One tradeoff that I have found to be common in embedded software engineering from my computer science curriculum is the need to heavily conserve space. The benefit of having smaller microchips that are cheaper and can make devices portable is incredible. However, the drawbacks of having these can cause three major tradeoffs:

Features: When developing a program for a system with less capabilities, you tend to sacrifice possibilities of new features for the end user. The limitations set on the developer to create a more barebones and simple program can often lead to more basic machinery that doesn’t break boundaries. Finding the sweet spot between saving memory and getting a lot out of your hardware isn’t always easy.

Completion Time: When trying to size down certain programs it is easy to get caugt up in how to simplify your code. Choosing routes at the starting point of your program that will lead to simpler code may make it easier to keep it simple down the road. However, this may also lead to issues in getting the few complex and harder to contain pieces of code done with ease.

Cost: The cost of the hardware your using is also a huge factor in the amount of code you right. Sometimes like mentioned before, you want more features. The need for new hardware is sometimes too great and eating the cost can become a burden. Compromising on both ends is the only way to get what you want while keeping your costs low.