Train Control Demo 1
CS452 - Spring 2014
Real-Time Programming

Team
Max Chen - mqchen
mqchen@uwaterloo.ca

Ford Peprah - hkpeprah
ford.peprah@uwaterloo.ca

Bill Cowan
University of Waterloo
Due Date: Tuesday, 8^{th}, July, 2014

Table of Contents

1 Program Description

1.1 Getting the Program

To run the program, one must have read/write access to the source code, as well as the ability to make and run the program. Before attempting to run the program ensure that the following three conditions are met:

- You are currently logged in as one of `cs452`, `mqchen`, or `hkpeprah`.
- You have a directory in which to store the source code, e.g. `~/cs452_microkern_mqchen_hkpeprah`.
- You have a folder on the FTP server with your username, e.g. `/u/cs452/tftp/ARM/cs452`.

First, you must get a copy of the code. To to this, log into one of the aforementioned accounts and change directories to the directory you created above (using `cd`), then run one of

```
git clone file:////u8/hkpeprah/cs452-microkern -b demo1 .
                        or
git clone file:////u7/mqchen/cs452/cs452-microkern -b demo1 .
```

You will now have a working instance of our demo1 source code in your current directory. To make the application and upload it to the FTP server at the location listed above (`/u/cs452/tftp/ARM/YOUR_USERNAME`), run `make upload`.

1.2 Running the Program

To run the application, you need to load it into the RedBoot terminal. Ensure you've followed the steps listed above in the "Getting the Program" settings to ensure you have the correct directories and account set up. Navigate to the directory in which you cloned the source code and run `make upload`. The uploaded code should now be located at

`/u/cs452/tftp/ARM/YOUR_USERNAME/kernel.elf`

To run the application, go to the RedBoot terminal and run the command

```
load -b 0x00218000 -h 10.15.167.4 ‘‘ARM/YOUR_USERNAME/kernel.elf’’; go
```

The application should now begin by running through the game tasks before reaching a prompt. The generated files will be located in `DIR/build` where `DIR` is the directory you created in the earlier steps. To access and download an existing version of the code, those can be found at `/u/cs452/tftp/ARM/mqchen/kernel.elf` and `/u/cs452/tftp/ARM/hkpeprah/kernel.elf`.

1.2.1 Command Prompt

After the startup tasks have finished running, the user will reach a command prompt where they will be able to enter commands. A list of available commands and the syntax can be found at run-time by entering either "?" or "help" followed by the "RETURN" key. All commands must be followed by the "RETURN" key for the `Shell` to interpret them.

2 Train Calibration

2.1 Path Finding

Dijkstra's is used to find path between a starting node and the destination. A wrapper structure is built for each provided `track_node` that contains meta information such as the predecessor node, the distance from the source, the number of nodes in the path so far and some additional information about its position with respect to the heap.
The nodes are stored in a binary heap implemented as a fixed size array. The heap is used by the algorithm to determine the next node to explore. For each node explored, the reverse and direct ahead nodes are always added, and in the case of branches, the curved destination is added as well.
The result of pathfinding is a list of nodes from the provided to the destination. The operation required to travel from a node in the path to the next is implicit in the relation between the two nodes (ie. n + 1 is the destination of n(DIR_STRAIGHT) $\rightarrow$ flip switch to straight, n + 1 is the reverse $\rightarrow$ reverse the train) and this information is used by the trains to navigate.

2.2 Measurements

Measuring the train speeds was done iteratively. First, we determined a singular path that all the trains would follow and measured the distance around that path in millimeters. We then used one sensor as a reference point, and allowed each train to do two laps around triggering that sensor, first lap to get rid of acceleration/deceleration effect and the second lap to measure the time, and took the time it took to go around the track as our time measurement. By dividing the total distance by the time, it gave us a pretty good approximation of the number of millimeters the train could cover per tick of the clock at each individual speed from 0 to 14. We used this measurement multipled by 1000 to turn our units into micrometers per tick as this was a much more usable number that would not involve floating point computation as our measurement of velocity. To then measure acceleration/deceleration, we timed the number of ticks it took from the trains at the various speeds to stop from full speed to 0, and the total distance covered stopping. This provided us two measurements for

each train for each speed: `stopping_distance` and `stopping_time`. These measurements allowed us to compuate when to send a stop command based on the distance that would be covered by stopping. We took acceleration to be equivalent to deceleration across all trains.

2.2.1 Track Distances

We used the provided track distance data to determine how far apart sensors are from each other. This was used in our calculation of the velocities of the trains.

