A Temperature-Compensated BLE Beacon and 802.15.4-to-BLE Translator on a Crystal-Free Mote

Titan Yuan*, Filip Maksimovic*, Brad Wheeler*, David C. Burnett*,
Lydia Lee*, Thomas Watteyne†, Kristofer S.J. Pister*
*University of California, Berkeley, CA, USA
†Inria, Paris, France
{titan, fil, brad.wheeler, db, lydia.lee, ksjp}@berkeley.edu, thomas.watteyne@inria.fr

***Abstract*— Crystal-free radios have the potential to revolutionize the IoT: due to their single-chip nature, they are both very cheap (no external components required) and very small (the size of a grain of rice). The Single-Chip Micro Mote (SCμM) is a 2×3×0.3 mm^3 crystal-free chip that can communicate with off-the-shelf transceivers over Bluetooth Low Energy (BLE) or IEEE 802.15.4. Setting its communication frequency is challenging because the crystal-free chip can rely only on internal oscillating circuits, which are very susceptible to temperature. Without compensation, a SCμM chip can no longer communicate with an off-the-shelf BLE receiver if the temperature changes by more than 1.25 °C. This paper introduces a two-step temperature compensation method, allowing SCμM to successfully send BLE frames over a 20 °C temperature range. After performing initial calibration during optical bootloading, we use an open-loop linear model to estimate the ambient temperature and continuously tune the mote's local oscillator (LO) frequency as the temperature changes. We show how the mote can use the intermediate frequency of 802.15.4 frames it receives from nearby off-the-shelf transceivers as a frequency reference to adjust its LO frequency. This compensation method enables SCμM to operate as a tiny BLE beacon, a BLE temperature sensor (for retail or medical applications), or a 802.15.4-to-BLE translation device.**

***Keywords*— Internet of Things, crystal-free radio, temperature compensation, wireless temperature sensor.**

I. Introduction

The single-chip micro mote (SCμM) [1] is the first crystal-free chip to be compatible with the Bluetooth Low Energy (BLE) and IEEE 802.15.4 standards, allowing it to interact with off-the-shelf hardware. The 2×3×0.3 mm^3 chip requires just three connections for operation: power, ground, and antenna. The main challenge of crystal-free architectures is that they must rely entirely on internal oscillating circuits for timekeeping and frequency selection. These oscillators are subject to large temperature coefficients, e.g., −40 ppm/°C for SCμM's local oscillator (LO) [2].

A growing number of studies focuses on software approaches to compensate for this drift and keep crystal-free motes functional over a wide temperature range. Using SCμM, Suciu *et al.* track the intermediate frequency (IF) of incoming 802.15.4 frames and use a recursive least-squares model to predict the tuning code based on temperature [3], [4]. Elsts *et al.* use a lookup table built from initial calibration and compensation history to calculate the required clock compensation in channel-hopping networks [5]. Song *et al.* use both a lookup table and phase-tracking of incoming frames to achieve a carrier stability better than ±100 ppm on their crystal-free 3.5×3.8 mm^2 MICS transceiver [6]. Chang *et al.* propose the QuickCal routine in which 16 dedicated off-the-shelf 802.15.4 transceivers regularly send frames for SCμM to receive and tune its frequency settings [7].

There have also been other approaches developed for crystal-based architectures. Martinez *et al.* apply quadratic least-squares to a periodically updated table of frequency drift measurements to build a drift-temperature model with a maximum drift of 10 ms over 24 h (close to 0.1 ppm) [8]. Lee *et al.* present a compensation circuit to program a 23-bit frequency divider based on measurements between −40 °C and 60 °C to achieve less than 1 ppm of frequency variation [9].

