Computer Science (CS) education, for some time now, has been accused of being outdated and failing to capture the hearts and minds of the next generation of leaders and innovators. Rekindling the pioneer spirit and excitement of the hacking culture of the 1970s and 1980s requires flexible and accessible open hardware and software in an environment that promotes experimentation and collaboration. Although the software community has a long history of building and sharing such platforms, open and flexible hardware platforms have been few and far between. Oregon State University developed the Oregon State Wireless Active Learning Device (OSWALD) to help close this loop. The OSWALD is an open, Linux-based, ultra-mobile personal computer incorporating many of today’s new technologies and gives students and enthusiasts full access to both its hardware and software. Based on discussions with the OSWALD development team, this article provides a brief overview of the platform and explores the lessons learned in re-inventing computer science through open source.
Open Source and the State of Computer Science Education

One of the most frustrating challenges CS educators face today is convincing the next generation of students that room still exists for innovation and making a difference in computing. With the advancement of computing and its proliferation into most aspects of our lives, many prospective students feel computing is “a solved problem”, that the days of innovation and chances to make a significant contribution have passed. We are victims of our own success. The rapid spread of cheap hardware and polished applications had removed one of the most compelling reasons many have had for studying computer science: the need to scratch our own itch. As a consequence, CS enrollment has dropped for more than a decade.

It doesn't help that the CS curriculum has, in many ways, failed to keep up with changes. Here I refer to the core curriculum, not the electives or advanced courses, where we often see a much richer and up-to-date set of topics. Little has changed in the core curriculum during the last 15 years short of programming language choices. In all fairness, the body of CS has been growing tremendously, and the fundamentals are still fundamental, but academia has failed to make sure this core stays relevant and visible to current and future students.

Open source provides a unique opportunity to help us bridge this gap. In an article titled “Computer Science Education in the 21st Century”, David A. Patterson, former president of ACM and Pardee Professor of Computer Science at UC Berkeley, explained that open source provided unique opportunities for injecting realism and
relevance into the classroom. One of Patterson's laments is that students can easily go
through their undergraduate education without real exposure to computing. Students
rarely read other people's code, and sharing of code is often the same as cheating. The
largest codebase students might see is a few thousand lines of their code written over
a period of 6–12 months with, at most, a half-dozen other students. This code will not
evolve or be maintained. In all likelihood, it will be discarded as soon as the class ends.
Open source allows educators to change this dynamic, opening a window into the
chaotic and complex, yet realistic world of software development. A place where code
is designed and written by many, and the context, metadata, discussion and
collaboration required is available to all. It's a unique resource for educators to
present an invaluable opportunity to students to learn real skills. Oregon State
University has been busy developing curriculum centered on open source, and it is
aware of how to navigate its sometimes discouraging waters.

Dr Carlos Jensen, an assistant professor at Oregon State University and driving force
behind this project, explains:

Students do not want to re-invent the wheel, and can be
intimidated by the quantity and quality of easily available
software. Many of us learned how to program by writing
small programs designed to meet some need other software
couldn't. Some of us attained glory by writing something
our fellow students also found useful and would circulate.
Today, chances are, there is already an app to meet most
needs, and students often see joining large open-source
communities as their only choice to make a difference in
computing. It can be a daunting and intimidating prospect
for someone learning how to code, and therefore, open
source must be introduced in the right context.