Application Report

Choose the Best out of Three Ultra-Low-Power Architectures for Your Wireless Sensor Tags and Data Loggers

Shuang Feng, Milen Stefanov

ABSTRACT

Once a day, once a minute, or even multiple times per second: Designers must find the optimum ultra-low power architecture for wireless sensor tags or data loggers for their use case. Traditional approaches utilize either power supply duty-cycling of the full system or the stand-by modes of the MCU and sensor for extending the battery lifetimes.

Texas Instruments provides a novel implementation, introduced in this report, that enables advanced sensing with a programmable, autonomous ultra-low power Sensor Controller CPU with fast wake-up capability. This dedicated peripheral achieves equal or lower-power consumption than regular MCU stand-by approach while off-loading sensor code from the Main MCU and storing sensor data into a few KB large buffer within its ultra-low-leakage (ULL) always-on SRAM memory. Collecting an array of sensor data and transmitting it at once reduces the wireless protocol overhead to the bare minimum and delivers further power savings compared to multiple shorter packets, each with redundant overhead byte.

The Sensor Controller CPU is found in SimpleLink™ high-performance wireless MCUs such as the CC1352P device.

Table of Contents

List of Figures

List of Tables

Trademarks

SimpleLink™, SmartRF™, BoosterPack™, and EnergyTrace™ are trademarks of Texas Instruments.
Arm®, Cortex®, are registered trademarks of Arm Limited.
All trademarks are the property of their respective owners.

1 Introduction

A wireless sensor tag for asset tracking monitors temperature, humidity, ambient light or other environmental parameters and typically transmits the data over a low-power short-range wireless network to a gateway. Support for global cold chain requires proper handling of time- and temperature-sensitive shipments and enables remote control for food, pharmaceuticals and other cargo with similar shipping requirements.

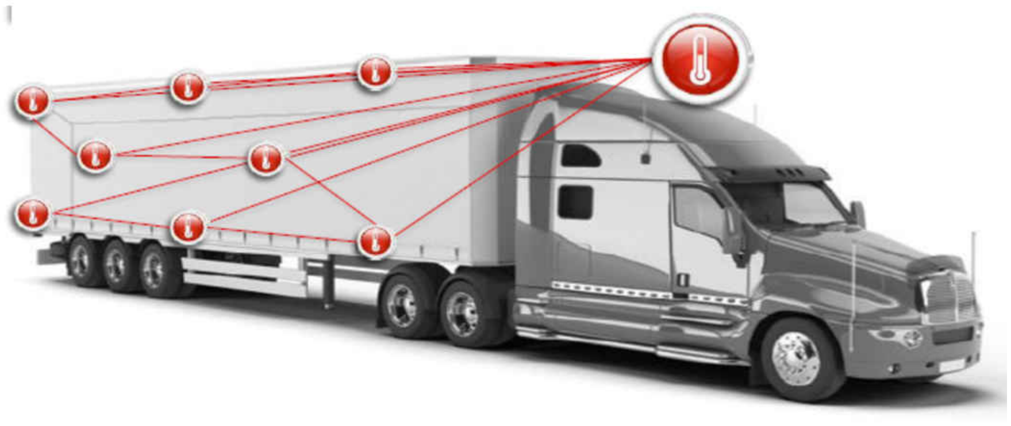


Figure 1-1. Sensor Tags Application in Cold Chain

In the application illustrated in Figure 1-1, the key functions are the readout of the various sensors as well as providing low power wireless connectivity in either Sub-1 GHz, 2.4 GHz or both RF bands. Ultra-low power dual-band wireless MCUs, such as the CC1352P device, integrate the function of a general-purpose microcontroller, a dedicated sensor controller engine peripheral, and wireless connectivity into a single device.

There are two traditional solutions for low-power sensing applications: most common is the standard *Standby* approach, where the MCU is running in a low-power mode when not accessing the sensor device, by maintaining a Real Time Clock (RTC) and its RAM contents all the time.

The second solution is the *Duty-cycle* approach, where the full system power is being switched on and off between two sensor readout cycles and the MCU loses its RAM and RTC contents, while a nano-timer device keeps track of the duty-cycle period. This method achieves ultra-low power in the switched-off (power-off) state, typically in the range of 50 nA, as documented in *Humidity and Temp Sensor Node for Sub-1GHz Star Networks Enabling 10+ Year Coin Cell Battery Life*.

This report introduces a new solution, also called the *Sensor Controller* approach, which is a combination of the *standard* and *duty-cycle* methods, with the following functionalities:

1. The CC1352P MCU runs in *standby* mode, directly powered by the battery. Thus no external devices, such as a nano-timer and an ultra-low leakage load switch, are required.
2. The *Sensor Controller Engine (SCE)* peripheral implements the *I2C communication and data transfer* to and from the sensor (in contrast to the other two solutions, where this task is handled by the main Arm® Cortex®-M4F MCU core).
3. Using one general purpose *IO pin*, the *SCE powers off* the two pullup resistors for the SCL and SDA lines when the sensor is off and no I2C communication is running.

4. SCE *stores the sensor data* in a user-*configurable buffer* (up to 3K bytes long) while in standby and passes a chunk of this buffer data at once to the Arm Cortex-M4F core.
5. The Arm Cortex-M4F core creates a data packet with a user-configurable payload length as per the data protocol being used, and sends it over-the-air.
6. *Optional:* Using one general-purpose *IO pin*, the *SCE powers on* or *off* the sensor device (depends on duty-cycle of the sensor readout).

Figure 1-2 shows the block diagram of the *Sensor Controller* (or SCE-based) approach. Note that even though this diagram looks identical to the popular *Standby* method, there are hardware differences to consider, due to the dedicated SCE ultra-low power peripheral handling the I2C protocol autonomously. Two dedicated IO pins are used to deliver the power supply to the sensor (here the HDC2010 device) and the I2C pullup resistors. These IO pins, controlled by the SCE, enable the duty-cycling of both the sensor power supply pin (if needed) and the resistors, leading to a truly optimized low-power solution.

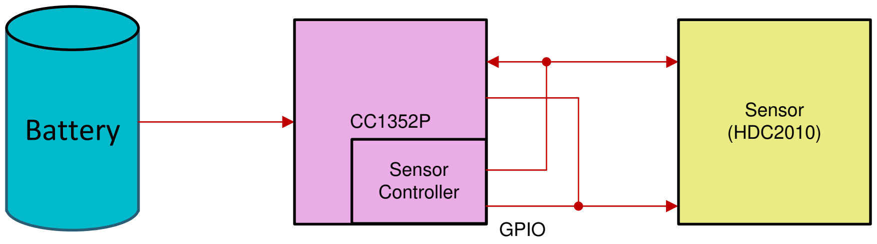


Figure 1-2. SensorTag or Data Logger Solution for Temperature and Humidity With Ultra-low Power (I2C-bus and GPIO Pins)

This report describes a solution, where the CC1352P device utilizes its Sensor Controller Engine to configure the HDC2010 for reading out temperature and humidity data periodically. This solution can be used as a development platform for both temperature and humidity monitoring or data logger devices. The period for monitoring of the temperature and humidity is configurable and spans from milliseconds to seconds if this is only dependent on the functionality of the HDC2010 or HDC2080 devices, as the CC1352P MCU is continuously powered and runs TI RTOS with a real-time clock. The wake up and sensor readout can occur after any user-defined period, as required by the application.

