Implementing NB-IoT: Communication with a Load Cell

Johannes Almroth

14th November 2019

Abstract

IoT technology has been proclaimed as a new technological prowess that will change our economy as well as our cities and way of living. Despite these bold statements, IoT is far from being easily implemented by companies not directly working with any of the enabling technologies, such as telecom. Narrowband-IoT (NB-IoT), a new radio protocol focusing on wide area coverage and low power consumption, is being heralded by the 3GPP as one of the key technologies necessary to push society into the age of IoT. NB-IoT networks are still extremely new in a lot of countries, and while the SIM-cards necessary to use these networks can be readily purchased from telecom companies, the lack of implemented projects might scare the everyday layman looking to implementing IoT within his/her business. The purpose of this paper is to provide an example for how an IoT device can be implemented in practicality, specifically with a scale. A micro-controller is hooked up to a load cell, from which the data produced is sent to the net via a cloud platform.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet of Things (IoT) has been lauded as a world-changing technology that will significantly affect our economy as well as our way of living. In a report by the GSMA, the total number of IoT devices is estimated to triple by 2025, bringing the total number of devices to $25.2 *billion*. Meanwhile, the global IoT revenue will fourfold from 2018, increasing it to $4.4 *billion*.[3] While there undeniably is a lot of excitement and potential economic impact associated with IoT, many consumers just associate the term with connecting a common toaster or coffee machine to a Wi-Fi network. While this technically fits the definition for an IoT device[4], the significant use cases will probably be implemented with different sensors, such as scales, thermometers, etc. that will further improve automatization & optimization processes. As an example, the key categories within the predicted growth is smart homes (e.g. security devices) and smart buildings (e.g. energy consumption sensors).[3] For the predicted growth to happen, businesses need to take a chance and work on projects that implements different IoT technologies, and to enable this, the GSMA has developed some LPWAN (Low-Power Wide-Area Network) protocols that focus on different key aspects that make IoT possible. Some of these aspect include long battery life, high connection density, indoor coverage and geo-tracking capabilities. However, it would be an educated guess to assume that the biggest hindrance to

This paper will focus on the implementation of an IoT device, and highlight the practical and structural challenges that need to be taken into account when starting this kind of project. The aim of this paper is to familiarize the reader with the basic workings and needs of such a device, and leave them with a rough idea of what they need to do to implement their own device with an appropriate sensor.

The IoT device that will be implemented is a basic microcontroller connected to a load cell. The practical work can be divided into two parts: enabling communication between the load cell and microcontroller, as well as connecting the microcontroller to a cloud service.

The signal output of a load cell is in mV/V[9]. Simply put, what this means is that when the loading cell is at full weight capacity, it will output x amount of mV per V the device is supplied. *e.g.,*a load cell with the full scale output of 3.0 mV/V, and the recommended excitation of 10V, will output 3.0 * 10 = 30 mV at full weight capacity. The remaining range of 0-30mV is used to signify the remaining range of the weight capacity from 0 to x kgs. To correctly tie these signals to a weight unit, a calibration is needed. To send this electric analog signal to a microcontroller, we will need some form of ADC (Analog-to-Digital Converter). Considering the system as a whole, we need to power the load cell, ADC and microcontroller with the same power source and ground, as to not cause any electrical interference throughout the different parts. Practically, this means that the choice of hardware will be affected by what voltage the power source is able to provide. Ideally, the microcontroller, ADC and load cell would all require the same amount of voltage as to not overcomplicate the model and make of the power source.

To enable wireless communication with some form of web service, the microcontroller will need to be fitted with a adequate wireless protocol. It would seem suitable that we would opt for one of the previously mentioned LPWAN technologies that the 3GPP (a telecom standard development organization[1]) developed for the specific purpose of being used with IoT devices. The two major flagships are NB-IoT (Narrowband IoT) and LTE-MTC (Long Term Evolution-Machine Type Communication [also known as CAT-M1]). The major differences between them can roughly be summed up as LTE-M having a higher data rate and NB-IoT enabling longer battery life. Additional differences will be explained in more detail in the next chapter. Both of these technologies enable wireless communications to the net via cell towers. From a hardware perspective, the only thing needed on the user end is a SIM-card from the telecom company that has enabled the corresponding technology in their network, as well as a compatible microcontroller.

