

Data Logging

Harjit Singh

Problem Statement

- When the mouse is running, and it does something unexpected, I've been left guessing "what just happened?"
- If you are really good, you can tell what happened.
- Or if you are really lucky, you have a reproducible situation.
- My strategy is to store what the mouse saw/did and then from that determine the issue.

Examples

- At practice, during APEC 2014, the maze had a 15 cell straight followed by a 180 degree turn.
 - The mouse would navigate down the straight but half way through the turn, do a “spin out”.
 - I had logging turned on and was able to see that the rotational position went from positive to negative!
 - The servo tried to track it but then couldn't and shut down the mouse as “must be crashed”.
 - The sign change was caused by an overflow in the turn profiler.
 - The logging enabled me to see what happened without trying to reproduce it.
 - I was able to focus on implementing a fix, rather than spending time to instrument and reproduce the issue.

Traditional Approach

- Select a few variables and setup triggers to enable logging in certain situations.
 - The log data is stored in RAM.
 - This requires that each time logging is to be done, the user has to re-setup the logging code.
 - If the processor crashes, the log is lost.
- If something else of interest happens and you weren't logging it, you are out of luck.

Proposed Approach

- Log
 - All the time.
 - Log to flash – it is non-volatile.

Logging to RAM VS Flash

- RAM logging is “easy”:
 - Declare an array.
 - Write data to it.
 - Write speed is multi-megabytes a second – not that you have enough RAM to do this.
 - When it is time to transfer it to the host, read the array and write it out over the serial port.
- Flash logging requires:
 - That you copy the data from a buffer to the flash device.
 - With proper part selection, the write speed can be 500KB/s.
 - You implement a simple file system.

Flash Device Options

- Serial Flash
 - There are parts that can program 256 bytes in 0.3 ms or 512 bytes in 0.4 ms.
 - 16 MB and 32 MB parts available in 6 mm X 8 mm package.
 - Erasing full device takes 170 seconds.
- FRAM
 - Neat write feature – send the write command and address, **once**, and then stream data!
 - No Erase required.
 - Largest size I've seen is 256 KB.
- Internal Flash
 - Flash needs to be multi-bank so that you can execute code from part of it while you are writing to a different part.
 - Check that it is fast enough.
 - Check endurance - typically 10K write cycles, which is enough.
 - Size depends on the part you are using.

Flash Information

- For the APEC 2014 contest, the log file size was 2.7MB.
- I use Micron N25Q128 which is a 16MB flash which can program 256 bytes in 0.5 ms.
- In the future, I might use the Spansion S25FL256S device because it has a lower erase time.

What To Log?

Command	Description / Payload
PrintString	Log the NULL terminated string.
Error	Error code. For example: Battery voltage is too low.
MarkedWall	Which walls were marked.
CellLocation	Current cell coordinate.
Orient	Current orientation.
Path	Path the mouse is attempting to execute.
ProfServo	FwdPosErr, DesFwdVel, DesRotPos, RotPosErr, Battery Voltage.
Sensor	Left Front Sensor, Right Front Sensor, Left Diag Sensor, Right Diag Sensor.
Corrector	Lateral and longitudinal corrector values.
Run	Run #, FwdAccel, MaxFwdVel, MaxDiagFwdVel, Run mode.
RunInfo	Run success/fail and run time.

Sensor or Motion Velocity & Error graphs

The screenshot shows the MouseDash v0.1 interface. On the left is a control panel with buttons for 'Load1', 'Load2', and 'Debug', and various checkboxes for 'ToCenter', 'Touched', 'Sensor', 'Forward', 'Rotation', and 'Unused'. Below these are input fields for 'Position', 'Angle', 'Vbat', and '# of Points To Graph'. A circular gauge shows the 'Mouse Angle' with a needle pointing to 360 degrees. At the bottom left is a 'Cell selector' table with columns 'turn', 'cell', 'cell', 'cell', 'cell'. The top right shows a 'Parse Progress' indicator at 0. The center contains two graphs: 'Front Sensors' and 'Diagonal Sensors', both plotting Power and Distance against Time. The right side features a maze grid with a red path and a yellow arrow. At the bottom is a data table with columns for sensor readings and a 'Comment' column.

turn	cell	cell	cell	cell
0				

mm	LFP	LDP	RDP	RFP	LFD	LDD	RDD	RFD	FVel	FErr	RVel	RPos	RErr	Comment

Current maze information

Current Heading

Cell selector

Motion & Sensor numerical information

www.Bandicam.com

File

Mouse State

Load1 Load2

Debug ToCenter

Touched

MotionSt Sensor

DiagSt Forward

SysRgn Sensor

Rotation

Unused

Position Offset Det

Angle Index

VBat CurKY

of Points To Graph Orient

Mode Grid Index

360 45 90 135 180 225 270 315

Mouse Angle

W	LDP	LDF	RCF	RFP	LFD	LDF	RDF	RFP	W	LFP	RFP	RDF	W	Connect

W	Cx12	Cx22	Cx12	Cx22

APEC 2014: First Speed Run

Mouse weaving down the maze.

