



**HAL**  
open science

## 6TiSCH on SC $\mu$ M: Running a Synchronized Protocol Stack without Crystals

Tengfei Chang, Thomas Watteyne, Brad Wheeler, Filip Maksimovic, David Burnett, Osama Khan, Sahar Mesri, Lydia Lee, Ioana Suciu, Xavier Vilajosana, et al.

► **To cite this version:**

Tengfei Chang, Thomas Watteyne, Brad Wheeler, Filip Maksimovic, David Burnett, et al.. 6TiSCH on SC $\mu$ M: Running a Synchronized Protocol Stack without Crystals. Sensors, 2020, 10.3390/s20071912 . hal-02616355

**HAL Id: hal-02616355**

**<https://inria.hal.science/hal-02616355v1>**

Submitted on 24 May 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Article

# 6TiSCH on SC $\mu$ M: Running a Synchronized Protocol Stack without Crystals

Tengfei Chang<sup>\*1</sup>, Thomas Watteyne<sup>1</sup>, Brad Wheeler<sup>2</sup>, Filip Maksimovic<sup>2</sup>, David Burnett<sup>2</sup>, Osama Khan<sup>2</sup>, Sahar Mesri<sup>2</sup>, Lydia Lee<sup>2</sup>, Ioana Suciuc<sup>3</sup>, Xavier Vilajosana<sup>3</sup> and Kris Pister<sup>2</sup>

<sup>1</sup> Inria-Paris, France; {tengfei.chang, thomas.watteyne}@inria.fr

<sup>2</sup> UC Berkeley, CA, USA; {brad.wheeler, fil, oukhan, smesri, lydia.lee, db, ksip}@berkeley.edu

<sup>3</sup> Univ. Oberta de Catalunya, Spain; {isuciu0, xvilajosana}@uoc.edu

\* Correspondence: tengfei.chang@inria.fr; Tel.: +33-180494143

Version March 26, 2020 submitted to Sensors

**Abstract:** We report the first time-synchronized protocol stack running on a crystal-free device. We use an early prototype of the Single-Chip micro Mote, SC $\mu$ M, a single-chip  $2 \times 3$  mm<sup>2</sup> mote-on-a-chip, which features an ARM Cortex-M0 micro-controller and an IEEE802.15.4 radio. This prototype consists of an FPGA version of the micro-controller, connected to the SC $\mu$ M chip which implements the radio front-end. We port OpenWSN, a reference implementation of a synchronized protocol stack, onto SC $\mu$ M. The challenge is that SC $\mu$ M has only on-chip oscillators, with no absolute time reference such as a crystal. We use two calibration steps – receiving packets via the on-chip optical receiver and RF transceiver – to initially calibrate the oscillators on SC $\mu$ M so that it can send frames to an off-the-shelf IEEE802.15.4 radio. We then use a digital trimming compensation algorithm based on tick skipping to turn a 567 ppm apparent drift into a 10 ppm drift. This allows us to run a full-featured standards-compliant 6TiSCH network between one SC $\mu$ M and one OpenMote. This is a step towards realizing the smart dust vision of ultra-small and cheap ubiquitous wireless devices.

**Keywords:** Crystal-free, 6TiSCH, SC $\mu$ M, smart dust.

## 1. Introduction

Low-power wireless networks are a key technology for applications ranging from industrial process monitoring to smart city and environmental monitoring. These networks combine time synchronization to achieve ultra-low power consumption, and channel hopping for high reliability. The resulting technology is known as Time Synchronized Channel Hopping (TSCH). TSCH is at the core of all main industrial standards, including WirelessHART [1], ISA100.11a [2] and IEEE802.15.4 [3]. 6TiSCH [4] is the latest such standardization efforts, lead by the Internet Engineering Task Force (IETF). Best-in-class commercial TSCH products today offer  $<50$   $\mu$ A average current draw and over 99.999% end-to-end reliability [5].

Today, these standards can run on virtually any IEEE802.15.4-compliant chip. All of the commercial chips use stable oscillators as a time reference. A typical design consists of a printed circuit board with the main chip, and 2 crystal oscillators: a *fast* crystal (typ. 16-20 MHz) which is used to accurately select the communication frequency and clock the modulation/demodulation, and a *slow* crystal (typ. 32 kHz) used as the main timing source for the synchronized state machine of TSCH. A crystal oscillator is a small fragment of lab-grown and cut quartz, enclosed in a package, and excited by circuitry that is typically on the chip. These devices have the property of oscillating at a frequency that is precise and characterized over temperature, supply voltage, and aging. Typical drift rates, i.e. the inaccuracy of the frequency, is in the 10-30 ppm (parts-per-million) range. That is, rather than oscillating at 32768 Hz a 10 ppm crystal oscillates somewhere between 32767.672 and 32768.328 Hz.



52 algorithms to limit the temperature effect to  $\pm 10$  ppm, using an embedded temperature sensor. The  
53 remaining  $\pm 50$  ppm budget is used to compensate for the effect of stress and aging on frequency error.

54 The single-chip design realizes the crystal-free vision entirely. The idea is to design a chip in  
55 which all oscillating circuits are inside the chip itself, and consists of different types of resonating  
56 electronic circuits (e.g. LC resonator, RC delay-based oscillator, ring oscillators). The result is a single  
57 chip which can operate without any external active components, and a key component for realizing  
58 the Smart Dust vision.

59 The Single Chip micro-Mote, or SC $\mu$ M, is a true crystal-free chip we taped out in 2019 [7]. It is a  
60  $2 \times 3$  mm<sup>2</sup> single-chip crystal-free mote-on-chip which contains an ARM Cortex-M0 micro-controller, a  
61 2.4 GHz IEEE802.15.4 radio, and an optical receiver for optical programming. Fig. 1 shows SC $\mu$ M on  
62 the board we use to develop/debug it.

