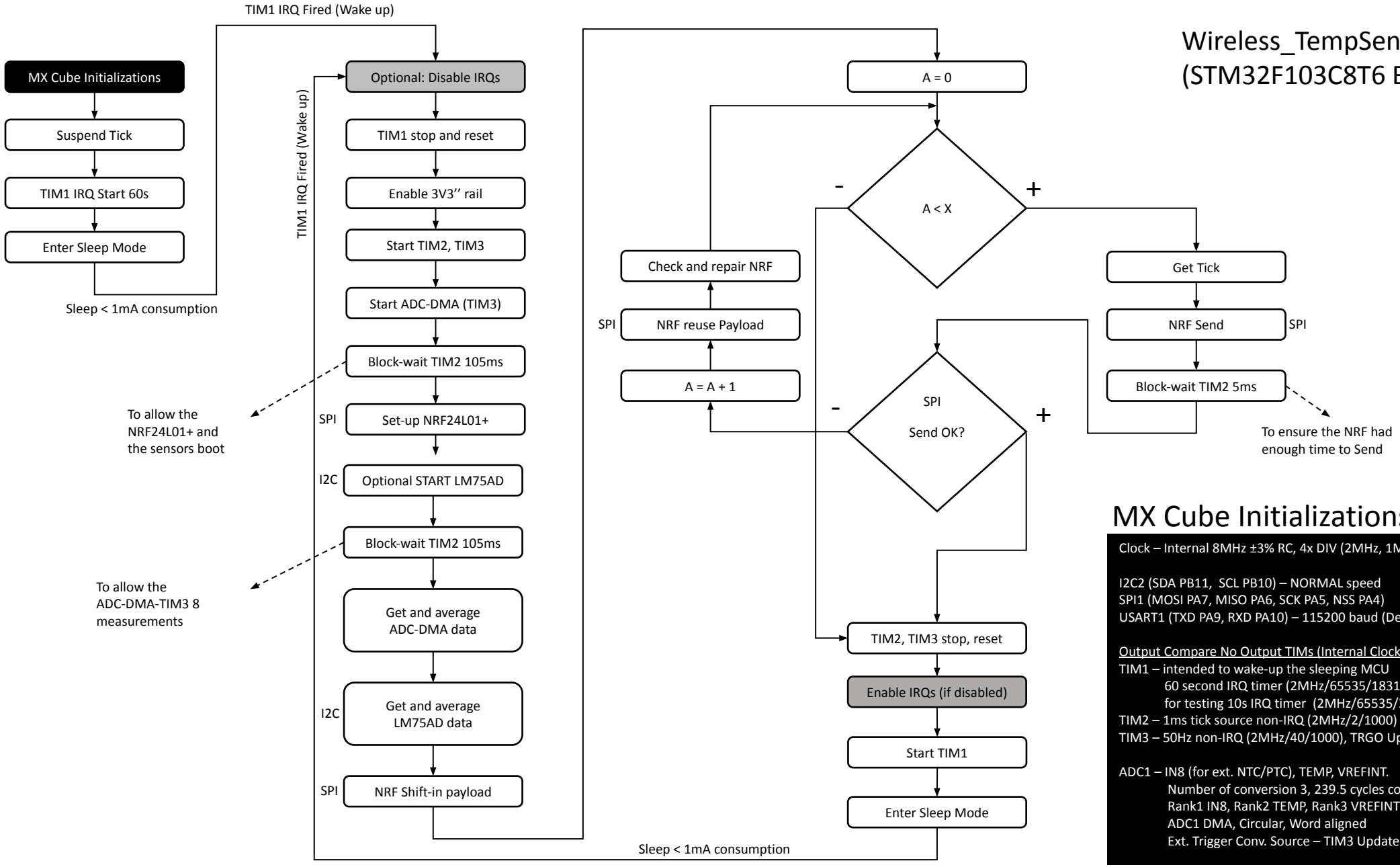
Wireless_TempSens client (STM32F103C8T6 BluePill)



MX Cube Initializations

Clock – Internal 8MHz ±3% RC, 4x DIV (2MHz, 1MHz ADC)