MouseDash v0.1

File | Parse Progress: 100

MouseState

Load1 | Load2 | ToCenter | Debug | Touched

MotionSt: Sensor
DiagSt: Forward
SysFlags: Sensor | Rotation | Unused

Position: None | None | Index: 112270
Angle: None | None | CurXY: 50
VBat: 7.75 | Orient: North
of Points To Graph: 1000 | Mode: Centert
Grid Index: 116716

Front Sensors

Power vs. Time (10⁻³)

Rotation Profiler And Servo

Velocity (deg/s) vs. Error (deg) vs. Time (10⁻³)

Maze Grid

mm	LFP	LDP	RDP	RFP	LFD	LDD	RDD	RFD	FVel	FErr	RVel	RPos	RErr	Comment
58	32	944	784	144	0.0	81.5	84.5	0.0	2.00	0.0	0.00	355.3	-1.1	Cell Location: 50
60	32	944	784	80	0.0	81.8	84.8	0.0	2.00	0.0	0.01	355.3	-1.1	LatCorr: Do
62	16	976	800	64	0.0	80.0	82.3	0.0	2.00	0.0	0.02	355.3	-1.1	Correction
64	16	976	800	96	0.0	80.5	82.5	0.0	2.00	0.0	0.00	355.3	-1.1	Correction
66	64	976	784	80	0.0	80.8	82.5	0.0	2.00	0.0	0.02	355.3	-1.1	Correction
68	16	992	768	96	0.0	79.5	83.8	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
70	16	976	784	112	0.0	80.0	84.3	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
72	0	992	784	80	0.0	79.8	83.3	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
74	80	976	768	80	0.0	79.3	83.0	0.0	2.00	0.0	0.00	355.3	-1.1	Correction
76	32	976	768	96	0.0	79.5	85.5	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
78	48	992	784	128	0.0	79.5	83.5	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
80	0	992	768	80	0.0	79.3	83.5	0.0	2.00	0.0	0.01	355.3	-1.1	Correction
82	96	992	752	80	0.0	79.3	84.8	0.0	2.00	0.0	0.01	355.4	-1.1	Correction
84	32	976	768	112	0.0	79.5	83.8	0.0	2.00	0.0	0.00	355.4	-1.1	Correction
86	32	976	752	144	0.0	80.5	84.3	0.0	2.00	0.0	0.01	355.4	-1.1	Correction
88	0	992	752	32	0.0	79.3	83.3	0.0	2.00	0.0	0.00	355.4	-1.1	Correction
90	64	992	752	112	0.0	79.3	84.5	0.0	2.00	0.0	0.01	355.4	-1.1	Correction

Log Window:

```

Feb2b
vParseLog: Unknown command d7 at
Feb2c
vParseLog: Unknown command d6 at
Feb2d
LatCorr: Correction 61.8
vParseLog: Unknown command d3 at
Feb30
vParseLog: Unknown command d4 at
Feb31
vParseLog: Unknown command c4 at
Feb32
vParseLog: Unknown command c6 at
Feb33
vParseLog: Unknown command a6 at
Feb34
vParseLog: Unknown command a7 at
Feb35
vParseLog: Unknown command 97 at
Feb36
vParseLog: Unknown command 98 at
Feb37
vParseLog: Unknown command 88 at
Feb38
    
```

