An Architecture for Sender-based Addressing for Selective Sensor Network Wake-Up Receivers

Johannes Blobel, Janis Krasemann and Falko Dressler
Distributed Embedded Systems (CCS Labs), Heinz Nixdorf Institute, Paderborn University, Germany
{blobel,dressler}@ccs-labs.org, janiskra@mail.uni-paderborn.de

Abstract—**Duty cycling has been the main concept for saving energy in sensor networks for a long time. Yet, additional overhead for synchronization and the fact that overhearing and idle listening cannot completely be prevented, motivated further research. Wake-up receiver, i.e., additional ultra-low power radios that are not switched off and can receive a so called wake-up signal, aim to fill this gap. The wake-up signal can either be a simple energy burst that wakes up all nodes in reception range or can include an address to wake up individual nodes. In this paper, we introduce a novel addressing scheme for wake-up receivers that supports broadcast, multicast, and unicast wake-ups, and which can be realized in hardware with little additional complexity at the receiver. We implemented a prototype based on a commercially available wake-up receiver to show the feasibility of our approach. We discuss the tradeoff between the length of the wake-up signal and the possible energy savings and show an application example for sending software updates to sensor nodes in an energy saving way.**

I. Introduction

A crucial property of Wireless Sensor Networks (WSNs) is the limitation of resources, foremost their restricted energy capacity and, thus, network lifetime [1], [2]. As a powerful concept, duty cycling is used for many of such networks where a node is in a low-power sleep mode for most of the time and only wakes up shortly to fulfill its task like collecting sensor data and exchanging information with other nodes. Since communication between nodes can only be successful if all participating nodes are awake at the same time, the nodes have to be synchronized. This synchronization can been achieved by specialized MAC protocols, like S-MAC, T-MAC, or WiseMAC [3]. However, the additional overhead used for synchronization is not acceptable in ultra low-power scenarios. Also, the existing protocols cannot prevent the waste of energy by overhearing or idle listening completely, which leaves room for further optimizations.

Wake-up Receivers (WuRxs) are a promising approach to solve these shortcomings [4]. They approach the problem of synchronization by adding the ability to wake-up a remote node using a radio signal. Of course, this requires the receiver of the wake-up signal to be turned on the whole time, which means that the main transceiver of a node cannot be used. Instead an additional receiver – the Wake-up Receiver – is added to a node which has very low power consumptions (typically a few µA down to nA [5]) and can therefore be in receiving mode all the time. The low power consumption comes at the cost of a low achievable data rate or low sensitivity. Since this receiver is not used to transmit the actual payload of a node this is an acceptable limitation.

The wake-up signal itself can also contain modulated data that can be used to include additional information to the signal [6]. Today, commercially available wake-up receivers already allow the user to choose between a simple broadcast mode and a pattern recognition mode. In the first mode, the node is woken up upon the reception of a simple energy burst on the carrier frequency. Since this is prone to false wake-ups due to other transmissions on the same frequency and noise, the pattern recognition mode only wakes up a node if a predefined pattern is received. The name "pattern" instead of address already suggests that this is mainly used to reduce the amount of false wake-ups rather than adding an addressing scheme.

In this paper, we show how the ability to encode data within the wake-up signal can be used to implement a versatile addressing scheme for wake-up receivers. With this technique it is possible to send a broadcast, multicast, or unicast wake-up signal. A key feature of our architecture is that the *sender* of the signal can precisely determine which nodes should be woken-up, which adds exiting new possibilities to the whole field of WSNs. We call this a *Selective Wake-up Receiver (SWuRx)*. One key application is the use for software updates in our BATS project. Here, ultra-low power sensor nodes are used to track and localize bats in their natural environment [7]. The developed SWuRx helps to wake-up only those nodes selectively that need to interact with the sender, e.g., for reconfiguration. We developed an hardware prototype for first direct measurements and also analyzed the performance of the resulting system.

Our key contributions can be summarized as follows:

- We introduce a novel addressing scheme for wake-up receivers supporting unicast, multicast, and broadcast and the corresponding hardware architecture (Section III).
- We develop a prototype hardware solution that can be combined with a variety of commercially available wake-up receiver chips and measured the resulting performance (Section IV).
- Using the BATS project example, we investigate the capabilities of the system for normal communication as well as software updates of our mobile systems using simulation (Section V).
- We analyze the potential energy savings in homogeneous networks (Section VI).

II. Related Work

One of the first papers on WuRxs already included a simple addressing scheme based on the utilization of multiple radio frequencies [8]. Since then many systems have been proposed, that differ in sensitivity (which determines the wake-up range), power consumption, delay, and addressing capabilities. An overview and comparison of different WuRx implementations along with a discussion of their benefits and drawbacks can be found in [9] and [4].

