PROJECT 2 REPORT

Michael Meli
ECE 561

Optimizations – Inclination Measurement and Calculations

Optimizations

The following table explains the optimizations performed and the corresponding execution time improvements:

Optimization Performed	Profile Ticks (10,000 runs)	Improvement (in ticks)	Time Per Single Run (µs)
Initial (no optimization)	10418	------	1041.8
Force optimization for time and highest optimization level (O3) using compiler options	10415	3	1041.5
Set the --fpmode=fast (fast floating point mode) compiler option	6674	3741	667.4
Change the calculation of atan2() to atan2f() and sqrt() to sqrtf(), which operate on floating point data instead of converting to double	3328	3346	332.8
Changed the I^2C clock speed to 1.2MHz using a SCL divider of 20 (0x00)	2051	1277	205.1
Square root approximation using online guide [4] given in the project spec	1992	59	199.2
Put parentheses around the expression “180/PI” to allow the compiler to precompute the value	1955	37	195.5
atan2 approximation using the polynomial approximations given by [1] and [2] in the project spec	1538	417	153.8

As the table above shows, the final runtime of an inclination measurement and calculation was approximately 153.8 microseconds.

Profiling Results at Each Step

The following tables list the top 5 functions in execution time profile for each of the above steps in the optimization process.

Initial (no optimization):

Function Name	Ticks
i2c_wait	3143
__aeabi_dmul	2603
_double_epilogue	1191
__aeabi_ddiv	788
__aeabi_llsl	418

Forced optimization for time and highest optimization level (O3):

Function Name	Ticks
i2c_wait	3192
__aeabi_dmul	2890
_double_epilogue	1070
__aeabi_ddiv	923
__aeabi_llsl	408

Setting the --fpmode=fast compiler option:

Function Name	Ticks
i2c_wait	2499
__aeabi_dmul	1775
_double_epilogue	355
_dsqrt	334
__aeabi_dadd	320

Changing the use of atan2() to atan2f() and sqrt() to sqrtf() to operate with floats:

Function Name	Ticks
i2c_wait	2837
_fsqrt	135
__aeabi_fmul	120
__aeabi_fadd	90
__aeabi_fdiv	65

Increasing the I^{2}C clock speed to the fastest possible:

Function Name	Ticks
i2c_wait	979
__aeabi_fmul	227
__aeabi_fdiv	124
__aeabi_fadd	123
_fsqrt	102

Implementing a square root approximation:

Function Name	Ticks
i2c_wait	909
__aeabi_fmul	376
__aeabi_fadd	204
__aeabi_fdiv	118
i2c_repeated_read	50

Put parentheses around (180/PI) to allow the compiler to precompute the value:

Function Name	Ticks
i2c_wait	842
__aeabi_fadd	268
__aeabi_fmul	131
i2c_repeated_read	129
__aeabi_fdiv	124

Implementing an atan2 approximation (**final** results):

Function Name	Ticks
i2c_wait	894
__aeabi_fdiv	401
i2c_read_setup	143
i2c_repeated_read	53
read_full_xyz	34

Optimizations – Magnetometer Calculations to Determine Tilt-Compensated Heading

Optimizations

The following table explains the optimizations performed and the corresponding execution time improvements:

Optimization Performed	Profile Ticks (10,000 runs)	Improvement (in ticks)	Time Per Single Run (µs)
Initial (only optimizations were setting the –O3 highest optimization flag and enabling optimize for time)	8750	------	875
Set the --fpmode=fast (fast floating point mode) compiler option	2098	6652	209.8
Change the calculation of cos(), sin(), and atan2() to cosf(), sinf(), and atan2f(), which operate on floating point data instead of converting to double	2056	42	205.6
Reused values for repeated trigonometric functions to reduce recalculation of the same values	1666	390	166.6
atan2 approximation using the polynomial approximations given by [1] and [2] in the project spec	1032	634	103.2
Implemented a cos approximation using a second degree Taylor Series polynomial	871	161	87.1
Implemented a sin approximation using a third degree Taylor Series polynomial	735	136	73.5

As the table above shows, the final runtime of a tilt-compensated heading calculation was approximately 73.5 microseconds.

Profiling Results at Each Step

The following tables list the top 5 functions in execution time profile for each of the above steps in the optimization process.

Initial (only optimizations were setting the O3 highest optimization option and enabling optimize for time):

Function Name	Ticks
__aeabi_dmul	10203
__aeabi_dadd	2313
_double_epilogue	1469
__aeabi_llsl	1415
__aeabi_ddiv	1346

Setting the --fpmode=fast (fast floating point mode) compiler option:

Function Name	Ticks
__aeabi_fmul	652
__aeabi_fadd	320
sinf	204
__aeabi_fdiv	110
cosf	103

Changing the calculations of cos, sin, and atan2 to cosf, sinf, and atan2f to operate with floats:

Function Name	Ticks
__aeabi_fmul	954
__aeabi_fadd	501
sinf	144
_float_round	114
cosf	113

Reusing values of calculations that were performed multiple times:

Function Name	Ticks
__aeabi_fmul	424
__aeabi_fadd	241
sinf	119
atan2f	93
__aeabi_fdiv	59

Implementing an atan2 approximation:

Function Name	Ticks
__aeabi_fmul	421
__aeabi_fadd	278
cosf	84
__aeabi_fdiv	84
sinf	57

Implementing a cos approximation using a Taylor Series polynomial:

Function Name	Ticks
__aeabi_fmul	332
__aeabi_fadd	160
sinf	111
calc_tilt_comp_heading	111
__aeabi_fdiv	80

Implementing a sin approximation using a Taylor Series polynomial (**final** result):

Function Name	Ticks
__aeabi_fmul	295
__aeabi_fadd	147
__aeabi_fdiv	101
sin_approx	93
calc_tilt_comp_heading	93

Development Effort Tracking

Estimated person-hours required: 25 hours

Actual person-hours spent: 22 hours

A good deal of development time (around 10 hours) was spent attempting to fix the error where the I^2C bus would lock up at high baud rates. Another approximately 5 hours was spent attempting to set up SPI communications with the V2Xe compass, which was unsuccessful. The rest of the time was spent optimizing code for speed.