx_rd_pos;
  }
  else
  {
    return -1;
  }

  /* Is this an HTTP version we support? */
  if ((version == NULL) || (strncmp(version, "HTTP/", 5) != 0))
  {
    return -1;
  }

  if (!isdigit(version[5]) || version[6] != '.' || !isdigit(version[7]))
  {
    return -1;
  }

  /* Before v1.1 we close the connection after responding to the request */
  if ( (((version[5] - '0')*10) + version[7] - '0') < 11)
  {
    conn->close = 1;
  }

  strcpy(conn->uri, uri);
  return 0;
}
```

```
/*
 * http_send_file_chunk()
 *
 * This routine will send the next chunk of a file during an open HTTP session
 * where a file is being sent back to the client. This routine is called
 * repeatedly until the file is completely sent, at which time the connection
 * state will go to "COMPLETE". Doing this rather than sending the entire
 * file allows us (in part) to multiplex between connections "simultaneously".
 */
int http_send_file_chunk(http_conn* conn)
{
  int chunk_sent = 0, ret_code = 0, file_chunk_size = 0, result = 0;
  alt_u8* tx_ptr;

  if(conn->data_sent < conn->file_length)
  {
    file_chunk_size = fread(conn->tx_buffer, 1,
      MIN(HTTP_TX_BUF_SIZE, (conn->file_length - conn->data_sent)),
      conn->file_handle);

    tx_ptr = conn->tx_buffer;

    while(chunk_sent < file_chunk_size)
    {
      result = send(conn->fd, tx_ptr, file_chunk_size, 0);
      /* Error - get out of here! */
      if(result < 0)
      {
        fprintf(stderr, "[http_send_file] file send returned %d\n", result);
        ALT_DEBUG_ASSERT(1);
        conn->state = RESET;
        return result;
      }

      /*
       * No errors, but the number of bytes sent might be less than we wanted.
       */
      else
      {
        conn->activity_time = alt_nticks();
        chunk_sent += result;
        conn->data_sent += result;
        tx_ptr += result;
        file_chunk_size -= result;
      }
    } /* while(chunk_sent < file_chunk_size) */
  } /* if(conn->data_sent < conn->file_length) */

  /*
   * We managed to send all of the file contents to the IP stack successfully.
   * At this point we can mark our connection info as complete.
   */
  if(conn->data_sent >= conn->file_length)
  {
    conn->state = COMPLETE;
  }

  return ret_code;
}

/*
 * http_send_file_header()
 *
 * Construct and send an HTTP header describing the now-opened file that is
 * about to be sent to the client.
 */
int http_send_file_header(http_conn* conn, const alt_u8* name, int code)
{
  int     result = 0, ret_code = 0;
  alt_u8* tx_wr_pos = conn->tx_buffer;
  fpos_t  end, start;
              const alt_u8* ext = strchr(name, '.');

  tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);

  switch(code)
  {
    /* HTTP Code: "200 OK\r\n" (we have opened the file successfully) */
    case HTTP_OK:
```

```
  {
   tx_wr_pos += sprintf(tx_wr_pos, HTTP_OK_STRING);
   break;
  }
  /* HTTP Code: "404 Not Found\r\n" (couldn't find requested file) */
  case HTTP_NOT_FOUND:
  {
   tx_wr_pos += sprintf(tx_wr_pos, HTTP_NOT_FOUND_STRING);
   break;
  }
  default:
  {
   fprintf(stderr, "[http_send_file_header] Invalid HTTP code: %d\n", code);
   conn->state = RESET;
   return -1;
   break;
  }
 }

 /* Handle the various content types */
 tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE);


 if (!strcasecmp(ext, ".html"))
 {
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_HTML);
 }
 else if (!strcasecmp(ext, ".jpg"))
```

```
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_JPG);
}
else if (!strcasecmp(ext, ".gif"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_GIF);
}
else if (!strcasecmp(ext, ".png"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_PNG);
}
else if (!strcasecmp(ext, ".js"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_JS);
}
else if (!strcasecmp(ext, ".min.js"))
 {
   tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_JS);
 }
else if (!strcasecmp(ext, ".css"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_CSS);
}
else if (!strcasecmp(ext, ".min.css"))
  {
    tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_CSS);
  }
```

```
else if (!strcasecmp(ext, ".swf"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_SWF);
}
else if (!strcasecmp(ext, ".ico"))
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_ICO);
}
else
{
  fprintf(stderr, "[http_send_file] Unknown content type: \"%s\"\n", ext);
  conn->state = RESET;
  ALT_DEBUG_ASSERT(1);
  return -1;
}

/* Get the file length and stash it into our connection info */
fseek(conn->file_handle, 0, SEEK_END);
fgetpos(conn->file_handle, &end);
fseek(conn->file_handle, 0, SEEK_SET);
fgetpos(conn->file_handle, &start);
conn->file_length = end - start;

/* "Content-Length: <length bytes>\r\n" */
tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_LENGTH);
tx_wr_pos += sprintf(tx_wr_pos, "%d\r\n", conn->file_length);
```

```
/*
 * 'close' will be set during header parsing if the client either specified
 * that they wanted the connection closed ("Connection: Close"), or if they
 * are using an HTTP version prior to 1.1. Otherwise, we will keep the
 * connection alive.
 *
 * We send a specified number of files in a single keep-alive connection,
 * we'll also close the connection. It's best to be polite and tell the client,
 * though.
 */
if(!conn->keep_alive_count)
{
  conn->close = 1;
}

if(conn->close)
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
}
else
{
  tx_wr_pos += sprintf(tx_wr_pos, HTTP_KEEP_ALIVE);
}

/* "\r\n" (two \r\n's in a row means end of headers */
tx_wr_pos += sprintf(tx_wr_pos, HTTP_CR_LF);
```

```
  /* Send the reply header */
  result = send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer),
             0);

  if(result < 0)
  {
    fprintf(stderr, "[http_send_file] header send returned %d\n", result);
    conn->state = RESET;
    return result;
  }
  else
  {
    conn->activity_time = alt_nticks();
  }

  return ret_code;
}



/*
 * http_find_file()
 *
 * Try to find the file requested. If nothing is requested you get /index.html
 * If we can't find it, send a "404 - Not found" message.
 */
int http_find_file(http_conn* conn)
{
```

```
alt_u8  filename[256];
int     ret_code = 0;

strncpy( filename, ALTERA_RO_ZIPFS_NAME, strlen(ALTERA_RO_ZIPFS_NAME));

/* URI of "/" means get the default, usually index.html */
if ( (conn->uri[0] == '/') && (conn->uri[1] == '\0') )
{
  strcpy(filename+strlen(ALTERA_RO_ZIPFS_NAME), HTTP_DEFAULT_FILE);
}
else
{
  strcpy( filename+strlen(ALTERA_RO_ZIPFS_NAME), conn->uri);
}
//new stuff
if ( strncmp(conn->uri,"/json",5) == 0)
                    {
            send(conn->fd,(void*)json_http_response,strlen(canned_http_response),0);


                    }

/* Try to open the file */
printf("\nFetching file:  %s.\n", filename );
conn->file_handle = fopen(filename, "r");

/* Can't find the requested file? Try for a 404-page. */
if (conn->file_handle == NULL)
```

```
{
  strcpy(filename, ALTERA_RO_ZIPFS_NAME);
  strcpy(filename+strlen(ALTERA_RO_ZIPFS_NAME), HTTP_NOT_FOUND_FILE);
  conn->file_handle = fopen(filename, "r");

