The proposed artefact is a development tool that is intended to alleviate the burden of programming across Internet of Things (IoT) devices, smart contracts, and other systems that integrate with these two. Each of these platforms require a distinct set of skills. The purpose of this tool is to mitigate some of the burdens associated with programming through the use of macro programming. The tool, named “ Smart Tool," enables users to create source code in.NET by following a template provided by the tool. After processing the programmers' source code, the proposed tool will generate executable code for IoT devices, smart contracts, and the main platform that will communicate with the IoT devices and smart contracts, as illustrated in Figure 1. The ultimate goal is to reduce the amount of low-level programming required and to enable programmers to create simple systems in a fraction of the time using a single platform. By avoiding the need to learn new languages and connecting the dots between how various systems communicate, programmers will save time. Without requiring extensive training, an intuitive tool enables non-embedded systems and blockchain specialist programmers to develop systems.

All three platforms, including the IoT platform, the blockchain (on which the smart contract is deployed), and the main platform, will operate independently, necessitating communication between them. Communication occurs via the main platform, where all function calls written by the user in the main application are encapsulated within another function called a "run-time call," which triggers a function on the other platforms and waits for a response. The run-time call is a function provided by "Smart Tool" that accepts three parameters pertaining to the function call. These include the platform's type, the function's name, and some platform-specific settings.

At the moment, "Smart Tool" does not include options for selecting different platforms, such as Ethereum (Blockchain) and Arduino (IoT). However, the tool is designed in such a way that it can accommodate other implementations. Both the Stratis and Raspberry Pi generators inherit an interface that determines the types of functions they can define. Therefore, if a different platform is required, an experienced programmer familiar with that platform must define an interface provided by "Smart Tool" and implement the functions in the specified language. Apart from the generator, an implementation would be needed within the previously discussed run-time call function (the function that allows platforms to communicate with each other). This code, like the generator, would need to be language-specific, and it would depend on which protocols would be most appropriate, for example, HTTP or TCP. Currently, the run-time call contains two cases, Stratis and Raspberry Pi, which communicate via http requests.

It is critical to note that the results, as well as the generated lines of code, are notably similar to those obtained using the traditional approach. Apart from the required knowledge and expertise, the code generated will be nearly identical, with significantly fewer lines of code required to define the system.