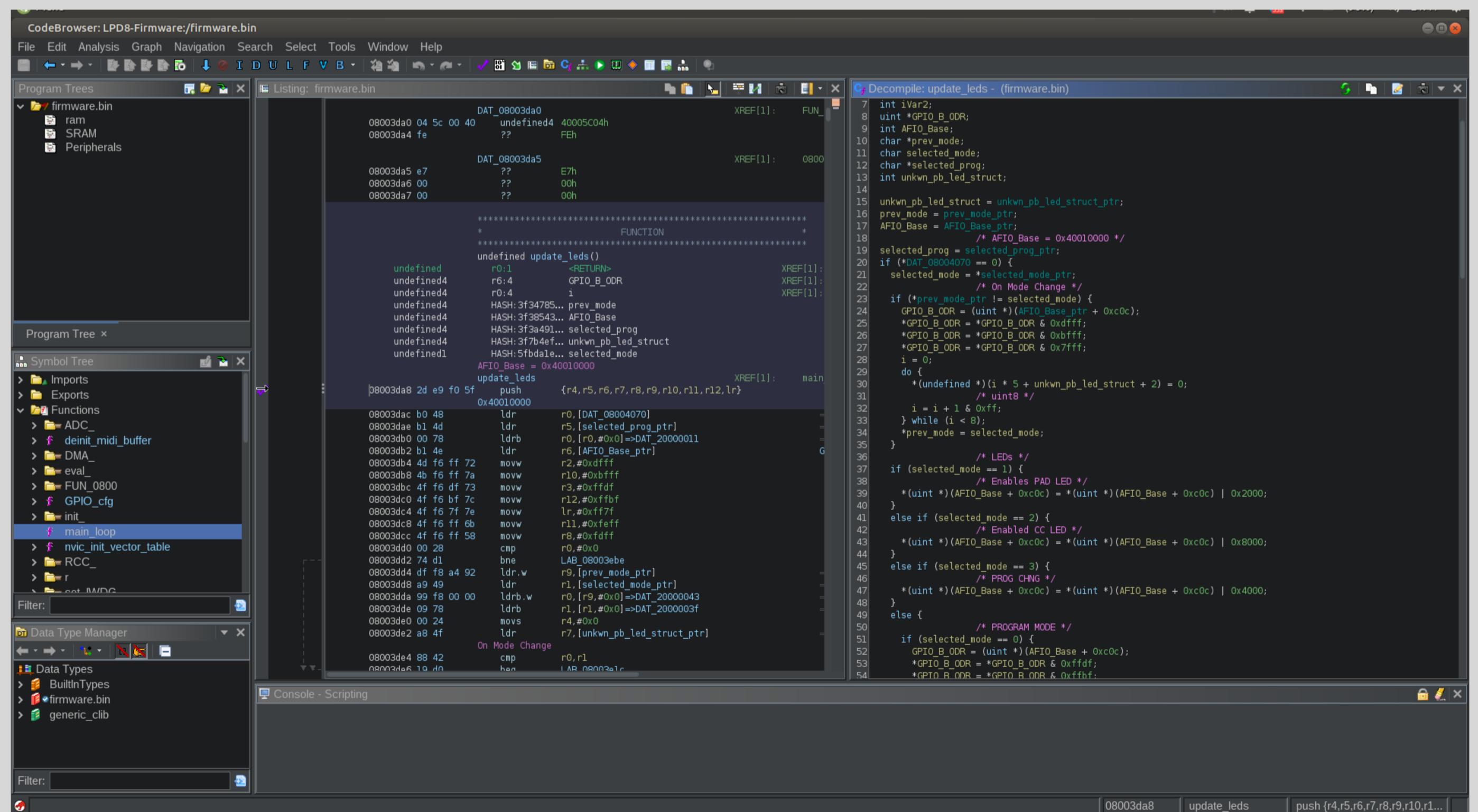


# 3. Reverse Engineering The Firmware

Reverse engineering the firmware is by far the most time-consuming task in this whole project. As of the time of writing, about one-third of the firmware code has been reverse-engineered. That said, this third covers arguably the most important and frequently used routines of the firmware (e.g., handling of pads, knobs, and LEDs). Ghidra is used to reverse engineer the firmware. Unfortunately, when reverse engineering embedded firmware, the code analysis lacks context, starting with minimal information. However, the bare-metal nature significantly reduces abstraction. This means that once a context is established, the reverse engineering efforts become significantly easier compared to, for instance, a multi-threaded application running on top of an OS. It's also worth mentioning that the ability to step through assembly instructions, inspect and alter memory, and modify register values via the SWD interface and GDB provided enormous help.



The screenshot shows the Ghidra software interface with the following windows:

- File Browser:** Shows the file structure with 'firmware.bin' selected.
- Program Tree:** Shows the program tree with nodes like 'Imports', 'Exports', 'Functions', and 'main\_loop'.
- Symbol Tree:** Shows symbols such as 'ADC\_cfg', 'DMA\_', 'eval', 'FUN\_0800', 'GPIO\_cfg', 'init', 'main', 'nvic\_init\_vector\_table', 'RCC', and 'set\_IWDG'.
- Code Browser:** Shows the assembly listing for 'firmware.bin' with the current function being 'update\_leds'. The assembly code includes instructions like 'ldr r0, [AFIO\_Base + 0xc0c]', 'movw r2, #0xffff', and 'bne LAB\_08003be'.
- Decompiler:** Shows the decompiled C-like pseudocode for the 'update\_leds' function.
- Data Type Manager:** Shows the data type manager with entries for 'BuiltInTypes', 'firmware.bin', and 'generic\_clib'.
- Console - Scripting:** Shows the command-line interface with the command 'push {r4,r5,r6,r7,r8,r9,r10,r11,r12,lr}'.



## Currently Reverse-Engineered functions

SysTick_Handler	RCC_set_SW
nvic_init_vector_table	RCC_set_USBPRE_BB
SYSTICK_set_RELOAD	RCC_get_SWS
SYSTICK_set_TICKINT	RCC_set_PLLON_BB
SYSTICK_action	RCC_set_APB2ENR
RCC_get_clock_freqs	RCC_set_AHBENR
init_SYSTICK	ADC_set_SMPR_SQR
set_IWDG_KR	init_analog
set_IWDG_PR	GPIOs_cfg
set_IWDG_RLR	write_midi_buffer
init_IWDG	eval_knobs
ADC_set_DMA	read_analog
ADC_set_ADON	eval_pads
ADC_set_RSTCAL	read_mode_pbs
ADC_read_RSTCAL	eval_mode_pbs
ADC_set_CAL	rst_pads
ADC_read_CAL	update_leds
ADC_set_EXTTRIG_SWSTART	reload_IWDG
DMA_CCR_set_EN	init_midi_buffer
DMA_clear_IFCR	deinit_midi_buffer
DMA_read_ISR	
DMA_set_IFCR	