Addressing is often done with the help of the microcontroller or with a correlator within the WuRx [9]. A different approach for address matching using the timing between two consecutive signals and the help of the microcontroller was proposed in [10]. The address is encoded within the time between two consecutive signals. Each signal wakes up the microcontroller very shortly to measure the timing and check for a match. The authors compare their addressing scheme to the correlator-based address matching and microcontroller-decoding of the signal. While this scheme only uses very little power, the duration of a wake-up signal is rather long (48 ms to 68 ms), since each address bit is encoded using a delay between two signals.

Bloom-filters for group based ID matching in a WuRx has been presented in [11]. The transmitted wake-up signal includes a bloom-filter to specify a group of nodes that should be woken up. Each node has its own bloom-filter, that specifies the groups it belongs to. If the received signal matches, i.e., the node belongs to the transmitted group, the node is woken up. This scheme cannot wake-up single nodes, but only groups of nodes.

Many WuRx implementations only have a very poor sensitivity, which limits the reception range to less then 10 m. The WuRx presented in [12], however, can reach a sensitivity of –83 dBm This allows for a wake-up distance of 1200 m (when sending with 10 dBm). However, the low current consumption of 3 µA and the good sensitivity comes at the cost of a very high latency of 484 ms.

III. Selective Wake-up Receiver

A. Addressing Concept

Today, commercially available wake-up receivers like the AS3933 from AMS can decode an On-Off-Keying (OOK) modulated signal and check whether this signal contains a certain predefined pattern. The pattern can be configured by the nodes microcontroller via an SPI interface and can be changed during runtime. Even though the main reason for using this pattern recognition is to prevent false wake-ups due to other signals on the same channel, it can also be used as an addressing scheme. If all nodes are set to the same pattern, a wake-up signal containing this pattern will wake up all nodes in communication range and can therefore be seen as a broadcast. If all nodes have a unique pattern configured, this pattern serves as an address and a unicast wake-up is possible. A mix of these two options would be possible if groups of nodes have common patterns configured which would allow a multicast wake-up. The problem with this existing solution is the fact that, the communication scheme has to be pre-configured by

Configured Pattern	1	0	1	1
Received Address	1	0	0	0
Received Mask	1	1	0	0
Match	1	1	1	1

Figure 1. Example of the pattern matching for a *multicast* wake-up

the *receivers*. A switch from broadcast to unicast wake-ups would require all nodes to wake-up and reconfigure their wake-up receiver to listen to a unique pattern. To switch back to broadcast each node would have to be woken up separately to change the pattern back to the common broadcast pattern.

Our selective wake-up receiver adds the possibility to choose the communication scheme by the *sender* of the wake-up signal without the need to reconfigure each node. Before we explain the logic design, let us consider the basic working principle:

- Each node has a unique pattern configured which serves as an address
- The wake-up signal contains an address *and* a mask
- A node is woken up if the bits of the received address determined by the mask match the configured pattern

Figure 1 illustrates a multicast wake-up. The unique pattern configured in the nodes WuRx is shown in the first row (blue). The received signal (address and mask) is shown in the next two rows (green). The first two bits of the mask are set to 1, therefore, the stored pattern and the received address have to be equal at this positions in order to get a match. The last two bits of the mask are 0, indicating that these bits are not relevant for the address matching, therefore these two bits will always match.

Using this technique, the sender can determine the communication scheme by sending a specific mask. If all mask bits are set to 1 only one node with the matching pattern will wake up, which is a unicast wake-up. For a broadcast all bits of the mask are set to 0 which means that every pattern will match any address and all nodes receiving this signal wake up. To send a multicast wake-up signal a subset of bits, that are relevant, are set to 1, which will only wake up a subgroup of nodes.

B. Logic Design

The above mentioned functionality can be added to existing techniques with only very little additional hardware. In fact, only a couple of additional logic gates are required to implement the aforementioned logic. A conventional correlator simply bit-wise compares the received address, that is stored in a register, with a predefined pattern. If all bits match, an interrupt pin is set high, waking up the microcontroller [9].

By using some more logic gates the correlator can be extended to also check the mask of the received signal. As before, the received signal is shifted into a register to store it and then parallel logic gates correlate the received signal with the predefined pattern.

