

# The Jetson Artificial Intelligence Tool Chain<sup>1</sup> (JAI-TC)

Behnam Saeedi  
(Saeedib@oregonstate.edu)  
June, 4<sup>th</sup> 2019



## Abstract

The Jetson Artificial Intelligence Tool chain (JAI-TC) is a set of packages, APIs and libraries for Artificial Intelligence applications to be deployed on the NVidia SOC, Jetson TX2. JAI-TC automates the installation of these items allowing for a wider set of users to leverage these technologies. Prior to this, the process of building and testing these tools has made them inaccessible to a wide range of users from researchers to hobbyists. There are various challenges with regards to setting up and creating AI applications on Jetson. The challenge this project addresses is with regards to commonly used packages, APIs and libraries in Artificial Intelligence, Computer Graphics and robotics for Jetson. These packages are often out of date, and difficult to build for a particular architecture. These packages include GPU accelerated OpenCV and the contributors repository for OpenCV, latest version of PyTorch, Caffe2, Latest GPU accelerated TensorFlow, and ROS. This project is aiming to reduce the complexity of building these tools by streamlining the build procedure for these tools in a single well documented and modular tool chain. This project could be a step in the direction of JAI-OS, an operating system build from ground up specifically for the purpose of performing GPU intensive AI computation.

## **ACKNOWLEDGEMENT**

I cannot express enough thanks to my major advisor Donald Lee Heer for his continuous support and encouragement. I offer my sincere appreciation for believing in me and the opportunities that he provided for me. Special thanks to my committee for being a part of this experience and giving me the opportunity to contribute back to this community in a meaningful way.

I would like to express my deep gratitude to my caring, loving and supportive wife, Jenna. It was a great comfort and relief to know that you were willing to go above and beyond to support me in my endeavors through some of the toughest days of our lives. Words cannot describe how grateful I am for having you in my life.

Last but not least, I am very grateful for my parents and their care, support, prayers and sacrifices to make this possible while thousands of miles away across oceans and seas. Likewise I thank my brother and sister for being so supportive.

## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Problem Statement</b>	<b>6</b>
2.1	Problem Definition . . . . .	6
<b>3</b>	<b>Requirements</b>	<b>7</b>
3.1	Solution Requirements . . . . .	7
3.1.1	Interface . . . . .	8
3.1.2	Packages . . . . .	8
3.1.3	Licensing . . . . .	8
3.1.4	Installation . . . . .	9
3.1.5	Documentation and artifacts . . . . .	9
3.1.6	Glossary . . . . .	9
3.1.7	Overview . . . . .	10
3.2	Overall Description . . . . .	11
3.2.1	Product Perspective . . . . .	11
3.2.2	Product Functions . . . . .	11
3.2.3	User Characteristics . . . . .	11
3.2.4	Constraints . . . . .	11
3.2.5	Assumptions and Dependencies . . . . .	12
3.2.6	Apportioning of the requirements . . . . .	13
3.3	Specific Requirements . . . . .	15
3.3.1	Functional Requirements . . . . .	16
3.3.2	Performance Requirements . . . . .	19
<b>4</b>	<b>Design</b>	<b>20</b>
4.0.1	Intended Audience . . . . .	20
4.0.2	Conformance . . . . .	20

4.1	System Architecture and Rationale . . . . .	20
4.1.1	Design Patterns . . . . .	20
4.1.2	Decomposition Description . . . . .	21
4.2	JAI-TC in Perspective . . . . .	21
4.2.1	Concerns . . . . .	21
4.3	Component Design . . . . .	22
4.3.1	Performance Switch . . . . .	22
4.3.2	Fan speed adjustment . . . . .	23
4.3.3	Internet Connection Check . . . . .	23
4.3.4	Display JAI-TC Licence . . . . .	24
4.3.5	Draw Menu . . . . .	25
4.3.6	Dynamic Menu Updater and Script Detection . . . . .	26
4.3.7	Quit with Reboot Option . . . . .	26
4.3.8	Package handler . . . . .	27
4.3.9	Package: Essentials . . . . .	28
4.3.10	Other Packages . . . . .	29
4.4	User Interface Design . . . . .	29
4.4.1	Overview of User Interface . . . . .	29
4.4.2	Screen Objects and Actions . . . . .	29
<b>5</b>	<b>User Manual</b>	<b>32</b>
5.1	Installation . . . . .	32
5.1.1	Downloading JAI-TC . . . . .	32
5.1.2	Setting Up JAI-TC . . . . .	32
5.2	JAI-TC Operations . . . . .	33
5.2.1	Package Selection . . . . .	33
5.2.2	License handling . . . . .	33
5.2.3	Help Menu . . . . .	33

5.2.4	Refreshing the Package List . . . . .	34
5.2.5	Exiting JAI-TC . . . . .	34
5.3	Customizing The Software . . . . .	34
5.3.1	Installation Scripts . . . . .	34
5.3.2	Offline License File . . . . .	34
5.3.3	Online License File . . . . .	35
5.4	Repairing Missing Licences . . . . .	35
<b>References</b>		36

## 1 INTRODUCTION

Late April 2014 NVidia released their high performance system on a chip (SoC). NVidia Jetson was initially intended for applications that require a high computational power on portable platforms. Later by introduction of TX1 SoC, Jetson SoCs made their way to mobile computers and entertainment systems such as NVidia Shield and Nintendo Switch.

