Algorithm to Detect Advertising Packets for Bluetooth Low Energy Wake-up Receivers

Remon Cents
Integrated Circuit Design, University of Twente, Enschede, The Netherlands
Email: r.b.a.cents@student.utwente.nl

***Abstract*—This paper presents the design of an algorithm to improve the detection of Bluetooth Low Energy advertising events in wake-up receivers. The goal is to decrease both latency and power consumption by reducing missed wake-ups and false wake-ups. The algorithm takes into account the Bluetooth Low Energy specifications including the packet length, packet interval, channel hopping, and a self-learning procedure of the advertising event interval. In the ideal case, the algorithm has a delay of 2 advertising events. Simulations show that in an environment with white Gaussian noise, the main receiver will wake-up consistently when required, with no false wake-ups, if the SNR is at least 0dB. Sensitivity can be sacrificed to decrease susceptibility to interference and noise power variations.**

I. Introduction

In internet of things (IoT) applications, low power consumption is important and most of the power is consumed in the communication link. Receivers have to monitor the channel all the time in order to receive the data as fast as possible, which results in wasted power if there is no data to receive. Wake-up receivers (WuRx), which have a far lower power consumption compared to the main receiver, can be used to monitor the channel continuously and wake-up the main receiver only when data is available. Bluetooth Low Energy (BLE) is known to have low power consumption, and this can be further improved by implementing a WuRx. The detection of BLE signals in the WuRx has to be optimized to prevent false wake-ups and reduce latency. Prior literature has already shown techniques on how to optimize this detection. This paper will use the research found in [1] as a foundation to further enhance the detection of BLE signals in the WuRx.

II. Specification

In order to initiate a connection in BLE, one device will start transmitting advertising packets on the three primary advertising channels 37, 38 and 39. This is called an advertising event and is shown in Fig. 1. The other device, which will contain a WuRx, will scan for these advertising events. If the WuRx has detected a device which is advertising, the main receiver is waken up and a connection can be established [2].

Regarding Fig. 1, it is important to mention that within the advertising event, the same packet is sent consecutively across the 3 advertising channels. The length of the packet itself is between 128 and 376 μs, and the time duration between each packet in a single advertising event is 10 ms at maximum in low duty cycle mode, and 3.75 ms at maximum in high duty cycle mode. This time duration is called the packet interval, and the low duty cycle mode is considered in the algorithm. If we are looking at multiple advertising events as seen in Fig. 2, there is a certain time observed between the advertising events. This time duration is called the advertising interval and is fixed or is manually adjustable on the transmitting device. This advertising interval can be set between 20 ms and 10.24 s, with steps of 625 μs. Furthermore, the device adds a pseudo-random delay of 0-10 ms to this advertising interval. Also, with regard to analyzing noise and setting the sampling frequency, it is important to know that all 3 primary advertising channels have a bandwidth of 2 MHz [1] [2].

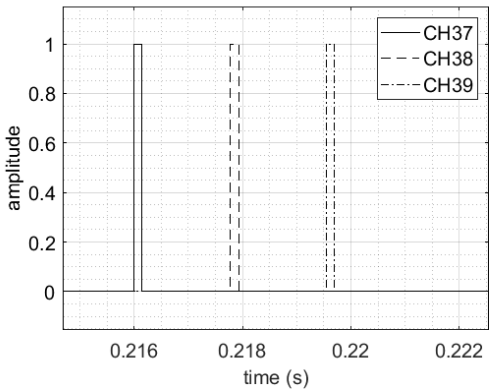

Fig. 1. Single BLE advertising event

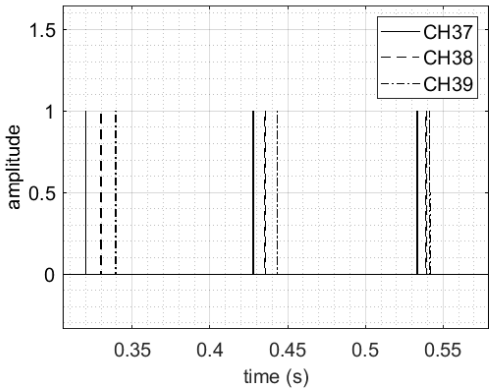

Fig. 2. Three BLE advertising events

III. Algorithm

In order to minimize latency and prevent false wake-ups of the main receiver, an algorithm will be implemented which will take the specifications mentioned in section II into account. The algorithm will be designed to operate with a receiver which is only capable of detecting the signal's energy. Therefore, it is only possible to use information regarding the amplitude and the time duration of the signal.