  /* We located the specified "404: Not-Found" page */
  if (conn->file_handle != NULL)
  {
    ALT_DEBUG_ASSERT(fd != NULL);
    ret_code = http_send_file_header(conn, filename, HTTP_NOT_FOUND);
  }
  /* Can't find the 404 page: This likely means there is no file system */
  else
  {
    fprintf(stderr, "Can't open the 404 File Not Found error page.\n");
    fprintf(stderr, "Have you programmed the filing system into flash?\n");
    send(conn->fd,(void*)canned_http_response,strlen(canned_http_response),0);

    fclose(conn->file_handle);
    conn->state = RESET;
    return -1;
  }
}
/* We've found the requested file; send its header and move on. */
else
{
  ret_code = http_send_file_header(conn, filename, HTTP_OK);
```

```
 }

 return ret_code;
}



/*
 * http_send_file()
 *
 * This function sends re-directs to either program_flash.html or
 * reset_sytem.html.
 */

void http_send_redirect( alt_u8 redirect[256] )
{
 printf ("Don't do anything....for now.\n");
}

/*
 * http_handle_post()
 *
 * Process the post request and take the appropriate action.
 */
int http_handle_post(http_conn* conn)
{
 alt_u8* tx_wr_pos = conn->tx_buffer;
 int ret_code = 0;
```

```
struct upload_buf_struct *upload_buffer = &upload_buf;
/*
tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);*/

if (!strcmp(conn->uri, mapping.name))
{
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
  send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);
  conn->state = CLOSE;
  mapping.func();
}

else if (!strcmp(conn->uri, sweep_field.name))
{


        tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
  send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);
  conn->state = CLOSE;
  sweep_field.func(conn);
 }
 // Added code
 else if (!strcmp(conn->uri, distance_field.name))
 {
        /*tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);// possibly remove
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_OK_STRING);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_KEEP_ALIVE);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
        tx_wr_pos += sprintf(tx_wr_pos, HTTP_CONTENT_TYPE_XML);
  send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);*/
  distance_field.func(conn);
  conn->state = CLOSE;
 }
 else if (!strcmp(conn->uri, lcd_field.name))
 {
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
         tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
  send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);
  conn->state = CLOSE;
  lcd_field.func(conn);
 }

 else if (!strcmp(conn->uri, upload_field.name))
 {
```

```
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
   conn->file_upload = 1;
   upload_buffer->rd_pos = upload_buffer->wr_pos = (alt_u8*) upload_buffer->buffer;
   memset(upload_buffer->rd_pos, '\0', conn->content_length );
   upload_field.func(conn);
  }
  else if (!strcmp(conn->uri, flash_field.name))
  {
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_VERSION_STRING);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_NO_CONTENT_STRING);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_CLOSE);
            tx_wr_pos += sprintf(tx_wr_pos, HTTP_END_OF_HEADERS);
   /* Kick off the flash programming.  */
   flash_field.func( conn );
  }
#ifdef RECONFIG_REQUEST_PIO_NAME
  else if (!strcmp(conn->uri, reset_field.name))
  {
   /* Close the socket.  */
   send(conn->fd, conn->tx_buffer, (tx_wr_pos - conn->tx_buffer), 0);
   reset_field.func();
  }
#endif
  return ret_code;
```

```
}



/*
 * http_prepare_response()
 *
 * Service the various HTTP commands, calling the relevant subroutine.
 * We only handle GET and POST.
 */
int http_prepare_response(http_conn* conn)
{
 int ret_code = 0;

 switch (conn->action)
 {
  case GET:
  {
   /* Find file from uri */
   ret_code = http_find_file(conn);
   break;
  }
  case POST:
  {
   /* Handle POSTs. */
   ret_code = http_handle_post(conn);
   break;
  }
```

```
    default:
    {
      break;
    }
  } /* switch (conn->action) */

  return ret_code;
}

/*
 * http_handle_receive()
 *
 * Work out what the request we received was, and handle it.
 */
void http_handle_receive(http_conn* conn, int http_instance)
{
  int data_used, rx_code;

  if (conn->state == READY)
  {
    rx_code = recv(conn->fd, conn->rx_wr_pos,
           (HTTP_RX_BUF_SIZE - (conn->rx_wr_pos - conn->rx_buffer) -1),
           0);

    /*
     * If a valid data received, take care of buffer pointer & string
     * termination and move on. Otherwise, we need to return and wait for more
     * data to arrive (until we time out).
     */
    if(rx_code > 0)
    {
      /* Increment rx_wr_pos by the amount of data received. */
      conn->rx_wr_pos += rx_code;
      /* Place a zero just after the data received to serve as a terminator. */
      *(conn->rx_wr_pos+1) = 0;

      if(strstr(conn->rx_buffer, HTTP_END_OF_HEADERS))
      {
        conn->state = PROCESS;
      }
      /* If the connection is a file upload, skip right to DATA.*/
      if(conn->file_upload == 1)
      {
        conn->state = DATA;
      }
    }
  }

  if(conn->state == PROCESS)
  {
    /*
     * If we (think) we have valid headers, keep the connection alive a bit
     * longer.
     */
```

```
conn->activity_time = alt_nticks();

/*
 * Attempt to process the fundamentals of the HTTP request. We may
 * error out and reset if the request wasn't complete, or something
 * was asked from us that we can't handle.
 */
if (http_process_request(conn))
{
 fprintf(stderr, "[http_handle_receive] http_process_request failed\n");
 conn->state = RESET;
 http_manage_connection(conn, http_instance);
}

/*
 * Step through the headers to see if there is any other useful
 * information about our pending transaction to extract. After that's
 * done, send some headers of our own back to let the client know
 * what's happening. Also, once all in-coming headers have been parsed
 * we can manage our RX buffer to prepare for the next in-coming
 * connection.
 */
while(conn->state == PROCESS)
{
 if(http_read_line(conn))
 {
  fprintf(stderr, "[http_handle_receive] error reading headers\n");
```

```
      conn->state = RESET;
      http_manage_connection(conn, http_instance);
      break;
    }
    if(http_process_headers(conn))
    {
      if( (conn->rx_rd_pos = strstr(conn->rx_rd_pos, HTTP_CR_LF)) )
      {
        conn->rx_rd_pos += 2;
        conn->state = DATA;
        conn->activity_time = alt_nticks();
      }
      else
      {
        fprintf(stderr, "[http_handle_receive] Can't find end of headers!\n");
        conn->state = RESET;
        http_manage_connection(conn, http_instance);
        break;
      }
    }
  } /* while(conn->state == PROCESS) */

  if( http_prepare_response(conn) )
  {
    conn->state = RESET;
    fprintf(stderr, "[http_handle_receive] Error preparing response\n");
    http_manage_connection(conn, http_instance);
    }

    /*
     * Manage RX Buffer: Slide any un-read data in our input buffer
     * down over previously-read data that can now be overwritten, and
     * zero-out any bytes in question at the top of our new un-read space.
     */
    if(conn->rx_rd_pos > (conn->rx_buffer + HTTP_RX_BUF_SIZE))
    {
      conn->rx_rd_pos = conn->rx_buffer + HTTP_RX_BUF_SIZE;
    }

    data_used = conn->rx_rd_pos - conn->rx_buffer;
    memmove(conn->rx_buffer,conn->rx_rd_pos,conn->rx_wr_pos-conn->rx_rd_pos);
    conn->rx_rd_pos = conn->rx_buffer;
    conn->rx_wr_pos -= data_used;
    memset(conn->rx_wr_pos, 0, data_used);
   }

  if (conn->state == DATA && conn->file_upload == 1 )
  {
    /* Jump to the file_upload() function....process more received data. */
    upload_field.func(conn);
  }
}

/*
```

```
 * http_handle_transmit()
 *
 * Transmit a chunk of a file in an active HTTP connection. This routine
 * will be called from the thread's main loop when ever the socket is in
 * the 'DATA' state and the socket is marked as available for writing (free
 * buffer space).
 */
void http_handle_transmit(http_conn* conn, int http_instance)
{
 if( http_send_file_chunk(conn) )
 {
   fprintf(stderr, "[http_handle_transmit]: Send file chunk failed\n");
 }
}