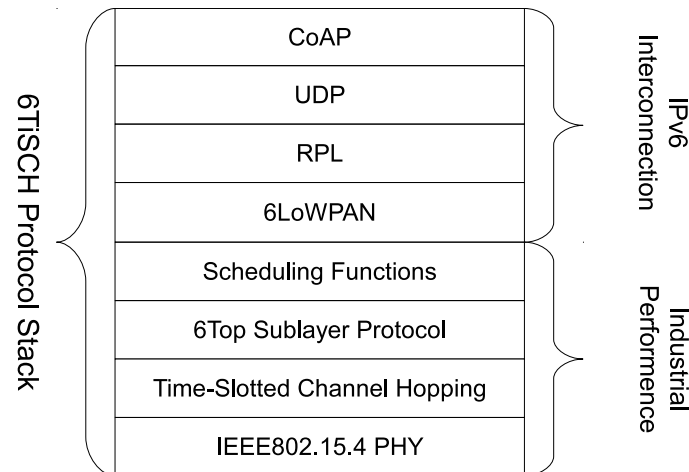
63 The Michigan Micro Mote (3M) partly realizes that vision. Known as “world’s smallest computer”,  
64 one 3M version measures only 0.04 mm<sup>3</sup> and is composed of a stack of dies wire-bonded together [8].  
65 They rely on visible light communication using an LED and a photodiode with a communication range  
66 of 15.6 cm. Another 3M mote, measuring  $3 \times 3 \times 3$  mm<sup>3</sup> does include a more traditional RF transmitter [9].  
67 Yet, because of the drift of its timing circuits, a highly-capable FPGA-based computer system is needed  
68 to receive the signals it sends, and mote-to-mote communication is not possible. A third  $4 \times 4 \times 4$  mm<sup>3</sup> 3M  
69 mote [10] is capable of mote-to-mote communication, but has to rely on a crystal oscillator (assembled  
70 as one layer of the stack) for accurate timekeeping. None of these are standards-compliant, i.e. they  
71 cannot communicate with off-the-shelf radios.

72 The challenge with SC $\mu$ M, as with any single-chip crystal-free device, is that its internal oscillators  
73 are far less accurate than crystal/MEMS-based external circuits. SC $\mu$ M has a drift up to 16,000 ppm  
74 over temperature [11], three orders of magnitude higher than crystal/MEMS-based oscillators. This  
75 makes it extremely difficult to (1) tune the frequency to communicate on, (2) set the correct rate to  
76 modulate/demodulate, and (3) keep a good sense of time to schedule communication in a TSCH  
77 network.

78 Why use complex networking with extremely constrained devices? TSCH does have important  
79 advantages. First, it is proven common in Industrial IoT applications, standardized in WirelessHART,  
80 ISA 100.11a and 6TiSCH, and commercialized for example in Analog Devices’ SmartMesh IP. Second,  
81 the micro-size of mote limits its energy storage capacity [12]. The synchronization based protocol,  
82 i.e. TSCH, provides ultra-low level of power consumption while mostly the other asynchronization  
83 protocol cannot satisfy. Third, SC $\mu$ M was designed for TSCH, in particular its timer structure and  
84 radio interface match our OpenWSN implementation. One important point is that, once the oscillators  
85 on SC $\mu$ M are calibrated for it to be able to communicate with regular motes such as the OpenMote, the  
86 same oscillators give the necessary timing to a TSCH implementation.

87 In [11], we showed a calibration algorithm to tune the oscillators on SC $\mu$ M so it can send and  
88 receive frames to the OpenMote, a popular off-the-shelf IEEE802.15.4 mote built around the CC2538  
89 chip [13]. In this paper, we go further and show an entire synchronized protocol stack running on  
90 SC $\mu$ M. Specifically, we show that, through 3 levels of calibration and compensation, we are able to  
91 have SC $\mu$ M and OpenMote drift by as little as 10 ppm, and stay synchronized with a maximum  
92 synchronization error of 300  $\mu$ s. Because it implements the full stack, SC $\mu$ M appears as a full-featured  
93 host in an IPv6 TSCH network.

94 As a fact that the environment changes, such as temperature, voltage or humidity, heavily  
95 influences the RC/LC oscillator frequency error, keeping sustainable frequency error while  
96 environment changes is a big challenge. For example, the LC oscillators of SC $\mu$ M drifts hundreds ppm  
97 when the temperature changes for 1 Celsius degree. The calibration and compensation techniques  
98 proposed in this article tune the oscillators to the desired frequency under constant room temperature.  
99 The goal is to express that SC $\mu$ M is capable to run a full standardized protocol stack under certain  
100 circumstance.



**Figure 2.** The 6TiSCH stack. The upper stack provides IPv6 connectivity. The lower stack, through IEEE802.15.4 TSCH, provides industrial-level performance.

101 The remainder of this paper is organized as follows. Section 2 introduces the 6TiSCH protocol  
 102 stack and the OpenWSN reference implementation of that stack. Section 3 details the main features  
 103 of SC $\mu$ M, including its clock system. Section 4 focuses on how we calibrate the clocks to allow  
 104 SC $\mu$ M to communicate with OpenMote. Section 5 explains the compensation algorithm we need for  
 105 porting OpenWSN onto SC $\mu$ M, and presents experimental synchronization results. Finally, Section 6  
 106 summarizes this paper and discusses ongoing and future research.

## 107 2. 6TiSCH Stack, OpenWSN Implementation

108 6TiSCH<sup>2</sup> is a working group which is standardizing the latest protocol stack based on TSCH. It  
 109 combines the industrial performance of IEEE802.15.4 TSCH, with the IETF upper stack for IoT devices.  
 110 As depicted in Fig. 2, this upper stack includes CoAP, UDP, RPL and 6LoWPAN.

111 At the core of the lower stack is IEEE802.15.4 TSCH. All nodes are synchronized to one another;  
 112 time is cut into timeslots, each typically 10 ms long. All communication is orchestrated by a schedule,  
 113 which indicates to each mote what to do in each slot: transmit, listen or sleep. This scheduled approach  
 114 allows for ultra low-power operation, as motes only turn their radio on when they know they need to  
 115 communicate with a neighbor, typically less than 1% of the time.

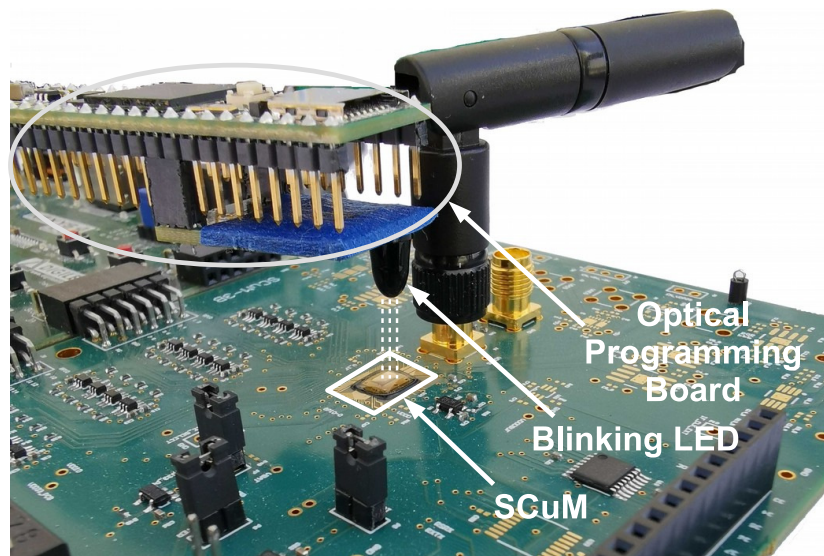
116 A pseudo-random hopping pattern is used for each transmission. The result is that, each time  
 117 mote *A* sends a frame to mote *B*, it does so on a different frequency. The resulting “channel hopping”  
 118 is effective at combating external interference and multi-path fading, and is also used by technologies  
 119 such as Bluetooth and cellular networks.