Figure 2 shows the conceptual structure of the SWuRx. The pattern $P = [p_1, p_2, \dots, p_n]$ is set by the microcontroller and is stored in a register. The address $A = [a_1, a_2, \dots, a_n]$ and mask $M = [m_1, m_2, \dots, m_n]$ are received by the wake-up receiver

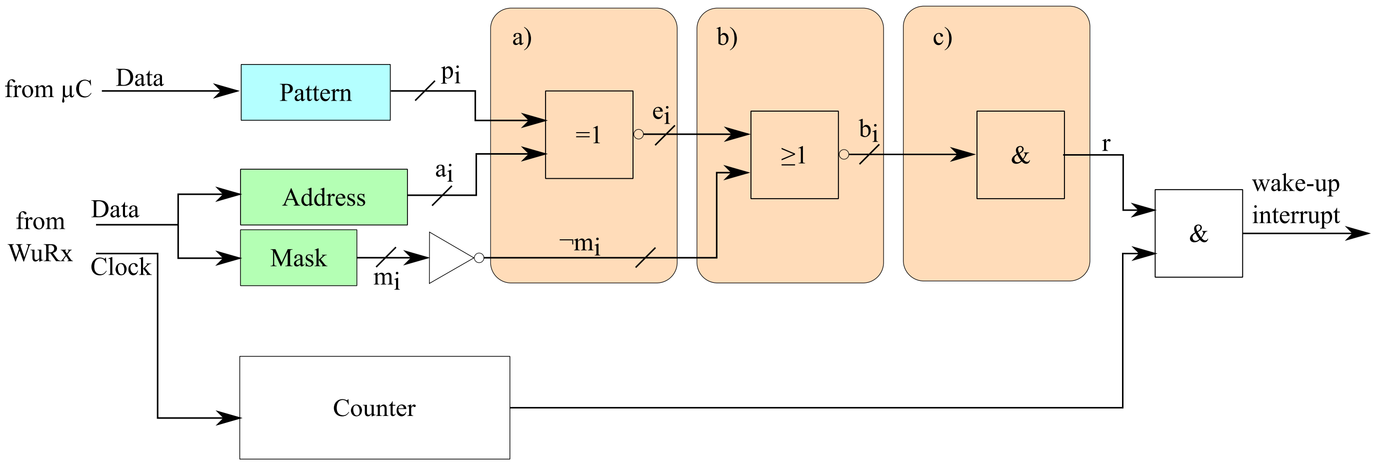


Figure 2. Logical Structure of the SWuRx. The data is shifted into the registers. From there the logic combinations are done in parallel for each bit and finally combined to the resulting signal r. After n clock cycles, the output of the counter goes high, signaling that r is now stable.

and are also shifted into registers. For all bits $i \in \{1 \dots n\}$, the following logical combinations are realized in three steps:

(a) $e_i = \text{XNOR}(a_i, p_i)$: True if pattern and address are equal.
(b) $b_i = \text{OR}(e_i, \neg m_i)$: True if either e_i is 1 or mask is 0.
(c) $r = \text{AND}(b_i \forall i \in \{1 \dots n\})$: True if all bits match.

In other words, the output of the correlator for bit i is either 1 if the mask is 1 *and* the address and pattern are equal or it is 1 if the mask is 0. In the end, all results b_i from phase (b) are merged with one n-AND gate. As a result, the wake-up interrupt will only be generated if for all bits the result of step (b) is 1.

During the shifting of the data into the registers, the result of the correlator can change and be in an undefined state, since the shift register has no latch functionality. Therefore, the clock signal from the WuRx is also send to a counter. After $2n$ clock cycles (n-bit for the address and n-bit for the mask), the output of the counter switches to 1. This signal signifies the end of the reception and therefore means, that the output of the matching logic in step (c) is now stable. If this timer signal and the logic signal are both 1, the interrupt of the microcontroller is triggered which wakes it up from the sleep state.

C. Broadcast Optimization and Mask Encoding

Based on this scheme some modifications are possible to reduce the required number of bits in the wake-up signal. When a broadcast signal is send, all mask bits are set to 0 and the content of the address is completely irrelevant. It is therefore not necessary to actually transmit these bits. If we change the architecture to send the mask first, an interrupt can be raised immediately if n consecutive 0s are received. This can be implemented by using the already existing counter which creates a signal after n clock cycles. This is combined with the inverted mask signals. Using this shortcut, the energy for using a broadcast wake-up signal can be halved.

Further reduction of the wake-up signal can be done, if it is not necessary to mask every address bit individually but only the first m bits. In this case, the mask can be send as a binary encoded number, specifying the number of masked bits. The received mask signal would then have to be decoded before being used in the aforementioned logic.