/*
 * WStask()
 *
 * This MicroC/OS-II thread spins forever after first establishing a listening
 * socket for HTTP connections, binding them, and listening. Once setup,
 * it perpetually waits for incoming data to either a listening socket, or
 * (if there is an active connection), an HTTP data socket. When data arrives,
 * the approrpriate routine is called to either accept/reject a connection
 * request, or process incoming data.
 *
 * This routine calls "select()" to determine which sockets are ready for
 * reading or writing. This, in conjunction with the use of non-blocking
 * send() and recv() calls and sending responses broken up into chunks lets
 * us handle multiple active HTTP requests.
 */
void WSTask()
{
  int    i, fd_listen, max_socket;
  struct  sockaddr_in addr;
  struct  timeval select_timeout;
  fd_set  readfds, writefds;
  static  http_conn    conn[HTTP_NUM_CONNECTIONS];

  /*
   * Sockets primer...
   * The socket() call creates an endpoint for TCP of UDP communication. It
   * returns a descriptor (similar to a file descriptor) that we call fd_listen,
   * or, "the socket we're listening on for connection requests" in our web
   * server example.
   */
  if ((fd_listen = socket(AF_INET, SOCK_STREAM, 0)) < 0)
  {
    die_with_error("[WSTask] Listening socket creation failed");
  }

  /*
   * Sockets primer, continued...
   * Calling bind() associates a socket created with socket() to a particular IP
   * port and incoming address. In this case we're binding to HTTP_PORT and to
   * INADDR_ANY address (allowing anyone to connect to us. Bind may fail for
   * various reasons, but the most common is that some other socket is bound to
   * the port we're requesting.
   */
  addr.sin_family = AF_INET;
  addr.sin_port = htons(HTTP_PORT);
  addr.sin_addr.s_addr = INADDR_ANY;

  if ((bind(fd_listen,(struct sockaddr *)&addr,sizeof(addr))) < 0)
  {
   die_with_error("[WSTask] Bind failed");
  }

  /*
   * Sockets primer, continued...
   * The listen socket is a socket which is waiting for incoming connections.
   * This call to listen will block (i.e. not return) until someone tries to
   * connect to this port.
   */
  if ((listen(fd_listen,1)) < 0)
  {
   die_with_error("[WSTask] Listen failed");
  }

  /*
   * At this point we have successfully created a socket which is listening
   * on HTTP_PORT for connection requests from any remote address.
   */
  for(i=0; i<HTTP_NUM_CONNECTIONS; i++)
  {
   http_reset_connection(&conn[i], i);
  }


  while(1)
  {
   /*
    * The select() call below tells the stack to return  from this call
    * when any of the events we have expressed an interest in happen (it
    * blocks until our call to select() is satisfied).
    *
    * In the call below we're only interested in either someone trying to
    * connect to us, or when an existing (active) connection has new receive
    * data, or when an existing connection is in the "DATA" state meaning that
    * we're in the middle of processing an HTTP request. If none of these
    * conditions are satisfied, select() blocks until a timeout specified
    * in the select_timeout struct.
    *
    * The sockets we're interested in (for RX) are passed in inside the
    * readfds parameter, while those we're interested in for TX as passed in
    * inside the writefds parameter. The format of readfds and writefds is
    * implementation dependant, hence there are standard macros for
    * setting/reading the values:
    *
    *   FD_ZERO  - Zero's out the sockets we're interested in
    *   FD_SET   - Adds a socket to those we're interested in
    *   FD_ISSET - Tests whether the chosen socket is set
    */
   FD_ZERO(&readfds);
   FD_ZERO(&writefds);
   FD_SET(fd_listen, &readfds);

   max_socket = fd_listen+1;

   for(i=0; i<HTTP_NUM_CONNECTIONS; i++)
   {
    if (conn[i].fd != -1)
    {
     /* We're interested in reading any of our active sockets */
     FD_SET(conn[i].fd, &readfds);

     /*
      * We're interested in writing to any of our active sockets in the DATA
      * state
      */
     if(conn[i].state == DATA)
     {
      FD_SET(conn[i].fd, &writefds);
     }

     /*
      * select() must be called with the maximum number of sockets to look
```

```
      * through. This will be the largest socket number + 1 (since we start
      * at zero).
      */
     if (max_socket <= conn[i].fd)
     {
       max_socket = conn[i].fd+1;
     }
   }
 }

 /*
  * Set timeout value for select. This must be reset for each select()
  * call.
  */
 select_timeout.tv_sec = 0;
 select_timeout.tv_usec = 500000;

 select(max_socket, &readfds, &writefds, NULL, &select_timeout);

 /*
  * If fd_listen (the listening socket we originally created in this thread
  * is "set" in readfds, then we have an incoming connection request.
  * We'll call a routine to explicitly accept or deny the incoming connection
  * request.
  */
 if (FD_ISSET(fd_listen, &readfds))
 {
  http_handle_accept(fd_listen, conn);
 }

 /*
  * If http_handle_accept() accepts the connection, it creates *another*
  * socket for sending/receiving data. This socket is independant of the
  * listening socket we created above. This socket's descriptor is stored
  * in conn[i].fd. Therefore if conn[i].fd is set in readfs, we have
  * incoming data for our HTTP server, and we call our receive routine
  * to process it. Likewise, if conn[i].fd is set in writefds, we have
  * an open connection that is *capable* of being written to.
  */
 for(i=0; i<HTTP_NUM_CONNECTIONS; i++)
 {
  if (conn[i].fd != -1)
  {
   if(FD_ISSET(conn[i].fd,&readfds))
   {
    http_handle_receive(&conn[i], i);
   }

   if(FD_ISSET(conn[i].fd,&writefds))
   {
    http_handle_transmit(&conn[i], i);
   }

   http_manage_connection(&conn[i], i);
```

```
    }
   }
 }/* while(1) */
}


/
*****************************************************************************
*                                          *
* License Agreement                                *
*                                          *
* Copyright (c) 2006 Altera Corporation, San Jose, California, USA.      *
* All rights reserved.                             *
*                                          *
* Permission is hereby granted, free of charge, to any person obtaining a   *
* copy of this software and associated documentation files (the "Software"),  *
* to deal in the Software without restriction, including without limitation   *
* the rights to use, copy, modify, merge, publish, distribute, sublicense,    *
* and/or sell copies of the Software, and to permit persons to whom the      *
* Software is furnished to do so, subject to the following conditions:      *
*                                          *
* The above copyright notice and this permission notice shall be included in  *
* all copies or substantial portions of the Software.                *
*                                          *
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR  *
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,    *
```

D.4 niosII_microc_fp.vhd

```
-- Robert Miller & Adam Eliason
-- rnmiller@ualberta.ca eliason1@ualberta.ca
-- Modified from file provided by Nancy Minderman
-- Changed port mappings from positional or ordered to explicit

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity niosII_microc_fp is

        port

        (



                -- the_dm9000a_inst

        ENET_CMD    :       OUT STD_LOGIC;
        ENET_CS_N   :       OUT STD_LOGIC;
        ENET_DATA   :       INOUT STD_LOGIC_VECTOR (15 DOWNTO 0);
        ENET_INT    :       IN STD_LOGIC;
        ENET_RD_N   :       OUT STD_LOGIC;
        ENET_RST_N :        OUT STD_LOGIC;
        ENET_WR_N  :        OUT STD_LOGIC;
                ENET_CLK    :       OUT STD_LOGIC;


                -- RPM Input Pin
                GPIO_1                  :       in                  std_logic_vector (35 downto
0);  -- from rpm GPIO_1[0],PIN_K25, pin1


                -- LOAD CONTROLLER Outputs
                GPIO_0                  :       out                 std_logic_vector (35 downto
0);


                -- LCD ports
                LCD_ON      :       out             std_logic;
                LCD_BLON    :       out             std_logic;
                LCD_EN      :       out             std_logic;
                LCD_RS      :       out             std_logic;
                LCD_RW      :       out             std_logic;
                LCD_DATA    :       inout   std_logic_vector (7 downto 0);