2.3 User Tasks

2.3.1 TrainController

A `TrainController` task is a server that is responsible for allocating portions of the track to trains, adding trains to the track, and routing trains to their desired locations as determined by the user. The use-case of the `TrainController` is to reserve peices of the track as resources; when a peice is unavailable, a train has the option of either re-routing or blocking on that resource if no alternate route is available. It provies the following methods:

Name	Prototype	Description
Nearest Sensor Edge	`NearestSensorEdge (char module, unsigned int id)`	Returns the nearest edge to the specified sensor.
Add a train to track	`AddTrainToTrack (unsigned int tr, char module, unsigned int id)`	Add train to track at specified sensor.
Move train	`MoveTrainToDestination (unsigned int tr, unsigned int id, unsigned int dist)`	Move train to the specified sensor and some distance after.

2.3.2 SensorServer

A `SensorServer` task is a server that is responsible for allowing tasks to query the state of the sensors on the track, waiting on particular sensors to trigger, or wait on a sensor with timeout. It keeps track of the previous state of the sensors and only reports on a change in the sensors from 0 to 1 then back down to 0 again; preventing it from alerting sensors that are always triggered or are being held down. It provides the following methods:

Name	Prototype	Description
WaitOnSensor	`int WaitOnSensor (char module, unsigned int id)`	Block waiting for the specified sensor to trigger.
WaitAnySensor	`int WaitAnySensor ()`	Block waiting for *any* sensor to trigger.
FreeSensor	`int FreeSensor (unsigned int sensor)`	Free the task waiting on the specified sensor.

2.3.3 TrainTask

A `TrainTask` is a reprsentation in our model of a train on the track. Each train (IDs 45, 47, 48, 49, 50, and 51) has a collection of velocities mapped to by speeds as defined as numeric constants from 0 to 14. These velocities tell us the micrometers that a train travels per tick of the clock. In addition, they have a collection of stopping distances (distances travelled when decelerating from a given speed to 0) and the time it would take the train to decelerate from a given speed. Each train has its own task that maintains information about the train, and all communication between the program and the train (both model and physical) is handled through the task. The task employs 2 couriers, waiting on its expected next sensor as well as a timeout in the case that the sensor malfunctions. When the courier returns, the train knows that it has reached the next sensor and updates its position accordingly. In between sensors, the train's position is computed periodically using its last known position, its speed and the times elapsed since the last position update.
When one courier wakes up, the other must be destroyed. This was a newly implemented kernel functionality that allows task descriptors to be recycled cleanly and reused. The two courier tasks are constantly being destroyed and recreated as a train traverses along the track.

2.4 Error Recovery

The `TrainTask` was designed to handle situations where sensors are broken on the track. To do this, the `TrainTask` spawns a task that times out waiting on the estimated time to arrival at the next sensor. If we do not trip the next sensor before the timeout expires, it assumes that the sensor is broken and uses the velocity and time to calculate where the train should be with respect to the previous sensor. This allows us to deal with situations where a given sensor never reports along a train by moving the train forward in tme in the model.
The `SensorServer` handles the situation of a sensor always reporting by checking for a sensor going from 0 to 1 and back down to 0 again. In the event that a sensor is always triggering (always 1), it will only be reported once by the `SensorServer` thus preventing it from being attributed to a train too early; since the trains compensate for missed sensors, this does not affect the ability of the trains to navigate along the track.