In this paper, we propose a method to allow SCμM to communicate with off-the-shelf BLE and 802.15.4 devices over a 20 °C temperature range. This is challenging for two reasons. First, without compensation, SCμM exceeds the maximum ±50 ppm BLE frequency drift requirement with a ±1.25 °C temperature change [10]. Second, due to its single-chip nature, SCμM is not equipped with a dedicated temperature sensor. In Section III, we describe how we use a flashing infrared LED and the mote's optical receiver to initialize the tuning parameters at a certain temperature. In Section IV, we apply the technique introduced in [11] to use the behavior of the on-chip oscillators' frequencies over temperature to indirectly measure the temperature. We build an open-loop linear model that uses the temperature estimate to tune the LO frequency. This turns the mote into a stand-alone BLE temperature sensor and is the first time open-loop temperature compensation is applied to a crystal-free mote. In Section V, we turn SCμM into an 802.15.4-to-BLE translator, simultaneously receiving 802.15.4 packets and transmitting the data as BLE packets. We use the IF of the incoming 802.15.4 packets to tune the BLE TX frequency around the bias temperature.

II. Single-Chip Micro Mote (SCμM)

The Single-Chip Micro Mote (SCμM) is a 2×3×0.3 mm^3 crystal-free chip capable of communicating with off-the-shelf hardware over BLE and 802.15.4. In order to operate without any attached wires, SCμM uses an optical receiver to load firmware for its Arm Cortex-M0 to execute [12]. We use an

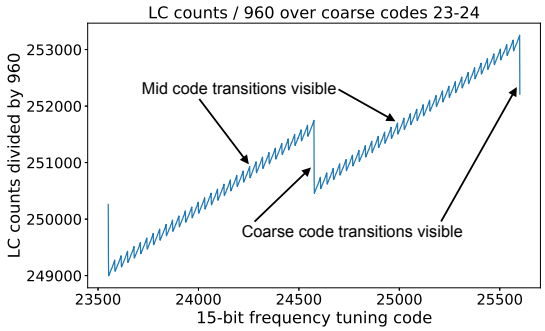


Fig. 1. SCμM's LC frequency is not monotonic with respect to the 15-bit frequency tuning code: there is a drop in frequency at the `coarse` and `mid` code transitions when the 5-bit `mid` and `fine` codes roll over, respectively. The counts were measured every 100 ms.

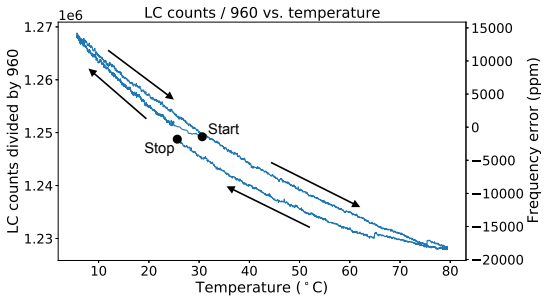


Fig. 2. The LC frequency for a given tuning code changes with temperature. Frequency is measured every 500 ms as SCμM is heated up and then cooled down. The hysteresis (the offset between start and stop points) is caused by the thermal mass of the development board to which the crystal-free mote is wirebonded.

infrared LED connected to a Teensy 3.6 microcontroller to generate the blink sequence that programs the mote.

The local oscillator (LO) consists of a 2.4 GHz LC tank whose frequency can be digitally tuned using three 5-bit overlapping capacitive DACs, called the `coarse`, `mid`, and `fine` DACs. As shown in Fig. 1, the tuning characteristic is not monotonic to ensure that there are no missed frequencies in the tuning range. In the absence of temperature changes, the LC tank has a frequency stability better than $\pm$40 ppm [13]. The LC frequency changes with temperature, as shown in Fig. 2. Note that the temperature coefficient shown in Fig. 2 is larger than the one reported in [2] because of the drift of the on-chip oscillator used to trigger the measurements [14].