Another potential use of Jetson SoCs are mobile AI applications. Mobile AI applications depend on other more powerful computer for processing data. It is important to consider some of the issues with regards to this approach. For example, what happens when the latency of the computation is important? What if the service provider goes out of business? Anki's Cozmo platform and the social robot Jibo are two good examples of this situation where users have spent hundreds of dollars for a system that is no longer going to be working due to manufacturer going out of business. Jetson on the other hand is independently capable of performing a wide range of process intensive AI computations. This could present a lot of potential in Robotics vision systems, path finding and autonomous vehicles that depend on real time decision making.

Unfortunately, developing for Jetson SoC presents a challenge. Although, many AI APIs are already available for Jetson platform, they are often out of date or missing key features. Therefore, developers need to build specific versions of those APIs in order to be able to effectively utilize Jetson SoCs. Another issue is the experimental nature of the Jetson developer hardware kits. Since the introduction of TK1 in 2014, four new iterations of this SoC has been introduced with major revisions. This issue substantially increases the complexity of development and deployment process on hardware that utilizes the Jetson SoC technology.

This project proposes a robust solution to combat these challenges. This document provides an overview of the Problem Statement, Software Requirements Specifications and Software Design Documentation. Finally this document provides a brief Users Manual to be used as a reference for JAI-TC users.

## 2 PROBLEM STATEMENT

Understanding the problem is a step in the direction of finding a solution. This section provides an in depth definition of the problem JAI-TC addresses, as well as a brief description of the proposed solution. Furthermore, it covers key concepts that need to be considered for proposing an effective solution. These concepts include interface, packages, licensing, installation, documentation and maintenance. Finally, this section, should serve as a reminder for the developers in order to stay true to the initial purpose of this project.

### 2.1 Problem Definition

Despite the amazing capabilities of the NVidia Jetson SoCs, there are a wide range of drawbacks that hinder the development process. The exact problems JAI-TC addresses are defined as series of challenges that developers experience during setup, build and deployment of their artificial intelligence, computer graphics or robotics applications this hardware.

The most important thing to consider is the purpose of this solution. This solution needs to make development process easier for the developers and not to add extra steps to an already convoluted procedure. There already are package managers available to developers that once they are setup, they are very effective in handling package installation. One of the key features for an effective solution for this problem is to have a simplified

setup procedure or even better, no setup procedure for the tool itself what so ever.

Application that are designed to solve complex problems cannot be any more efficient as the underlying libraries and packages that are used in their component development. Having access to the most up to date versions of those libraries and packages are very important to ensure access to the newest, most efficient and most secure components available to developers. This is where traditional package managers on NVidia Jetson platform fail. The latest versions of libraries and packages available to Jetson platform are generally out dated. In some cases such as the PyTorch package, it is as much as 1 major iteration older. An effective solution Needs to address this problem by providing means of installing the latest possible versions on those packages and libraries for the user.

Building process for many packages and libraries from scratch is generally convoluted and includes many hidden steps. Furthermore, these steps are not necessarily identical for different packages. Some packages could be installed as easily as running a setup command or as complicated as having to patch certain kernel components. The proposed solution needs to be capable of handling a wide range of different approaches to build and installation of packages. It is very important for this tool to be flexible and customizable.

Another issue to consider is the deployment of developed tool. While setting up these packages could be simple enough for the developers, it could pose many challenges for users of that tool. Users might not necessarily have the required expertise to figure out the installation of the required libraries and packages. The proposed solution needs to be user friendly enough to allow less skilled users to navigate through the installation procedure. The success of such solution is dependant on its interface and the user experience.

One of the important steps in any setup procedure is the acknowledgement of the end user license agreement or EULA. It is very important for any solution that is designed to streamline setup procedures to communicate the license of the tool to the user. An effective solution needs to ensure that the user is aware of license, and have access to them at any given time. An effective tool chain, needs to have safeguards in order to ensure that the user is aware of the licensing.

Finally, the process of building and deploying libraries and packages are generally not well documented. Developers of those tools usually assume a high knowledge of programming that is not necessarily in line with their audience's level of expertise. An effective solution for this problem needs to come bundled with an extensive documentation to instruct the usage of this tool for the newer, less skilled users. At the same time, it also need to explain its more advanced features for users who want to get more out of this tool.

