SoRTES report

Pieter Schumacher Seppe Plomteux

December 2019

1 Introduction

This report discusses in detail how our system satisfies the requirements specified in the assignment for the SoRTES project of 2019. Section *Reliability and real-time constraints* discusses how the system adheres to real-time constraints with absolute consistency. Section *Synchronization and database consistency* describes how the system ensures database consistency between and during gateway transmissions. Finally, section *Minimizing power consumption* details the techniques used to minimize power consumption between consecutive transmissions and after receiving twenty beacons.

2 Reliability and real-time constraints

In the timeframe between receiving and sending a gateway transmission, a number of tasks must be executed. FreeRTOS tasks are scheduled using *priority based scheduling*. Therefore, a task with the highest priority is most suitable for meeting a deadline. A task with the highest priority uses FreeRTOS library calls to wait for 100 milliseconds less then the supplied delay. After this delay, the task activates the LoRa antenna and puts it in receiving mode. By waking up preemptively by 100 milliseconds, the offset between the startup time of the LoRa module and receiving a packet is mitigated. This premature wake-up also ensures no feasible clock drift between the gateway and temperature sensor can cause a missed deadline.
Once the LoRa antenna receives a packet, an interrupt occurs. The ISR connected to this interrupt then parses the packet and sends the received delay to 2 separate queues. One queue provides the highest priority process with a delay so that this process can block until the next beacon is expected. The other queue provides a lower priority process with the same delay, which will be written to EEPROM. After the lower priority process receives the delay, it enables the ADC and starts a conversion. An interrupt is triggered when a conversion is complete. The ISR connected to this interrupt sends the data from the ADC to yet another queue. The task waiting for this queue simply takes the temperature reading, sends it via the LoRa module and stores it in EEPROM. It is in this task that the delay is also stored in EEPROM. If 20 beacons have passed,

this task gives a semaphore that will allow another task to enter the deep sleep mode.
An important note to make is that it isn't necessary to account for the time spent on lower priority tasks, because the delay is started right after parsing the packet. In other words, the delay is already running during execution of lower priority tasks.

3 Synchronization and database consistency

When writing new data to EEPROM, race conditions may occur. For example, two concurrent writes to the same address will result in only one correctly being stored, thus leading to data corruption. This is known as the *lost update problem*. Additionally, if a request to print the latest or all existing records occurs, a write concurrent to this operation will lead to incorrect information being provided (also known as the *incorrect summary problem*).
To ensure such problems don't arise, a semaphore provided by FreeRTOS designated for mutating the database is created on setup. The sempahore is taken on each higher order operation which requests a read or write to the database. This essentially locks the use of the requested behaviour until the semaphore is given back. Since we use a LIFO data structure, the semaphore is taken before a write request to EEPROM and released once the request is fulfilled and the stack pointer is incremented. Similarly, if a command to print all data existing in the database occurs, the semaphore is taken before the first read and released after the last.

4 Minimizing power consumption

To minimize the overall power consumption of the provided chip, a lot of functions are disabled in the setup code. All unused timers, the analog comparator, two wire interface (TWI) and USART are shut off. Initially, also the ADC is turned off. SPI cannot be turned off, since it is used by the LoRa library. The ADC is turned back on when a temperature measurement is needed and is turned off again in the interrupt service routine triggered when a conversion is complete.
To minimize power consumption between beacons, the micro is put in ADC noise reduction mode. This is the lowest power state that still allows timer interrupts to wake up the chip. Anytime the micro is in an idle state, it enters this sleep mode. This is achieved by the `vApplicationIdleHook` provided by FreeRTOS.
After receiving 20 beacons, the chip disables the ADC and SPI and goes into power down mode. In this mode, among some others, an SPI adress match can wake up the microchip. Since SPI is disabled, this functionality cannot draw unwanted power. Pin 2 is configured to trigger an interrupt when it is LOW, this allows the chip to wake up when needed.