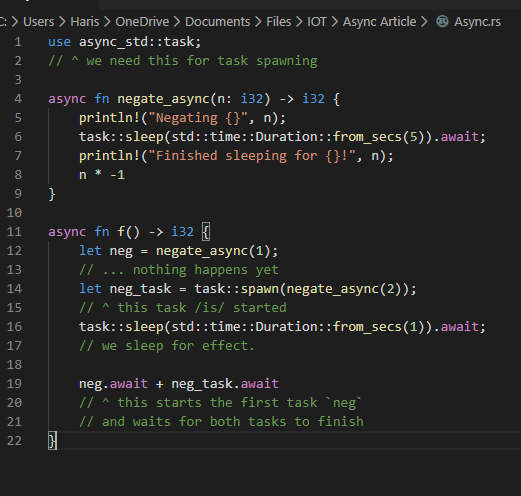
**Async Rust:**

One of the most valuable features in any programming language is the ability to run code simultaneously along with another piece of code and this method reduces the amount of time taken to perform any task. If we observe the hardware side of things, a CPU has multiple cores and threads to concurrently perform calculations. Rust has just been introduced to this new feature and it widens the functionality of rust even more. To obtain the outcome of an asynchronous calculation we use **.await.** These are called futures in Rust.

In rust async is different as it requires a function to start it. An executor is required to launch a task. Until then we have future which is not yet started.



We need an external library to use async hence we import the library first, after that the async function takes an integer and sleeps for 5 seconds, and returns the negated value. But the more interesting part is the f() async function which creates a future of the previous function yet doesn’t start. The next line task spawn executes the future returned by negate sync. We add delay so the difference can be seen. By awaiting neg in the next line we start executing the future.