2 System Overview

2.1 Software

The following software is used to create and work with the Sensor Controller CPU in the CC1352P MCU (these revision numbers are used for performance testing in this document).

- Sensor Controller Studio (SCS) v2.5.0
- Code Composer Studio (CCS) 9.1.0
- SDK3.20.00.20 (SDK for the CC1352P device, code example rfPacketTX)
- SWRC367 Firmware

First, the Sensor Controller code for interfacing to the HDC2010 sensor was developed and verified by reading out manufacturer and vendor ID pre-defined values, as well as monitoring the I2C-bus transactions as Figure 3-3 shows.

Next, the Sensor Controller I2C-bus functions are integrated into the existing rfPacketTX code example from the SDK with a focus on ultra-low power and periodic RF transmissions. All data packets were monitored using a CC1350 LaunchPad under SmartRF™ Studio in "Packet RX" mode to capture and visualize the data content.

Finally, a second CCS project was developed, based on rfPacketTX, now using the main MCU for controlling the HDC2010 device over I2C-bus and leaving the SCE peripheral unused.

2.2 Hardware

For the measurements, the following hardware (available at TI.com) was used:

- CC1352P LaunchPad
- BOOSTXL-BASSENSORS (includes the HDC2010 sensor, the TMP116 sensor, and other sensors)

HW modifications: the I2C-bus pullup resistors on the CC1352P LaunchPad were removed as these are in parallel to the I2C pullup resistors found on BOOSTXL-BASSENSORS. The pullup resistors on the BoosterPack™ plug-in module are wired to a CC1352P DIO pin (extra wire); the existing copper trace to the +3.3V pin of the CC1352P LaunchPad was cut on the PCB. In this power-optimized design, the new wiring allows for powering the I2C pullup resistors only when data is transferred.

As the ADDR pin of the HDC2010 device in BOOSTXL-BASSENSOR is connected to GND, its slave i2C address is **0x40** (see Figure 2-1). The 0x40 address becomes **0x80 (Write)** or **0x81 (Read)** when used in the Sensor Controller Studio, which inserts the read/write bit as the last bit after the 7-bit address, as Table 2-1 shows.

Table 2-1. HDC2010 I^{2}C Slave Address

ADDR	ADDRESS (7-BIT ADDRESS)	BOOSTXL-BASSENSORS (8-BIT ADDRESS)
GND	**1000000**	**0x80(W)/0x81(R)**
VDD	1000001	0x82(W)/0x83(R)

Note

The HDC_V+ pin uses active low level voltage to control the P-MOSFET which gates the power to the HDC2010 sensor.

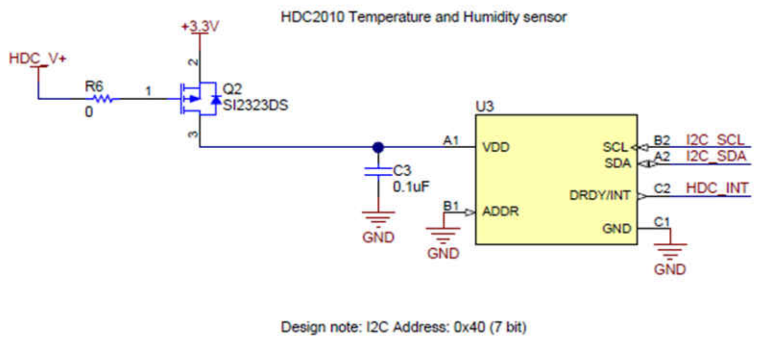


Figure 2-1. Interfacing to HDC2010, Power Supply Line is P-MOSFET Gated (Low Active)

After power up, the HDC2010 device can be configured to a sampling frequency from one sample every two minutes down to five samples every second. The raw values provided by the HDC2010 sensor have to be calculated or converted into a human readable format for temperature and humidity, see Equation 1 and Equation 2.

$$\text{Temperature }(°\text{C}) = \left(\frac{\text{TEMPERATURE }[15:0]}{2^{16}}\right) \times 165 - 40 \tag{1}$$

$$\text{Humidity }(\%\text{RH}) = \left(\frac{\text{HUMIDITY }[15:0]}{2^{16}}\right) \times 100 \tag{2}$$

Both of these equations are implemented in the Sensor Controller code example for demonstration purposes; however, the additional code reduces the available buffer memory for storing sensor data and negatively impacts the amount of data to be transmitted over-the-air.

For lowest power consumption, TI recommends to read and transmit the raw 16-bit humidity and 16-bit temperature sensor values over-the-air (after a packet with the desired payload length is created), while the conversion into a human readable format Equation 1 and Equation 2 can be done at the receiving or data display node.

3 System Operation

3.1 HDC2010 Sensor Readout

Using the Sensor Controller Studio in debug mode, a simple way to test the HDC2010 functionality is to exhale in front of the HDC2010 sensor, which leads to a very large change for the humidity value, while the temperature is barely raising. Putting a finger for a few seconds on the HDC2010 sensor, raises the temperature to about 30.8°C (all tests were run in an office environment at room temperature). The x axis is time and the unit is second. The y axis is humidity and temperature value. Use the equations in Section 2.2 to calculate the real value.