Our goal is for SCμM to reliably transmit BLE advertising packets on channel 37 (2.402 GHz) to a smartphone. The BLE specifications define a data bit rate of 1 Mbps and a maximum frequency drift of $\pm$50 ppm [10]. We deal with the non-monotonicity of the LO frequency by only tuning the 5-bit `fine` DAC around a bias temperature; this allows for frequency compensation over a range of around 20 °C. We use the optical bootloading process to tune the `coarse` and `mid` codes at the bias temperature (Section III). To tune the `fine` code in the face of temperature variations, we either estimate the temperature and compensate for it (Section IV) or use incoming 802.15.4 frames as a frequency reference

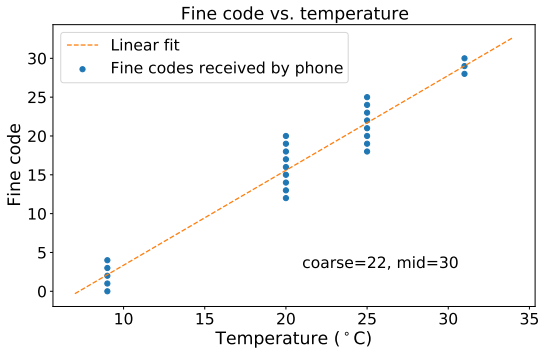


Fig. 3. The `fine` codes resulting in successful BLE packet reception on a smartphone at four temperatures and for a fixed `coarse` and `mid` code value.

(Section V).

III. Tuning `coarse` and `mid` Optically

Immediately after the optical programming operation, we program the Teensy board to blink its infrared LED every 100 ms for 15 s. The firmware that was previously loaded turns on the LC divider and calibrates the LC frequency by measuring the number of LC counts within each of the 100 ms periods. This one-time operation is sufficient to determine the `coarse` and `mid` tuning codes. Further tuning of the `fine` code (Sections IV and V) allows SCμM to keep operating within a 20 °C range around the temperature at which optical calibration was done.

IV. Tuning `fine` by Estimating Temperature

Using the `coarse` and `mid` codes from Section III, we vary the temperature across four values and have SCμM transmit a BLE packet for each of the `fine` code settings at each temperature value. We record which `fine` code settings result in successful BLE packet reception on a smartphone that is using the Qualcomm Snapdragon 835 chipset. Fig. 3 shows how varying the 5-bit `fine` code while keeping `coarse` and `mid` constant allows SCμM to keep communicating over a temperature range of around 20 °C. We note that the smartphone's BLE receiver allows for a frequency error of almost 400 kHz, considerably larger than the $\pm$50 ppm frequency requirement stipulated by the BLE specifications.

SCμM is not equipped with a dedicated temperature sensor. Instead, SCμM features a 2 MHz chipping clock and and a 32 kHz sleep timer. The frequencies of these oscillators vary differently over temperature changes, but by characterizing these variations, SCμM can determine the temperature between 0 °C and 100 °C with an accuracy of 2 °C [11].

We use this temperature estimate and a least-squares regression on the mean `fine` code received at each temperature from Fig. 3 to derive the temperature compensation. This relationship is expressed in (1) for a particular SCμM chip.

$$\texttt{fine code} = 1.2[1/°\text{C}] \cdot \text{temperature}[°\text{C}] - 18.4 \quad (1)$$

We experimentally verify this approach by placing the crystal-free mote into a temperature chamber, sweeping its

Fig. 4. The BLE sniffer app running on a Google Pixel 2 XL displaying a BLE advertising packet received from a SCμM mote while the mote is in a temperature chamber.

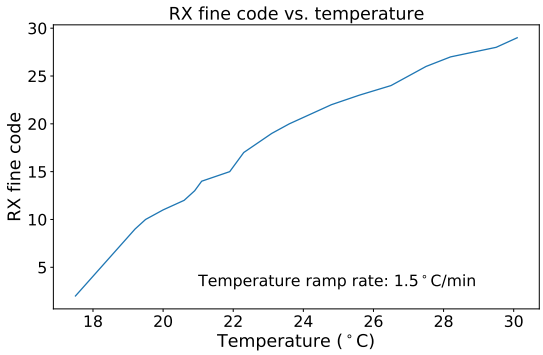


Fig. 5. The 802.15.4 RX `fine` code over temperature after IF compensation as the temperature is increased from 16 °C to 35 °C at a ramp rate of 1.5 °C/min.

temperature, and checking that a smartphone can still receive SCμM's BLE packets across temperature. A screenshot of the smartphone's BLE sniffer application is shown in Fig. 4.