APEC 2014: Second Speed Run

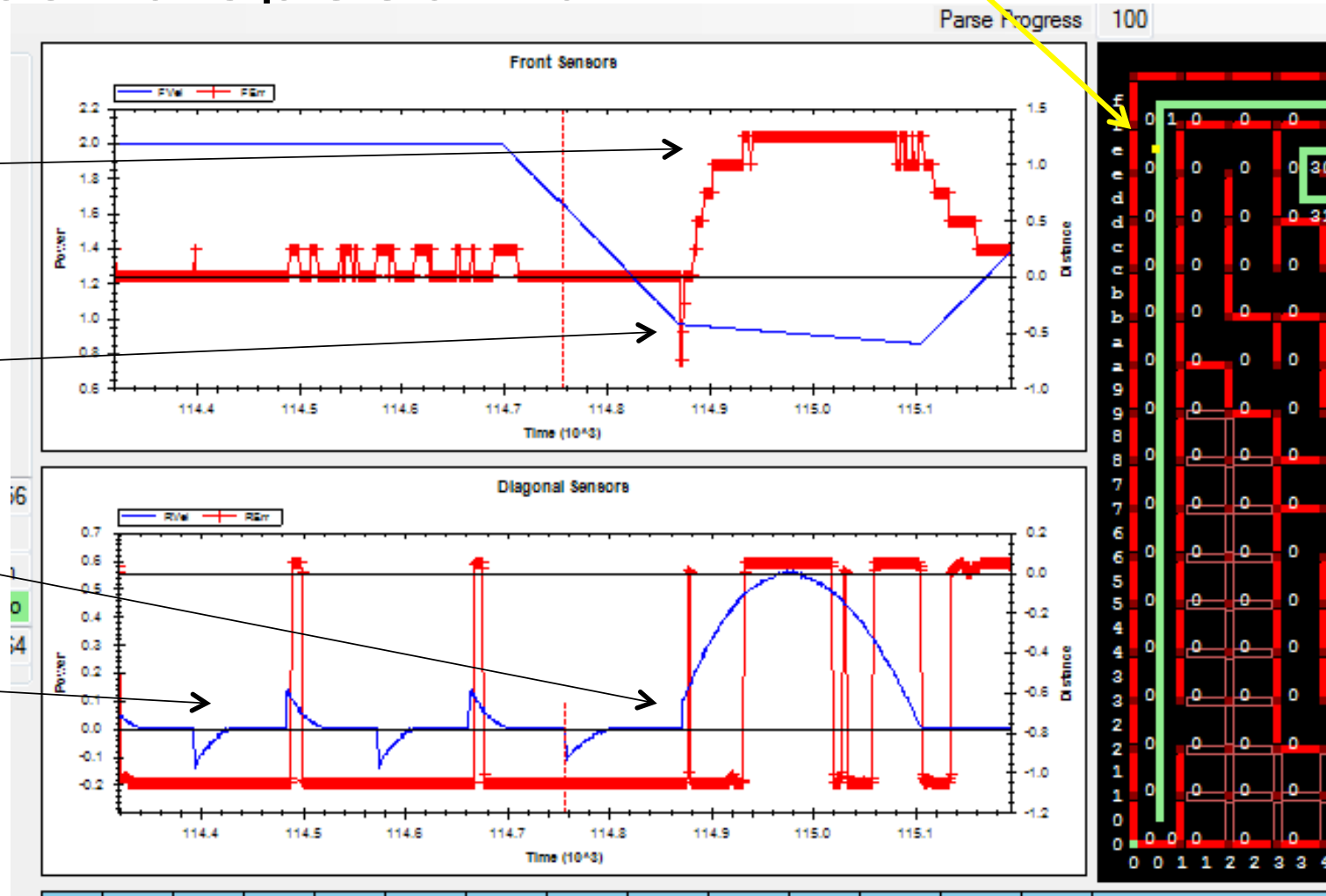
Mouse is here.

Forward Servo Error is 1.25 mm.

Decelerating through the turn.

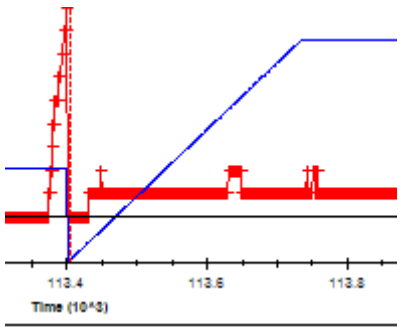
Turn started LATE!