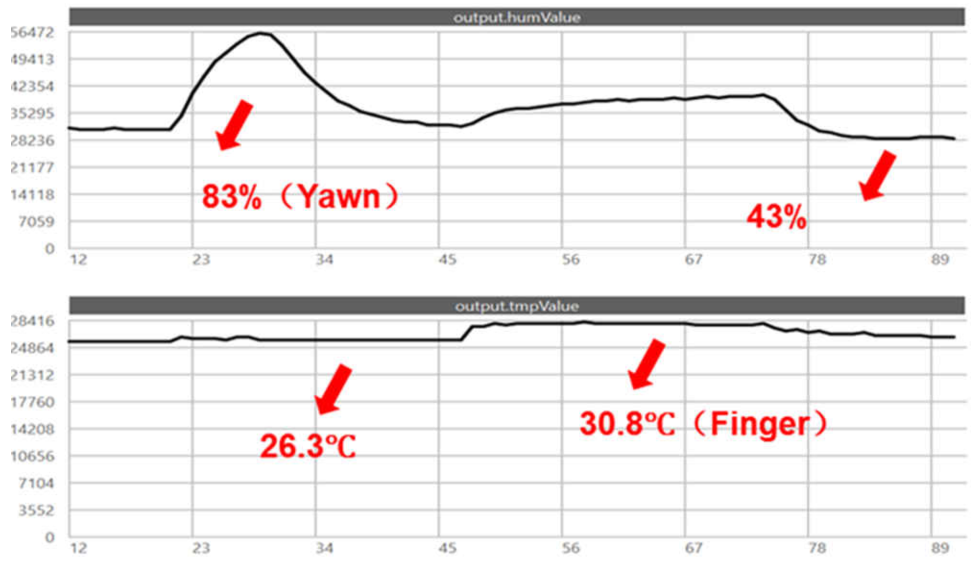


Figure 3-1. Temperature and Humidity Curve Under Sensor Controller Studio

The software flow chart in Figure 3-2 describes the HDC2010 sensor configuration. First, the DIO pin is set to LOW state which opens the P_MOSFET device and the HDC2010 sensor is powered on, then after the minimum start-up time delay several configuration registers are initialized. These register values set the sensor sampling frequency (how often a measurement is made) and the data resolution, here the maximum 14-bit accuracy is set. Finally, the 0x0F register is set to 0x01 to start the measurement. Only 1.24 ms later, both the temperature and humidity value are read out and stored into a data buffer inside the Sensor Controller ULL RAM memory.

The final version of the firmware always keeps the HDC2010 sensor powered as it consumes typically 50 nA in standby mode between temperature and humidity measurements. The complete power-off of the HDC2010 sensor was also implemented and tested. In that case, every HDC2010 readout requires the full initialization cycle and the delay to do the measurement. Depending on the sensor readout period, especially for very long periods of inactivity, power-off can be more power efficient. For the 60 s period analyzed in this report, the standby mode achieved lower power.

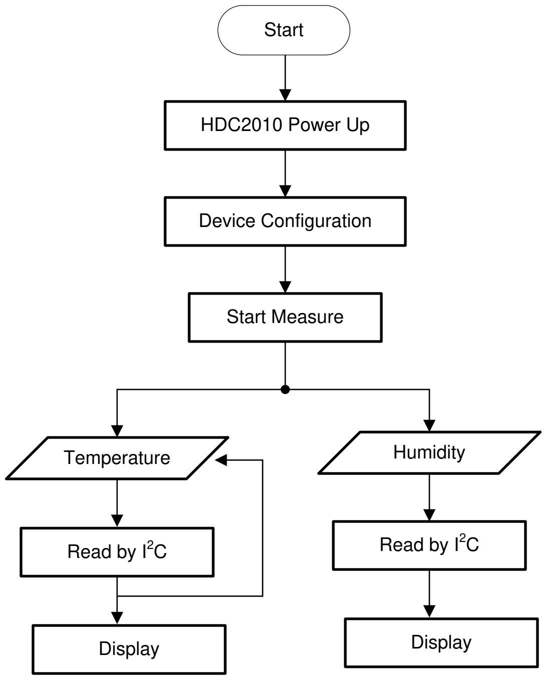


Figure 3-2. Flow Chart of Sensor Controller Engine Firmware

3.2 I2C Protocol and Data Buffering for Low Power

In this design the multibyte read functionality of the HDC2010 sensor is used to reduce the I2C communication time; the difference between single-byte mode and multibyte mode is described in Table 3-2. The multibyte mode requires only one read command with continuous ACK request to read out 4 bytes, which are starting at address 0x00. This helps reduce the code size as well as the I2C communication activity time and the overall system power consumption.

Table 3-1. Read Single Byte

Master	START	Slave address(W)		Address		Start	Slave address (R)			NACK	STOP
Slave			ACK		ACK			ACK	DATA		

Table 3-2. Read Multi Byte

Master	START	Slave address (W)		Address		Start	Slave address (R)			ACK		ACK		NACK	STOP
Slave			ACK		ACK			ACK	DATA		DATA				

In Figure 3-3, the upper image of a *Saleae LogicPro16* analyzer displays the multibyte read, with only one write register address command and 4 bytes (Humidity and Temperature values) read out with a total duration of 0.1738 ms. The lower image utilizes a separate write register address command for each byte value to be read out, which lasts for 0.409 ms in total.

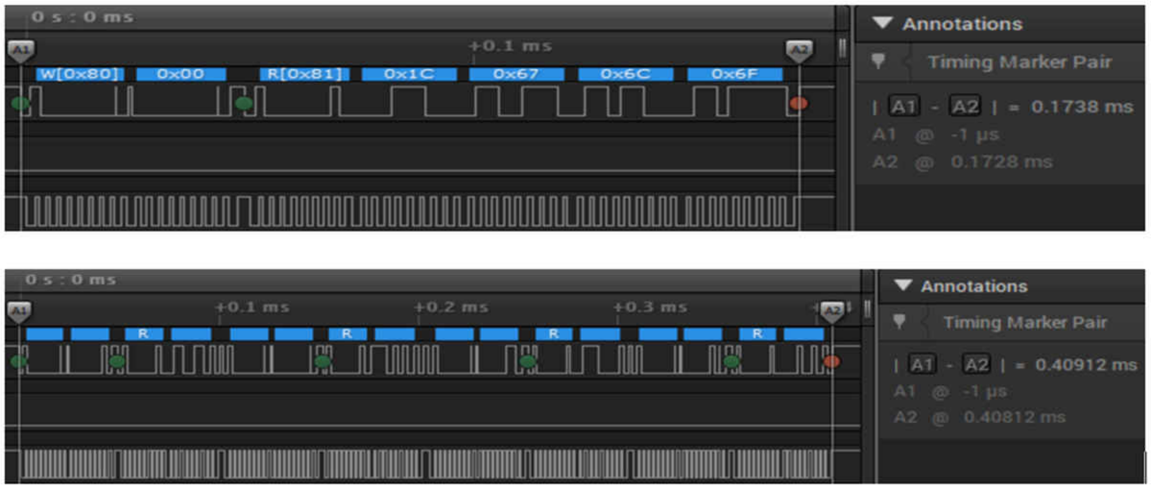


Figure 3-3. Comparison of Four Byte Read Modes (Multibyte Read on Top)

The collected sensor data is written to an user-configurable data buffer area, with the 2 bytes temperature and 2 bytes humidity values stored one after the other. When the desired buffer length is filled up with data, for example, 80 bytes after 20 minutes with 1 readout every minute, the Arm Cortex-M4F MCU is notified to get the data buffer and transmit it. This significantly reduces the protocol overhead, with only the same 8 bytes for preamble and SYNC but now with 80 bytes payload at once.