This frequency compensation method requires two calibrations over temperature for every chip: one to find the linear model for the temperature estimate [11] and one to find the linear model for the `fine` code at the measured temperature (Eq. (1)).

V. Tuning `fine` by Receiving 802.15.4 Frames

SCμM was designed to receive 802.15.4 frames, so the idea is to use 802.15.4 frames sent by nearby off-the-shelf transceivers as a frequency reference for SCμM. Our setup consists of an OpenMote placed next to SCμM; we have the OpenMote transmit an 802.15.4 frame every 62.5 ms on 802.15.4 channel 11 (2.405 GHz).

We program SCμM as a "translator": it receives 802.15.4 frames and re-transmits their contents in a BLE advertising packet sent on BLE channel 37 (2.402 GHz) every 400 ms. We keep track of both the 802.15.4 RX and the BLE TX frequency tuning codes and assume that a change in the RX tuning code corresponds to an equal change in the TX tuning code, which proves to be an acceptable assumption. We start with the `coarse` and `mid` codes from Section III and only adjust SCμM's `fine` code.

As described in [3], [15], we can perform temperature compensation by measuring the intermediate frequency (IF) as the mote receives 802.15.4 packets. The hardware measures the IF by counting the number of zero-crossings within 100 μs. If the LO frequency is correctly tuned, we expect an IF of 2.5 MHz with 500 zero-crossings per 100 μs. Otherwise, we can use the non-zero IF offset from the expected value to tune the LO frequency. We only use the IF offsets from packets with a sufficiently low link quality indicator (LQI) error rate to reduce interference, and we convolve the IF offsets with a smoothing Gaussian filter [15].

Every 800 ms, if we have received more correct 802.15.4 packets than the number of taps in the Gaussian filter, we calculate the filtered IF offset in terms of the number of zero-crossings. Each `fine` code corresponds to around 100 kHz in the LO frequency, which equals an IF offset of 20 zero-crossings. We adjust the LO frequency if the filtered IF offset has an absolute value greater than 12, a little more than halfway to the next frequency tuning code.

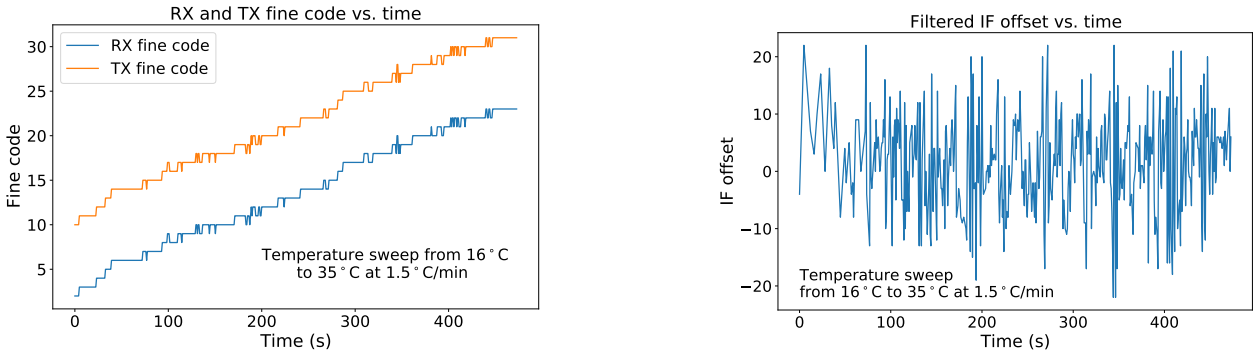
We place SCμM with an OpenMote inside a temperature chamber and increase its temperature from 16 °C to 35 °C at a ramp rate of 1.5 °C/min. We use a Nubee NUB8500H infrared thermometer to measure the on-board temperature. The `fine` code evolving over temperature is shown in Fig. 5.