### **3 REQUIREMENTS**

The purpose of this section is to provide a detailed description of the system requirements that were considered during the development of JAI-TC. This section covers the description of modules, components, sub-components and connectors of this solution. This section is essentially an overview of the tasks that needed to be done for the completion of JAI-TC project.

#### **3.1 Solution Requirements**

As discussed in problem definition, there are key features that need to be present in our solution. These key features are simplicity, efficiency, flexibility, ease of use, EULA compliance and documentation. This subsection

dives deeper in describing the requirements that need to be met for producing a solution that has those features. This subsection provides some guidelines for an ideal design for various components.

### *3.1.1 Interface*

There are two options for the UI design. First option is to develop a graphical user interface (GUI) for JAI-TC. GUIs are much easier to understand and navigate by users with less technical expertise. However, they are often time consuming, complicated and unreliable during design process. The second option is to develop a text based UI. Text base UIs could be a lot more flexible, lightweight, and reliable. They could be developed fast and used by more professional audience of our tool. This comes at the cost of ease of use. Comparably GUIs are much easier to use than their text-base alternatives.

For the GUI option there are more things to take into consideration. For example, there are a wide range of GUI development tools available to various languages and platforms. Each package has its own unique pros and cons and provide specific features for the GUI element of JAI-TC. The final point to consider is that a GUI significantly reduces maintainability, re-usability and customizability of the tool. Similarly, for the Text based UI option, we need to consider other factors. There are many different approaches to text based UI design such as Command line, indexed menu, and text based GUI. The pros and cons of each solution needs to be taken into consideration such as design complexity, reliability, tool availability, usability and maintainability. The final point to consider for this solution to be effective is to ensure that the UI element supports display of manuals, links, licenses and other important text and non text based information for the user.

### *3.1.2 Packages*

JAI-TC includes installation scripts for a large number of different packages. The solution needs to include a selection of these tools that have the largest user base. These packages include SKLearn, OpenCV, PyTorch, Tensorflow and caffe2 in AI, ROS, Kinect and Realsense in robotics and OpenGL, GLM and GLSL in computer graphics.

Furthermore, there are another group of packages that need to be included. These packages by themselves are not used by the developer for their AI, CG or Robotics applications. These packages are used to run the installation script and setup the development environment for the users. For the purpose of this documentation, these packages are categorized as Essentials. The Essentials include Git, python 2.7 and 3, screen, cmake, pip and many GCC and other development libraries.

### *3.1.3 Licensing*

There are several things that needs to be addressed with regards to licensing.

- **JAI-TC License:** JAI-TC itself requires a proper license since it is going to be widely used by developers who might chose to monetize their product. Furthermore, JAI-TC might have an active modding community with a large repository of costume installation and build scripts. An extensive research on this topic was necessary to identify the appropriate license for JAI-TC.
- **Package Licences:** Each package that JAI-TC installs, has its own licensing agreements and conditions. One of the important aspects of installation scripts is that they streamline the process of setup and can potentially obscure the licenses that those tools hold. These licences need to be provided and maintained.



- **Contributor Licenses:** The developers of the tools that JAI-TC installs are not the only contributors to this procedure. Many setup procedures also require various patches and modifications. Furthermore, some setup procedures are provided by various organizations. Those contributors are also entitled to their distribution and re-distribution rights.
- **License Display:** As mentioned earlier, one of the important elements to consider is to ensure that the licenses are not obscured, or ignored through the process of the installation. This could be achieved by providing means of displaying the licenses to the user. This has several benefits. One of these benefit is that it allows the user to have an easy access to the licenses in case they wanted to review them. Another advantage is that even if the user is not aware of the license, this could act as a reminder to them. Last but not least, it allows the developer to remove the responsibility of the usage of the tool from the writer f the installation script and onto the developer by preventing the script to accept the license on the developer's behalf.
- **Handling Missing Licences:** It is important to ensure that the licenses are not missing from the tool set when the users attempt to install the packages. This could be achieved by checking for the license before setup begins.
- **Getting the Most Recent Licenses:** Te solution needs to stay relevant as newer versions of the packages are introduce. Tool licenses could change due to various reasons. The latest licence is available either on the repository or on the official website for these packages. The solution needs to have means of displaying most recent licenses available to the developers.

#### 3.1.4 *Installation*

For this subsection concepts such as installation and distribution of the product were considers. There are a few methods of distributing this tool that are already available to the users. These methods include the NVidia developer portal, NVidia Jetson forums, and Jetson independent communities such as JetsonHacks. Another thing to consider was to whether have JAI-TC packaged and installed or made available as a repository clone. The conclusion was that the most convenient mode of distributing this tool is to make the repository link available to various Jetson Communities.