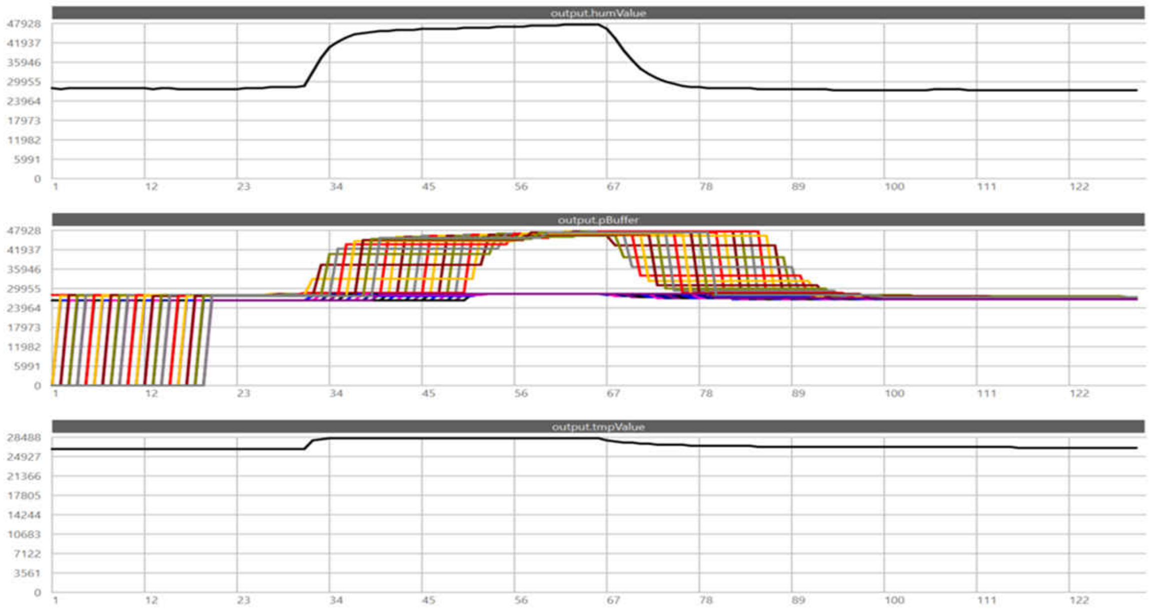


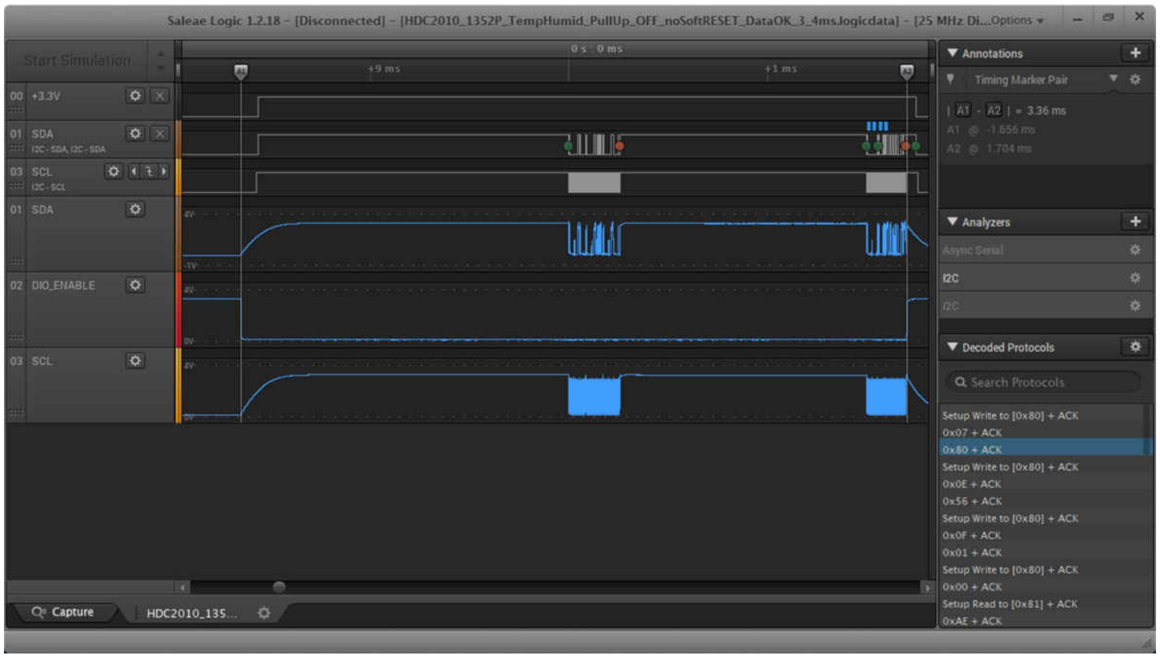
Figure 3-4. Temperature and Humidity Data Buffer in Sensor Controller Studio

The data buffer size with 80 bytes is arbitrary and was selected to have a reasonable size for testing purposes. The system designer may increase the buffer size to utilize the maximum available ULL SRAM inside the SCE, which amount to almost 3K bytes, see the resource allocation for the provided CC1352P + HDC2010 Sensor Controller Engine project.

If the HDC2010 device is read out every minute, then the buffer needs to store 4 bytes × 60 minutes = 240 data bytes in one hour. By adding 4 bytes for preamble, 4 bytes for SYNC word and 2 bytes for a sequential packet number, the total data packet length sums up to 250 bytes. Multiple wireless protocols, including wM-Bus and Bluetooth Low Energy (BLE), use a packet length of up to 256 Bytes.

Figure 3-5 shows the SCE interaction with the HDC2010 sensor over the I2C bus, showing the positive pulse on Channel 0 (or 3.3 V) to the I2C pullups and the power on pulse (active low) for the HDC2010 device on Channel

2. The total time to start from power-off, configure the HDC2010 registers, and read out the sensor data is only 3.36 ms.