The practical reasons for enabling wireless communications with load cell sensor in particular are many. Any form of application where weight needs to be measured overtime can benefit from such a device, whether it's about simply measuring trends over time or sending an alarm at a specific threshold value. With the addition of LPWAN technology, use cases in remote areas such as forests and deep underground locations are possible. Examples might range from measuring which sand depots that need refills when salting roads during winter, measuring the amount of lime depot that's being fed into a watercourse to affect the pH value, or reporting activity at a fodder station in hunting season.

1.1 Purpose and Goals

The goal of this paper is to familiarize the reader with an IoT device. Practically, this will be done by establish a working internet connection with a load cell through a microcontroller and documenting the process. The data sent from the load cell should be functionally identical to the data produced if the load cell was offline. Disregarding problems due to a internet service provider, data speeds and losses should not be abnormal. Using the same components, replication of the project should be feasible with the documentation provided in this thesis, assuming similar software and service providers remain. Additionally, some software design choices will be discussed in regards to aspects that benefit an IoT device, specifically issues such as battery life.

1.2 Delimitations

The end goal of the paper is to document and outline the steps needed to implement a functioning data upload from a load cell to the internet via NB-IoT. The final implementation will not be a functional product ready to be taken into commercial use. Any extra improvements upon a NB-IoT enabled load cell will only be done if time remains after the implementation and the completion of the thesis. The reason for this is due to a limited time budget, since this project is done within the framework for a bachelor’s thesis.

Chapter 2

Background

This section aims to give the reader a bit more background into the circumstances of the technologies used in the paper. We will discuss the background of the NB-IoT protocol and how it affected the groundwork of this paper, as well as expand on some of the concepts mentioned in the previous chapter. This is relevant information for people seeking to replicate the paper in some part, whether it's about acquiring and working with similar hardware or making basic software design choices about energy efficacy.

TODO: Is the Vetek paragraph even worth mentioning?Vetek is a Swedish scale supplier located in Väddö, situated approx. 100 kilometers north of Stockholm. Vetek constructs their own scales and weighing systems, as well as reselling products from other manufacturers.[11] Vetek aims to improve their services, and as such are interested in the possible use cases of IoT (Internet of Things) technology, and ultimately see how that can be applied to their own products. In a pilot project, Vetek wants to see how this connectivity can be implemented in a energy-efficient and effective manner. This would entail investigating factors such as power consumption, range and data rate.

As mentioned in the previous chapter, there are pretty high hopes regarding the expansion and profitability of the IoT industry as a whole. To enable this growth and functionality, the 3GPP developed various new radio technologies, the two most prominent being NB-IoT and LTE-M. LTE-M has the most functionality, including voice capabilities and device positioning. Thanks to its wider bandwidth frequency it also has lower latency and boasts a data rate up to 1 Mbps.[10] In return, device complexity and costs are higher compared to NB-IoT. The focus of NB-IoT was to enable indoor coverage, low cost development, long battery life and high connection density, which makes the technology ideally suited for low data rate applications in extremely challenging radio conditions.

Because of the novelty of IoT and the NB-IoT, the technology isn't easily accessible, and the required SIM-cards are only available to purchase for a hefty sum. As of writing this paper, a trial-kit from Telia (in Sweden), containing 5 SIM-cards for a period of 6 months currently costs €450.[8] Compared to other communication protocols, they are not as simple to implement out of the box, but the potential benefits should be enticing enough for many innovators to start experimenting.