                -- LEDS
                LEDG        :       out             std_logic_vector(7 downto 0);
                LEDR        :       out             std_logic_vector(17 downto 0);
```

```
--SD Card
SD_DAT          :       in                  std_logic; --data in
SD_DAT3         :       out         std_logic; --chip select
SD_CMD          :       out         std_logic; --data out
SD_CLK          :       out         std_logic; --clock



-- DRAM ports
DRAM_CLK        :       out         std_logic;
DRAM_CKE   :       out         std_logic;
DRAM_ADDR :       out         std_logic_vector(11 DOWNTO 0);
DRAM_BA_1  :       buffer  std_logic;
DRAM_BA_0  :       buffer  std_logic;
DRAM_CS_N  :       out         std_logic;
DRAM_CAS_N      :       out         std_logic;
DRAM_RAS_N      :       out         std_logic;
DRAM_WE_N       :       out         std_logic;
DRAM_DQ    :       inout   std_logic_vector(15 DOWNTO 0);
DRAM_UDQM       :       buffer  std_logic;
DRAM_LDQM       :       buffer  std_logic ;

-- 7 Segment PIO
HEX0, HEX1  :       OUT STD_LOGIC_VECTOR(6 DOWNTO 0);

-- Flash
FL_ADDR : OUT STD_LOGIC_VECTOR (21 DOWNTO 0);
FL_DQ : INOUT STD_LOGIC_VECTOR (7 DOWNTO 0);
```

```
        FL_OE_N : OUT STD_LOGIC;
        FL_CE_N : OUT STD_LOGIC;
        FL_WE_N : OUT STD_LOGIC;
                FL_RST_N : OUT STD_LOGIC;



                -- CLOCK port
                CLOCK_50    :       in              std_logic;

                -- Switch
                SW                          :       in              std_logic_vector(0
downto 0);

                -- RESET key
                KEY                     :       in              std_logic_vector(3 downto 0)
   );

end niosII_microc_fp;


-- Library Clause(s) (optional)
-- Use Clause(s) (optional)

architecture structure of niosII_microc_fp is

-- A component declaration declares the interface of an entity or
-- a design unit written in another language.  VHDL requires that
-- you declare a component if you do not intend to instantiate
-- an entity directly.  The component need not declare all the
-- generics and ports in the entity.  It may omit generics/ports
-- with default values.

component niosII_system

    port
    (

                LCD ports
                LCD_E_from_the_lcd_display : out std_logic;
                LCD_RS_from_the_lcd_display : out std_logic;
                LCD_RW_from_the_lcd_display : out std_logic;
                LCD_data_to_and_from_the_lcd_display : inout std_logic_vector (7 downto
0);

        -- PLL ports
                altpll_0_c0_out : out std_logic;
        altpll_0_c1_out : out std_logic;
                altpll_0_c2_out : out std_logic;
        locked_from_the_altpll_0 : out std_logic;

                -- the_dm9000a_inst
         signal ENET_CMD_from_the_dm9000a_inst : OUT STD_LOGIC;
         signal ENET_CS_N_from_the_dm9000a_inst : OUT STD_LOGIC;
         signal ENET_DATA_to_and_from_the_dm9000a_inst : INOUT
STD_LOGIC_VECTOR (15 DOWNTO 0);
```

```
        signal ENET_INT_to_the_dm9000a_inst : IN STD_LOGIC;
        signal ENET_RD_N_from_the_dm9000a_inst : OUT STD_LOGIC;
        signal ENET_RST_N_from_the_dm9000a_inst : OUT STD_LOGIC;
        signal ENET_WR_N_from_the_dm9000a_inst : OUT STD_LOGIC;


                -- the_tri_state_bridge_0_avalon_slave
        signal address_to_the_ext_flash : OUT STD_LOGIC_VECTOR (21 DOWNTO 0);
                 signal tri_state_bridge_0_data : INOUT STD_LOGIC_VECTOR (7
DOWNTO 0); --new
        signal read_n_to_the_ext_flash : OUT STD_LOGIC;
        signal select_n_to_the_ext_flash : OUT STD_LOGIC;
        signal write_n_to_the_ext_flash : OUT STD_LOGIC;


        -- LEDS
                out_port_from_the_green_leds : out std_logic_vector (7 downto 0);
                out_port_from_the_red_leds :  out STD_LOGIC_VECTOR (17 DOWNTO 0);


                -- SD CARD
                spi_clk_from_the_sd_controller_0 : OUT STD_LOGIC;
                spi_cs_n_from_the_sd_controller_0 : OUT STD_LOGIC;
                spi_data_in_to_the_sd_controller_0 : IN STD_LOGIC;
                spi_data_out_from_the_sd_controller_0 : OUT STD_LOGIC;




                -- Sensors
                coe_RPMSensor_to_the_RPM_counter_0 : IN STD_LOGIC_VECTOR (0
downto 0);
                in_port_to_the_LeftTurn_sensor_0 :  IN STD_LOGIC;
```

```
    in_port_to_the_RightTurn_sensor_0 :  IN STD_LOGIC;
            in_port_to_the_loadIncrease : IN STD_LOGIC;
            in_port_to_the_loadDecrease : IN STD_LOGIC;



            -- Load Controller
            out_port_from_the_loadController : OUT STD_LOGIC_VECTOR (11
DOWNTO 0);


    -- One more PLL port
     signal phasedone_from_the_altpll_0 : out std_logic;-- chnged to signal


    -- DRAM ports
    zs_addr_from_the_sdram_0 : out std_logic_vector (11 DOWNTO 0);
    zs_ba_from_the_sdram_0 : out std_logic_vector (1 DOWNTO 0);
    zs_cas_n_from_the_sdram_0 : out std_logic;
            zs_cke_from_the_sdram_0 : out std_logic;
    zs_cs_n_from_the_sdram_0 : out std_logic;
    zs_dq_to_and_from_the_sdram_0 : inout std_logic_vector (15 DOWNTO 0);
    zs_dqm_from_the_sdram_0 : out std_logic_vector (1 DOWNTO 0);
    zs_ras_n_from_the_sdram_0 : out std_logic;


            zs_we_n_from_the_sdram_0 : out std_logic;


            -- the_seven_seg_pio
    signal out_port_from_the_seven_seg_pio : OUT STD_LOGIC_VECTOR (15
DOWNTO 0);
```

```
        -- CLOCK port
        clk_0       : in  std_logic;


                -- SWITCH to change message
                in_port_to_the_switch : in std_logic;


        -- RESET Key
        signal reset_n : in  std_logic; -- changed to signal
                signal in_port_to_the_reset_button : IN STD_LOGIC;


                -- Local increase/decrease loads
        signal in_port_to_the_local_loadDecrease : IN STD_LOGIC;
        signal in_port_to_the_local_loadIncrease : IN STD_LOGIC
    );
end component;


        signal BA : std_logic_vector (1 downto 0);
        signal DQM : std_logic_vector (1 downto 0);


                -- I have no use for these signals from the pll so I just
                -- connect them with signals
                -- I specifically asked the PLL to omit these signals and yet
                -- they are generated anyway


        signal pll_c1:std_logic;
        signal pll_locked:std_logic;
        signal pll_phase: std_logic;
                signal seven_seg_16 : STD_LOGIC_VECTOR(15 downto 0);



                begin
                – The SOPC-created IP does not turn the LCD on at all.
                – I've explicitly turned it on with the next LOC. Note that LCD_ON is the
                – VCC power line not the programmaticly controlled version that is bit-banged.
        LCD_ON <= '1';


                – Set the pair of seven segment displays as a 16 bit
                – vector where the 8th and 16th bit are ignored
                – (this is the convention that the altera examples use,
                – usually the 8th and 16th bits would be decimal points)
                HEX0 <= seven_seg_16(6 downto 0);
                HEX1 <= seven_seg_16(14 downto 8);