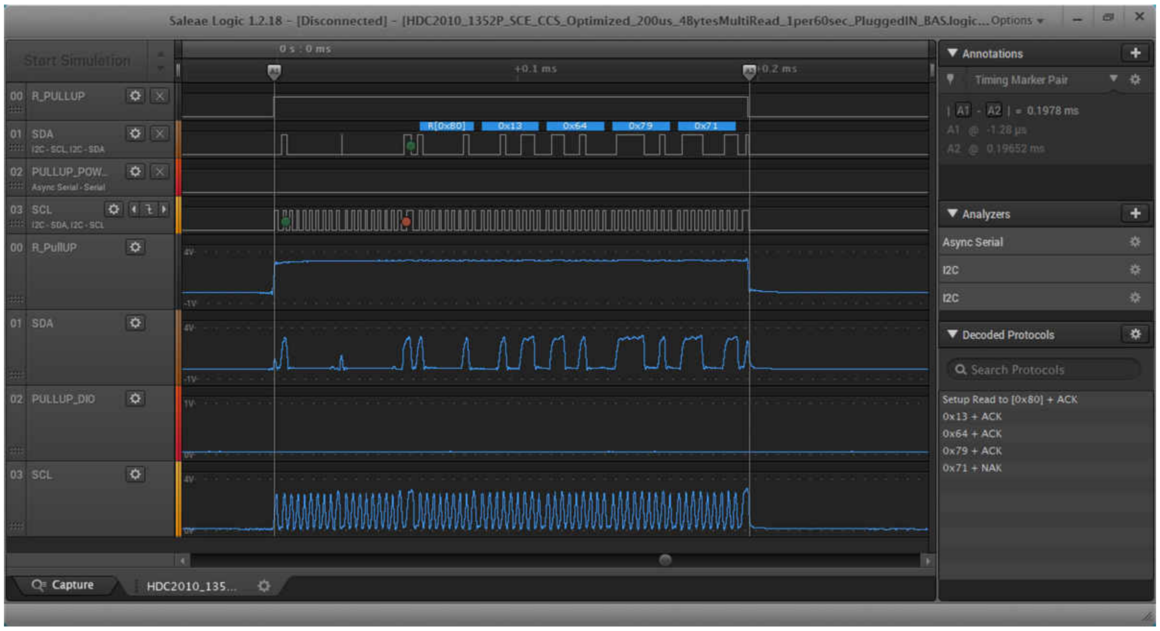
Saleae Logic 1.2.18 – [Disconnected] – [HDC2010_1352P_TempHumid_PullUp_OFF_noSoftRESET_DataOK_3_4ms.logicdata] – [25 MHz Di...

| A1 - A2 | = 3.36 ms
A1 @ -1.656 ms
A2 @ 1.704 ms

Setup Write to [0x80] + ACK
0x07 + ACK
0x80 + ACK
Setup Write to [0x80] + ACK
0x0E + ACK
0x56 + ACK
Setup Write to [0x80] + ACK
0x0F + ACK
0x01 + ACK
Setup Write to [0x80] + ACK
0x00 + ACK
Setup Read to [0x81] + ACK
0xAE + ACK

Figure 3-5. Complete Initial SCE Access to HDC2010 (Power on of I2C Pullups, Configuration, Data Readout, and Power Off)

Note that all subsequent data readouts are shorter, if the HDC2010 sensor is not powered off every time but kept in standby between measurements. Both solutions were tested and the final firmware keeps the HDC2010 sensor in standby mode between measurements. This leads to a very fast sensor data readout with a duration of only 198 µs due to the multibyte read mode, as Figure 3-6 shows.



Saleae Logic 1.2.18 – [Disconnected] – [HDC2010_1352P_SCE_CCS_Optimized_200us_4BytesMultiRead_1per60sec_PluggedIN_BAS.logic...

| A1 - A2 | = 0.1978 ms
A1 @ -1.28 µs
A2 @ 0.19652 ms

Setup Read to [0x80] + ACK
0x13 + ACK
0x64 + ACK
0x79 + ACK
0x71 + NAK

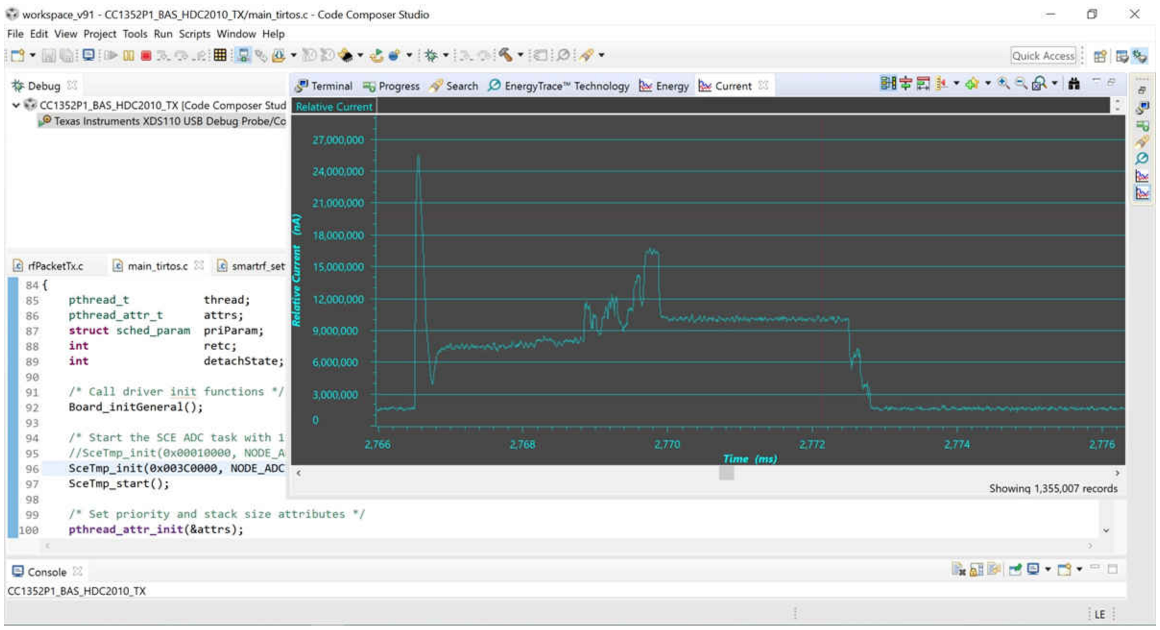
Figure 3-6. Periodic Sensor Readout Using the SCE With 198 µs

4 Test and Verification

To allow for accurate comparison between *Standby* and *Sensor Controller* system solutions, the HDC2010 sampling code with periodic RF transmission was implemented also on the main Arm Cortex-M4F MCU instead, while the SCE is left unused. Using the same duty-cycle of 60 seconds and same RF parameters and packet length, also the *Duty-cycling* approach as in the TIDA-00484 reference design was added.