The RX and TX frequency tuning codes over time as the temperature increases are shown in Fig. 6. The IF offset is always within $\pm$20: this indicates that the RX frequency is within $\pm$40 ppm of 802.15.4 channel 11. We confirm with a smartphone that we are receiving BLE packets from the SCμM mote in the chamber.

Since we only use the IF offsets from packets with a low LQI error rate, our approach stops working if the RX LO frequency is tuned far from the desired channel frequency as most of the received packets have a high LQI error rate.

A complementary approach in that case is to set the `fine` code to the one on which SCμM receives the most 802.15.4 frames. Every few seconds, we program the mote to dither its RX frequency while listening for 802.15.4 frames. When the mote listens for 802.15.4 frames, we sweep its RX frequency tuning code within a range of $\pm$2 `fine` codes of the current RX tuning code. This corresponds to listening for 802.15.4 frames on five different frequency settings in a range of approximately 500 kHz. At each of the five `fine` codes, we listen for incoming 802.15.4 frames for 800 ms, recording how many frames are received at each frequency setting. We find the weighted average of the `fine` codes of all received frames within the last 4 s, which indicates the best `fine` code to listen on for incoming 802.15.4 frames. We then adjust the RX tuning code accordingly and change the TX tuning code by the same amount.

In a temperature sweep from 16 °C to 35 °C, we verify that this method also allows us to compensate for any frequency drift in the LO and that we continuously receive BLE frames



(a) (b)

Fig. 6. The mote's 802.15.4 RX and BLE TX `fine` codes (a) and the corresponding filtered IF offset (b) during a temperature sweep from 16 °C to 35 °C at a ramp rate of 1.5 °C/min. Both the `fine` codes and the filtered IF offset are recorded every 800 ms. The TX `fine` code is adjusted by the same amount as the RX `fine` code to keep the filtered IF offset at 0. An IF offset of 20 corresponds to around 40 ppm of deviation from the channel frequency.

during the entirety of the sweep. This method can be used as a first step if the initial LO frequency is further off from the correct setting.

VI. Conclusion

Without an external frequency reference, the local oscillator (LO) of a crystal-free mote such as SCμM is subject to a high temperature coefficient. Without compensation, if the temperature changes by more than 1.25 °C, the BLE packets sent by SCμM are no longer within the frequency specifications.

This paper introduces a two-step compensation solution for SCμM to successfully communicate to a BLE receiver over a range of 20 °C. We tune the `coarse` and `mid` frequency codes using optical frequency calibration and then tune the `fine` code through a combination of estimating the ambient temperature and tracking the IF of received 802.15.4 frames.

The methods presented in this paper allow SCμM to be used as a BLE beacon or a 802.15.4-to-BLE translator. For example, a SCμM chip can be embedded in wearables or assembled with a tiny coin cell battery onto the back of a flying insect as a tracker.

We are currently working on using SCμM as a BLE beacon between 0 °C and 100 °C. We can generalize to a wider temperature range by determining the `coarse` and `mid` codes at multiple bias temperatures spaced 20 °C apart. When the temperature crosses into another 20 °C bin, we adjust the `coarse` and `mid` codes correspondingly, and within each 20 °C bin, we continue using the two `fine` code compensation methods described in this paper. The 2 MHz chipping clock used to clock the BLE TX FIFO at 1 MHz has a temperature coefficient of around 160 ppm/°C, for which we need to compensate as well [15].

References

[1] F. Maksimovic, B. Wheeler, D. Burnett, O. Khan, S. Mesri, I. Suciu, L. Lee, A. Moreno, A. Sundararajan, B. Zhou, R. Zoll, A. Ng, T. Chang, X. Vilajosana, T. Watteyne, A. Niknejad, and K. Pister, "A Crystal-Free Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, sub-mW BLE Compatible Beacon Transmitter, and Cortex M0," in *Symposium on VLSI Circuits*, 2019.