Still Weaving

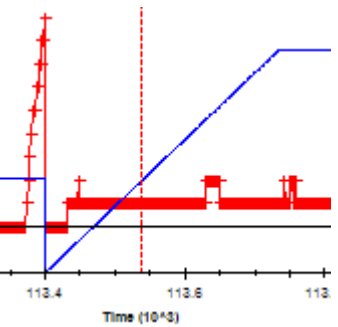


Battery Voltage During Motion

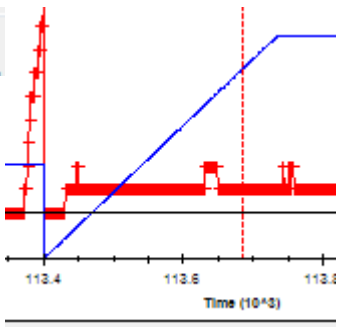
Angle
 VBat
 # of Points To Graph



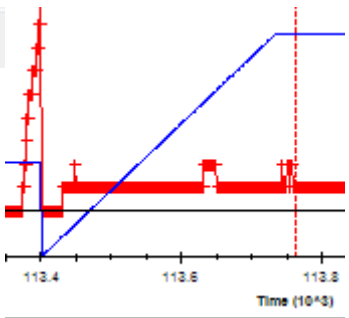
VBat



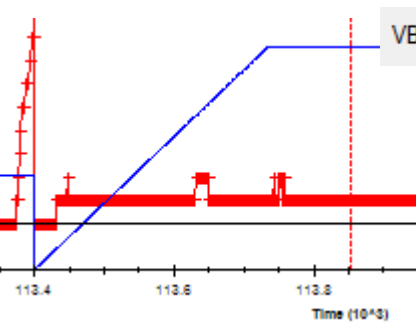
VBat



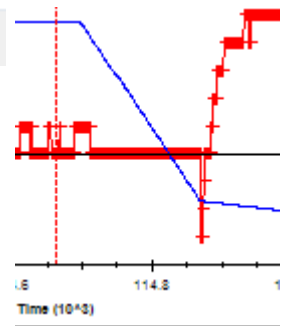
VBat



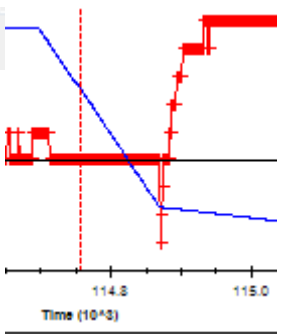
VBat



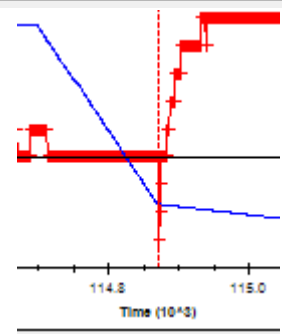
VBat



VBat

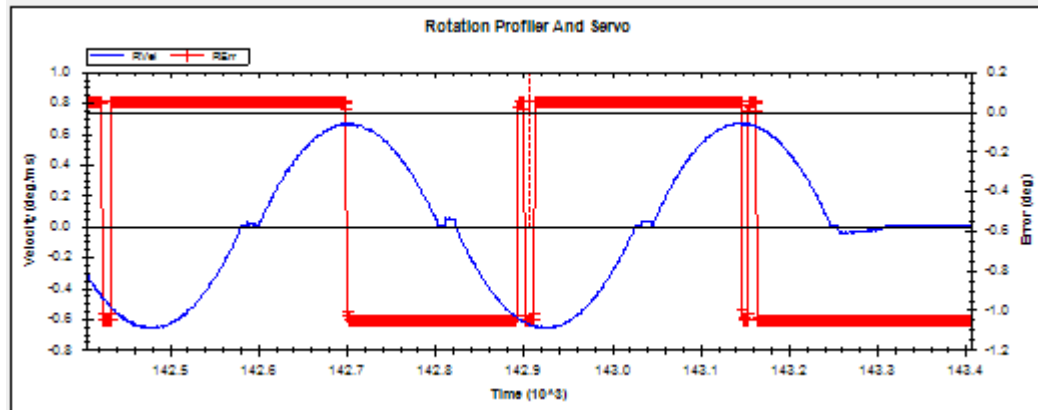
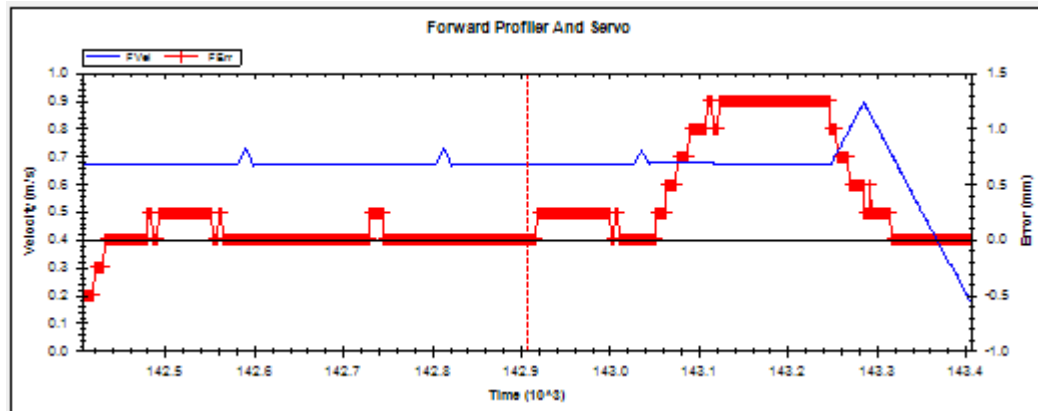


VBat



APEC 2014: Oddities

Notice acceleration blips between turns.



Mouse was decelerating, then it decided to turn and for some reason the deceleration is insane. And a little bit later, during the turn, it accelerates!