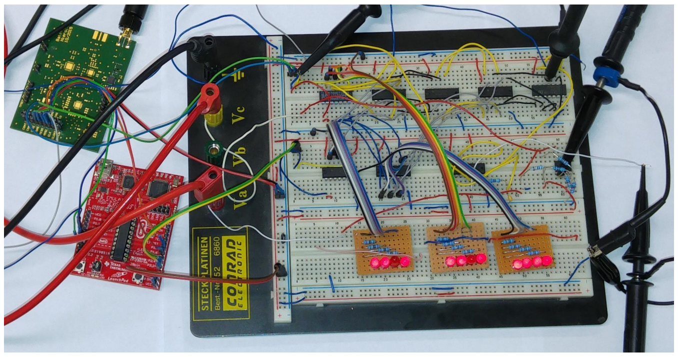
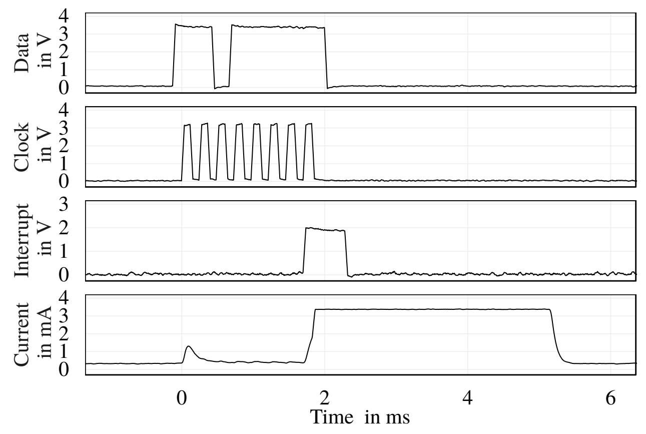


Figure 3. Our SWuRx prototype using standard logic ICs. The green PCB on the top left side contains the WuRx, the red board the microcontroller. Debug LEDs show the content of the registers.

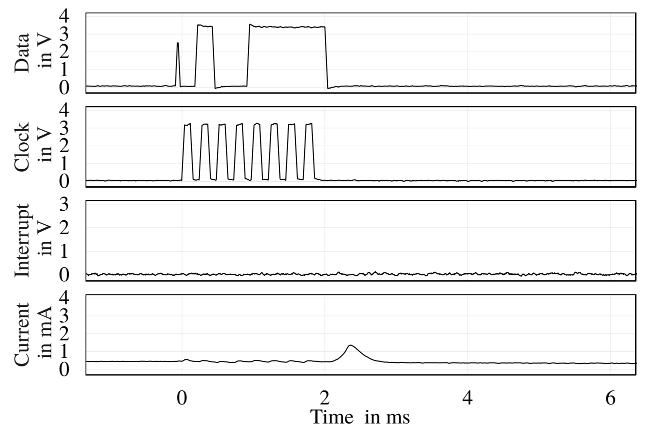
IV. Hardware Prototype

We have built a hardware prototype of the SWuRx using discrete standard logic gates and shift registers. A picture of the prototype can be seen in Figure 3.

Since most standard ICs are available as quadruple gates, our prototype has an address and mask width of 4 bits. The ICs used are from the 74HCxx series, the AND gate from step (c) in the previous section, for example, is a 74HC08 quad 2-input AND gate. As Wake-up Receiver, we used an AS3933 which can receive an OOK modulated, Manchester encoded wake-up signal. In order to be able to use standard transceivers, the antenna input of the AS3933 is preceded by an envelope detector that extracts the envelope of the received RF-signal (868 MHz) and feeds a 125 kHz signal to the WuRx. By using Manchester coding for the wake-up data, the chip can not only retrieve the data signal itself, but can also recover the clock



(a) The received address *does* match the preconfigured pattern: an interrupt is raised.



(b) The received address does *not* match the preconfigured pattern: no interrupt is raised.

Figure 4. Signals and current consumption of the testbed during reception of a unicast wake-up signal.

signal. Both signals, data and clock, are fed to the SWuRx prototype circuit.

The received address and mask are stored in a shift register (74HC164). The outputs of the shift registers are connected to the corresponding logic gates. As a counter we used the 74HC193 4-bit binary counter. The fourth output Q3 of this counter goes high after 8 clock cycles. This counter signal is also fed to an RC delay filter, which resets the whole circuit after 1 ms.

As microcontroller, we used an MSP430G2553 on a TI Launchpad. The microcontroller configures the AS3933 WuRx over its SPI interface and writes a pattern to the corresponding shift register of our circuit. It then goes into the low-power mode LPM4, from which it can only be woken up via an external interrupt. If such an interrupt occurs, the microcontroller runs for 3 ms and then goes to the sleep mode again.

The wake-up signal is generated using GNU Radio, a real-time signal processing framework for use in SDR platforms and send with an Ettus B210 USRP. This allows for a precise control of the wake-up signal and the transmitted data.

A. Qualitative Measurements of Prototype

The developed prototype shows the feasibility of our proposed architecture. We conducted experiments to investigate how the wake-up receiver reacts to different signals and measured the current that was used by the system.