4.1 EnergyTrace™ Results

In this test both the sensor readout (HDC readout in automatic mode) and the RF packet transmit period are set to once per minute. The RF parameters are the default settings for 50 kbps, 2-GFSK in SmartRF Studio at +1 dBm. All these parameters are equal to the settings of the TIDA-00484 reference design to enable comparison between the different system solutions. Figure 4-1 shows the current profile, captured by the EnergyTrace function in CCS with a total transmit time of nearly 6.2 ms. The last 2.5 ms of this pulse with a stable current of approximately 9.4 mA shows the wireless data transmission by the CC1352P device (note that EnergyTrace reports relative measurement values, the CC1352P data sheet value is 9 mA).



```
{
    pthread_t           thread;
    pthread_attr_t      attrs;
    struct sched_param  priParam;
    int                 retc;
    int                 detachState;

    /* Call driver init functions */
    Board_initGeneral();

    /* Start the SCE ADC task with 1
    //SceTmp_init(0x00010000, NODE_A
    SceTmp_init(0x003C0000, NODE_ADC
    SceTmp_start();

    /* Set priority and stack size attributes */
    pthread_attr_init(&attrs);
```

Figure 4-1. Current Profile for +1-dBm Transmit With SCE and Arm® Cortex®-M4F Activity

4.2 Average Current Consumption

Further testing was done, using a TI proprietary tool, which uses the EnergyTrace™ method on more advanced hardware and software, resulting in less than 3 percent deviation from an Agilent Technologies N6705B power anlyzer.

This TI tool captured current profiles for 100 seconds and calculated the average values over a user-defined period, either 60 seconds with one RF transmission included or any time slot between transmissions. Tests were done with 3.0- and 2.0-V power supplies to emulate the initial and end-of-life voltages of a CR2032 coin cell or any other primary LiMnO2 battery. A special firmware was developed and tested, where the HDC sensor was read out every 10 seconds but the RF transmission happens every 60 seconds. This was needed to evaluate the effect of the Sensor Controller as well as the sensor data buffering prior to transmission. The *duty-cycling* solution is represented by the TIDA-00484 design, with its results documented in the *Humidity and Temperature Sensor Node for Sub-1GHz Star Networks Enabling 10+ Year Coin Cell Battery Life* design guide.

The test results shown in columns 1 and 2 of Table 4-1 were delivered by a N6705B power analyzer with Keysight 14585A software.

Table 4-1. Comparison of the Three Ultra-low Power Architectures

	Standby (main MCU reads HDC2010), $I_{average\ [60\ s]}$	*Sensor Controller* (SCE reads HDC2010), $I_{average\ [60\ s]}$	*Duty-cycling* (measured in TIDA-00484)

Table 4-1. Comparison of the Three Ultra-low Power Architectures (continued)

HDC2010 every **10 s**, RF every **60 s** [**3.0 V**]/[*2.0*]	**2181 nA** / *3040 nA*	**1666 nA** / 2565 *nA*	na
HDC2010 every **60 s**, RF every **60 s** [3.0 V]/[2.0]	**1686 nA** / *2542 nA*	**1693 nA** / 2576 *nA*	**1957 nA**

Note

The results show that using the *Sensor Controller* only once per 60 seconds is not achieving any power savings compared to the *Standby* solution (see Table 4-1). The power consumption values of these two solutions are identical when comparing 1686 and 1693 nA for the 60 s sensor read-out and wireless transmission period. The values in Table 4-1 are the average result of three test runs per each combination of supply voltage, sensor sampling rate and StandBy or Sensor Controller.

If the sensor is read every 10 seconds then the advantage of using the SCE peripheral becomes visible. Reading the sensor even more often further increases the benefit of utilizing the SCE peripheral. The duty-cycling approach at 60 seconds sensor sampling period is less power efficient as the inactivity time is too short and restarting and reinitializing the full system needs more energy than it saves in between.