Further in this section, the design and analysis of the algorithm are explained. This algorithm consists of two main parts: The event detection and a self-learning procedure of the advertising interval. First, the detection of an event in a noise-free environment is discussed, which is followed by an explanation of the self-learning procedure of the advertising interval and finally, the detection of events in a noisy environment is analyzed.

A. Event detection

The event detection procedure is based on the assumption that the transmitter will operate in low duty cycle mode, as described in section II. Also, it is mainly based on the work that is discussed in [1], and is described in detail in this subsection. A visualization of the event detection algorithm is shown in Fig. 3.

Starting at channel 37, when the signal is above a certain threshold, the system will start counting to calculate how long the signal is above this threshold. When this measured packet length is $< 128\ \mu s$, this means the received packet is not a BLE advertising packet and the system keeps monitoring channel 37. However when the measured packet length is $> 376\ \mu s$, the system will switch to channel 38 as this packet can be a BLE advertising packet with interference extending it. In addition, this switch will happen as soon as the signal is exceeding the maximum specified BLE packet length of 376 μs. This will ensure the next packet on channel 38 is not missed. In case the signal is within correct time bounds, the system will also switch to channel 38 and remember a BLE advertising packet has been detected.

On channel 38, the situation is almost the same. However, if the next packet (within bounds or too long) does not arrive within the specified 10 ms packet interval, the system will switch back to channel 37 instead of moving to channel 39, as there is no (potential) BLE advertising packet observed. Moving back to channel 37 prevents a false wake-up when interference is detected in channel 39, as well as preventing missing a Bluetooth packet in channel 37 because the system is watching channel 39 for an additional 10 ms.

On channel 39, the situation is exactly as in channel 38. After the detection of a packet which is either $> 376\ \mu s$ or within bounds, the system will evaluate whether the length of two or more of the detected packets in the three channels was within bounds. If this is the case, the system has detected this as a BLE advertising event. However, the main receiver will not wake up yet. The time of the start of this detected advertising event is saved and analyzed by the self-learning advertising interval procedure, which is part of the algorithm.