Figure 4a shows the incoming signals *Clock* and *Data* from the AS3933, the wake-up interrupt signal, and the current consumption of the whole testbed. In this example a unicast wake-up signal was received (the last 4 bit are 1) which matches the pattern that was preconfigured by the microcontroller. Therefore at the end of the reception, the wake-up interrupt is raised and the microcontroller wakes up. The total power consumed was 46 mJ. It can be seen that during the reception of the signal, the current consumption is slightly higher. The large discrete logic gates and the charging of the capacitor of the RC delay filter consume considerable power. If the logic would be directly integrated into a silicon chip the additional power required would be much lower.

When a signal is sent that does not match a nodes address, the microcontroller is not woken up. This can be seen in Figure 4b where a unicast signal with a wrong address is sent. As before, the logic circuit consumes a little power during the reception of the signal, but since no interrupt is raised, the microcontroller is not woken up. The total power consumed was 14 mJ.

V. SWuRx for Software Updates of Low-Power Sensor Nodes

In this section we introduce an application example for our proposed SWuRx. It was developed in the context of our current Dynamic Adaptable Applications for Bats Tracking by Embedded Communicating Systems (BATS) project [7]. The project's goal is to enable biologists to observe bats in their natural habitat. This is done by attaching sensor nodes to the animals' bodies, which send collected data to several ground nodes that are distributed within the observation region. The constraints regarding weight and energy usage of these sensor nodes are very strict, since bats are rather small animals and the nodes must not influence their natural behavior. The total weight of the sensor including the battery must not exceed 2 g. Since the bats move very quickly, the communication channel is highly unreliable. For the downlink from the sensor nodes to the ground stations, we successfully used Erasure Codes (ECs) as a forward error correction mechanism [13].

In the other direction, we investigated the usage of rateless ECs, namely fountain codes, to send firmware updates and reconfigurations to the mobile nodes. We developed a communication protocol that uses a SWuRx and fountain codes [14] in order to offer a reliable uplink to the sensor nodes while being as energy efficient as possible. When using fountain codes, a message is divided into several source packets. In each encoding step, a subset of these source packets is then encoded using a generator function and send to the receivers. Since this is a rateless EC, arbitrary many packets can be generated. This approach eliminates the need for retransmissions when a certain packet cannot be received by one of the mobile nodes. A node only has to receive enough other packets in order to decode the original message. The drawback of fountain codes is that the recipient may not be able to decode the message as soon as the minimum number of packets is received (this depends on the combination of source packets that were used by the generator function to generate each packet). To send an update, a ground station just has to send fountain code encoded packets until all nodes acknowledged the successful reception of the update.

Because of packet erasure (we assume a binary erasure channel here, i.e., a packet is either entirely lost or successfully transmitted), some nodes may finish decoding the message sooner than others. If a node receives a packet that belongs to an update it has already decoded, we consider this a false wake-up. The goal of the update protocol is to use the addressing and masking capabilities of the SWuRx in order to minimize such false wake-ups.

The ground station protocol starts sending packets to all nodes by first waking them up with a broadcast wake-up signal and then sending the packet over the main transceiver. If nodes start to successfully decode the update message and signal this by sending an acknowledgment, the algorithm changes the wake-up signals to multi- or even unicast signals. This prevents nodes that already received the update from waking up unnecessarily, which can save a lot of energy.

In order to minimize the number of false wake-ups, we developed an algorithm that tries to narrow down the set of possible recipients every time an acknowledgment arrives by creating new multicast groups that address a smaller number of nodes than before. The quality of this grouping depends on the length of the wake-up signal. If it is too small, only imperfect groups can be created, which leads to false wake-ups. If the length of the wake-up signal is sufficient, the sender can address every node individually and therefore reduce the number of false wake-ups to zero. This only works if the sender *knows* that the specific node successfully decoded the message, i.e., false wake-ups are still possible if acknowledgments are lost due to packet erasure.

We evaluated the performance of this protocol and the SWuRx using OMNeT++ [15] and MiXiM [16]. In the experiment, a single ground station sends an update message, that requires at least ten packets to be decoded, to 128 nodes. We measured the amount of false wake-ups when using different wake-up signal lengths and under different packet loss probabilities. In our simulations, the SWuRx uses a binary encoded mask instead of the direct representation as proposed in Section III-C. We used the following wake-up pattern:

- 1 bit (1 bit for the mask, 0 bit for the address): Only broadcasts are possible; the results for this configuration are used as a reference as to how energy efficient the SWuRx really is.
- 2 bit (1 bit for the mask, 1 bit for the address): This allows the protocol to do broadcasts and multicasts to two groups with 64 nodes.
- 5 bit (2 bit for the mask, 3 bit for the address): The protocol allows multicasts to groups of 16.
- 10 bit (3 bit for the mask, 7 bit for the address): This is the optimal configuration for $2^7 = 128$ nodes, as it allows the sender to address every node individually, if necessary.

