Software composition for embedded systems exhibits unique challenges. Embedded computers are surrounded by physical processes: they receive their inputs from sensors and send their outputs to actuators. Embedded computing devices, viewed from their sensor and actuator interfaces, act like physical processes with dynamics, noise, fault, size, power and other physical characteristics. The role of the embedded software is to “configure” the computing device so as to meet physical requirements. It is not surprising that using current software technology, logical/functional composability does not imply physical composability. In fact, physical properties are not composable, rather, they appear as cross-cutting constraints in the development process. The effects of cross-cutting constraints can be devastating for the design. Meeting specifications in one part of the system may destroy performance in others, and, additionally, many of the problems will surface at system integration time. Consequently, we need to go beyond conventional software technology to a model-based system design technology, which addresses the design of the whole system with its many interdependent physical, functional and logical aspects. This talk describes Model-Integrated Computing, which comprises modeling, model analysis and model-based software generation technologies as foundation for embedded software composition.