#### 3.1.5 *Documentation and artifacts*

With the consultation of the project advisor there were several decisions made with regards of various artifacts to be produced for JAI-TC. The artifacts need to explain the project on a detailed level and make it possible for others to replicate, modify or improve the tool. Fortunately, There already exists a method for producing such documentation. The artifacts include, Problem Statement, SRS, Design Document and User Manual. A copy of these documents would be made available in PDF form on the JAI-TC repository.

#### 3.1.6 *Glossary*

- **AI:** Artificial Intelligence, deals with the simulation of intelligent behavior through mathematical models. [1]
- **Bash:** Bourne-Again SHell is the shell, or scripting language, for the GNU operating system. [2]
- **Caffe:** Caffe is a deep learning framework and is developed by Berkeley AI Research (BAIR) and by community contributors. [3]

- **CG:** Computer Graphics is a branch of computer science that studies methods for digitally synthesizing and manipulating visual content.
- **Computer Vision:** CV is the field that studies methods of allowing computers analyze content of a digital image. [4]
- **GCC:** GNU Compiler Collection, is a compiler produced by the GNU project and distributed by FSF. [5]
- **GLM:** "OpenGL Mathematics (GLM) is a header only C++ mathematics library for graphics software based on the OpenGL Shading Language (GLSL) specifications." [6]
- **GLSL:** GLSL or Graphics Language's Shaders Language is a C++ like language description for describing a kernel program that is designed to be compiled in the graphics driver and run on the GPU. [7]
- **GUI:** Graphical User Interface is a component that allows users to interact with a software by manipulating visual graphical elements.
- **JAI-TC:** Jetson Artificial Intelligence Tool Chain is a solution proposed in order to simplify the process of package handling and installation on Jetson SoC.
- **Jetson:** NVidia Jetson is a powerful System on a chip computer that is targeted toward high performance portable computing purposes. [8]
- **Kinect:** Microsoft Kinect is a series of sensors designed to capture depth data within a specified range.[9]
- **License:** Software license is a legal agreement between the software developer and the user that dictates the terms and condition of sale and redistribution of that software.[10]
- **Man pages:** Linux Man pages is a tool that allows users to find out more about Linux Commands, Functions or files. [11]
- **Realsense™:** Are sensors that allow computers understand the environment better through providing the depth map of the scene.[12]
- **ROS:** Robot Operating System is a framework for designing and developing software for robotic platforms. [13]
- **script:** For the purpose of this document scripts are series of bash commands designed to perform a specific installation task.
- **SoC:** System on a Chip are series of integrated circuits that include various parts of a full computer on a single chip.
- **UI:** User interface is a tool that allows the user to interact with a software.

### 3.1.7 Overview

The following subsections provide an overview of the system functionality and system interaction with other libraries and user custom package installation scripts. Subsection two introduces different types of users and their interaction with the system. Furthermore, it also mentions the system constraints and assumptions. The third Subsection provides the requirement specifications in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences. Finally, subsection 4 provides the performance metrics used to assess the overall

effectiveness of this solution.

## 3.2 Overall Description

### 3.2.1 Product Perspective

This software is a management tool for series of installation scripts designed to increase the utility of the NVidia Jetson SoCs. JAI-TC provides means of displaying and managing licenses for its package installation scripts. Furthermore, JAI-TC allows users to include custom installation scripts and make them available for the Jetson user community.

### 3.2.2 Product Functions

The Jetson Artificial Intelligence - Tool chain provides a setup solution for a wide range of AI, Robotics and CG packages by utilizing the installation scripts that are presented in the tool chain. Furthermore, users can easily get access to the latest version of package End user license agreements. JAI-TC users can make, or download custom package installers as long as they follow JAI-TC's license display guidelines. This tool allows users to save a significant amount of time during setup and dedicate more time to actual development of their software.

### 3.2.3 User Characteristics

JAI-TC is targeted towards a wide range of users. This user base can be categorized into two different user groups:

- **Professionals:** This group includes researchers and industry developers in the fields of artificial intelligence, computer graphics and robotics. The result of their work is going to be included in research papers, studies or consumer grade electronics and services. JAI-TC targets this group because of their limited time and high cost of man power. Generally this group of audience are more comfortable with figuring different installation approaches and are more likely to take advantage of the more advanced features of JAI-TC such as custom installation scripts.
- **Hobbyists:** This group includes individuals or groups of users that are developing technologies for their personal use. These users usually use our tools for purposes such as entertainment, smart home IoT and developing home-brew applications for consumer grade hardware. The reasons for this group to take advantage of JAI-TC is unlike the professional category. Hobbyists have a lot of time to figure things out, but they might lack the expertise needed to utilize the Jetson hardware. These users can use JAI-TC in its most basic form, or over time, learn to take advantage of its more advanced features.

Each one of these two user bases will benefit from JAI-TC on different levels and utilize its capabilities on different scales. So it is important to ensure that JAI-TC does not limit the capabilities and use cases it provides to one audience group for the interest the other.