                – Hard wire the flash to always be on (not reset)
                FL_RST_N <= '1';


                – I need to make the mapping between the BA and DQM lines of the
                niosII_system
                – and the physical DRAM
        DRAM_BA_1 <= BA(1);
                DRAM_BA_0 <= BA(0);
        DRAM_UDQM <= DQM(1);
        DRAM_LDQM <= DQM(0);
```

```
-- Component Instantiation Statement (optional)
NIOSII : niosII_system port map (
                    -- FOR GPIO pins
                    coe_RPMSensor_to_the_RPM_counter_0(0) => GPIO_1(13), -
                    in_port_to_the_LeftTurn_sensor_0 => GPIO_1(15),
                    in_port_to_the_RightTurn_sensor_0 => GPIO_1(12),
                    in_port_to_the_loadDecrease => GPIO_1(10),
                    in_port_to_the_loadIncrease => GPIO_1(11),
                    -- FOR GPIO Output for load controller
                    out_port_from_the_loadController(0) => GPIO_0(1), -- pin1
                    out_port_from_the_loadController(1) => GPIO_0(5),
                    out_port_from_the_loadController(2) => GPIO_0(9),
                    out_port_from_the_loadController(3) => GPIO_0(0),
                    out_port_from_the_loadController(4) => GPIO_0(8),
                    out_port_from_the_loadController(5) => GPIO_0(4),
                    out_port_from_the_loadController(6) => GPIO_0(3),
                    out_port_from_the_loadController(7) => GPIO_0(7),
                    out_port_from_the_loadController(8) => GPIO_0(2),
                    out_port_from_the_loadController(9) => GPIO_0(11), --pin 10
                    out_port_from_the_loadController(10) => GPIO_0(10), --pin
13
                    out_port_from_the_loadController(11) => GPIO_0(6), --pin 14
                    -- FOR SD CARD
                    spi_clk_from_the_sd_controller_0 => SD_CLK,
                    spi_cs_n_from_the_sd_controller_0 => SD_DAT3,
                    spi_data_in_to_the_sd_controller_0 => SD_DAT,
                    spi_data_out_from_the_sd_controller_0 => SD_CMD,
                    LCD_E_from_the_lcd_display => LCD_EN,
                    LCD_RS_from_the_lcd_display => LCD_RS,
                    LCD_RW_from_the_lcd_display => LCD_RW,
                    LCD_data_to_and_from_the_lcd_display => LCD_DATA,
                    altpll_0_c0_out => DRAM_CLK,
                    altpll_0_c1_out => pll_c1,
                    altpll_0_c2_out => ENET_CLK,
                    locked_from_the_altpll_0 => pll_locked,
                    phasedone_from_the_altpll_0 => pll_phase,
                    --LEDs
                    out_port_from_the_green_leds => LEDG,
                    out_port_from_the_red_leds => LEDR,
                    --SDRAM
                    zs_addr_from_the_sdram_0 => DRAM_ADDR,
                    zs_ba_from_the_sdram_0 => BA,
                    zs_cas_n_from_the_sdram_0 => DRAM_CAS_N,
                    zs_cke_from_the_sdram_0 => DRAM_CKE,
                    zs_cs_n_from_the_sdram_0 => DRAM_CS_N,
                    zs_dq_to_and_from_the_sdram_0 => DRAM_DQ,
                    zs_dqm_from_the_sdram_0 => DQM,
                    zs_ras_n_from_the_sdram_0 => DRAM_RAS_N,
                    zs_we_n_from_the_sdram_0 => DRAM_WE_N,
                    clk_0 => CLOCK_50, --
                    --DE2 Buttons
                    in_port_to_the_switch => SW(0),
                    reset_n => KEY(0),--
```

```
                    in_port_to_the_reset_button => KEY(1),
                    in_port_to_the_local_loadDecrease => KEY(3),
                in_port_to_the_local_loadIncrease => KEY(2),
                    --Ethernet dm9000
                    ENET_CMD_from_the_dm9000a_inst => ENET_CMD,
                    ENET_CS_N_from_the_dm9000a_inst => ENET_CS_N,
                    ENET_DATA_to_and_from_the_dm9000a_inst =>
ENET_DATA,
                    ENET_INT_to_the_dm9000a_inst => ENET_INT,
                    ENET_RD_N_from_the_dm9000a_inst => ENET_RD_N,
                    ENET_RST_N_from_the_dm9000a_inst => ENET_RST_N,
                    ENET_WR_N_from_the_dm9000a_inst => ENET_WR_N,
                    out_port_from_the_seven_seg_pio => seven_seg_16,
                    address_to_the_ext_flash => FL_ADDR,
                    tri_state_bridge_0_data => FL_DQ,
                    read_n_to_the_ext_flash => FL_OE_N,
                    select_n_to_the_ext_flash => FL_CE_N,
                    write_n_to_the_ext_flash => FL_WE_N
        );
end structure;
```

D.5 RPM_Counter.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity RPM_counter is

  port (
    -- inputs:
                              csi_clk :         IN STD_LOGIC;
                              csi_reset : IN STD_LOGIC;
                              avs_read : IN STD_LOGIC;
                              coe_RPMSensor : IN STD_LOGIC;


    -- outputs:
     avs_readdata : OUT STD_LOGIC_VECTOR (15 DOWNTO 0)
    );
end entity RPM_counter;

architecture Behavioral of RPM_counter is
signal c : std_logic_vector(15 downto 0) :=(others => '0');  --initializing count to zero.
begin
process(coe_RPMSensor,csi_reset)

begin
if(coe_RPMSensor'event and coe_RPMSensor='1') then --positive edge trigger
c <= c+'1';  --increment count at every positive edge of coe_RPMSensor.
end if;

if(csi_reset='1') then  --when reset equal to '1' make count equal to 0.
c <=(others => '0');  -- c ="0000000000000000"
end if;
end process;

process(csi_clk,avs_read)

begin
if(avs_read='1') then
avs_readdata  <= c;
end if;
end process;

end Behavioral;
```