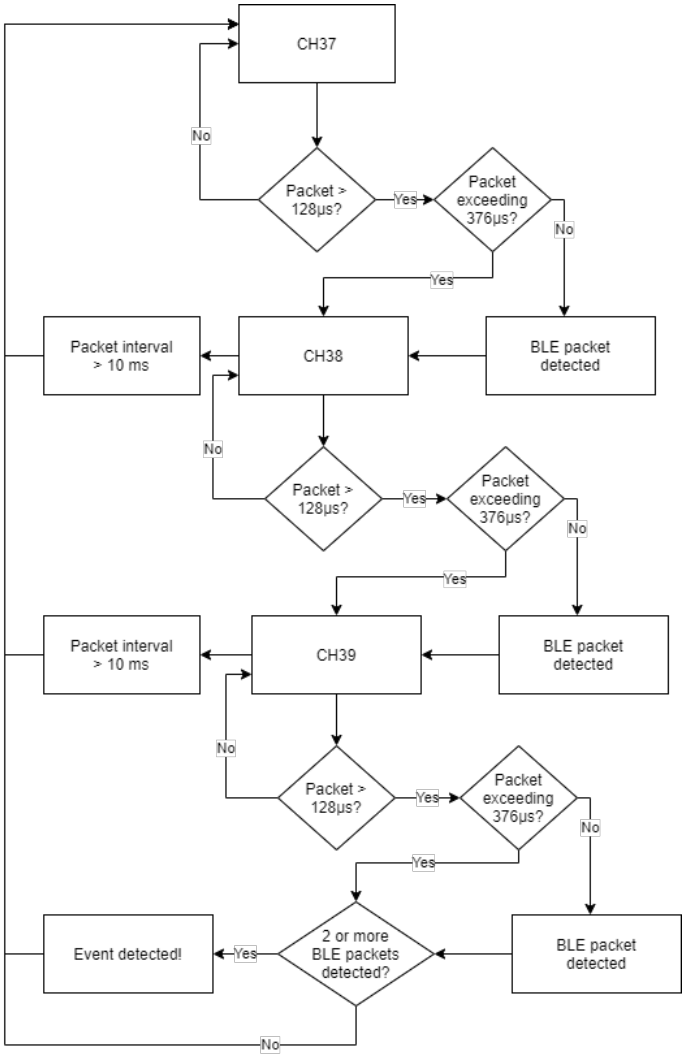

Fig. 3. Block diagram of the event detection procedure

B. Self-learning advertising interval

To further reduce the number of false wake-ups, a self-learning function is implemented which will measure the time between detected advertising events. This is done by making use of the saved timestamps when every detected advertising event started. Also, note that the time of the start of an event is saved instead of the actual timestamp it was detected. This is because the set advertising interval is determined from the start and not the end of an advertising event [3]. In terms of the requirements, this system should be able to function when multiple BLE devices are simultaneously advertising. This means multiple intervals have to be detected out of the single number sequence containing all the timestamps at which advertising events were detected. An interval can only be detected when there are at least three advertising events,

as with two advertising events, it is far less certain if those two events were from the same device. The downside is that every time a new device is within range, the first 2 advertising packets are not causing a wake-up, and thus an unavoidable delay of 2 advertising events is observed. An example of the detection of such an interval is visually shown in Fig. 4, and can be done by looking at the earlier mentioned number sequence. This will be done every time a new advertising event

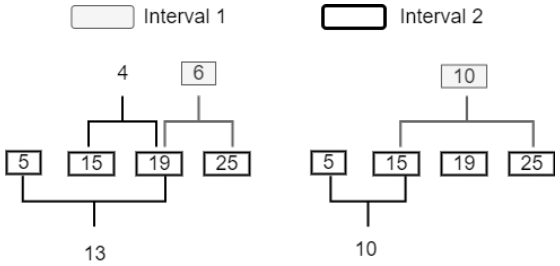

Fig. 4. Visualisation of the advertising event interval calculation

has been detected. Let us consider $x[n]$ the sequence of the timestamps of all detected events and N the total number of detected events. We will first look at the interval from $x[N]$ to $x[N-1]$, called "interval 1", which is "6" in Fig 4. After that, the intervals from $x[N-1]$ to all earlier events are calculated, called "interval 2" ($\{4, 13\}$ in Fig 4). If there exists an "interval 2" that matches "interval 1", a pattern has been detected and the main receiver wakes up. However if no match is found, as in the left part of Fig. 4, the system will repeat the same steps but now "interval 1" is the interval between $x[N]$ and $x[N-2]$ and "interval 2" the interval between $x[N-2]$ and all earlier events. In the example in Fig. 4, this results in a match (both intervals are "10") and the main receiver wakes up. If no match is found, this process will repeat until "interval 1" is defined between $x[N]$ and $x[2]$. This process is mathematically shown in equation 1 and 2. Equation 1 shows the calculation of "interval 1", with $y_1[n]$ being "interval 1".

$$y_1[n] = x[N] - x[N-n], n = 1, ..., N-2, N \geq 3 \quad (1)$$

The calculation of "interval 2" is mathematically shown in equation 2, with $y_2[k]$ being "interval 2".

$$y_2[k] = x[N-n] - x[N-n-k], k = 1, ..., N-n-1, N \geq 3 \quad (2)$$

"Interval 1" and "interval 2" match if the difference between the two intervals is at most 10 ms, as the advertising interval will be 0-10 ms longer compared to the predefined value, as described in section II. After a match has been found, the 3 corresponding timestamps will be deleted and are no longer be saved as an event detection timestamp. This will decrease the likelihood that event combinations from multiple devices or interference can be recognized as an interval. For further prevention of false wake-ups, events older than $2 \cdot (10.24\,s + 10\,ms)$ will be deleted, as the advertising interval can not be longer than $(10.24s + 10ms)$ as stated in section II. Furthermore, detected intervals that are shorter than 20 ms (the lower limit described in section II) will not cause a wake-up to occur. The timestamp of the 3 advertising events associated with this interval will also be deleted.