120 6TiSCH builds on top of IEEE802.15.4 TSCH. Each mote in a 6TiSCH network starts with a minimal  
 121 schedule [14]. The Minimal Scheduling Function [15] is used to track the amount of frames sent to a  
 122 particular neighbor, and uses the 6top Protocol [16] to negotiate additional cells to that neighbor when  
 123 needed. All communication is secured, and the Constrained Join Protocol [17] is used by a node to  
 124 securely join a network, through mutual authentication between the network and the joining node.

125 OpenWSN [18] is the reference open-source implementation of 6TiSCH. It consists of two parts:  
 126 the firmware running on the motes and OpenVisualizer, a Python-based application running on a PC.  
 127 The firmware implements the entire 6TiSCH protocol stack; OpenVisualizer acts as the bridge between  
 128 the 6TiSCH low-power wireless network and the Internet. OpenWSN has been ported to 10 hardware  
 129 platforms. In this paper, we present a port of OpenWSN onto SC $\mu$ M.

<sup>2</sup> <https://tools.ietf.org/wg/6tisch/charters>





**Figure 3.** The blinking pattern of the LED on the optical programming board (*top*) is used to switch SC $\mu$ M into bootloading mode and transfer the binary image to be executed onto SC $\mu$ M (*bottom*) [19].

130 OpenWSN has very limited requirements for the hardware it runs on. All it needs is a single  
 131 32 kHz timer with a single compare register. As detailed in Section 3, SC $\mu$ M was designed with  
 132 OpenWSN in mind, and its timer structure is perfectly suited to run the OpenWSN TSCH state  
 133 machine. The challenge is that SC $\mu$ M has no stable time reference.

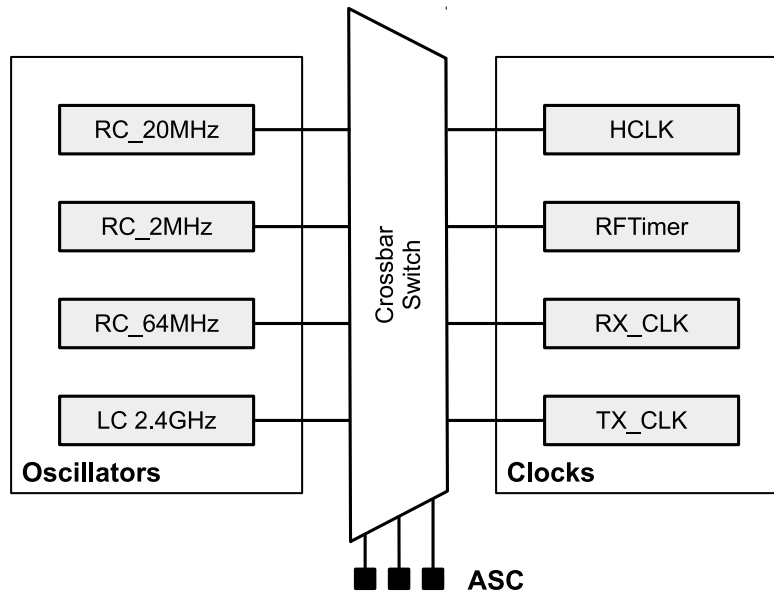
### 134 3. The Single Chip micro-Mote (SC $\mu$ M)

135 SC $\mu$ M is a true single-chip low-power wireless mote-on-chip which combines an ARM Cortex-M0  
 136 core, and an IEEE802.15.4 radio. It measures  $2 \times 3 \text{ mm}^2$ , roughly the size of a grain of rice. On top of that,  
 137 it features a radio timer (RFTimer) which is designed specifically for implementing time synchronized  
 138 communication protocols such as 6TiSCH (see Section 2). Loading code into the chip is done optically  
 139 by using an external board which blinks an LED close to the optical receiver on SC $\mu$ M [19]. Fig. 3  
 140 shows the optical bootloading process. It is single-chip by design, and replaces external clock sources  
 141 (typically crystal-based) by an internal clock system described below.

142 SC $\mu$ M operates at 1.5 V. It consumes 1.6 mW while transmitting, with an output power of -10 dBm,  
 143 and 1.4 mW while receiving, with a sensitivity of -83 dBm. While the radio consumption is optimized,  
 144 the same level of optimization hasn't been implemented for the full chip, yet. We measured 150-200  $\mu$ A  
 145 of leakage current from SRAM and analog circuits that cannot be shut off in this revision of the chip.  
 146 In addition, we measure 200-250  $\mu$ A of current drawn by the different peripherals (including the ARM  
 147 Cortex-M0 micro-controller) which use HCLK as their clock source (which cannot be turned off). This  
 148 results in approximately 400  $\mu$ A of current.

149 To perform time-slotted communication with 6TiSCH, a slot with maximum length packet  
 150 transmission, which takes 4.256ms, plus the acknowledgment reception (0.8ms) costs 4.9  $\mu$ C for SC $\mu$ M  
 151 running at 1.5V. To receive a maximum length packet (4.256ms), plus guard time (1ms) and send the  
 152 acknowledgment (0.8ms), SC $\mu$ M costs 4.9  $\mu$ C. For idle listen slot, which takes 0.4ms, SC $\mu$ M costs 0.5  $\mu$ C.  
 153 Assuming the leakage is reduced to a reasonable level, with 1mA current for Tx/Rx radio that turns  
 154 on/off in under 100ns, there is no doubt we can get the current below 1  $\mu$ A while running 6TiSCH  
 155 protocol stack. We expect the next revision of the chip to implement low-power modes for the entire  
 156 chip.

157 Fig. 4 shows the clock system of SC $\mu$ M. There are 4 main oscillators: three RC oscillators (64 MHz,  
 158 20 MHz, 2 MHz), one LC oscillator (2.4 GHz). A "crossbar switch" is used for routing the 4 oscillators  
 159 to be used as clock sources by the rest of the chip, including the micro-controller and the RFTimer.



**Figure 4.** SC $\mu$ M contains 4 main oscillators. A “crossbar switch” maps physical oscillators to clock sources that are used by the different peripherals in the chip. The Analog Scan Chain (ASC) is the mechanism for configuring this mapping.