According to Swedish telecom company Telia, they were the first to introduce the NB-IoT technology in Sweden, as well as the Nordic countries overall.[8] They further claim that their network will be in range for over 99.9% of Sweden's population, as well as provide a speed of 200 kb/s in more than 95% of the country.[7]

The grand opening of the network was on the 24th of May, and pilot projects were conducted as early as a year before this, in multiple locations across the country. Telia currently offers a starter kit for any actor interested in the technology, with a trial period of 6 months that includes access to Telia's IoT portal and APIs as well as 5 SIM cards, each with a 30MB data cap per month. Telia doesn't seem have many competitors when it comes to the Swedish IoT market, though Tele2 have partnered up with Nokia to offer similar services, and according to a press release from 2018, they have rolled out both LTE-M and NB-IoT across their networks.[5] Telenor has launched a IoT network in Norway with NB-IoT functionality in 2018[6], and according to an exchange with their customer support, followed suite in Sweden in the beginning of October. **TODO: How do I reference a private email conversation?** The fact that Telia already has partnered up with a multitude of cities and companies give the indication that they have a head start in the market.

2.1 Requirements

Aside from security, one of the biggest challenges regarding IoT devices relate to limitations arising from energy infrastructure. As mentioned earlier, one of the core issues NB-IoT aims to achieve is to be a low-power technology, thus decreasing the maintenance needed for battery-powered devices. A claim often paraded with NB-IoT is that it enables a battery-time of up to 10-years[2], though it's worth mentioning that over such a period of time the underlying IoT technology (in the form of microcontrollers/sensors) will probably require more frequent maintenance than the batteries themselves. However, if an IoT device constantly transmitted data for days on end, its energy supply would run out rather quickly. Therefore, it's also important that energy consumption is something that's accounted for when writing the code for an IoT application. Questions worth considering are **how often** and **when** to send data, in order to ensure device effectiveness while still maintaining energy efficiency.

After questioning Vetek about possible use cases that a battery-powered IoT device would fit into, the most common examples boiled down into monitoring weights that would change in a linearly decreasing fashion.

TODO: Explain why the three following subsections were chosen

2.1.1 Data Reading Interval

In most IoT devices the relevant data is provided by some form of sensor, whether it be a scale, thermometer or something entirely different. The type of sensor being used has a huge impact on the IoT device, especially when considering that they have to be powered by the same energy source. However, it's safe to assume that in most cases (depending on the sensor), the data transmissions will be the part of the device that will consume the most amount of energy during the lifetime of the device. Nonetheless, it's also important to factor in how often the device polls the sensor for data. The simplest way of deciding when to poll data from a sensor is to let it do so at a fixed and constant rate, often enough to be relevant, and seldom enough as to not waste precious energy. The same can be said for the actual transmission of the data, though this will be discussed elsewhere in the paper **TODO: Where?**

However, if within the context of the application we can conclude that no data needs to be polled (for a while), then subsequently no data will need to be sent, and thus we save energy on both ends of the system. For some applications there might even be longer periods of downtime where it's not relevant to conduct monitoring on the given sensor, *e.g.,*during nighttime, closing hours, etc. Another interesting angle is modifying the reading rate depending on the data itself. A simple example of this would be to have a slower reading interval at stable values, and increase it when experiencing large enough changes. Given the conditions of an IoT device powered by batteries, it's not unreasonable to assume that readings might not always be accurate at times. Depending on the sensor, spikes and drops of false values might occur, and not taking these scenarios into account would be prudent. In the following chapter we will explore a possible implementation regarding the reading rate of sensor data depending on the output of the data values.

2.1.2 Sensor Failure

In this paper, we define sensor failure as a sensor giving too many unreliable or false data values to be considered functional. The goal of identifying such a state in an IoT device is to prevent unstable data from being interpreted as valid, which in turn can save the end user from unwanted consequences. Depending on the longevity and purpose of the device, the threshold of when to declare a sensor as failing may differ, especially as this state can be quite fluid. A functional sensor means different things for different devices and applications. A simple way might be to conclude that if x% of data is considered invalid during the last 24hrs, an alarm should be raised to the device administrator. Complications arise when failures need to be reported quickly, or estimated more thoroughly. It's also possible that the sensor can be temporarily unreliable due to external circumstances, and given enough time, these circumstances might **TODO: vanish/recede/pass**. On one extreme you can have a device that reports failures too frequently and bogs down whatever dashboard is handling it's status report. On the other, you can have a device taking too long to determine a sensor failure that otherwise useful data could have been monitored if an error had been raised in time. In the following chapter we will look at possible way to handle sensor failure in a somewhat fluid manner, with the goal of being responsive while still allowing the sensor some leeway.