3 MD5 Hashes

```
8afa04fd4ff12bc483271286d52dfa00  /u8/hkpeprah/cs452-microkern/bin/cs452-upload.sh
de6700ffc18bb2c8f15a491fc2929d13  /u8/hkpeprah/cs452-microkern/bin/md5.sh
7d3d938f3360ca46d07b07d6fed3711c  /u8/hkpeprah/cs452-microkern/bin/profiler.sh
40e6f5862869392d9733ea2d6defbb68  /u8/hkpeprah/cs452-microkern/include/bwio.h
bfcb67b08f11ead2eb221f89218918ff  /u8/hkpeprah/cs452-microkern/include/mem.h
fd85b3c0c6c81624eaae7329af22e801  /u8/hkpeprah/cs452-microkern/include/string.h
ebc4454525ebfb20da056b665ba30e17  /u8/hkpeprah/cs452-microkern/include/syscall.h
dd612e94d212df2ff0ef275c4aa7d922  /u8/hkpeprah/cs452-microkern/include/task.h
91425c50507432ecf2bfc92ae70589c4  /u8/hkpeprah/cs452-microkern/include/ts7200.h
266f306d9159e549873df452a1e52194  /u8/hkpeprah/cs452-microkern/include/types.h
d898edf77661ac9a98e2a0c6b9d9b9a6  /u8/hkpeprah/cs452-microkern/include/vargs.h
def804066ee9f47e9a9bbad0d1f84249  /u8/hkpeprah/cs452-microkern/include/k_syscall.h
1624fa508a85025bed31a50a05048f6d  /u8/hkpeprah/cs452-microkern/include/kernel.h
eac9f764270af5a8683da7e4ac5ee3c3  /u8/hkpeprah/cs452-microkern/include/stdio.h
204c87fe6abdf4a362f6c46ffd825091  /u8/hkpeprah/cs452-microkern/include/stdlib.h
cdfa26635677328255ed78474340bd38  /u8/hkpeprah/cs452-microkern/include/syscall_types.h
c37781e47b522db11fe4fbaf39957880  /u8/hkpeprah/cs452-microkern/include/term.h
bff22a8329a113c8bf64d0607d71cb55  /u8/hkpeprah/cs452-microkern/include/utasks.h
d00e25b002757c4b33b7aaac5618b990  /u8/hkpeprah/cs452-microkern/include/calibration.h
ec29cb9d7e1429fd87c601722c81f326  /u8/hkpeprah/cs452-microkern/include/clock.h
61d6b97adcf0e26750da24be235220cf  /u8/hkpeprah/cs452-microkern/include/controller.h
02da5c5ed64ddf5a5ff08090e1d407cf  /u8/hkpeprah/cs452-microkern/include/hash.h
c4bee24fcb42fadd00dca64817d6c87c  /u8/hkpeprah/cs452-microkern/include/idle.h
7cb6397fc4af9f54ac2bf6ba897ee4bf  /u8/hkpeprah/cs452-microkern/include/interrupt.h
cd31bb05b0e8cdd7f5fc4e216615d9a2  /u8/hkpeprah/cs452-microkern/include/logger.h
3296682e40d4b19c236a01b0b5c20427  /u8/hkpeprah/cs452-microkern/include/null.h
10b2fae3ddfba67aee37174d3ec8fbab  /u8/hkpeprah/cs452-microkern/include/path.h
717b31cadb3b90f022dc0690530d1aab  /u8/hkpeprah/cs452-microkern/include/perf_test.h
65b51124e5ee634ebdbba3664aa22a63  /u8/hkpeprah/cs452-microkern/include/random.h
7ff8faa9d929453fa8cd82d0e7c32b19  /u8/hkpeprah/cs452-microkern/include/rps.h
01ef05317987bed323982f018efcc688  /u8/hkpeprah/cs452-microkern/include/sensor_server.h
21519cac55397ad4c69970d00978d733  /u8/hkpeprah/cs452-microkern/include/server.h
ce8542472ac5ee1d570df56c365f4f12  /u8/hkpeprah/cs452-microkern/include/shell.h
1512795a5385a5e631e672e6d97fe228  /u8/hkpeprah/cs452-microkern/include/sl.h
c3088482a9ce253fdccb2575abef3246  /u8/hkpeprah/cs452-microkern/include/track_node.h
f7ef9d4c517b0412f48001a86e2ae72c  /u8/hkpeprah/cs452-microkern/include/train.h
1f7664d69e067ddf7dd63bb5f795c95d  /u8/hkpeprah/cs452-microkern/include/track_data.h
3bcdc8bfd4dab89d14d7782d48b907a5  /u8/hkpeprah/cs452-microkern/include/train_speed.h
7e5aa7e7c4ec675928bbe18e6925ecae  /u8/hkpeprah/cs452-microkern/include/train_task.h
89e35e2cf35d247f24787d12aaaa55e3  /u8/hkpeprah/cs452-microkern/include/uart.h
ec340a6df8d3a5537318f799b3824e3f  /u8/hkpeprah/cs452-microkern/include/util.h
8402c682ff31b15549689baa00ea45b6  /u8/hkpeprah/cs452-microkern/lib/libbwio.a
a2ca387ea8e0c6ee6945940dba79f28e  /u8/hkpeprah/cs452-microkern/Makefile
8da00e714e5f8f92edb2fdce848f750b  /u8/hkpeprah/cs452-microkern/src/mem.c
f0ae274d5e80356c7ed6a5933177136c  /u8/hkpeprah/cs452-microkern/src/orex.ld