160 There are 4 clocks: HCLK used as the master clock of the micro-controller, RFTimer used by the TSCH  
 161 state machine, RX\_CLK and TX\_CLK for generating the DSSS chip rate. The crossbar switch is configured  
 162 by a series of registers called the Analog Scan Chain (ASC).

163 Though the frequency stability of RC/LC oscillators are not comparable to the crystal oscillator,  
 164 in term of combating with the influence of temperature and voltage, it is possible to calibrate the clock  
 165 through software to meet the requirements.

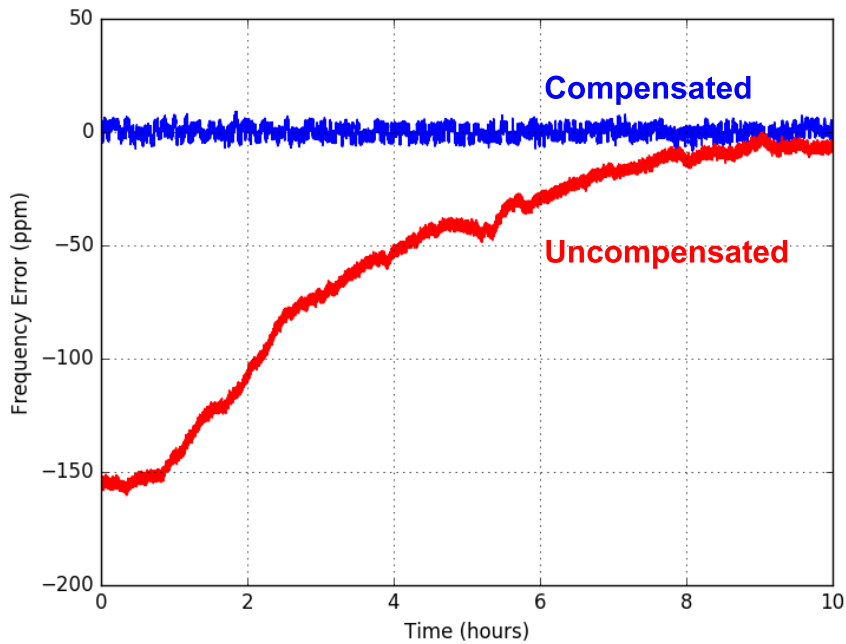
166 The previous works presented in [20] shows the LC tank oscillator of SC $\mu$ M drifts less than 40  
 167 ppm over 13 hours in the absence of temperature changes, which meets the  $\pm 40$ ppm specification of  
 168 IEEE802.15.4. Over 50°C temperature changing range, SC $\mu$ M could drift over 4000ppm of variation .  
 169 By adding a feedback mechanism through the incoming packet to calibrate the frequency, the effect of  
 170 the temperature variation is reduced from 150 ppm to less than 10 ppm indoor over the duration of  
 171 the test, which is over 10hours, as indicated in Fig. 5. [21] did a thorough analysis of the stability of  
 172 various clocks used in SC $\mu$ M as well. For the 32kHz RC oscillator, to keep the time offset within 1ms,  
 173 which is the minimal offset allowing two TSCH motes to communicate each other, SC $\mu$ M is capable to  
 174 re-synchronize every 20 seconds.

175 The RFTimer is designed specifically for TSCH. It orchestrates the transition between the different  
 176 states of the TSCH state machine, as shown in Fig. 6. The RFTimer comes with multiple compare  
 177 registers, which allows the entire sequence of events to happen during a slot to be programmed at the  
 178 beginning of the slot. The RFTimer then works hand-in-hand with the radio. For example, a frame can  
 179 be loaded into the transmit buffer of the radio automatically at a specific time, without needing code  
 180 to be executed on the micro-controller.

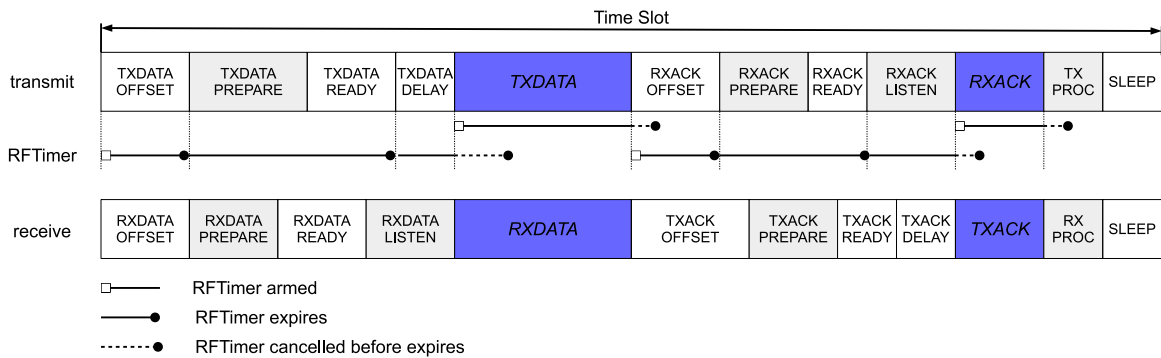
#### 181 4. Frequency Synthesis and Clock Calibration

182 The goal of this paper is to show a 6TiSCH network composed of one SC $\mu$ M and one OpenMote.  
 183 The challenge is that SC $\mu$ M does not have an accurate sense of time, and therefore derives its time  
 184 reference from OpenMote. This section describes how SC $\mu$ M tunes the frequency it communicates on,  
 185 and how we calibrate its clocks.

186 We need to give SC $\mu$ M a rough time reference so it can send frames that OpenMote can receive.  
 187 The frequency of each of the oscillators is tunable. We designed the code running on the optical



**Figure 5.** The LC oscillator frequency drift can be compensated through a feedback mechanism with incoming packets. With the feedback mechanism, the effect of the temperature variation is reduced from 150ppm (red trace) to less than 10ppm (blue trace) indoor, by testing over night. [20].