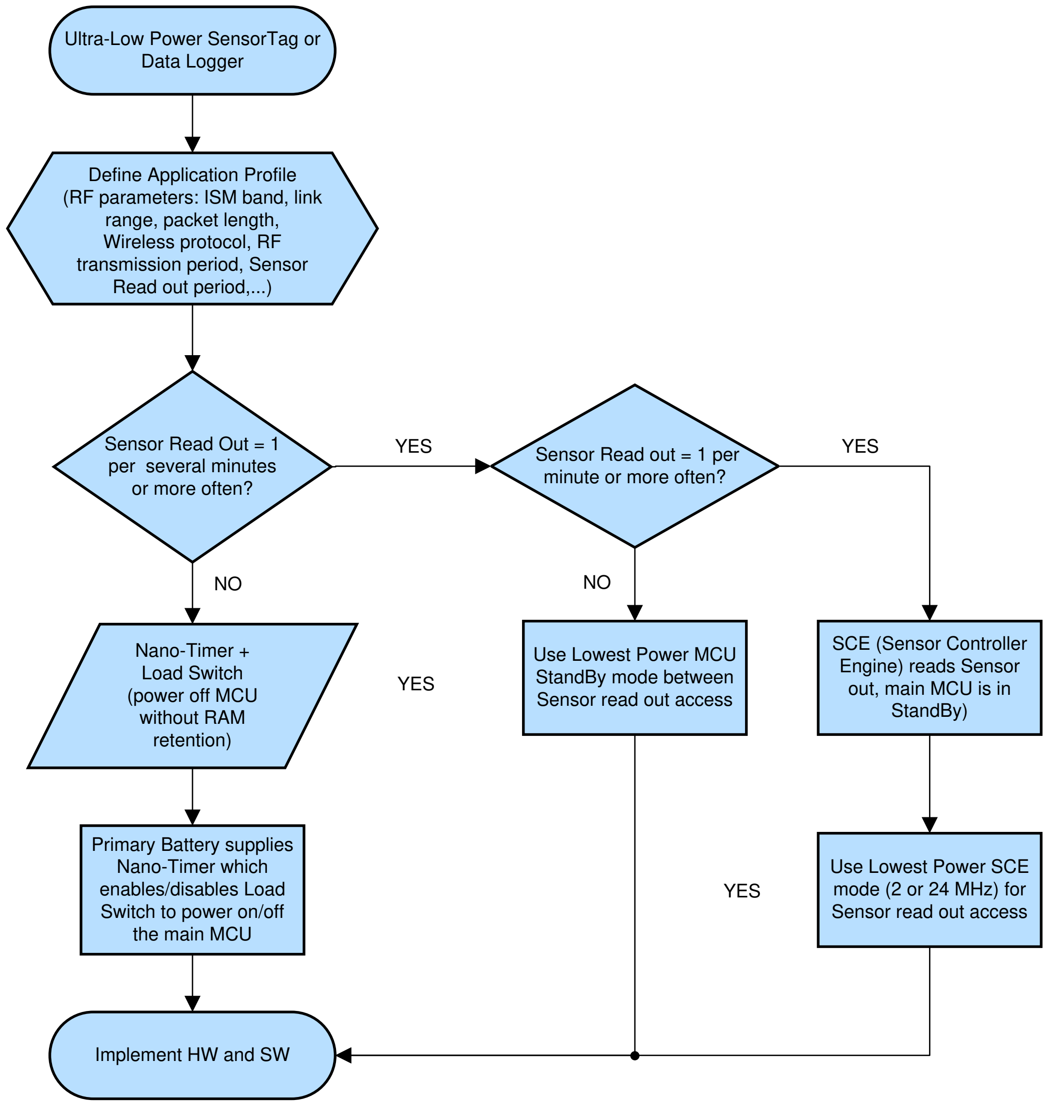

Ultra-Low Power SensorTag or Data Logger
Define Application Profile (RF parameters: ISM band, link range, packet length, Wireless protocol, RF transmission period, Sensor Read out period,...)
Sensor Read Out = 1 per several minutes or more often?
YES
NO
Sensor Read out = 1 per minute or more often?
YES
NO
Nano-Timer + Load Switch (power off MCU without RAM retention)
YES
Use Lowest Power MCU StandBy mode between Sensor read out access
SCE (Sensor Controller Engine) reads Sensor out, main MCU is in StandBy)
Primary Battery supplies Nano-Timer which enables/disables Load Switch to power on/off the main MCU
YES
Use Lowest Power SCE mode (2 or 24 MHz) for Sensor read out access
Implement HW and SW

Figure 4-2. Ultra-low Power Decision Tree for *Standby*, *Sensor Controller* and *Duty-Cycling*

The flow chart in Figure 4-2 is based on the test results and is a decision tree for easy estimation of which system approach to consider when ultra-low power is the key application requirement. *Duty-cycling* (with nano-timer and load switch) is most efficient with a low duty cycle of at least 1-2 minutes or longer inactivity times.

StandBy is the easiest to implement (no SCE code development needed) and performs better than *Duty-cycling* for 1 minute sensor sampling time.

The *Sensor Controller* performs equal to *StandBy* when the sensor readout occurs every minute (confirmed with N6705B measurement data in Table 4-1) and delivers power savings with higher sensor readout frequency, such as every 10 seconds.

A graphical representation of the three solutions with the average current during inactivity (Sensor and Wireless MCU are in standby or completely powered off) and during sensor readout and subsequent RF packet transmission is shown in Figure 4-3. The current values herein were *measured with TI's Energy Tool*.

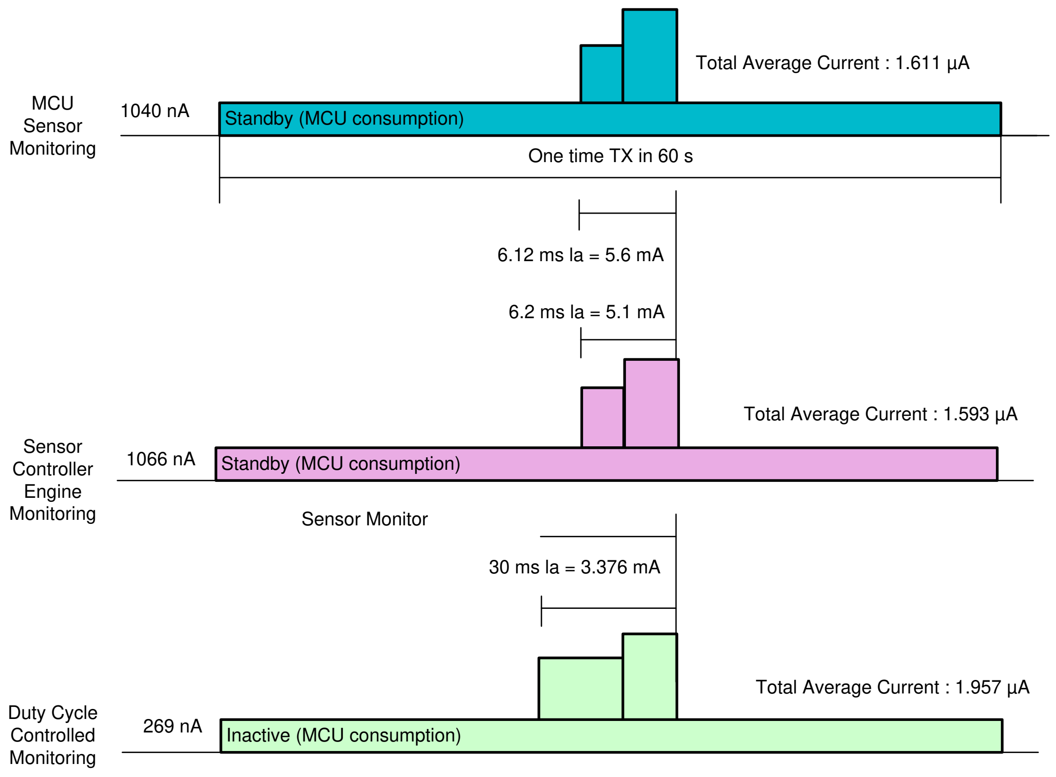
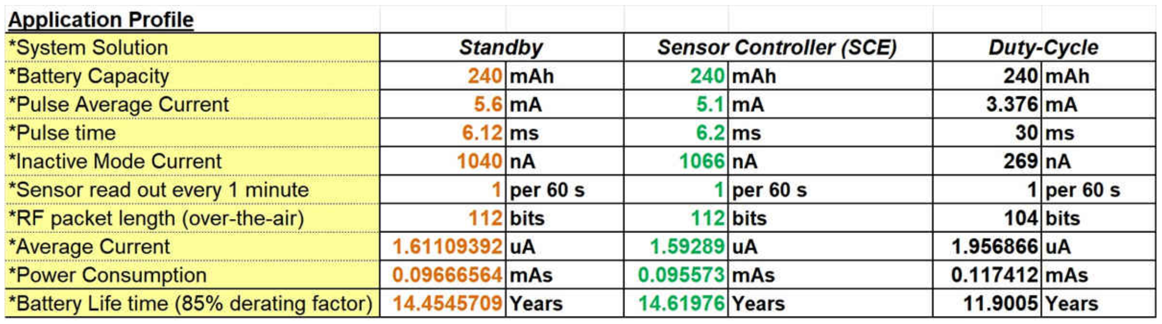

Total Average Current : 1.611 µA
MCU Sensor Monitoring
1040 nA
Standby (MCU consumption)
One time TX in 60 s
6.12 ms Ia = 5.6 mA
6.2 ms Ia = 5.1 mA
Total Average Current : 1.593 µA
Sensor Controller Engine Monitoring
1066 nA
Standby (MCU consumption)
Sensor Monitor
30 ms Ia = 3.376 mA
Total Average Current : 1.957 µA
Duty Cycle Controlled Monitoring
269 nA
Inactive (MCU consumption)

Figure 4-3. Simplified Current Profiles (Comparison)

The average current of duty cycle controlled monitoring is 1.957 µA at 3 V, as the TIDA-00484 design shows. The newly-introduced Sensor Controller Engine approach draws 1.593 µA at 3 V, or nearly 20% less power consumption in this application.