### 3.2.4 Constraints

Due to the nature of development kits, JAI-TC will suffer from certain constraints. There were certain steps taken in order to minimize some of the constraints as it will be covered in the Design section. This subsection will briefly mention some of these constraints and steps taken to prevent them.

JAI-TC is dependant on the operating system that it is running on. This tool is specifically designed to run on Jetpack's Ubuntu 16.04 LTS distribution. This means the tool is not guaranteed to work on other distributions of Linux. Despite this constraint for certain packages, there are promising results from running JAI-TC on the same distribution of Linux on other architectures. In the future, JAI-TC might support a wider range of CPU and GPU architectures, But as it stands, there are no plans to make this a reality.

In an ideal system, most of the constraints are the ones imposed by its users. JAI-TC has design decisions in place to encourage its user-base to utilize the tool chain as intended. However, it still might be too advanced for one end of the user spectrum and not support enough features for the other end. Simplified user design and inclusion of custom scripts were two of the ensures that were put in place to combat this constraint.

JAI-TC is designed for the Jetson platform which is taking advantage of some proprietary NVidia drivers and hardware. This means there are features that JAI-TC cannot exploit due to lack of information about the underlying architecture of the hardware. This is why from very early stages of development of JAI-TC, NVidia developer community was consulted. This however, still implies that the JAI-TC is bound by the knowledge of its community and heavily depends on NVidia Hardware experts to lead the script development in the fields that closely work with the NVidia drivers and proprietary hardware. This problem could also be experienced in building libraries that have built in assembly instructions, or require special drivers and patches to work.

One of the unfortunate constraints that JAI-TC will suffer from is the lack of storage. The hardware has limited flash storage and external storage devices generally do not support certain files such as symbolic links. This limits the JAI-TC's capabilities to tools that would fit on its flash memory. There are limited solutions to this issue and due to lack of time, it was decided that the solution needs to be addressed by the users.

### *3.2.5 Assumptions and Dependencies*

During the development of this tool there were assumptions made with regards to three different aspects of this tool. These assumptions were with regards to the software, users and the support community.

Despite the wide range of intended audience for this tool, there were some assumptions made with regards to their knowledge, skill level, and intended use cases. This tool assumes that the user has a good enough technical understanding of the Linux terminal environment that they would be able to navigate their way around the solution. JAI-TC also assumes that the user will honor the tool licenses and would not use this tool with any intentions to harm other users. It is also assumed that the user has access to internet and would be able to use this tool when it is connected to the internet.

JAI-TC's long term support heavily depends on the community which will maintain it. This tool assumes that there is an active community interested in this particular solution. It assumes that the users are willing to put time and effort to provide support for JAI-TC.

There were certain assumptions that were made with regards to the software aspect of JAI-TC. The first important assumption is that the latest versions of packages have the most demand from JAI-TC's user base. It is assumed that the software is run on the Jetpack platform and compilations all happen on the intended architecture. In other words, in its current state JAI-TC does not support cross compilation.

### 3.2.6 Apportioning of the requirements

The following Gantt chart, illustrates the time-line of JAI-TC's development and describes the required dead-lines for each section. These charts are presented in figures 1, 2 and 3.