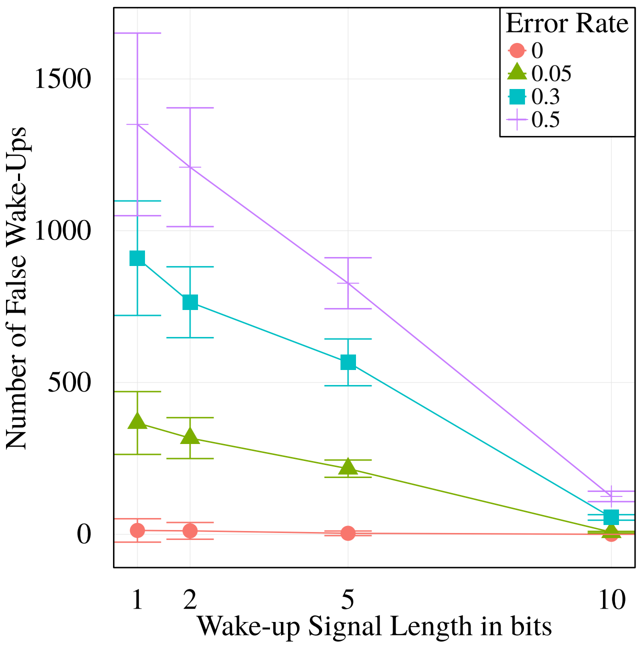


Figure 5. Number of false wake-ups during the update process.

Other OOK lengths (such as 8 bit) are not considered, as the protocol could not make full use of the mask because there are not enough address bits in the wake-up message. We used the mobility model from [17] for the bats in the simulation, which means that it is possible that a node temporarily moves out of communication range of the ground node.

Figure 5 shows the mean and the standard deviation of the number of false wake-ups depending on the packet error rate and the wake-up signal length. The results indicate that the number of false wake-ups decreases the more bits are used for the wake-up signal, with the best value being reached for 10 bit. If the communication channel is not perfect, the number of false wake-ups using the broadcast scheme (signal length of 1 bit) is very high. But when using a sufficiently large wake-up signal this effect is drastically reduced. We also observe that the positive impact of a longer wake-up signal is larger for a worse communication channel. The relative difference in false wake-ups between the 1 bit and 10 bit configurations increases with the packet error rate.

Our simulations show that the use of a SWuRx can greatly reduce the number of false wake-ups in a system, where information should be send to individual nodes or groups of nodes. This can save energy on the nodes that are not woken up unnecessarily. This system could also be used to reconfigure only a subset of the mobile nodes.

In the BATS project, the mobile nodes send their data to the ground nodes when they are woken up. The transmitted RF signal is not only used for sending the data, it also allows to track the position of the bat. Using a SWuRx for waking up the nodes makes it possible to track different individuals with different time resolutions. If the biologists notice something interesting during a running experiment, they can increase the rate of wake-up signals for single individuals without increasing the energy consumption for the other nodes.

VI. Analysis of Energy Savings in Homogeneous Networks

The main incentive for using a wake-up receiver in sensor networks is to save energy by avoiding idle listening and overhearing while keeping the communication latency low. So far, we only focused on the reception of a wake-up signal which can be done with very little additional energy. Yet, the power required for the transmission of a signal also plays an important role for the applicability and usefulness of such a system. In the presented application example, the wake-up signal was send out by the ground stations that serve as data collectors and do not have such tight energy budgets as the mobile nodes. This is a typical scenario in WSNs and it has been shown that the energy savings using a WuRx can be very high in comparison to a duty-cycle based system [18].

On the other hand, when wake-up signals should be send by sensor nodes as well, the energy consumption for sending this signal has to be taken into consideration. The achievable data rate of a wake-up receiver is often much lower than that of the main transceiver [9]. That means that the transmitter has to be powered on for a rather long time to send the wake-up signal. In order to save energy within a network using the SWuRx, the total energy saved by *not* waking up nodes unnecessarily must be greater than the energy used for sending the signal.

For the analysis, we consider the following scenario. In a WSN one initiating node sends a wake-up signal in order to wake up nodes it wants to communicate with. When using a WuRx without addressing capabilities, all nodes receiving this signal wake up, enable their main transceiver and receive some data from the initiating node. If the data was not destined for a node, this means that it was woken up unnecessarily. This node would then discard the received message and return to sleep mode. When using our SWuRx, such false wake-ups can be prevented, which saves energy. The number of false wake-ups highly depends on the considered scenario and the density of the network. Also the power consumption required for transmitting and receiving determines the usefulness of our proposed solution.