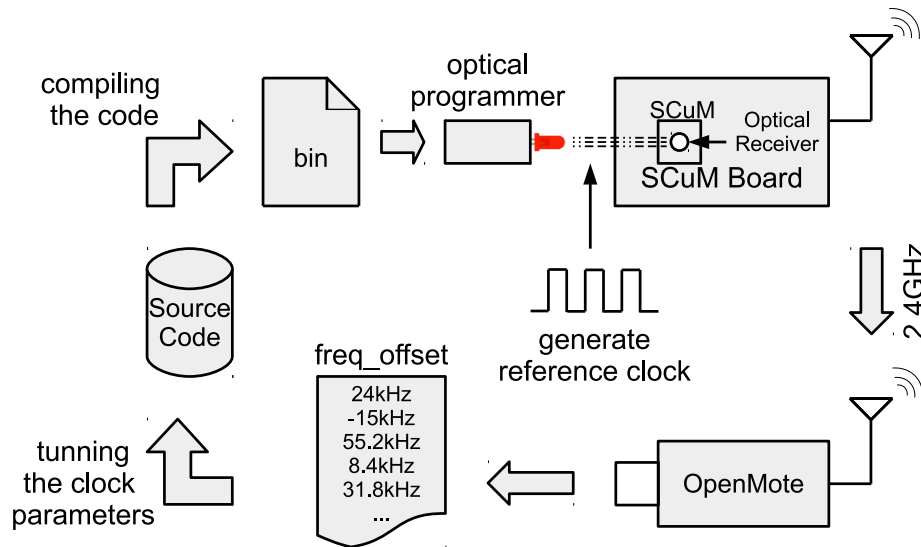


**Figure 6.** A TSCH slot is implemented as a state machine. The different states of a transmit and a receive slot are shown at the top. The RFTimer is used to transition from one state to the next, kicking off different actions in the radio (e.g. loading a frame in the transmit buffer) without intervention from the micro-controller.

188 programmer board in such a way that, at the end of the bootloading process, the optical programmer  
 189 repeatedly sends a sequence that causes a OPTICAL\_ISR interrupt to be generated on SCμM. This  
 190 interrupt fires every 100 ms for 2.5 s. While this is happening, on SCμM, all the clocks are running. By  
 191 recording the counter value of each of the clocks, and knowing the interval between interrupts, SCμM  
 192 calibrates each of the oscillators.

193 Following this coarse calibration using the optical programmer, SCμM can also calibrate against  
 194 the OpenMote. We do this offline, i.e. this calibration is done once, the result of which is reused the  
 195 next time SCμM is programmed. For this calibration, SCμM sends frames on channel 11 (2.405 GHz) to  
 196 OpenMote. OpenMote is programmed to listen on that channel, and print over its serial port the value  
 197 of its XREG\_FREQEST register, which indicates the frequency offset of the incoming signal. According to  
 198 that value, we manually tune the LC oscillator of SCμM, to minimize that frequency offset. The goal of





**Figure 7.** Setup used to tune the communication frequency of SC $\mu$ M. SC $\mu$ M transmits frames to OpenMote, which logs the frequency offset for each frame it receives. These offsets are then used to manually tune the LC oscillator of SC $\mu$ M, which is used to select the transmit frequency, to minimize the mean frequency offset.

199 this calibration is the same with what is done in [20]. The difference is that, in [20] the calibration is  
 200 done on SC $\mu$ M side through the intermediate frequency estimation.

201 This procedure is repeated for each SC $\mu$ M board, as each has slightly different tuning parameters.  
 202 This is illustrated in Fig. 7.

203 While SC $\mu$ M is running the 6TiSCH stack, it keeps synchronized to the OpenMote. Part of that  
 204 is making sure the boundaries of its TSCH slots are aligned in time with that of OpenMote. This is  
 205 done natively in the OpenWSN implementation. OpenWSN uses a 32 kHz timer with a compare  
 206 value set so it fires at each slot boundary. The accuracy of the clock used by this timer influences  
 207 synchronization accuracy. RFTimer runs at 500 kHz, not 32 kHz as OpenWSN assumes. This means  
 208 the OpenWSN port to SC $\mu$ M divides down the 500 kHz RFTimer so it appears as a 32 kHz clock  
 209 source to the otherwise unmodified OpenWSN stack implementation. Since  $500/32=15.625$ , the integer  
 210 division applied in the port results in a rounding error. This means the slot length on SC $\mu$ M is slightly  
 211 different than the slot length of OpenMote. As is, this difference in slot length results in an apparent  
 212 relative drift between OpenMote and SC $\mu$ M. We therefore implement a digital trimming (tick skipping)  
 213 compensation algorithm, detailed in Section 5.

214 In the implementation presented in this paper, a limitation of the FPGA/SC $\mu$ M setup presented  
 215 in Fig. 1 is that we cannot use the 20 MHz on-chip RC oscillator (RC\_20MHz in Fig. 4) to source the  
 216 RFTimer. Instead, we use a 20 MHz crystal oscillator of the FPGA. The results in this paper carry over  
 217 to using the on-chip RC\_20MHz, *except* that (1) the FPGA's crystal oscillator has a much smaller drift over  
 218 temperature, (2) the on-chip RC delay-based oscillator is expected to have higher jitter than the FPGA's  
 219 crystal oscillator. How much this impacts the overall stability of our implementation (including over  
 220 temperature) when running on SC $\mu$ M is left for future work.

## 221 5. Implementation and Experimental Results

222 The port of OpenWSN on SC $\mu$ M has a footprint of 54 kB. This includes the full protocol stack and  
 223 drivers. It takes the optical bootloader 2-3 s to load that code onto SC $\mu$ M.

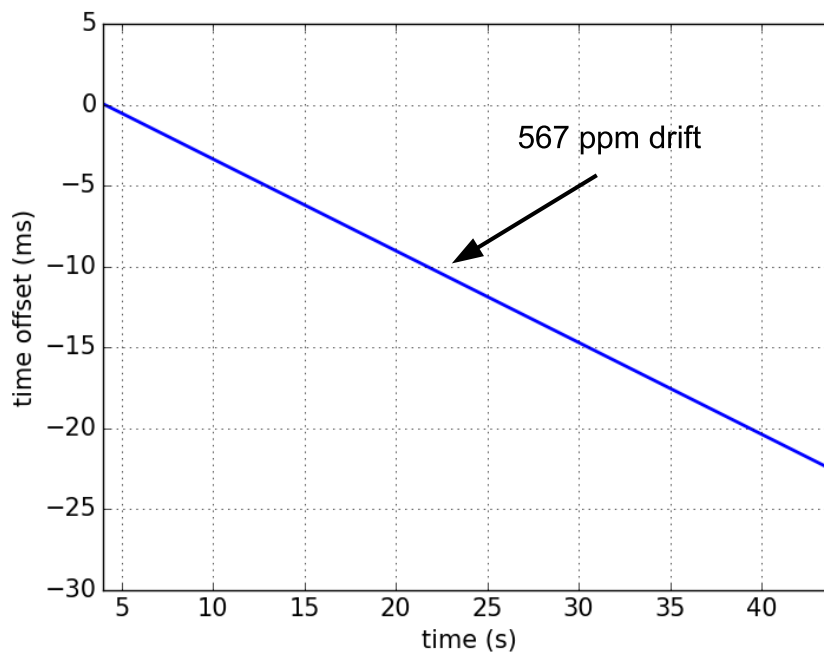
224 One of the goals of porting OpenWSN onto SC $\mu$ M is to show that this platform is perfectly capable  
 225 of running an off-the-shelf completely standards-based full stack. As a result, we made as little changes  
 226 as possible to the OpenWSN implementation. There are, however, the following changes that we had  
 227 to make.