D.6 web_server.c

```
*******************************************************************************

* Copyright (c) 2006 Altera Corporation, San Jose, California, USA.          *
* All rights reserved. All use of this software and documentation is         *
* subject to the License Agreement located at the end of this file below.    *
**************************************************************************
*                                              *
* Modified to work with Interniche (week of 9/22/06) - BjR                  *
*                                              *
* This is an example webserver using NicheStack on the MicroC/OS-II RTOS.     *
* It is in no way a complete implementation of a webserver, it is enough to   *
* serve up our demo pages and control a few board elements and that's it.     *
*                                              *
* This example uses the sockets interface. A good introduction to sockets     *
* programming is the book Unix Network Programming by Richard Stevens.        *
*                                              *
* Please refer to file ReadMe.txt for notes on this software example.         *
**************************************************************************
************
* NicheStack TCP/IP stack initialization and Operating System Start in main()
* for this example.
*
* This example demonstrates the use of MicroC/OS-II running on NIOS II.
* In addition it is to serve as a good starting point for designs using
* MicroC/OS-II and Altera NicheStack TCP/IP Stack - NIOS II Edition.
*
* Please refer to the Altera NicheStack Tutorial documentation for details on
* this software example, as well as details on how to configure the NicheStack
* TCP/IP networking stack and MicroC/OS-II Real-Time Operating System.
*/
```

```
/* MicroC/OS-II definitions */
#include "includes.h"

#include <stdio.h>
#include <errno.h>
#include <ctype.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include <unistd.h>

/* Web Server definitions */
#include "alt_error_handler.h"
#include "web_server.h"

/* Nichestack definitions */
#include "ipport.h"
#include "libport.h"
#include "osport.h"
#include "tcpport.h"
#include "net.h"

#include "sd_controller.h" // Added SD controller
#include <string.h>
#include "io.h" //for IORD IOWR
#include "sys/alt_irq.h"
#include "altera_avalon_timer_regs.h" //for ALTERA_AVALON_TIMER_STATUS_REG
```

```
#include "bike.h"



#ifndef ALT_INICHE
  #error This Web Server requires the Interniche IP Stack Software Component.
#endif

#ifndef __ucosii__
  #error This Web Server requires the UCOS II IP Software Component.
#endif

#ifndef  RO_ZIPFS
  #error This Web Server requires the Altera Read only Zip file system.
#endif

#ifdef LCD_DISPLAY_NAME
FILE* lcdDevice;
#endif /* LCD_DISPLAY_NAME */

extern int current_flash_block;

extern void WSTask();
static void WSCreateTasks();

//For ISR
static void rpm_isr( void * context, alt_u32 id);
static void loadincrease_isr( void * context, alt_u32 id);
static void loaddecrease_isr( void * context, alt_u32 id);
static void localloadincrease_isr( void * context, alt_u32 id);
static void localloaddecrease_isr( void * context, alt_u32 id);
static void turnleft_isr( void * context, alt_u32 id);
static void turnright_isr( void * context, alt_u32 id);
static void reset_isr( void * context, alt_u32 id);

/* NicheStack network structure. */
extern struct net netstatic[STATIC_NETS];

//extern int distance;



/* Declarations for creating a task with TK_NEWTASK.
 * All tasks which use NicheStack (those that use sockets) must be created this way.
 * TK_OBJECT macro creates the static task object used by NicheStack during operation.
 * TK_ENTRY macro corresponds to the entry point, or defined function name, of the task.
 * inet_taskinfo is the structure used by TK_NEWTASK to create the task.
 */

TK_OBJECT(to_wstask);
TK_ENTRY(WSTask);

struct inet_taskinfo wstask = {
    &to_wstask,
    "web server",
    WSTask,
   HTTP_PRIO,
   APP_STACK_SIZE,
};

/* Function which displays the obtained (or assigned) IP
 * Address on the LCD Display.
 */
#ifdef LCD_DISPLAY_NAME
void lcd_ip_addr()
{
  /* Declare local ipaddr variable. */
  ip_addr* ipaddr;

  /* Assign ipaddr to the network interface's IP Address.
   * NOTE:  This code assumes that only a single network
   * interface exists
   */
  ipaddr = &nets[0]->n_ipaddr;

  /* Display the IP Address (initially) on the LCD Display. */
  lcdDevice = fopen( "/dev/lcd_display", "w" );
  fprintf(lcdDevice, "\nIP Address\n%d.%d.%d.%d\n",
     ip4_addr1(*ipaddr),
     ip4_addr2(*ipaddr),
     ip4_addr3(*ipaddr),
     ip4_addr4(*ipaddr));
  fclose( lcdDevice );
```

```
}
#endif

/* Function which resets the system.  Initiated by sending a string to the "LCD"
 * or through the Reset web form.
 */
#ifdef RECONFIG_REQUEST_PIO_NAME
void trigger_reset()
{
  /* Drive a 0 out to the configuration PLD reconfig_request pin. */
  IOWR_ALTERA_AVALON_PIO_DATA( RECONFIG_REQUEST_PIO_BASE, 0x0 );
  /* Set BIDIR PIO to drive out. */
  IOWR_ALTERA_AVALON_PIO_DIRECTION( RECONFIG_REQUEST_PIO_BASE, 1 );
  usleep ( 1000000 );
  /* Drive out a 1....probably won't reach this point!!! */
  IOWR_ALTERA_AVALON_PIO_DATA( RECONFIG_REQUEST_PIO_BASE, 0x1 );
}
#endif

/* WSInitialTask will initialize the NichStack TCP/IP stack and then initialize
 * the rest of the web server example tasks.
 */

void WSInitialTask(void* pdata)
{
  INT8U error_code = OS_NO_ERR;
```

```
/*
* Initialize Altera NicheStack TCP/IP Stack - Nios II Edition specific code.
* NicheStack is initialized from a task, so that RTOS will have started, and
* I/O drivers are available.  Two tasks are created:
*    "Inet main"  task with priority 2
*    "clock tick" task with priority 3
*/
alt_iniche_init();
/* Start the Iniche-specific network tasks and initialize the network
 * devices.
 */
netmain();
/* Wait for the network stack to be ready before proceeding. */
while (!iniche_net_ready)
  TK_SLEEP(1);
/* Create the main network task.  In this case, a web server. */
TK_NEWTASK(&wstask);
/* Application specific code starts here... */
/*Create Tasks*/
WSCreateTasks();
printf("\nWeb Server starting up");
/* Application specific code ends here. */
/* Display the IP Address on the LCD Display. */
#ifdef LCD_DISPLAY_NAME
lcd_ip_addr();
#endif
/*This task deletes itself, since there's no reason to keep it around, once
```

```
   * it's complete.
   */
  error_code = OSTaskDel(OS_PRIO_SELF);
  alt_uCOSIIErrorHandler(error_code, 0);
  while(1); /* Correct Program Flow should not reach here. */
}
/*
 * A MicroC/OS-II message box will be used to communicate between telnet
 * and board LED control tasks.
 */
//static void board_control_init();
void LED_task(void* pdata);
void SSD_task(void* pdata);
#ifdef SEVEN_SEG_PIO_NAME
static void sevenseg_set_hex(alt_u8 hex);
#endif
void board_control_task(void *pdata);
/*void Init_Load(void);
void Init_SD(void);
void RPM_task(void *pdata);
void LOADINCREASE_task(void *pdata);
void LOADDECREASE_task(void *pdata);
void TURNLEFT_task(void *pdata);
void TURNRIGHT_task(void *pdata);
float CalcSpeed(int RPM, int circumference);
int CalcRPM(int RPM);
int CalcDistance(int RPM, int Circumference);
```

```
void IncreaseLoad(void);
void DecreaseLoad(void);
void PrintLoad(void);*/
/* Definition of Task Stacks for tasks not using networking.*/
OS_STK   WSInitialTaskStk[TASK_STACKSIZE];