2.1.3 Sensor Disconnect

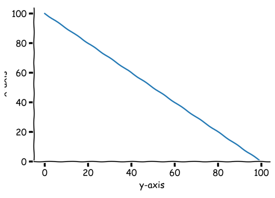
We define a sensor disconnect as when no credible data is being produced at all. If the sensor doesn't recover, immediate maintenance is needed for any continued functionality. If

Chapter 3

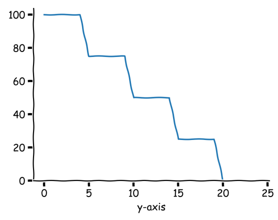
Design & Implementation

3.1 Data Reading Rate

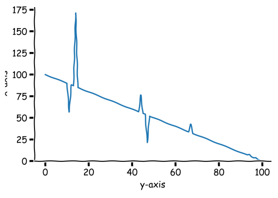
3.1.1 Behaviors



(a) Graph with linear decrease



(b) Graph with plateaus



(c) Graph with spikes

To determine what we want to achieve in this aspect, it's useful to first define some behaviors from which we will base our assumptions on. In figure 3.1a we see our base case, with a simple linear decline in data values. In figure 3.1b we see data the goes through some changes periodically, but stabilizes itself between the changes. In figure 3.1c we see a clear linear decrease with some spikes in data values here and there, which we can assume to be faulty measurements.

3.1.2 Implementation

We're interested in modifying the data reading rate depending on the changes in data values. This can be done in a number of ways, but a simple start might be to measure the delta in the last x amount of values and increase/decrease the rate depending on it's relation to delta.

To do this we need a *short term buffer* of a fixed size that contains x amount of recently read values in chronological order. This buffer should act as a FIFO-queue, so that the first element added to the buffer is the first one removed during an overflow. Measuring the delta of all the values in the buffer, we compare it to a threshold value that will either decrease, increase or maintain the current interval of reading data. When we say the delta of *all* values in the buffer, we mean the delta between each data point summed up. Assuming a buffer of 10 values, we would calculate the sum as follows:

$$\sum_{n=0}^{8} x_n - x_{n+1}$$

Starting with figure 3.1a, the values go from (100-0) in 100 data points on the x-axis. Calculating the delta for the first 10 values would be simple. The short term buffer would be: [100, 99, ..., 91]. The total delta would be (100-99) + (99 - 98) + ... + (92 - 91) = 10. For this example we can say that our determined threshold for decreasing our reading interval is ≥ 15, and an increase at ≤ 5. In our current example the reading rate is maintained.

In figure 3.1b the slopes are steeper, and would thus generate a larger total delta. Assuming a buffer length of 10, the first 5 values would be 100, and the other 5 would be 75. Our delta calculation would give us a value of 25, which would warrant a increase in our reading rate. In figure 3.1c the calculation would work the same as as in 3.1a, with the exception of some outlying data points. In regards to data reading rate, we can handle these values in two ways. Either we filter them in some way, or include them. Depending on the application, filtering can be easy. For example, perhaps we know that values $\geq$ 150 are impossible for our sensor, and thus we can manually check each values to see if they adhere to our specific bounds. Perhaps we know that such rapid changes in said values aren't possible, and we can filter them based on that. One way of doing this might be to calculate the average value of the short term buffer, and only allow new data within a range of *x* amount of units. Given the short term buffer [100, 99, ... 91], the average value would be 95.5. In our application we know that data points can only reasonably change with about 20 units, assuming a minimum reading rate. We set the limit to 30 units to give the program some leeway. In either way, extreme data points can be filtered.