228 First, the port of OpenWSN on SC $\mu$ M does not come with link-layer security, nor secure joining.  
 229 This is because this version of SC $\mu$ M does not come with an AES-128 cipher.

230 Second, we were able to significantly simplify the OpenWSN TSCH state machine thanks to the  
 231 RFTimer. On all other platforms OpenWSN is ported to, the time used by the TSCH state machine  
 232 only has one compare register, and cannot automatically trigger radio actions. This means that, on any  
 233 other platform, at the start of a transmit slot, the code schedules the timer to fire at the beginning of the  
 234 TXDATAPREPARE state (see Fig. 6). At that time, the micro-controller is woken up again, and loads the  
 235 frame into the transmit buffer of the radio, and arms the timer again, this time to fire at the starts of the  
 236 TXDATADELAY state. In contrast, on SC $\mu$ M, RFTimer provides multiple timer compare registers, so the  
 237 code arms the RFTimer to fire at the start of the TXDATAPREPARE, TXDATADELAY states, etc. states, all at  
 238 once at the start of a slot. Moreover, using the RFCONTROLLER\_REG register of SC $\mu$ M [22], the RFTimer  
 239 directly triggers radio actions, significantly reducing the load on the micro-controller.

240 Third, we had to increase the slot length to 82 ms. This is because we are using, on this revision  
 241 of SC $\mu$ M, an FPGA for the digital side of the chip. Each time the FPGA configures the radio (at most  
 242 twice per slot), it takes 16 ms for it to completely use the ASC. This is a limitation of the FPGA version  
 243 only, which the next revision of SC $\mu$ M will not have.

244 Because of the rounding error in clock division detailed in Section 4, the slot duration of SC $\mu$ M  
 245 and OpenMote are slightly different, resulting in apparent relative drift. To measure this, we program  
 246 the devices to toggle a pin at the beginning of each slot. We connect both pins to a logic analyzer,  
 247 and have the devices run without communicating (i.e. without resynchronizing). Fig. 8 shows the  
 248 evolution of the time offset between SC $\mu$ M and OpenMote, over time. The drift (the slope of the line in  
 249 Fig. 8) is 567 ppm.



**Figure 8.** Time offset between free-running SC $\mu$ M and OpenMote. No communication is taking place. SC $\mu$ M emulates a 32 kHz clock source by dividing down the 500 kHz RFTimer source. The rounding error in this integer division results in an apparent drift between SC $\mu$ M and OpenMote of 567 ppm.

250 The default guard time of the OpenWSN implementation is 1 ms. This is the maximum offset  
 251 between two motes, beyond which they cannot communicate. With a drift of 567 ppm, it takes less  
 252 than 2 s for perfectly synchronized motes to de-synchronize beyond the guard time. Typical drift rates  
 253 of crystal oscillators are in the 10-30 ppm, the apparent drift rate observed here is hence much larger,

254 and must be compensated. To do so, we implement a digital trimming compensation algorithm. This  
 255 algorithm uses “tick skipping”: it periodically adds or subtracts a tick from the lengths of the slot.

256 As the slot communication feature of TSCH, the offset between SC $\mu$ M and OpenMote is measured  
 257 as the offset of their slot boundaries. Because of the frequency error, the slot length of SC $\mu$ M is longer  
 258 or shorter than OpenMote, which leads to the time offset between them. After a certain duration, the  
 259 offset will accumulate to 1 tick (e.g. 30.5 $\mu$ s at 32768Hz). Tick trimming is applied then, to extend or  
 260 short the current slot length to compensate the offset.

261 Alg. 1 shows how the digital trimming works in pseudo-code.  $TC_x$  indicates the time correction  
 262 in ticks at synchronizing time  $T_{Sync_x}$ .  $I_{sync}$  indicates the synchronization interval in seconds.  $T_{numSlots}$   
 263 indicates the synchronization interval in number of slots. *Drift* indicates the number of slots drifting  
 264 for one tick. The digital trimming procedure is shown in the second part of the algorithm. More details  
 265 that how the trimming compensation is implemented is explained in [23].

---

#### Algorithm 1: Digital trimming algorithm

---

**Result:** Calculate Drift

$TC_x = TimeCorrection(T_{Sync_x})$  where  $x = 1, 2, 3, \dots$ ;

$I_{sync} = T_{Sync_x} - T_{Sync_{x-1}}$ ;

$T_{numSlots} = I_{sync} / S_{duration}$ ;

$Drift = T_{numSlots} / (TC_x - TC_{x-1})$ ;

**Result:** Digital Trimming

$SlotCounter = Drift$ ;

**if**  $IsNewSlot$  **then**

$SlotCounter = SlotCounter - 1$ ;

**if**  $SlotCounter = 0$  **then**

$SlotDuration = SlotDuration \pm 1tick$ ;

**end**

$IsNewSlot = 0$ ;