[2] F. Maksimovic, "Monolithic Wireless Transceiver Design," Ph.D. dissertation, Univ. of California, Berkeley, 2020.

[3] I. Suciu, F. Maksimovic, D. Burnett, O. Khan, B. Wheeler, A. Sundararajan, T. Watteyne, X. Vilajosana, and K. Pister, "Experimental Clock Calibration on a Crystal-Free Mote-on-a-Chip," in *IEEE INFOCOM, CNERT Workshop*, 2019.

[4] I. Suciu, F. Maksimovic, B. Wheeler, D. Burnett, O. Khan, T. Watteyne, X. Vilajosana, and K. Pister, "Dynamic Channel Calibration on a Crystal-Free Mote-on-a-Chip," *IEEE Access*, vol. 7, 2019.

[5] A. Elsts, X. Fafoutis, S. Duquennoy, G. Oikonomou, R. Piechocki, and I. Craddock, "Temperature-Resilient Time Synchronization for the Internet of Things," *IEEE Trans. on Indust. Informatics*, vol. 14, 2018.

[6] M. Song, M. Ding, E. Tiurin, K. Xu, E. Allebes, G. Singh, P. Zhang, S. Traferro, H. Korpela, N. Van Helleputte, R. Staszewski, Y. Liu, and C. Bachmann, "A 3.5mm×3.8mm Crystal-Less MICS Transceiver Featuring Coverages of ±160ppm Carrier Frequency Offset and 4.8-VSWR Antenna Impedance for Insertable Smart Pills," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020.

[7] T. Chang, T. Watteyne, F. Maksimovic, B. Wheeler, D. Burnett, T. Yuan, X. Vilajosana, and K. Pister, "QuickCal: Assisted Calibration for Crystal-Free Micro-Motes," *IEEE Internet of Things Journal*, 2020.

[8] B. Martinez, X. Vilajosana, and D. Dujovne, "Accurate Clock Discipline For Long-Term Synchronization Intervals," *IEEE Sensors Journal*, vol. 17, no. 7, 2017.

[9] D.-S. Lee, S.-J. Kim, D. Kim, Y. Pu, S.-S. Yoo, M. Lee, K. C. Hwang, Y. Yang, and K.-Y. Lee, "A Design of Fast-Settling, Low-Power 4.19-MHz Real-Time Clock Generator With Temperature Compensation and 15-dB Noise Reduction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 6, 2018.

[10] D. Burnett, B. Wheeler, L. Lee, F. Maksimovic, A. Sundararajan, O. Khan, and K. Pister, "CMOS Oscillators to Satisfy 802.15.4 and Bluetooth LE PHY Specifications without a Crystal Reference," in *Computing and Communication Workshop and Conf. (CCWC)*, 2019.

[11] T. Yuan, F. Maksimovic, D. Burnett, B. Wheeler, L. Lee, and K. Pister, "Temperature Calibration on a Crystal-Free Mote," in *IEEE World Forum on Internet of Things (WF-IoT)*, 2020.

[12] B. Wheeler, A. Ng, B. Kilberg, F. Maksimovic, and K. Pister, "A Low-Power Optical Receiver for Contact-free Programming and 3D Localization of Autonomous Microsystems," in *IEEE Ubiquitous Computing, Electronics Mobile Comm. Conference (UEMCON)*, 2019.

[13] B. Wheeler, F. Maksimovic, N. Baniasadi, S. Mesri, O. Khan, D. Burnett, A. Niknejad, and K. Pister, "Crystal-Free Narrow-Band Radios for Low-Cost IoT," in *Radio Freq. Integrated Circuits Symp. (RFIC)*, 2017.

[14] T. Yuan, "Temperature-Compensated BLE Transmission from a Crystal-Free Mote," Master's thesis, Univ. of California, Berkeley, 2020.

[15] B. Wheeler, "Low Power, Crystal-Free Design for Monolithic Receivers," Ph.D. dissertation, Univ. of California, Berkeley, 2019.