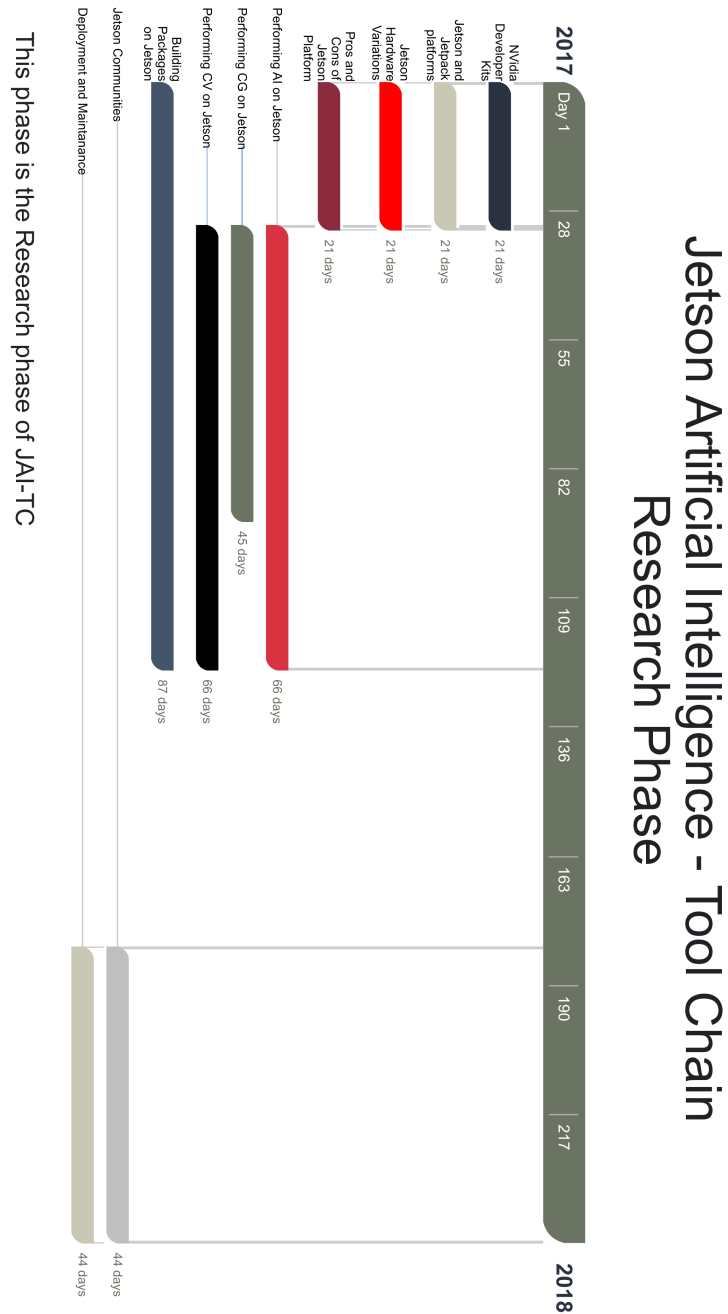
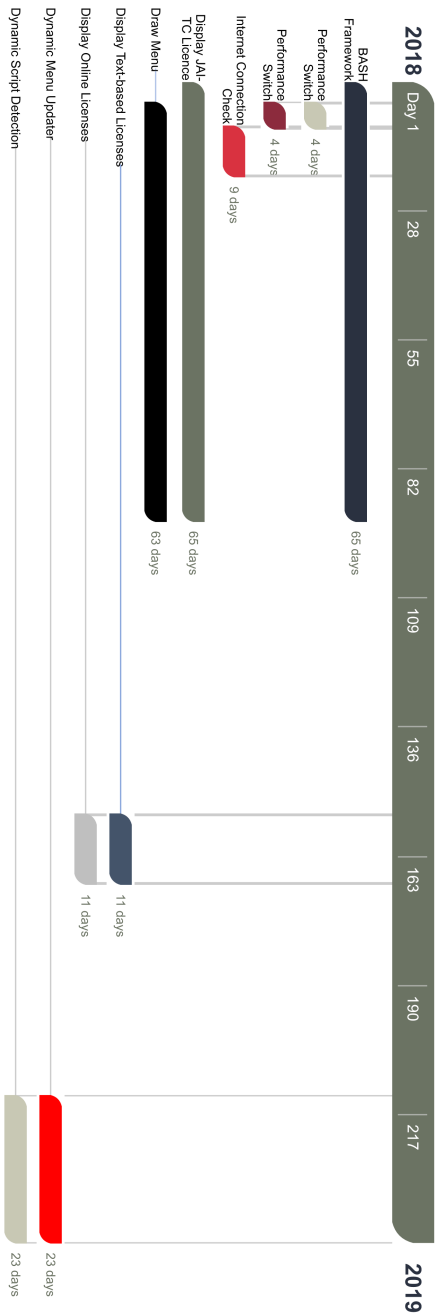


Figure 1: The Research Phase started in September 2017.

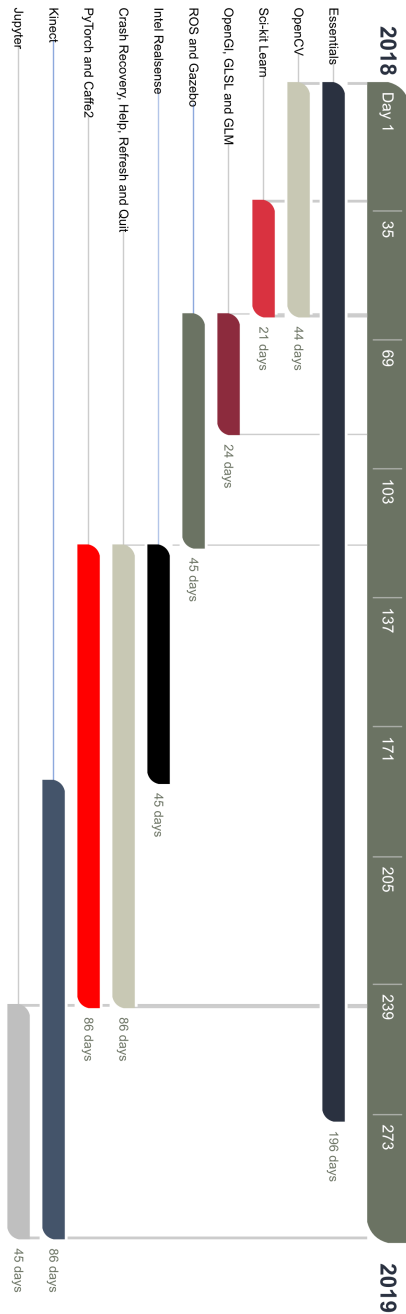
# Jetson Artificial Intelligence - Tool Chain Framework Phase



This phase is the development of the mainframe

Figure 2: Framework Development Phase started in July 2018.