**end**

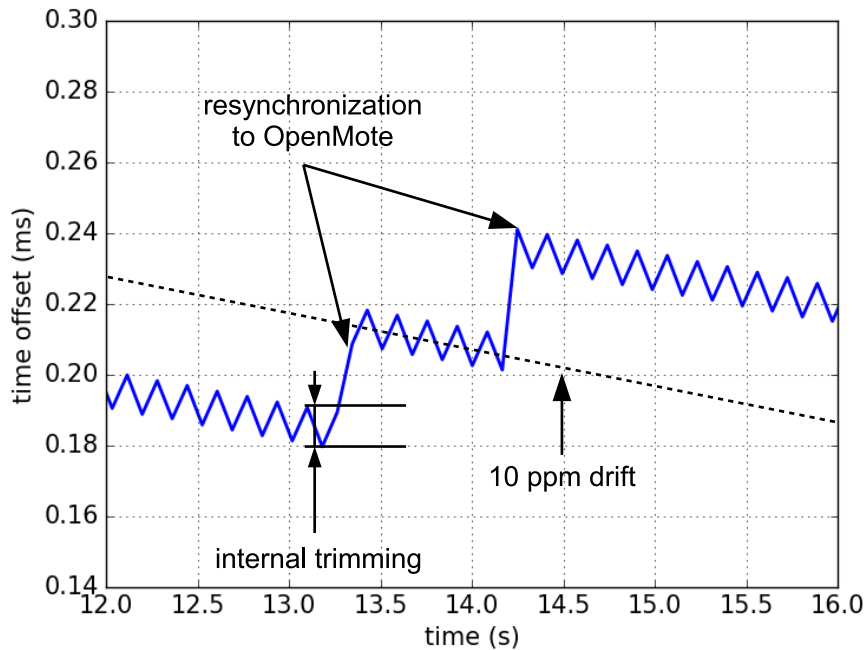
---

266 An I/O pin is toggled at the beginning of each slot on both SC $\mu$ M and OpenMote side. By  
 267 connecting logic analyzer to that I/O pin on OpenMote and SC $\mu$ M, we calculate the time offset  
 268 between SC $\mu$ M and OpenMote. Fig. 9 is the resulting offset when applying digital trimming. It shows  
 269 two effects working together. First, the digital trimming causes the rapid saw-tooth like compensation,  
 270 resulting in a much more manageable 10 ppm apparent drift. Second, SC $\mu$ M regularly synchronizes  
 271 to OpenMote, causing larger jumps. Overall, at constant temperature, SC $\mu$ M and OpenMote remain  
 272 synchronized with a synchronization error not exceeding 300  $\mu$ s.

273 This compensation allows us to build a proof-of-concept 6TiSCH network composed of one  
 274 OpenMote and one SC $\mu$ M. Since both implement the full protocol stack, the full functionality is  
 275 available. By setting the OpenMote as dagroot, it starts to send Enhance Beacons (EB), to which SC $\mu$ M  
 276 synchronizes. SC $\mu$ M then receives RPL DIOs packets [24], which allow SC $\mu$ M to identify OpenMote  
 277 as its routing parent. Using the 6TiSCH 6top protocol, SC $\mu$ M reserves a transmit cell to OpenMote.  
 278 We are able to issue an ICMPv6 echo request (ping) from the PC running OpenVisualizer to the IPv6  
 279 address of SC $\mu$ M and see SC $\mu$ M respond. To the best of our knowledge, this is the first example of a  
 280 synchronized protocol stack (in this case 6TiSCH) running on a crystal-free chip.

## 281 6. Conclusion

282 This paper details the first example of a synchronized network protocol running on a crystal-free  
 283 device. We use the Single Chip micro-Mote (SC $\mu$ M), a state-of-the-art 2 $\times$ 3 mm<sup>2</sup> crystal-free  
 284 mote-on-a-chip, which features an ARM Cortex-M0 micro-controller and an IEEE802.15.4 radio. We use  
 285 the OpenWSN protocol stack, the reference open-source implementation of 6TiSCH, a synchronized



**Figure 9.** Time offset between SC $\mu$ M and OpenMote when SC $\mu$ M periodically re-synchronizes to OpenMote. SC $\mu$ M uses a digital trimming (tick skipping) algorithm to compensate for the drift shown in Fig. 8. The result is that SC $\mu$ M stays synchronized to OpenMote within 300  $\mu$ s of synchronization error.

286 protocol stack being standardized at the IETF. The challenge is that SC $\mu$ M has no stable clock source,  
 287 making synchronized communication hard.

288 In our solution, SC $\mu$ M first listens to a blinking LED to provide coarse calibration of its oscillators.  
 289 Using an OpenMote, which can measure and report the frequency offset, provides a second level of  
 290 more precise tuning. Finally, as SC $\mu$ M and OpenMote are communicating, the OpenWSN port on  
 291 SC $\mu$ M uses a digital trimming compensation algorithm based on tick skipping to turn a 567 ppm  
 292 apparent drift due to a rounding error into a 10 ppm apparent drift. This allows a synchronized fully  
 293 functional 6TiSCH network to form between SC $\mu$ M and OpenMote.

294 This is only a first step, with several avenues for follow-up work. First, in the platform used for  
 295 this paper, the RFTimer is actually driven by the FPGA's crystal oscillator. This, in and of itself, does  
 296 not conceptually break the crystal-free nature of this work, because the frequency source used by the  
 297 analog part of the chip comes from an oscillator not locked in hardware to the FPGA's crystal oscillator.  
 298 A new version of SC $\mu$ M is about to be tested in which this shortcoming is lifted, and on which we can  
 299 use the on-board RC oscillator to drive the RFTimer.

300 Second, all communication in this paper is done on a single frequency, 2.405 GHz. The 6TiSCH  
 301 stack is meant for channel hopping, in which the devices hop across all 16 frequencies of the 2.4 GHz  
 302 ISM band in a pseudo-random fashion. Doing so on SC $\mu$ M would mean repeating the calibration on  
 303 each of the frequencies, either keeping track of individual tuning parameters of each frequencies, or  
 304 designing a methodology for finding one factor given that of another frequency. We have started that  
 305 work, which was presented in a previously published paper [11].

306 Third, the network presented here is a first step only, the ultimate goal being to build a true  
 307 multi-hop mesh network composed of a combination of SC $\mu$ M and OpenMotes, and eventually only  
 308 SC $\mu$ M chips. The challenge with that is that SC $\mu$ M-to-SC $\mu$ M communication is significantly harder  
 309 than SC $\mu$ M-to-OpenMote, because of the lack of stable clock now on both devices communicating.

310 The promise of true crystal-free architectures is, however, enormous. They are the last stepping  
311 stone to realize the “smart dust” vision, allowing for ubiquitous, extremely small and cheap wireless  
312 devices.