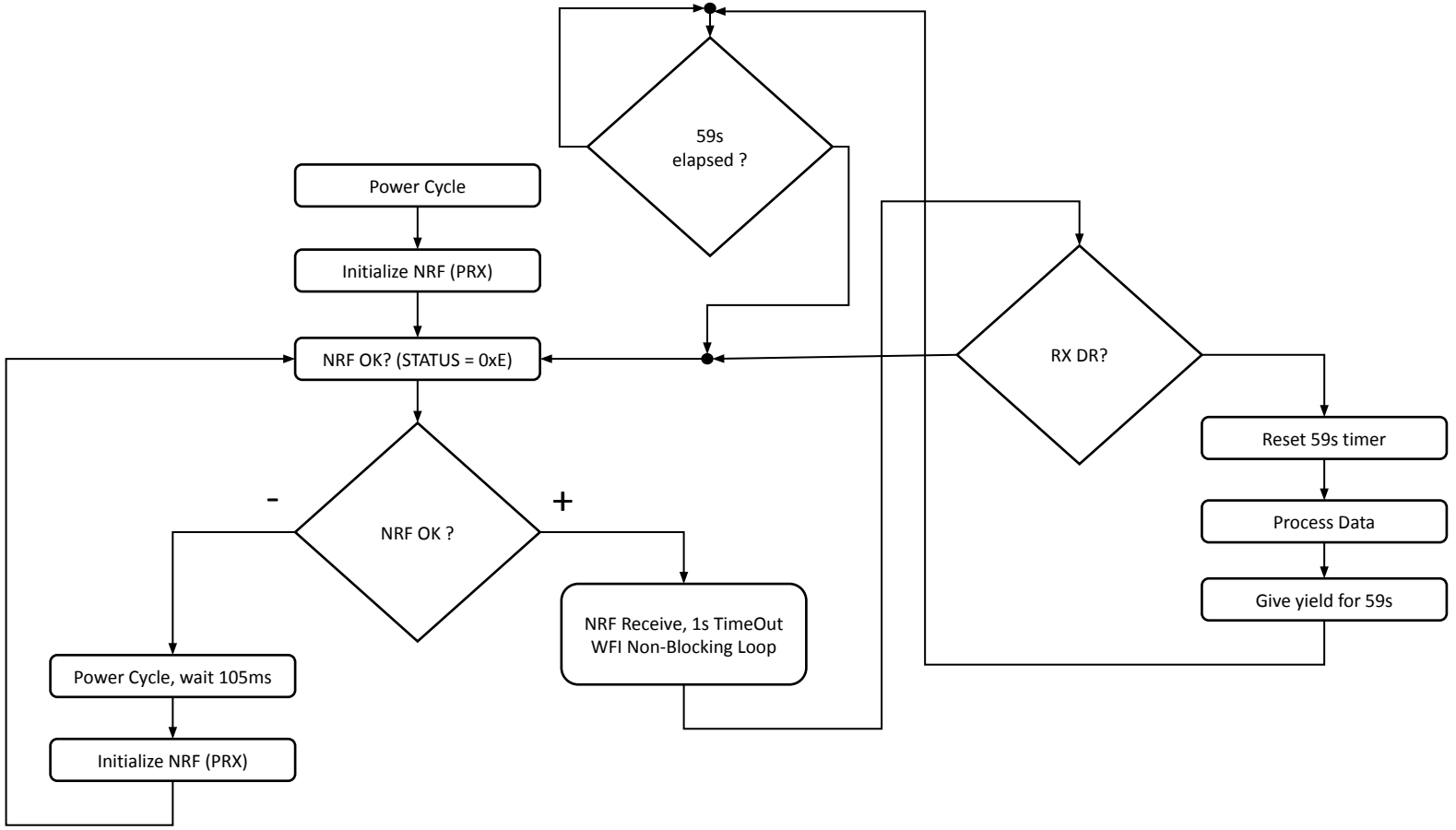
I2C2 (SDA PB11, SCL PB10) – NORMAL speed
SPI1 (MOSI PA7, MISO PA6, SCK PA5, NSS PA4)
USART1 (TXD PA9, RXD PA10) – 115200 baud (Debug)

Output Compare No Output TIMs (Internal Clock Sources)
TIM1 – intended to wake-up the sleeping MCU
60 second IRQ timer (2MHz/65535/1831)
for testing 10s IRQ timer (2MHz/65535/305)
TIM2 – 1ms tick source non-IRQ (2MHz/2/1000)
TIM3 – 50Hz non-IRQ (2MHz/40/1000), TRGO Update Event (ADC)

ADC1 – IN8 (for ext. NTC/PTC), TEMP, VREFINT.
Number of conversion 3, 239.5 cycles conversion
Rank1 IN8, Rank2 TEMP, Rank3 VREFINT
ADC1 DMA, Circular, Word aligned
Ext. Trigger Conv. Source – TIM3 Update Event

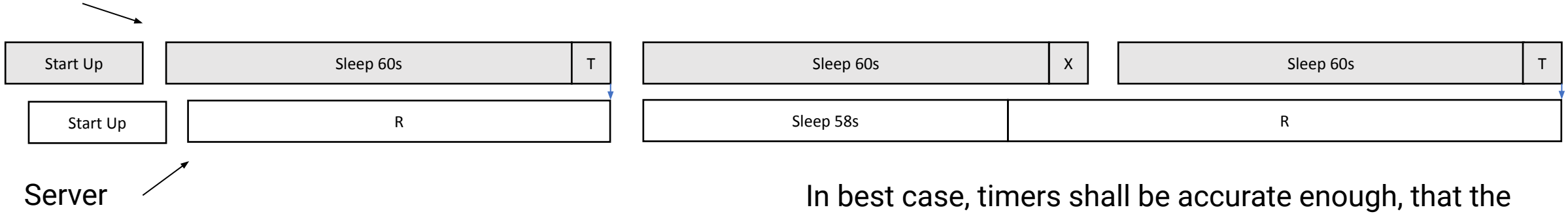
GPIO PC13 Output Push-Pull, Low Speed, Default High (LED)