ec93960c40d851aab4a4035658c1599d  /u8/hkpeprah/cs452-microkern/src/string.c
97270e6142e040f3041388491e2af2dd  /u8/hkpeprah/cs452-microkern/src/syscall.c
87a6557aeb942ef877eaf85e4653244d  /u8/hkpeprah/cs452-microkern/src/task.c
9f2f4e7adf88f06e580e271394edcfd8  /u8/hkpeprah/cs452-microkern/src/k_syscall.c
15a593a056e79d16ebcfa0c4b9e797fb  /u8/hkpeprah/cs452-microkern/src/kernel.c
0d41b435c56be1e3da0913253c353e0c  /u8/hkpeprah/cs452-microkern/src/main.c
5a1a6706d5d6adf25e1378eea2afc648  /u8/hkpeprah/cs452-microkern/src/stdio.c
18d5bbcbd3c6a18215642750f7883bed  /u8/hkpeprah/cs452-microkern/src/stdlib.c
fb02e0ba097afaabae2927e17634270b  /u8/hkpeprah/cs452-microkern/src/hash.c
82366d900b909a582d51062d4945fb83  /u8/hkpeprah/cs452-microkern/src/idle.c
bc8e7b0214401e5e8d53295ed9d84ed1  /u8/hkpeprah/cs452-microkern/src/interrupt.c
7c409e716ad13a48ff8b7de0a98a4964  /u8/hkpeprah/cs452-microkern/src/logger.c
```

```
1c567e71a3e68cc7940831dfe2bfbe36  /u8/hkpeprah/cs452-microkern/src/path.c
fa6ec92dd89273313a0cc47b3cda1b94  /u8/hkpeprah/cs452-microkern/src/path.h
ab4b0499e37884b85ff6e4b30a5c1d4e  /u8/hkpeprah/cs452-microkern/src/random.c
9453e45434cea32c42a71bbec56c9bf8  /u8/hkpeprah/cs452-microkern/src/servers/clock.c
b94492e56975c6e54412139d3a6ba492  /u8/hkpeprah/cs452-microkern/src/servers/controller.c
be6f5004650bd42670811b4889ad4ae9  /u8/hkpeprah/cs452-microkern/src/servers/sensor_server.c
cc8ef3fd964f0a056444b32cd753eb74  /u8/hkpeprah/cs452-microkern/src/servers/server.c
580a5d1969f25fd509abe3a5e530777a  /u8/hkpeprah/cs452-microkern/src/servers/uart.c
b0f93cbd90b6f9378dbf2591d9d10387  /u8/hkpeprah/cs452-microkern/src/track_data.c
7b1ef60d5774d3e466b4170b7ec08a84  /u8/hkpeprah/cs452-microkern/src/tasks/null.c
e410a8b26d9827107f987c0a6cb9e23a  /u8/hkpeprah/cs452-microkern/src/tasks/rps.c
4e5d94fb0dff67725964aab0961913e8  /u8/hkpeprah/cs452-microkern/src/tasks/shell.c
3147221764035267d64c481374950044  /u8/hkpeprah/cs452-microkern/src/tasks/train_task.c
fa5c39d4ebb0f792127dbccdd419c072  /u8/hkpeprah/cs452-microkern/src/tasks/utasks.c
0f372fa76f789d4d54d641035bbf7c5f  /u8/hkpeprah/cs452-microkern/src/train.c
092256cc9e551ee62e900d5bbd71bb8b  /u8/hkpeprah/cs452-microkern/src/train_speed.c
4066b901583ac62dc7e1fd3bb8c6d92a  /u8/hkpeprah/cs452-microkern/src/ui/calibration.c
68f2a5a042592a3c06294aba007b9fc1  /u8/hkpeprah/cs452-microkern/src/ui/term.c
609e301851dc2196f4f7e6416148fa0a  /u8/hkpeprah/cs452-microkern/src/util.c
0c9c7e1968f6fa419d4342138e197149  /u8/hkpeprah/cs452-microkern/tests/tasks1.c
5dfb150de46a48f1cf5907abf7151196  /u8/hkpeprah/cs452-microkern/tests/fig8test.c
8e52c7e550ffc488a5a49616da1d099c  /u8/hkpeprah/cs452-microkern/tests/log_viewer.c
6ee1f24aac0ad5f8c8ca40b74954bd87  /u8/hkpeprah/cs452-microkern/tests/logtest.c
9595204a548aa5e2dc24a46a378e6857  /u8/hkpeprah/cs452-microkern/tests/modeltest.c
df30f4165a4e3ed0c4059e9ea3aa14dd  /u8/hkpeprah/cs452-microkern/tests/pathtest.c
7bdc3688319a8fc4f27da5e99f76c313  /u8/hkpeprah/cs452-microkern/tests/perf_test.c
5f5461846b8af8c1a18b82cc4c85176a  /u8/hkpeprah/cs452-microkern/tests/sensortest.c
cc584ca632e63d06fff15334d78bacd8  /u8/hkpeprah/cs452-microkern/tests/sltest.c
1f57d189e7320c01c9fbc862d177042b  /u8/hkpeprah/cs452-microkern/tests/speedtest.c
f1f8c6fd425032444162c32949728fee  /u8/hkpeprah/cs452-microkern/tests/tasks2.c
8ee655142dc66971e1479e506155655b  /u8/hkpeprah/cs452-microkern/tests/tasks3.c
0200506bcfd95aff7fa73153931ee2a0  /u8/hkpeprah/cs452-microkern/tests/test_create_destroy.c
82e6a6dcc6b5a9ec57de149f94d6948c  /u8/hkpeprah/cs452-microkern/tests/traintest.c
7393bef53f92842bc08ba2dcd0ac530d  /u8/hkpeprah/cs452-microkern/tests/uarttest.c
```