C. Event detection with white Gaussian noise

By adding noise, the event detection based on a threshold becomes unreliable at low signal to noise ratios, as the noise fluctuates above and below the threshold. Therefore, the averaging of samples is implemented. This leaves the mean unaltered but decreases the variance of the noise. As the noise power of white Gaussian noise is defined as σ^2, the SNR can be written as: $SNR = \frac{P_{Signal}}{P_{Noise}} = \frac{P_{Signal}}{\sigma^2}$. Whenever the signal averaging window is doubled, the variance is halved. So the larger the averaging window, the higher the effective SNR. However, averaging too much has a downside, as it increases the window in which a packet length is in BLE range. This is illustrated in Fig. 5 and Fig. 6. As the bandwidth is 2 MHz and the minimum packet length is 128 μs (as specified in section II), the sampling rate is assumed to be 4 MHz, resulting in a minimum packet length of is 512 samples. Looking at Fig. 5, it

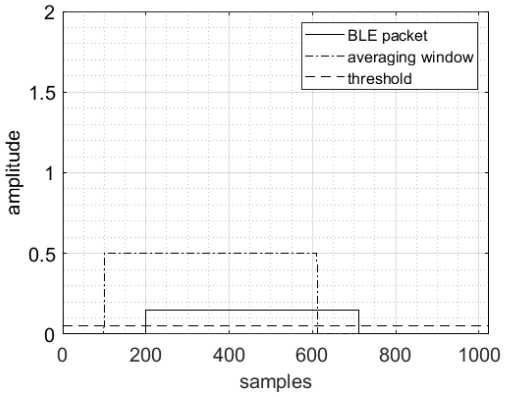

Fig. 5. averaging window with a long duration low power signal

can be seen that for a very low power signal, it can be possible that the complete averaging window has to cover up the whole packet length in order to result in an amplitude which is higher than the detection threshold. Generally, this means the minimum packet length for detection can be expressed as seen in equation 3.

$$\text{Length}_{min} = \text{packetlength}_{min} - (\text{averagingwindow} - 1) \quad (3)$$

So with an averaging window of 512, this means a minimum detection length of 1.

For short high power signals, the opposite is true as depicted in Fig. 6. When a short and high power signal is observed, it is possible that the amplitude is already above the threshold when only one sample of this signal is a part of the averaging window. This means that the maximum packet length for detection can be expressed as shown in equation 4.

$$\text{Length}_{max} = \text{packetlength}_{max} + (\text{averagingwindow} - 1) \quad (4)$$

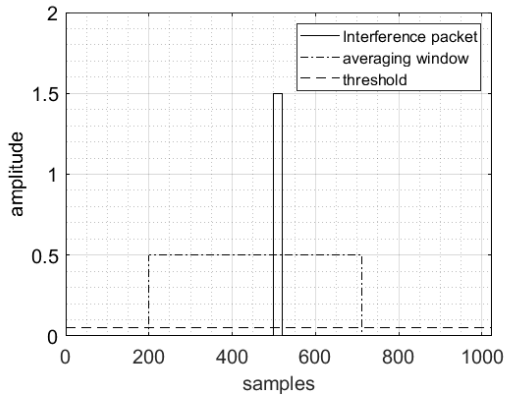

Fig. 6. Averaging window with a short duration high power signal

An unfortunate side effect of this is that a short powerful signal with a length of 10 samples for example, which is not in BLE range and thus not a BLE-packet, at an averaging window of 512, can be measured as a signal of length $10 + (512 - 1) = 521$ samples, which is considered to be a BLE-packet, even without the adjusted lower mentioned limit of 1 mentioned in equation 3. This means that with averaging there is a trade-off between good low-SNR performance and susceptibility to high power interference signals shorter than 128 μs or low power interference signals slightly longer than 376 μs.