OS_STK   LEDTaskStk[TASK_STACKSIZE];
OS_STK   SSDTaskStk[TASK_STACKSIZE];
OS_STK   BCTaskStk[TASK_STACKSIZE];
// This is our SD card
extern sd_card_info_struct* sd_card_global;
// Initializes SD card on board, returns FAILED or OK
void Init_SD(void)
{
  int volumes_mounted;
  //printf("test before printing to LCD in SD");
  char* msg = "\x1B[2J";
  char* msg2 = "Please Wait.... \nMounting SD Card";
  FILE* fp;

  fp = fopen (LCD_DISPLAY_NAME, "w");
  if (fp!=NULL)
  {
    fprintf(fp, "%s%s",msg,msg2);
    fclose (fp);
  }
   // Initialize and mount the filesystem.
 printf("\nInitializing Filesystem            ");
```

```
   // Since the sd card controller is not on the cpu clock, we need
   // to re-set it's clock divider using the actual avalon slave clock freq.
  sd_set_clock_to_max( 50000000 );//change to 50mhz
  volumes_mounted = sd_fat_mount_all();
  if( volumes_mounted <= 0 )
  {
   printf( " - FAILED\n" );
  }
  else
  {
   printf(" - OK\n");
  }
  sd_print_card_info(sd_card_global);
  printf( "\nTesting to see if \"database\" is a folder." );
  char* foldername = "/database";
  int test = sd_isdir(foldername);


  if(test==1){printf("\nFolder exists!\n");}
  else if (test == 0){printf("\nFolder does NOT exist :(\n");}
  else{printf("\nSomething went wrong checking folder. returned value was: %d\n",test);}//
Code should never reach here unless error
  char* buf;
  test = sd_list(foldername, buf);
  printf("\tList = %d\n", test);
}
/*
 * Mailbox to control board features
 *
 */
OS_EVENT *board_control_mbox;
int main (int argc, char* argv[], char* envp[])
{
  /* Initialize the current flash block, for flash programming. */
		current_flash_block = -1;
		INT8U error_code;
		char err = OS_NO_ERR;
		commQ = OSQCreate(&commMsg[0], 1);
		leftQ = OSQCreate(&leftMsg[0], 1);
		rightQ = OSQCreate(&rightMsg[0], 1);
		increaseQ = OSQCreate(&increaseMsg[0], 1);
		decreaseQ = OSQCreate(&decreaseMsg[0], 1);
		resetQ = OSQCreate(&resetMsg[0], 1);
		//Register IRQs
		if (alt_irq_register(RPM_TIMER_IRQ, NULL, rpm_isr) == 0)
		{printf("RPM ISR Registered\n");}
		if (alt_irq_enable(RPM_TIMER_IRQ) == 0)
		{printf("RPM Timer IRQ enabled\n");}
		if (alt_irq_register(LOADDECREASE_IRQ, NULL, loaddecrease_isr) == 0)
		{printf("Load Decrease ISR Registered\n");}
		if (alt_irq_enable(LOADDECREASE_IRQ) == 0)
		{printf("Load Decrease IRQ enabled\n");}
		if (alt_irq_register(LOADINCREASE_IRQ, NULL, loadincrease_isr) == 0)
		{printf("Load Increase ISR Registered\n");}
		if (alt_irq_enable(LOADINCREASE_IRQ) == 0)
```

```
        {printf("Load Increase IRQ enabled\n");}
        if (alt_irq_register(LOCAL_LOADDECREASE_IRQ, NULL, localloaddecrease_isr)
== 0)
        {printf("Local Load Decrease ISR Registered\n");}
        if (alt_irq_enable(LOCAL_LOADDECREASE_IRQ) == 0)
        {printf("Local Load Decrease IRQ enabled\n");}
        if (alt_irq_register(LOCAL_LOADINCREASE_IRQ, NULL, localloadincrease_isr)
== 0)
        {printf("Local Load Increase ISR Registered\n");}
        if (alt_irq_enable(LOCAL_LOADINCREASE_IRQ) == 0)
        {printf("Local Load Increase IRQ enabled\n");}
        if (alt_irq_register(LEFTTURN_SENSOR_0_IRQ, NULL, turnleft_isr) == 0)
        {printf("Left Turn ISR Registered\n");}
        if (alt_irq_enable(LEFTTURN_SENSOR_0_IRQ) == 0)
        {printf("Left Turn IRQ enabled\n");}
        if (alt_irq_register(RIGHTTURN_SENSOR_0_IRQ, NULL, turnright_isr) == 0)
        {printf("Right Turn ISR Registered\n");}
        if (alt_irq_enable(RIGHTTURN_SENSOR_0_IRQ) == 0)
        {printf("Right Turn IRQ enabled\n");}
        if (alt_irq_register(RESET_BUTTON_IRQ, NULL, reset_isr) == 0)
        {printf("Reset ISR Registered\n");}
        if (alt_irq_enable(RESET_BUTTON_IRQ) == 0)
        {printf("Reset IRQ enabled\n");}
        printf("\nFinished registering interrupts.\n");
 /* Clear the RTOS timer */
 OSTimeSet(0);


 /* Initialize load to 0 */
 Init_Load();

 /* Initialize SD card */

 //Init_SD();

 /* WSInitialTask will initialize the NicheStack TCP/IP Stack and then

  * initialize the rest of the web server's tasks.

  */


 error_code = OSTaskCreateExt(WSInitialTask,

                     NULL,

                     (void *)&WSInitialTaskStk[TASK_STACKSIZE-1],

                     WS_INITIAL_TASK_PRIO,

                     WS_INITIAL_TASK_PRIO,

                     WSInitialTaskStk,

                     TASK_STACKSIZE,

                     NULL,

                     0);

 alt_uCOSIIErrorHandler(error_code, 0);

 //write 7 (0111) to status register to enable ITO bit (enable IRQs), CONT bit (tells timer to run
forever), START bit (starts timer)

 IOWR(RPM_TIMER_BASE, ALTERA_AVALON_TIMER_CONTROL_REG, 0x7);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(LOADINCREASE_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(LOADDECREASE_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(LOCAL_LOADINCREASE_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(LOCAL_LOADDECREASE_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(LEFTTURN_SENSOR_0_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(RIGHTTURN_SENSOR_0_BASE, 1);
 IOWR_ALTERA_AVALON_PIO_IRQ_MASK(RESET_BUTTON_BASE, 1);


 /*

  * As with all MicroC/OS-II designs, once the initial thread(s) and
  * associated RTOS resources are declared, we start the RTOS. That's it!
  */
 OSStart();
 while(1); /* Correct Program Flow never gets here. */
 return -1;
}
static void WSCreateTasks()
{
 INT8U error_code = OS_NO_ERR;
  /* Start LED Task. */
 error_code = OSTaskCreateExt(LED_task,
                    NULL,
                    (void *)&LEDTaskStk[TASK_STACKSIZE-1],
                    LED_PRIO,
                    LED_PRIO,
                    LEDTaskStk,
                    TASK_STACKSIZE,
                    NULL,
                    0);
alt_uCOSIIErrorHandler(error_code, 0);
  /* Start SSD Task. */
#ifdef SEVEN_SEG_PIO_NAME
 error_code = OSTaskCreateExt(SSD_task,
                    NULL,
                    (void *)&SSDTaskStk[TASK_STACKSIZE-1],
                    SSD_PRIO,
                    SSD_PRIO,
                    SSDTaskStk,
                    TASK_STACKSIZE,
                    NULL,
                    0);
alt_uCOSIIErrorHandler(error_code, 0);
 #endif
  /* Start Board Control Task. */
 error_code = OSTaskCreateExt(board_control_task,
                    NULL,
                    (void *)&BCTaskStk[TASK_STACKSIZE-1],
                    BOARD_PRIO,
                    BOARD_PRIO,
                    BCTaskStk,
                    TASK_STACKSIZE,
                    NULL,
                    0);
alt_uCOSIIErrorHandler(error_code, 0);

  /* Start RPM Task. */
 error_code = OSTaskCreateExt(RPM_task,
                    NULL,
                    (void *)&RPMTaskStk[TASK_STACKSIZE-1],
                    RPM_PRIO,
                    RPM_PRIO,
                    RPMTaskStk,
                    TASK_STACKSIZE,
                    NULL,
                    0);
 alt_uCOSIIErrorHandler(error_code, 0);

  /* Start Load Increase Task. */
  error_code = OSTaskCreateExt(LOADINCREASE_task,
                     NULL,
                     (void *)&LoadIncreaseTaskStk[TASK_STACKSIZE-1],
                     LOADINCREASE_PRIO,
                     LOADINCREASE_PRIO,
                     LoadIncreaseTaskStk,
                     TASK_STACKSIZE,
                     NULL,
                     0);