An excel sheet is used to create the table comparing the three solutions, based upon the same 240 mAh CR2032 coin cell capacity and battery life formula as described in the TIDA-00484 design.



Application Profile						
*System Solution	*Standby*		*Sensor Controller (SCE)*		*Duty-Cycle*	
*Battery Capacity	240	mAh	240	mAh	240	mAh
*Pulse Average Current	5.6	mA	5.1	mA	3.376	mA
*Pulse time	6.12	ms	6.2	ms	30	ms
*Inactive Mode Current	1040	nA	1066	nA	269	nA
*Sensor read out every 1 minute	1	per 60 s	1	per 60 s	1	per 60 s
*RF packet length (over-the-air)	112	bits	112	bits	104	bits
*Average Current	1.61109392	uA	1.59289	uA	1.956866	uA
*Power Consumption	0.09666564	mAs	0.095573	mAs	0.117412	mAs
*Battery Life time (85% derating factor)	14.4545709	Years	14.61976	Years	11.9005	Years

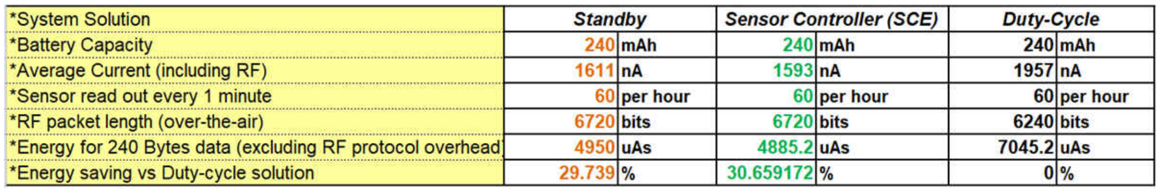
Figure 4-4. System Solutions Comparison (Sensor Readout and Data Transmission Once per Minute)

4.3 Power-Saving Effect of Data Buffering in RAM

It is important to understand the effect of reading out and buffering the sensor data in ultra-low leakage RAM, which enables transmission of one much longer data packet with less protocol overhead. This is possible with both *Standby* and *Sensor Controller* solutions, as two dedicated RAM areas are available: one for the Arm Cortex-M4F and a smaller one for the SCE peripheral.

The duty-cycle approach cannot buffer the data, as it gets powered off and all RAM content is lost. Hence, the sensor data must be sent out wirelessly before the power-off occurs leading to the transmission of additional wireless overhead, which is at the beginning of each packet sent over-the-air. In this report the Sub-1 GHz protocol overhead consists of 4 bytes preamble (0101.. sequence) followed by 4 bytes synchronization pattern (also called SYNC word) and 2 bytes packet sequence number (or just one byte to code the packet length).

Assuming the total packet length is 250 bytes (including 10 bytes overhead), see Section 3.2, both the *StandBy* and *Sensor Controller* solutions can send the data of 60 humidity and temperature values collected in 1 hour, when the sensor readout happens once per minute. In contrast, the *Duty-cycling* approach will transmit *redundantly 59 times* the identical wireless protocol overhead of 9 or 10 bytes, with nearly the highest power consumption, as these bytes are sent over-the-air.



*System Solution	*Standby*		*Sensor Controller (SCE)*		*Duty-Cycle*	
*Battery Capacity	240	mAh	240	mAh	240	mAh
*Average Current (including RF)	1611	nA	1593	nA	1957	nA
*Sensor read out every 1 minute	60	per hour	60	per hour	60	per hour
*RF packet length (over-the-air)	6720	bits	6720	bits	6240	bits
*Energy for 240 Bytes data (excluding RF protocol overhead	4950	uAs	4885.2	uAs	7045.2	uAs
*Energy saving vs Duty-cycle solution	29.739	%	30.659172	%	0	%

Figure 4-5. Power Efficiency due to Data Buffering and no Unnecessary Wireless Overhead

Obviously, the data buffering with longer data packets delivers additional savings, which amount to 30%, see Figure 4-5. This is a significant factor and must considered, when using the decision tree, see Figure 4-2. Depending on the amount of extra wireless overhead in a given application, the *Duty-cycling* solution may even require a much longer inactivity period than initially estimated to deliver the lowest power consumption.

Using the Sensor Controller Engine in the CC1352P MCU (and the CC13xx and CC26xx family devices) for sensor readout achieves equal or lower power consumption than using the main Arm Cortex-M4F MCU. If the application has long inactivity periods of many minutes or more, then Duty-cycling of the full system can deliver even lower power consumption with the addition of a nano-timer and load switch devices.

5 Summary

This application note shows an ultra-low power optimized design by leveraging the *Sensor Controller Engine* (SCE) in the CC1352P MCU for reading out the HDC2010 humidity and temperature sensor. Multiple options were tested, including HDC2010 readouts every 10 s and every 60 s, the latter is compared to the nano-timer based Duty-cycling solution described in the TIDA-00484 design. Alternatively, a *StandBy* solution with sensor readout by the Arm Cortex-M4F MCU instead of SCE was developed and tested. By comparing the advantages and disadvantages of the three system solutions a decision tree for selecting the most efficient ultra-low power architecture was derived.

Knowing the length of the inactivity period (or sensor readout frequency) and the RF parameters and the RF protocol overhead, the application designer has guidance on how to select the lowest power system architecture.

The newly introduced *Sensor Controller Engine* solution is proven to perform equal or better than the standard *StandBy* approach, where the MCU is used to read out the sensor.

Duty-cycling is highly recommended for long inactivity application periods of typically many minutes or even hours, where a nano-timer device and load switch deliver the best power efficiency.

Designers must adapt the considerations in this report to their application profile and wireless protocol parameters.

6 References

- Texas Instruments, *HDC2010 Low-Power Humidity and Temperature Digital Sensors Data Sheet*
- Texas Instruments, *CC1352P SimpleLink™ High-Performance Dual-Band Wireless MCU With Integrated Power Amplifier Data Sheet*
- Texas Instruments, *Humidity and Temperature Sensor Node for Sub-1GHz Star Networks Enabling 10+ Year Coin Cell Battery Life Design Guide*
- Texas Instruments, Code Example for SCE and StandBy HDC2010 Sensor Read-out

7 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (January 2020) to Revision A (June 2021) **Page**

- Updated the numbering format for tables, figures and cross-references throughout the document..................2