IV. Method

The performance of the described algorithm has been evaluated in a MATLAB simulation. For generating the BLE packets, the timing specifications mentioned in section II regarding the packet length and packet interval, are uniform random generated within the specified range. The packet length has been considered the same length on all three channels for one event, as it can be assumed the same data is transmitted. The advertising interval can be set and a uniform random delay of 0-10 ms will be added. Noise has been modeled with a fixed level of white Gaussian noise. The SNR can be adjusted which will alter the signal strength. This signal with noise will be squared in order to obtain the signal power. The algorithm itself has been implemented as described in section III, while making sure the analysis is done in real-time and only one channel is monitored at the same time, as this would also be the case in a real receiver. For verifying the performance, a successful detection is considered within $\pm$ 10 ms of the time an advertisement has been sent in the simulation. Any wake-up outside of this time interval is considered a false wake-up.

In order to verify that the simulation has been set up correctly, the algorithm has temporarily been simplified to using only 1 channel. The real performance will be compared to the theoretical performance by the means of probability calculations. Equations 34 until 37 from [4] have been used to perform these calculations.

In order to verify whether the self-learning algorithm is functioning as expected, three tests can be performed.
for all tests the noise will be removed in the simulation.
1. A single device transmitting with a 100 ms advertising interval. It is expected that the receiver wakes up at the third event.
2. Two 'devices' transmitting at the same time with different advertising intervals of 1 s and 100 ms, while removing the 0-10 ms random delay to prevent collisions. It is expected that the receiver wakes up at the third event of both "devices".
3. A single device transmitting with a 1s advertising interval, but increasing the random added delay to 0-500 ms. A relatively high missed wake-up rate is expected as a large portion of the advertisements violate the $\pm$ 10 ms difference compared to the calculated interval.

For measuring the performance of the algorithm, multiple tests will be performed. The performance will be measured in terms of false wake-ups and missed wake-ups. Note that the self-learning advertising interval procedure always adds a delay of 2 advertising events when no missed wake-up is observed. The key variables that will be changed to tune the performance are the detection threshold and the size of the averaging window. The impact on false wake-ups and missed advertisements of these variables will be analyzed for a wide range of thresholds and averaging window sizes for a fixed SNR. Finally, with optimal values for these variables, the performance dependency on the SNR will be evaluated and also compared with an implementation that does not contain the self-learning interval.

For modeling interference, short bursts will be created with a power 10 times higher than the noise power, with a length of 50 μs and a uniform random interval between 1 and 2 ms.

V. Results

The performed probability calculations mentioned in section IV, result in less than 5% difference compared to theory, so it can be assumed the simulation is valid.

Looking at the tests to verify the behaviour of the self-learning function, mentioned in section IV, the results are shown in table I.

TABLE I
VERIFICATION OF THE BEHAVIOUR OF THE SELF-LEARNING INTERVAL FUNCTION, USING THE TESTS DESCRIBED IN SECTION IV

Test No.	Expected	Observed
1	2 out of 3 events do not cause wake-up	382 events caused 127 correct wake-ups
2	2 out of 3 events do not cause a wake-up for both devices	440 events caused correct 146 wake-ups
3	more than 2 out of 3 events do not cause a wake-up	only 15 out of 156 events caused a wake-up (sum of 5 simulations)

In table I it can be seen that the results match the expectations and thus it can be concluded the self-learning interval function behaves as expected.

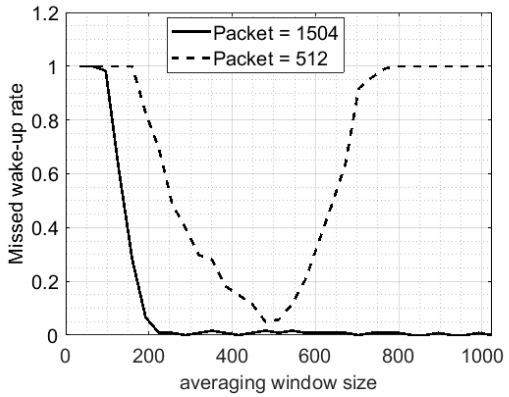

Fig. 7. Simulation showing missed wake-up rate compared to the averaging window size. Simulation time = 40s with advertising event interval = 100ms, SNR = -1dB and threshold = $1.6 \cdot P_{noise}$. No interference.