## Jetson Artificial Intelligence - Tool Chain Package Development Phase



This phase is the package development phase of JAI-TC

Figure 3: Package installation script development Phase started in August 2018.

### 3.3 Specific Requirements

This subsection contains all of the functional and quality requirements of this solution. It provides a detailed description of the system and all its core features.

### 3.3.1 Functional Requirements

In this section, a list of all Functional requirements are provided. The Use Case format seem to be the best fit in describing these functional requirements.

**ID:** FR0

**Title:** BASH Framework

**Description:** JAI-TC's framework written in BASH.

**Rationale:** This is the overall framework of the JAI-TC. Every other method operates within this framework

**Dependencies:** None

**ID:** FR1

**Title:** Performance Switch

**Description:** This function switches the Jetson to high performance mode when it starts up, and back to normal performance mode on exit.

**Rationale:** This will ensure a smooth build experience by the user and can decrease the build time.

**Dependencies:** FR0

**ID:** FR2

**Title:** Fan speed adjustment

**Description:** This controls the fan behavior once the tool is launched.

**Rationale:** If the Jetson is set to high performance mode, the fan needs to be ramped up to help with the heat deception and prevent throttling.

**Dependencies:** FR0, FR1

**ID:** FR3

**Title:** Internet Connection Check

**Description:** This method checks for the internet connection. It shuts the program down if there is no internet connection.

**Rationale:** The tool depends on the internet. There is no point in running the program if there is no internet.

**Dependencies:** FR0

**ID:** FR4

**Title:** Display JAI-TC Licence

**Description:** This function displays the license of JAI-TC to the user and asks for their name to be entered.

**Rationale:** This is used to acknowledge that the user is made aware of the terms and conditions of JAI-TC.

**Dependencies:** FR0

**ID:** FR5

**Title:** Draw Menu

**Description:** This method acquires a list of available scripts and presents them in an enumerated fashion to the user. Then it asks for the user input.

**Rationale:** This requirement is the back bone of JAI-TC's user interface.

**Dependencies:** FR0

**ID:** FR6

**Title:** Display Text-based Licenses



**Description:** This function displays the co-responding “;script name;.license” file to the user.

**Rationale:** This particular license usually contains the license of the person who has made the scripts as well as the tool’s license. It is very customizable.

**Dependencies:** FR0

**ID:** FR7

**Title:** Display Online Licenses

**Description:** This function displays the package’s license by getting it directly from the package’s home website or other online means.

**Rationale:** This ensures that the latest license of the package that is being installed is displayed to the user.

**Dependencies:** FR0, FR3

**ID:** FR8

**Title:** Dynamic Menu Updater

**Description:** This function rechecks the list of available scripts, adds the new scripts to the list and re-displays the list to the user.

**Rationale:** There is a chance that the user would not restart the tool after creating a new script. This function allows the user to see the new script without restarting the program.

**Dependencies:** FR0, FR5, FR9

**ID:** FR9

**Title:** Dynamic Script Detection

**Description:** This function automatically detects the newly added scripts

**Rationale:** This lets the menu updater know that there are new scripts to be added to the list.

**Dependencies:** FR0

**ID:** FR10

**Title:** Help Menu

**Description:** This function displays a brief help article to the user.

**Rationale:** This helps the user to figure out how to utilize the tool in case they needed some help with that procedure.

**Dependencies:** FR0, FR5

**ID:** FR11

**Title:** Quit with Reboot Option

**Description:** This function allows the user to gracefully close the JAI-TC. It also allows the user to chose to reboot the tool.

**Rationale:** Certain packages might require a reboot in order to work. Also, JAI-TC needs to close gracefully.

**Dependencies:** FR0, FR5

**ID:** FR12

**Title:** Crash Recovery

**Description:** JAI-TC is capable of recovering from crashes and build failures.

**Rationale:** The program should not shut down when there is a problem with the installation script. This would give user a chance to try other options without having to re-launch the tool.