A valid question at this point is *how often* we should perform this delta measurement, and of course the answer depends on the needs and workings of the application. Some applications might benefit from adjusting their reading rate very tightly, while others might only want to do this periodically.

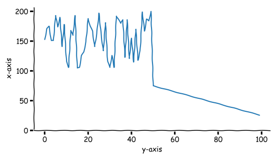
To sum up the important design choices when considering how often the sensor should be polled, we considered:

- What values we **compare** our short-term delta to. When should we decrease, increase or maintain our current data reading rate.
- What values are **valid** in the context of our application.
- How **often** we should adjust our reading rate interval.

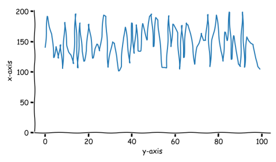
3.2 Sensor Failure

3.2.1 Behaviors

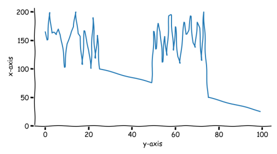
In the previous chapter we defined a sensor failure as a state where the sensors measures invalid data for such a period of time that the sensor isn't deemed functional. Functional can be different things for different applications, such as that one successful reading an hour is enough for one case, while one successful read per second might not even be enough for another. The question of how to determine successful readings versus unsuccessful ones are beyond the scope of this paper and varies greatly from sensor to sensor. We will aim to implement one way of handling the device state when these faulty readings do occur.



(a) Graph with some faulty data



(b) Graph with only faulty data



(c) Graph with two intervals of faulty data

In fig 3.2a, the sensor first reads faulty data but then recovers and starts producing valid data halfway through. In fig 3.2b all the data produced is faulty. In fig 3.2c there is one interval of faulty data followed by recovery, and then it repeats this pattern one more time.

3.2.2 Implementation

3.3 Sensor Disconnect

Chapter 4

Results

4.1 Discussion

Chapter 5

Conclusions

5.1 Future Work

Bibliography

[1] 3GPP. 3gpp, 2019. URL https://www.3gpp.org.

[2] GSMA. Narrowband – internet of things (nb-iot), 2019. URL https://www.gsma.com/iot/narrow-band-internet-of-things-nb-iot/.

[3] GSMA Intelligence. The mobile economy. GSMA's Mobile Economy report series https://www.gsma.com/r/mobileeconomy/, 2019.

[4] Steve Ranger. What is the iot? everything you need to know about the internet of things right now, 2018. URL https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/.

[5] Tele2. Tele2 iot och nokia ger företag större makt med ny modell för globala iot-implementeringar, 2018. URL https://www.tele2.com/sv/Media/pressmeddelanden/2018/tele2-iot-och-nokia-ger-foretag-storre-makt-med-ny-modell-for-globala-iot-implementeringar

[6] Telenor. Telenor launches nationwide iot network over 4g, 2018. URL https://www.telenor.com/media/announcement/telenor-launches-nationwide-iot-network-over-4g.

[7] Telia. Telia först i sverige med rikstäckande nät för sakernas internet - narrowband iot, 2018. URL http://press.telia.se/pressreleases/telia-foerst-i-sverige-med-rikstaeckande-naet-foer-sakernas-internet-narrowband-iot-2517174

[8] Telia. Nb-iot, 2018. URL https://www.teliacompany.com/en/about-the-company/internet-of-things/nb-iot/.

[9] Dara Trent. Understanding load cell specifications, 2019. URL https://www.800loadcel.com/white-papers/377.html.

[10] Shanquing Ullerstig, Ali Zaidi, and Christian Kuhlins. Know the difference between nb-iot vs. cat-m1 for your massive iot deployment, 2019. URL https://www.ericsson.com/en/blog/2019/2/difference-between-NB-IoT-CaT-M1.

[11] Vetek. Om vetek, 2019. URL https://www.vetek.se/om-vetek/content.