Fig. 7 shows that with regard to the averaging window size, the lowest missed wake-up rate is observed with an averaging window size of around 500 samples (with packet length = 512 samples). When the packet length is at the highest possible value of 1504 samples, the missed wake-up rate does not start to increase above 512. Furthermore, with a packet length of 1504 samples, a lower missed wake-up rate is observed for the same averaging window compared to a packet length of 512. False wake-ups were not observed and are therefore not plotted. Expected is that the larger the averaging window, the better the performance, as the effective SNR increases. However, given that the packet length of a BLE packet is between 512 and 1504 samples and the low power, the performance is expected to decrease when the averaging window exceeds the packet length. If the BLE packet signal drops below the threshold due to the low power, this would cause a higher missed wake-up rate for the shorter packet compared to the longer packet. This is because with the longer packet, the BLE packet signal can drop below the threshold multiple times, while the observed packet length is still longer than the calculated minimum from equation 3.

As it can be seen in Fig. 8, the threshold greatly impacts the performance. When the threshold approaches the noise power from either side, a number of false wake-ups are observed. The missed wake-up rate has decreased to almost 0 between 1.4 and 1.8 multiples of the noise power. A threshold approaching $2 \cdot P_{noise}$ will increase the missed wake-up rate, as the signal starts to occasionally drop below this threshold. Furthermore, the false wake-ups are caused by the averaged noise being above the threshold for the right amount of time (given the averaging window of 512 samples, this is between 1 and 2015 samples). The false wake-ups are not observed very close to the noise power as the events are detected in

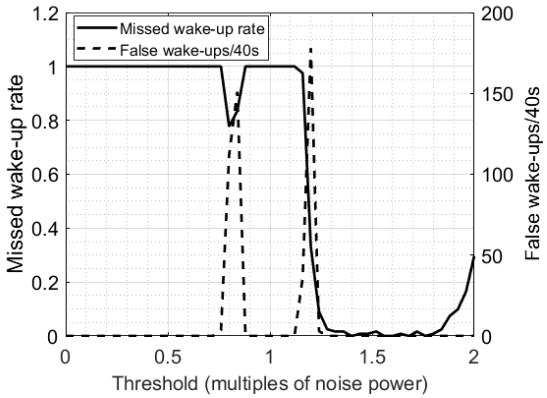

Fig. 8. Simulation showing missed wake-up rate and false wake-ups/40s compared to the threshold. Simulation time = 40s with advertising event interval = 100ms, SNR = 0dB and averaging window = 512. No interference.

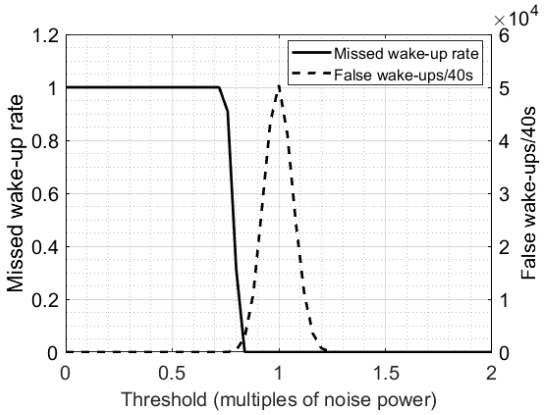

Fig. 9. Identical setup as in Fig. 8, with the self-learning function disabled.

an interval shorter than 20 ms, thus a wake-up is prevented by the self-learning function (Fig. 9 illustrates the result with the self-learning function disabled). The dip in the missed wake-up rate at around $0.7 \cdot P_{noise}$ is caused by these false wake-ups occurring within the time the advertisements were sent $\pm$ 10 ms (and thus considered as a correct detection). It can be concluded that the threshold should always be greater than approximately $1.4 \cdot P_{noise}$ to prevent false wake-ups and a high missed wake-up rate. Also the threshold should not be too large as the missed wake-up rate will increase for low SNR signals.

Looking at Fig. 10, it can be seen that when optimizing the averaging window and threshold parameters, the missed wake-up rate has decreased to almost 0 above a 0dB SNR, regardless of the self-learning function being enabled or disabled. False wake-ups were not observed and therefore not plotted. However when enabling the self-learning function, the missed wake-up rate approaches 1 at a slightly lower SNR compared to the situation where this self-learning function is disabled. With the self-learning interval enabled, 3 consecutive advertising events have to be detected correctly to cause a wake-up, which explains this small difference.