  alt_uCOSIIErrorHandler(error_code, 0);

  /* Start Load Decrease Task. */
   error_code = OSTaskCreateExt(LOADDECREASE_task,
                      NULL,
                      (void *)&LoadDecreaseTaskStk[TASK_STACKSIZE-1],
                      LOADDECREASE_PRIO,
                      LOADDECREASE_PRIO,
                      LoadDecreaseTaskStk,
                      TASK_STACKSIZE,
                      NULL,
                      0);
   alt_uCOSIIErrorHandler(error_code, 0);

   /* Start Left Turn Task. */
   error_code = OSTaskCreateExt(TURNLEFT_task,
                      NULL,
                      (void *)&TurnLeftTaskStk[TASK_STACKSIZE-1],
                      TURNLEFT_PRIO,
                      TURNLEFT_PRIO,
                      TurnLeftTaskStk,
                      TASK_STACKSIZE,
                      NULL,
                      0);
   alt_uCOSIIErrorHandler(error_code, 0);

   /* Start Right Turn Task. */
    error_code = OSTaskCreateExt(TURNRIGHT_task,
                      NULL,
                      (void *)&TurnRightTaskStk[TASK_STACKSIZE-1],
                      TURNRIGHT_PRIO,
                      TURNRIGHT_PRIO,
                      TurnRightTaskStk,
                      TASK_STACKSIZE,
                      NULL,
                      0);
    alt_uCOSIIErrorHandler(error_code, 0);

    /* Start Reset Task. */
     error_code = OSTaskCreateExt(RESET_task,
                      NULL,
                      (void *)&ResetTaskStk[TASK_STACKSIZE-1],
                      RESET_PRIO,
                      RESET_PRIO,
                      ResetTaskStk,
                      TASK_STACKSIZE,
                      NULL,
                      0);

     alt_uCOSIIErrorHandler(error_code, 0);

 /* Suspend both the LED and SSD tasks on start. */

 OSTaskSuspend(LED_PRIO);
 OSTaskSuspend(SSD_PRIO);

 /* The web server task is started by the Interniche stack, as the "main" network servicing task.
*/
}

static void rpm_isr( void * context, alt_u32 id)
{
 OSQPost(commQ, (void*)1);
 //write zero to status register to clear TO bit to aknowledge interrupt
 IOWR(RPM_TIMER_BASE, ALTERA_AVALON_TIMER_STATUS_REG, 0x0);
}

static void loadincrease_isr( void * context, alt_u32 id)
{
 OSQPost(increaseQ, (void*)1);
 //write zero to status register to clear TO bit
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(LOADINCREASE_BASE, 1);
}

static void loaddecrease_isr( void * context, alt_u32 id)
{
 OSQPost(decreaseQ, (void*)1);
 //write zero to status register to clear TO bit
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(LOADDECREASE_BASE, 1);
}

static void localloadincrease_isr( void * context, alt_u32 id)
{
 OSQPost(increaseQ, (void*)1);
 //write zero to status register to clear TO bit
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(LOCAL_LOADINCREASE_BASE, 1);
}

static void localloaddecrease_isr( void * context, alt_u32 id)
{
 OSQPost(decreaseQ, (void*)1);
 //write zero to status register to clear TO bit
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(LOCAL_LOADDECREASE_BASE, 1);
}

static void turnleft_isr( void * context, alt_u32 id)
{
 OSQPost(leftQ, (void*)1);
 //write 1 to edgecapture register to clear interrupt
 //IOWR(LEFTTURN_SENSOR_0_BASE, ALTERA_AVALON_TIMER_STATUS_REG,
0x0);
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(LEFTTURN_SENSOR_0_BASE, 1);
}



static void turnright_isr( void * context, alt_u32 id)

{

 OSQPost(rightQ, (void*)1);

 //write zero to status register to clear TO bit
 //IOWR(RIGHTTURN_SENSOR_0_BASE, ALTERA_AVALON_TIMER_STATUS_REG,
0x0);
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(RIGHTTURN_SENSOR_0_BASE, 1);
}


static void reset_isr( void * context, alt_u32 id)
{
 OSQPost(resetQ, (void*)1);
 IOWR_ALTERA_AVALON_PIO_EDGE_CAP(RESET_BUTTON_BASE, 1);
}


#ifdef LCD_DISPLAY_NAME
void lcd_output_text( char text[20] )
{
 //printf("Reached LCD OUTPUT TEXT task"); // Used for debugging
 /* If the incoming string is "ip_address" (case insensitive)
  * output the IP Address.  Otherwise, dump whatever's passed
  * into the text[] array.
  */
 if ( stricmp( text, "ip_address" ) == 0 )
 {
  lcd_ip_addr();
 }
#ifdef RECONFIG_REQUEST_PIO_NAME
 else if ( stricmp( text, "reset" ) == 0 )
 {
  trigger_reset();
 }
#endif
 else
 {
   lcdDevice = fopen( "/dev/lcd_display", "w" );
   fprintf(lcdDevice, "\n\n%s", text);
   fclose( lcdDevice );
 }
}
#endif


void board_control_task(void *pdata)
{
 INT8U error_code = OS_NO_ERR;
 board_control_mbox = OSMboxCreate((void *)NULL);


 struct http_form_data* board_control_mbox_contents;


 while(1)
 {
		//printf("\nReached Board Control Task"); // Used for debugging
    board_control_mbox_contents = (void*)OSMboxPend(board_control_mbox, 0,
&error_code);


    if (board_control_mbox_contents->LED_ON)
    {
      OSTaskResume(LED_PRIO);
}
else
{
 /* Suspend the task and clear the LED. */
 OSTaskSuspend(LED_PRIO);
 //IOWR_ALTERA_AVALON_PIO_DATA( LED_PIO_BASE, 0 );
 IOWR_ALTERA_AVALON_PIO_DATA( GREEN_LEDS_BASE, 0 );
 IOWR_ALTERA_AVALON_PIO_DATA( RED_LEDS_BASE, 0 );
}

if (board_control_mbox_contents->SSD_ON)
{
 OSTaskResume(SSD_PRIO);
}
else
{
 /* Suspend the task and set SSD to all zeros. */
 OSTaskSuspend(SSD_PRIO);
            #ifdef SEVEN_SEG_PIO_NAME
 sevenseg_set_hex(0);
            #endif
}

/* Always dump text to the LCD... */
      #ifdef LCD_DISPLAY_NAME
lcd_output_text( board_control_mbox_contents->LCD_TEXT );
      #endif
```

```
    //OSTimeDlyHMSM(0,0,0,50);

 }

}


void LED_task(void* pdata)

{


  alt_u8 led = 0x2;

  alt_u8 dir = 0;


  /*

   * Infinitely shift a variable with one bit set back and forth, and write

   * it to the LED PIO.  Software loop provides delay element.

   */

  while (1)

  {

    if (led & 0x81)
    {
      dir = (dir ^ 0x1);
    }
    if (dir)
    {
      led = led >> 1;
    }
    else
    {

      led = led << 1;

    }
    IOWR_ALTERA_AVALON_PIO_DATA(GREEN_LEDS_BASE, led);
    IOWR_ALTERA_AVALON_PIO_DATA(RED_LEDS_BASE, led);
    OSTimeDlyHMSM(0,0,0,50);
 }
}

#ifdef SEVEN_SEG_PIO_NAME
static void sevenseg_set_hex(alt_u8 hex)

{

 static alt_u8 segments[16] = {

  0x81, 0xCF, 0x92, 0x86, 0xCC, 0xA4, 0xA0, 0x8F, 0x80, 0x84, /* 0-9 */

  0x88, 0xE0, 0xF2, 0xC2, 0xB0, 0xB8 };                 /* a-f */

 alt_u32 data = segments[hex & 15] | (segments[(hex >> 4) & 15] << 8);

 IOWR_ALTERA_AVALON_PIO_DATA(SEVEN_SEG_PIO_BASE, data);

}

void SSD_task(void* pdata)
{

 alt_u32 count;

  while (1)
 {
  for (count = 0; count <= 0xff; count++)
  {
   sevenseg_set_hex(count);
   OSTimeDlyHMSM(0,0,0,50);
  }
 }
}

#endif

/
******************************************************************
*                                             *
* License Agreement                                        *
*                                             *
* Copyright (c) 2006 Altera Corporation, San Jose, California, USA.          *
* All rights reserved.                                  *
*                                             *
* Permission is hereby granted, free of charge, to any person obtaining a     *
* copy of this software and associated documentation files (the "Software"),  *
* to deal in the Software without restriction, including without limitation   *
* the rights to use, copy, modify, merge, publish, distribute, sublicense,     *
```

Robert Miller • email: rnmiller@ualberta.ca • Adam Eliason • email: eliason1@ualberta.ca