Wireless_TempSens server
STM32F767ZI (Nucleo-144)
FreeRTOS based (single task)



Wireless_TempSens client-server timing

R – Receive task
T – Transmit task
X – error transmit task



The client “does not care” whether is synced with the server or not, simply tries to send 1x per 60 seconds something. This fact shall reduce the system’s power consumption.

The server is someone who tries to be synced to have a rest for the 58 seconds. When is the server not synced, it tries to just continuously receive with until it reaches synced state (long enough to cover problem when clients start later but the server started to listen earlier). If there is a 60 seconds interval, the server must start to receive a bit earlier, e.g., 2 second (58 seconds).

In best case, timers shall be accurate enough, that the server is not slower than the client and thus is not late for receiving and therefore does not un-sync every time in order to silly spend another 60 seconds by listening to get “synced” again. Also, the client must not be faster than agreed, because the server will seem to be slow.

Best approach would be to get the clock tolerance characteristics of the both participants and have the listening gap (in this case 1s) increased by the error of the both.
E.g., if client can be ±1s faster and server ± 1s slower than the server, the server must start to listen at time
T_CLIENT – T_ERR_CLIENT – T_ERR_SERVER =
60s – 1s – 1s = 58s (that’s why 58 second)

Timing calculation

Ideal world calculation					
Clock [Hz]	Divider	total cnt	freq [Hz]	period [s]	Meaning
2000000	65535	1831	0,016667	59,997293	TIM1_WAKEUP
2000000	65535	305	0,100059	9,9940875	TIM1_WAKEUP_SHORT
2000000	2	1000	1000	0,001	TIM2_1MS
2000000	40	1000	50	0,02	TIM3_ADC1_TRGO
Tick [Hz]					
1000	1	60000	0,016667	60	RX_START
1000	1	10000	0,1	10	RX_START_SHORT
Error calculation for the client clock source (Internal RC HSI)					
RC Clock [Hz]	Divider	Err. min [%]	Err max [%]		*from datasheet STM32 HSI RC clock..
8000000	4	-3	3		
Real world maximum deviation caculations (client)					
Clock [Hz]	Divider	total cnt	freq [Hz]	period [s]	Meaning
1940000	65535	1831	0,016167	61,852879	TIM1_WAKEUP (MAX)
2060000	65535	1831	0,017167	58,249799	TIM1_WAKEUP (MIN)
Error calculation for the server clock source (BYPASS MCO PA8-U2)					
RC Clock [Hz]	Multiplier	Err. min [ppm]	Err max [%]		* from the Nucleo-144 Schematic NX3225GD-8.000M-EXS00A-CG048
8000000	13	-50	50		*jitter may occur
Real world maximum deviation calculations (server)					
Clock [Hz]	Divider	total cnt	freq [Hz]	period [s]	Meaning
1,04E+08	104	1000	999,995	0,001	1ms TICK MIN
1,04E+08	104	1000	1000,005	0,001	1ms TICK MAX
999,995	1	60000	0,016667	60,0003	RX_START (MIN)
1000,005	1	60000	0,016667	59,9997	RX_START (MAX)

Faster by 1.8s
Slower by 1.9s

Error <1 ms

From the error calculation:

The client might be faster by the 1.8s, thus the
server must start to receive at least 1.8s prior to the period. As the server might be faster by less than a millisecond and the client “faster error” is already rounded, the server error does not need to be taken in account.

The client might be slower by the 1.9s, thus the server must be in the receive state also up to 1.9s after the period point. As the wireless transaction takes circa 5ms and the server timing error is less than 1ms, both these values does not need to be taken in account, because the 1.9s is already a rounded value.

Voltage, current and timings

nRF24L01+

TX max current 0dBm 11.3mA
RX max current 12.3mA

T_Power_ramp_up 100ms
T_Power_on_reset 10.3ms
Vcc = 1.9V to 3.6V
Standby consumption 22uA
Power down consumption 900nA

STM32F103C8T6 (BluePill) with demounted LDO (U1)

sleep mode < 1mA @ 3V3
normal operation ~ 3mA @ 3V3

Temperature Sensors

LM75AD (2°C precision)
Vcc = 2.8V to 5.5V
Normal mode temperature conversion every 100ms
I2C operational 100uA
I2C operational 1mA

AD7414 (0.5° precision)
Vcc = 2.7 to 5.5V
Conversion operation 1.2mA peak
Non-conversion operation peak 0.9mA