The energy saved per wake-up event E depends on the number of nodes n that are not falsely woken up (prevented false wake-ups) by using the selective wake-up receiver and the energy used for sending the wake-up signal:

$$E = \underbrace{(P_{RX} \cdot T_{RX} + P_{\mu C} \cdot T_{ON}) \cdot n}_{\text{energy saved by not waking up}} - \underbrace{P_{TX} \cdot T_{\text{WuRx}}}_{\text{wake-up signal}} , \quad (1)$$

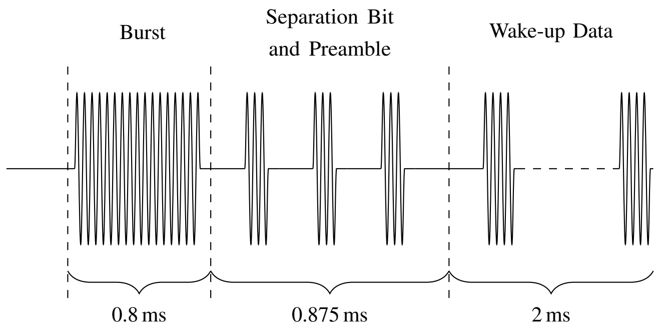


Figure 6. Structure of a wake-up signal for the AS3933 WuRx. The signal consists of a preceding burst, followed by a preamble and the actual data.

where P_{RX} and P_{TX} are the power consumptions for receiving and transmitting over the main transceiver, T_{ON} is the time a falsely woken up node is powered on and uses a power of $P_{\mu C}$. T_{RX} determines the time that a node requires for receiving a message after wake-up. The energy required to send a wake-up signal is given by the transmit power consumption P_{TX} and the duration of the wake-up signal T_{WuRx}

As an example we use the values from our mobile nodes in the BATS project which can be found in [19]. The AS3933 that we used for our hardware prototype can receive at most 4 kbit (Manchester symbols) per second which corresponds to a symbol duration of 250 µs. The wake-up signal has to follow the protocol determined in Figure 6. At the start of the signal a carrier burst of 0.8 ms is send followed by a separation bit and a preamble of 6 bit. Each bit of the preamble has a duration of half a Manchester symbol which gives a total of 0.875 ms. For our prototype we then have to send 8 bit of data which takes additional 2 ms. In total the transmission of a wake-up signal takes $T_{\text{WuRx}} = 3.675\,\text{ms}$.

Following Equation (1), we can calculate the minimum number of false wake-ups that have to be prevented with our system in order to actually save energy (the break even point):

$$n = \left\lceil \frac{P_{TX} \cdot T_{\text{WuRx}}}{P_{RX} \cdot T_{RX} + P_{\mu C} \cdot T_{ON}} \right\rceil \quad (2)$$

We already fixed the time for sending a wake-up signal to $T_{\text{WuRx}} = 3.675\,\text{ms}$. Based on our previous work [19] using the CC430, we assume the power required by the microcontroller in active mode to be $P_{\mu C} = 10.5\,\text{mW}$. The power consumption for sending is $P_{TX} = 99\,\text{mW}$ and for receiving $P_{RX} = 45\,\text{mW}$. The reception duration T_{RX} depends on the number of received bytes and the datarate of the main transceiver. For this example we use a data rate of 200 kbit/s. We assume that a falsely woken up node will be turned on for $T_{ON} = 2\,\text{ms}$.

Table I
Minimal number of prevented wake-ups required in order to reach the break even point.

Received Bytes per wake-up	2	8	16	32
Number of prevented false wake-ups required	16	10	7	5

Given these values, we can calculate the minimum number of prevented false wake-ups in order to reach the break even point depending on the amount of received data per wake-up, which is shown in Table I. The analysis shows, that the additional energy used for sending a selective wake-up signal only pays off, if sufficiently many nodes are not falsely woken up. If a node only receives few data, the energy losses for a false wake-up are not that high compared to the energy used for sending the wake-up signal. In networks with low density or if only little data is exchanged, it could be better to use a shorter wake-up signal without addressing, that requires less energy. Another option would be to use the proposed broadcast shortcut from Section III-C if this would cause only few nodes to wake up unnecessarily. This would reduce the energy for sending the wake-up signal by 50 %.

VII. Discussion and Conclusion

We presented a new approach for using wake-up receivers in a more flexible way. Instead of waking up all nodes within reception range, our architecture also allows to selectively wake up single nodes or groups of nodes. The decision which wake-up scheme should be used is done by the sender of the signal and does not require any reconfiguration of the mobile nodes. This great increase in flexibility makes it possible to use the wake-up technique in a more fine granular way which can save even more energy. To implement this functionality only little additional logic is required on the wake-up receiver. We have built a hardware prototype to show the feasibility of our approach and conducted first experiments and measurements.