313

- 314 1. WirelessHART Specification 75: TDMA Data-Link Layer. Technical report, HART Communication  
315 Foundation, 2008.
- 316 2. ISA-100.11a-2011: Wireless Systems for Industrial Automation: Process Control and Related Applications.  
317 Technical report, International Society of Automation (ISA), 2011.
- 318 3. IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, IEEE Computer  
319 Society, 2015.
- 320 4. Vilajosana, X.; Watteyne, T.; Vučinić, M.; Chang, T.; Pister, K. 6TiSCH: Industrial Performance for IPv6  
321 Internet-of-Things Networks. *Proceedings of the IEEE* **2019**, pp. 1–13.
- 322 5. Watteyne, T.; Weiss, J.; Doherty, L.; Simon, J. Industrial IEEE802.15.4e networks: Performance and  
323 Trade-offs. IEEE International Conference on Communications (ICC), Internet of Things Symposium; ,  
324 2015.
- 325 6. Wisser, B.; Sankaragomathi, K.; Schauer, J.; Korhummel, S.; Kavousian, P.; Yeager, D.; Arumugam,  
326 N.; Pletcher, N.; Barkin, D.; Parker, R.; Callaghan, L.; Ruby, R.; Otis, B. A 1.53 mm<sup>3</sup> Crystal-less  
327 Standards-compliant Bluetooth Low Energy Module for Volume Constrained Wireless Sensors. Symp. on  
328 VLSI Circuits. IEEE, 2019.
- 329 7. Maksimovic, F.; Wheeler, B.; Burnett, D.C.; Khan, O.; Mesri, S.; Suci, I.; Lee, L.; Moreno, A.; Sundararajan,  
330 A.; Zhou, B.; Zoll, R.; Ng, A.; Chang, T.; Vilajosana, X.; Watteyne, T.; Niknejad, A.; Pister, K. A Crystal-Free  
331 Single-Chip Micro Mote with Integrated 802.15.4 Compatible Transceiver, sub-mW BLE Compatible Beacon  
332 Transmitter, and Cortex M0. Symp. on VLSI Circuits. IEEE, 2019.
- 333 8. Wu, X.; Lee, I.; Dong, Q.; Yang, K.; Kim, D.; Wang, J.; Peng, Y.; Zhang, Y.; Saligane, M.; Yasuda, M.; Kumeno,  
334 K.; Ohno, F.; Miyoshi, S.; Kawaminami, M.; Sylvester, D.; Blaauw, D. A 0.04mm<sup>3</sup> 16nW Wireless and  
335 Batteryless Sensor System with Integrated Cortex-M0+ Processor and Optical Communication for Cellular  
336 Temperature Measurement. IEEE Symposium on VLSI Circuits (CLSI); IEEE, , 2018.
- 337 9. Chuo, L.X.; Shi, Y.; Luo, Z.; Chiotellis, N.; Foo, Z.; Kim, G.; Kim, Y.; Grbic, A.; Wentzloff, D.; Kim, H.S.;  
338 Blaauw, D. A 915MHz Asymmetric Radio Using Q-Enhanced Amplifier for a Fully Integrated 3x3x3mm<sup>3</sup>  
339 Wireless Sensor Node with 20m Non-Line-of-Sight Communication. International Solid-State Circuits  
340 Conference (ISSCC); IEEE, , 2017.
- 341 10. Chuo, L.X.; Kim, Y.; Chiotellis, N.; Yasuda, M.; Miyoshi, S.; Kawaminami, M.; Grbic, A.; Wentzloff, D.;  
342 Kim, H.S.; Blaauw, D. A 4x4x4-mm<sup>3</sup> Fully Integrated Sensor-to-Sensor Radio using Carrier Frequency  
343 Interlocking IF Receiver with -94 dBm Sensitivity. IEEE Radio Frequency Integrated Circuits Symposium  
344 (RFIC); IEEE, , 2019.
- 345 11. Suci, I.; Maksimovic, F.; Burnett, D.; Khan, O.; Wheeler, B.; Sundararajan, A.; Watteyne, T.; Vilajosana, X.;  
346 Pister, K. Experimental Clock Calibration on a Crystal-Free Mote-on-a-Chip. Computer and Networking  
347 Experimental Research using Testbeds (IEEE CNERT), 2019.
- 348 12. Warneke, B.; Last, M.; Liebowitz, B.; Pister, K. Smart Dust: Communicating with a Cubic Millimeter  
349 Computer. *Computer* **2001**, pp. 44–51.
- 350 13. Vilajosana, X.; Tuset, P.; Watteyne, T.; Pister, K. OpenMote: Open-Source Prototyping Platform for the  
351 Industrial IoT. International Conference on Ad Hoc Networks; Springer, , 2015.
- 352 14. Vilajosana, X.; Pister, K.; Watteyne, T. Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH)  
353 Configuration. Technical report, IETF, 2017. RFC8180.
- 354 15. Chang, T.; Vučinić, M.; Vilajosana, X.; Duquenois, S.; Dujovne, D. 6TiSCH Minimal Scheduling Function  
355 (MSF). Technical report, IETF, 2019. draft-ietf-6tisch-msf (Work-in-Progress).
- 356 16. Wang, Q.; Vilajosana, X.; Watteyne, T. 6TiSCH Operation Sublayer (6top) Protocol (6P). Technical report,  
357 IETF, 2018. RFC8480.
- 358 17. Vučinić, M.; Simon, J.; Pister, K.; Richardson, M. Minimal Security Framework for 6TiSCH. Technical  
359 report, IETF, 2019. draft-ietf-6tisch-minimal-security (Work-in-Progress).



- 360 18. Watteyne, T.; Vilajosana, X.; Kerkez, B.; Chraim, F.; Weekly, K.; Wang, Q.; Glaser, S.; Pister, K. OpenWSN: a  
361 Standards-based Low-power Wireless Development Environment. *Emerging Telecommunications Technologies*  
362 **2012**, pp. 480–493.
- 363 19. Wheeler, B.; Ng, A.; Kilberg, B.; Maksimovic, F.; Pister, K. A Low-Power Optical Receiver for Contact-free  
364 Programming and 3D Localization of Autonomous Microsystems. *Ubiquitous Computing, Electronics &*  
365 *Mobile Comm. Conf. (UEMCON)*. IEEE, 2019.
- 366 20. Wheeler, B.; Maksimovic, F.; Baniasadi, N.; Mesri, S.; Khan, O.; Burnett, D.; Niknejad, A.; Pister, K.  
367 Crystal-free narrow-band radios for low-cost IoT. *IEEE Radio Frequency Integrated Circuits Symposium*  
368 *(RFIC)*; IEEE, , 2017.
- 369 21. Burnett, D.; Wheeler, B.; Lee, L.; Maksimovic, F.; Sundararajan, A.; Khan, O.; Pister, K. CMOS oscillators  
370 to satisfy 802.15.4 and Bluetooth LE PHY specifications without a crystal reference. *IEEE 9th Annual*  
371 *Computing and Communication Workshop and Conference (CCWC)*; IEEE, , 2019.
- 372 22. Mesri, S. Design and User Guide for the Single Chip Mote Digital System. Master's thesis, Electrical  
373 Engineering and Computer Sciences, University of California at Berkeley, 2016.
- 374 23. Chang, T.; Watteyne, T.; Wang, Q.; Pister, K. Adaptive synchronization in multi-hop TSCH networks.  
375 *Computer Networks, Elsevier* **2015**, pp. 165–176.
- 376 24. Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Vasseur, J.; Alexander, R. RPL: IPv6 Routing  
377 Protocol for Low-Power and Lossy Networks. Technical report, IETF, 2012. RFC6550.

378 © 2020 by the authors. Submitted to *Sensors* for possible open access publication under the terms and conditions  
379 of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).