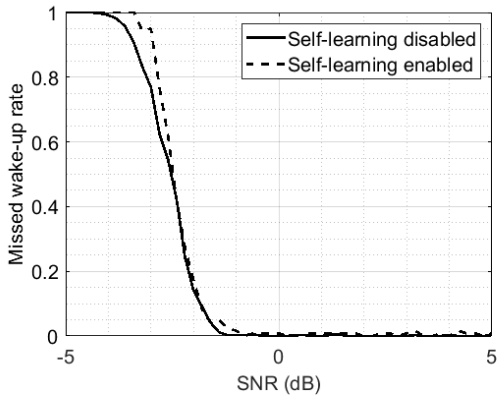

Fig. 10. Simulation showing missed wake-up rate with regard to the SNR. Simulation time = 40s with event interval = 100ms, threshold = 1.6·noise-power and averaging window = 512. No interference.

TABLE II
SIMULATION WITH INTERFERENCE AS DESCRIBED IN SECTION IV, SIMULATION TIME = 40S WITH EVENT INTERVAL = 100MS, SNR = 10DB, AVERAGING WINDOW = 512, THRESHOLD = $1.6 \cdot P_{noise}$. SELF LEARNING INTERVAL ENABLED.
NOTE 1: WAKE-UPS TOO EARLY (20 MS MAX) BUT ASSOCIATED WITH AN ADVERTISING EVENT.

Interference on channels	**Missed wake-up rate (%)**	**false wake-ups/100s**
37	11	0
38	40	0
39	0	0
37 & 38	22	0
37 & 39	11	0
38 & 39[1]	0	0
All (avg. window = 512 samples)	100	0
All (self-learning disabled)	0	7132
All (avg. window = 100 samples)	0	0

Table II shows an increased missed wake-up rate when the interference is on CH37, CH38 or both. When the interference is present on all channels, no wake-ups are observed at all. Disabling the self-learning function results in a very high number of false wake-ups. When lowering the averaging window, the interference does not have an effect at all on the detections. From table II, it can be concluded that with an averaging window of 512, the interference is detected as a BLE packet. A downside also described in section III-C. This is indicated by the large number of false wake-ups when all channels contain interference and the self-learning interval is disabled. This is also indicated by the higher error rate when interference is present only on CH37 or CH38 (receiver switches to next channel too early), but no error is observed when the interference is at CH39 (interference causes the detection in CH39, after a correct detection in CH37 and CH38). the fact that no wake-ups are observed at all when the self-learning function is enabled is because events are detected with an interval smaller than $20ms$, which means the self-learning function prevents wake-ups. When lowering the averaging, there are no missed detections or false wake-ups at all when the interference is present on all 3 channels. This can be explained by equations 3 and 4, and means the interference packet is not recognized as a BLE packet anymore.

VI. CONCLUSION

The designed algorithm is able to consistently wake up the main receiver when required, with a delay of 2 advertising events and practically no false wake-ups, up to a SNR of 0dB, assuming no variations in the noise power or interference. By sacrificing the sensitivity a threshold can be adjusted to improve robustness against variations in noise power and an averaging window can be made smaller to improve interference susceptibility in practice. The self-learning advertising interval function has the potential to drastically reduce the number of false wake-ups if they occur, at the cost of a delay of 2 advertising events.

REFERENCES

[1] P. Wang and P. Mercier, "A 220μW -85dBm sensitivity BLE-compliant wake-up receiver achieving -60dB SIR via single-die multi- channel FBAR-based filtering and a 4-dimentional wake-up signature," *Proc. IEEE Int. Solid-State Circuits Conf. - Dig. Tech. Papers (ISSCC)*, 2019.
[2] M. Afaneh, "Intro to bluetooth low energy," 2018.
[3] "Bluetooth core specification v5.2," p. 2939, Dec. 2019.
[4] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proceedings of the IEEE*, vol. 55, no. 4, Apr. 1967.