ESPRUINO TOOLS - AN ASYNCHRONOUS JAVASCRIPT LIBRARY TO CONTROL REMOTE EMBEDDED DEVICES [JAVASCRIPT, PROMISE, ARDUINO]

Callum McLuskey - 2469682M

December 14, 2022

1 Status report

1.1 Proposal

1.1.1 Motivation

Remote embedded systems is a sector of engineering which provides developers with a hands on experience allowing for code to be developed into a tangible object providing developers with a rewarding end product. However the currently implementations of this such as Arduino have a high learning curve using C++ as the language to program their devices or micro python which while removing the high learning curve leaves developers with non-transferable skills due to its high abstraction from the core python language. By working with the Espruino catalog of devices we by pass these issues. Despite this development with Espruino devices still provides a closed in understanding of how JavaScript works leaving the developer with non transferable skills. This project helps prepare future programmers and software engineers for work within the embedded systems allowing them to achieve working systems as well as allowing for the development of in demand skills through the usage of modern JavaScript with easy access to work with any modern JavaScript of their choice.

1.1.2 Aims

This project will provide an ecosystem of packages to reduce the entry barrier to remote embedded systems programming whilst allowing experienced developers to still benefit from the ease of life additions brought forth with them. On top of this these packages are built with web technologies at the fore front allowing both experienced and new developers to improve their grasp on current web technologies while developing their embedded systems.

1.2 Progress

- A core package has been built to allow for communication between a web app and an espruino device, providing preset commands for general device functionality alongside an easy structure to allow for developers to create new custom devices.
- A revamped UART package has been created. This package takes inspiration from a previously implemented package which enables UART connection through the WebBluetooth API, this new package

improves upon this old one by providing an improved user experience and user interface as well as re-implements the package to a modern WebBluetooth standard through the introduction of modern JavaScript(ES6).

- An NPX tool has been created. This allows users to get started using the espruino-tools ecosystem by running a single command in their terminal, this in turn will create a project with a local web server to start development in seconds. On top of this the package provides tags to allow users to get started with their choice of front end framework such as react or vue, or just use typescript if they want. This package also includes an easy starting project for the peer package mentioned below.
- A peer package has been created. This package extends upon the peerjs package to improve the development of peer to peer connection (using WebRTC) between web browser and mobile device whilst creating web apps for espruino devices, enabling users to benefit from touch screen and portable devices to control their espruino devices.
- A demo site has been created to show off exactly what can be done with these packages, with an easy submission process for developers to show off their own projects.
- A documentation site has been created to accurately show off any syntax used within theses packages whilst keeping it all in one place to allow for a much faster development time, this incorporates tutorials as well as quick start guides for each package.
- Each package has been given its own test suite and pipeline to ensure no broken code reaches the production packages.

1.3 Problems and risks

1.3.1 Problems

- The UART package is using preset timeouts instead of JavaScript promises. This provides an extended wait on the developers inputs.
- The core package parser does not work with nested functions, its unclear why this is happening.
- The peer package is not robust and does not include any testing.

1.3.2 Risks

- The development of a transpiler is not well documented.
  - **Mitigation:** will research how current transpilers work such as React, Vue and Typescript.
- Unsure on how to evaluate the user tester data to provide a good reason for using this package.
  - **Mitigation:** will perform research on how to improve this process to gather meaningful data.

1.4 Plan

Dissertation will be written weeks 1 through 10 during the second semester.

- *Week 1 - 3*: Develop a transpiler.
  - **deliverable:** A package containing a working transpiler that converts the developed espruino tools language into native espruino code along with tests to ensure working functionality.
- *Week 4 - 6*: Build online IDE using the transpiler.
  - **deliverable:** A web app that allows for device connection, code transfer and direct code writing, accepting both espruino tools via the transpiler and native espruino code.
- *Week 7*: Finalise UART implementation.
  - **deliverable:** polished UART package with up to date JavaScript and WebBluetooth methods used, using promises to remove any need for timer delays.
- *Week 8 - 10*: Writing dissertation.
  - **deliverable:** creating a fully fleshed out paper with enough time for multiple drafts ensuring the supervisor has enough time to read over my second to last draft.