As a first application example we used the new scheme to distribute a software update to sensor nodes in an energy efficient way. With the possibility to select nodes very fine granular only those, that have not received the complete update yet have to be woken up. This saves energy on all other nodes and can therefore prolong the network's lifetime. In heterogeneous scenarios, where the wake-up signal is send out by nodes with sufficient energy available (like data collectors, base stations), the energy savings can be very high. If the wake-up signal is also send by energy constrained mobile nodes the energy savings depend on the scenario. We plan to do further measurements and experiments and investigate other application areas.

VIII. Acknowledgments

This work has been supported by the German Research Foundation (DFG) under grant no. FOR 1508.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[2] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, pp. 1–39, Feb. 2009.

[3] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The Evolution of MAC Protocols in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 101–120, Feb. 2013.

[4] I. Demirkol, C. Ersoy, and E. Onur, "Wake-up receivers for wireless sensor networks: benefits and challenges," *IEEE Wireless Communications*, vol. 16, no. 4, pp. 88–96, Aug. 2009.

[5] S. Marinkovic and E. Popovici, "Nano-Power Wireless Wake-Up Receiver With Serial Peripheral Interface," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 8, pp. 1641–1647, Sep. 2011.

[6] R. De Francisco and Y. Zhang, "An Interference Robust Multi-Carrier Wake-up Radio," in *IEEE Wireless Communications and Networking Conference (WCNC 2011)*, Cancun, Mexico: IEEE, Mar. 2011, pp. 1265–1270.

[7] F. Dressler, S. Ripperger, M. Hierold, T. Nowak, C. Eibel, B. Cassens, F. Mayer, K. Meyer-Wegener, and A. Koelpin, "From Radio Telemetry to Ultra-Low Power Sensor Networks - Tracking Bats in the Wild," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 129–135, Jan. 2016.

[8] L. Gu and J. A. Stankovic, "Radio-triggered wake-up for wireless sensor networks," *Real-time Systems*, vol. 29, pp. 157–182, 2005.

[9] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. Gamm, and L. Reindl, "Performance Evaluation and Comparative Analysis of SubCarrier Modulation Wake-up Radio Systems for Energy-Efficient Wireless Sensor Networks," *MDPI Sensors*, vol. 14, no. 1, pp. 22–51, Dec. 2013.

[10] J. Oller, I. Demirkol, J. Paradells, J. Casademont, and W. Heinzelman, "Time-Knocking: A novel addressing mechanism for wake-up receivers," in *8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2012)*, Barcelona, Spain: IEEE, Oct. 2012, pp. 268–275.

[11] S. Ishida, T. Takiguchi, S. Saruwatari, M. Minami, and H. Morikawa, "Evaluation of a wake-up wireless module with Bloom-filter-based ID matching," in *8th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT 2010)*, Kuching, Malaysia: IEEE, Jun. 2010, pp. 1–6.

[12] H. Milosiu, F. Oehler, M. Eppel, D. Fruhsorger, S. Lensing, G. Popken, and T. Thones, "A 3-μW 868-MHz wake-up receiver with -83 dBm sensitivity and scalable data rate," in *ESSCIRC 2013*, Sep. 2013, pp. 387–390.

[13] F. Dressler, M. Mutschlechner, B. Li, R. Kapitza, S. Ripperger, C. Eibel, B. Herzog, T. Hönig, and W. Schröder-Preikschat, "Monitoring Bats in the Wild: On Using Erasure Codes for Energy-Efficient Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 1, Feb. 2016.

[14] D. MacKay, "Fountain codes," English, *IEE Proceedings - Communications*, vol. 152, no. 6, 1062–1068(6), Dec. 2005.

[15] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *European Simulation Multiconference (ESM 2001)*, Prague, Czech Republic, Jun. 2001.

[16] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. K. Haneveld, T. Parker, O. Visser, H. S. Lichte, and S. Valentin, "Simulating Wireless and Mobile Networks in OMNeT++ – The MiXiM Vision," in *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008): 1st ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2008)*, Marseille, France: ACM, Mar. 2008.

[17] M. Mutschlechner, B. Li, R. Kapitza, and F. Dressler, "Using Erasure Codes to Overcome Reliability Issues in Energy-Constrained Sensor Networks," in *11th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2014)*, Obergurgl, Austria: IEEE, Apr. 2014, pp. 41–48.

[18] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm, and L. Reindl, "Has Time Come to Switch From Duty-Cycled MAC Protocols to Wake-Up Radio for Wireless Sensor Networks?" *IEEE/ACM Transactions on Networking*, 2015, to appear.

[19] F. Dressler, B. Bloessl, M. Hierold, C.-Y. Hsieh, T. Nowak, R. Weigel, and A. Koelpin, "Protocol Design for Ultra-Low Power Wake-Up Systems for Tracking Bats in the Wild," in *IEEE International Conference on Communications (ICC 2015)*, London, UK: IEEE, Jun. 2015, pp. 6345–6350.