**Dependencies:** FR0

**ID:** FR13

**Title:** Package: Essentials

**Description:** This installation script includes the tools that are essential to developing on Jetson platform.

**Rationale:** Using a single script the user can setup important tools such as python, dev libraries, git, screen and many more development tools.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9

**ID:** FR14

**Title:** Package: OpenCV

**Description:** This installation scripts builds the OpenCV package from the source.

**Rationale:** Allows the user to develop Image Processing and Computer Vision tasks.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR15

**Title:** Package: OpenGL

**Description:** Installs the open source graphics language on the Jetson.

**Rationale:** Allows the user to produce images using the graphics hardware.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR16

**Title:** Package: GLM

**Description:** Installs the graphics language's Math library.

**Rationale:** This is a very important tool used in conjunction with OpenGL and GLSL.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR17

**Title:** Package: PyTorch

**Description:** This script build from source and installs the PyTorch AI package for the python.

**Rationale:** This allows the user to develop Machine learning, Deep learning and Computer vision applications on Jetson hardware.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR18

**Title:** Package: Caffe2

**Description:** This script build from source and installs the Caffe2 AI package.

**Rationale:** This allows the user to develop Machine learning, Deep learning and Computer vision applications on Jetson hardware.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR19

**Title:** Package: Science Kit Learn

**Description:** This script Installs the Science Kit Learn AI package.

**Rationale:** This allows the user to train and infer AI models using highly optimized AI implementations on the Jetson hardware.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR20

**Title:** Package: ROS

**Description:** This scripts installs the Robot Operating System on Jetson platform.

**Rationale:** Allows developers to take advantage of hundreds of tools and APIs designed for robotic applications.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR21

**Title:** Package: ROS II

**Description:** This scripts installs the Robot Operating System 2 on Jetson platform.

**Rationale:** Allows developers to take advantage of hundreds of tools and APIs designed for robotic applications.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR22

**Title:** Package: Jupyter

**Description:** Installs the Jupyter IDE on the Jetson.

**Rationale:** Allows the user to take advantage of one of the most user friendly python environments that is available for developers.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR23

**Title:** Package: Intel Realsense <sup>TM</sup>

**Description:** Installs all the required drivers and APIs needed for developing for the Intel Realsense <sup>TM</sup>Technology.

**Rationale:** Realsense <sup>TM</sup>Technology is a smaller more lightweight alternative to Microsoft Kinect.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

**ID:** FR24

**Title:** Package: Kinect

**Description:** This script installs the required drivers and samples for the Microsoft Kinect by building the OpenKinect from source.

**Rationale:** This allows the users to capture depth data for use in their various applications on Jetson.

**Dependencies:** FR0, FR4, FR5, FR6, FR7, FR9, FR13

### *3.3.2 Performance Requirements*

This section covers the performance requirements of JAI-TC. The following requirements describe the quality requirements which JAI-TC delivers.

**ID:** QR1

**Title:** Easy to Use

**Description:** The tool is easy to install and use.

**Rationale:** The point of having this tool is to make it easier to install these packages. JAI-TC is not trying to add more complexity to setup procedure.

**Dependencies:** None

**ID:** QR2

**Title:** Easy to Customize

**Description:** JAI-TC Allows users to customize their experience by adding their custom package installers or modifying the already existing ones.

**Rationale:** Users with different skill level should be able to get different experiences from this tool. This flexibility will allow the more advanced users to get much more out of JAI-TC.

**Dependencies:** None

## 4 DESIGN

This section covers the overall design of Jetson Artificial Intelligence - Tool chain. It provides an in depth description of the implementation with the design rational behind it. Furthermore, it provides the metrics to assess the success of the development. This section provides UML diagrams of the under-laying components and connectors.

### 4.0.1 *Intended Audience*

This section's intended audience are developers who would contribute, replicate or maintain this tool. Furthermore, this document targets the JAI-TC community. This includes people who are interested in improving the tool and use JAI-TC as a basis for more advanced and robust solutions.

### 4.0.2 *Conformance*

This section conforms to Jetson developer community's needs for various installation solutions. Furthermore, it conforms to the AI community's needs for robust installation scripts for some of the most widely used packages in this practice. It takes the needs of the robotics community in to consideration for a fast and reliable way of installing tools used for robotic applications. Finally, this tool conforms to the needs of the hobbyists with varying degrees of experience with this hardware.

## 4.1 System Architecture and Rationale

### 4.1.1 *Design Patterns*

Design patterns allow standardized approaches to providing services to the end user. Selecting the correct design patterns depend on the user base, purpose of the solution and the platform. JAI-TC is designed on the basis of two design patterns. These two design patterns are Facade and Visitor.

The first design pattern is the Facade. Facade pattern provides a unified high-level interface by hiding and abstracting away the complexity of the tool chain. This abstraction happens on two levels. The first level is the abstraction of the bash scripts and the second level is the abstraction of license compliance.

The second Pattern is the Visitor design pattern. Visitor pattern allows for creation of operation sets for various types of situations that need to be handled. JAI-TC has 3 different approaches to installing packages. The first approach is to get the desired package from standardized package managers such as Aptitude and Pip. The second approach is building the packages from the developer's source code. The final approach is by extracting pre-compiled binaries. Furthermore, there are two different methods of license handling that JAI-TC supports. These two methods are referred to as Offline and Online license displays. JAI-TC needs a visitor design pattern

in order to allow these different operations. The overall system component is also described in the UML figure 4.

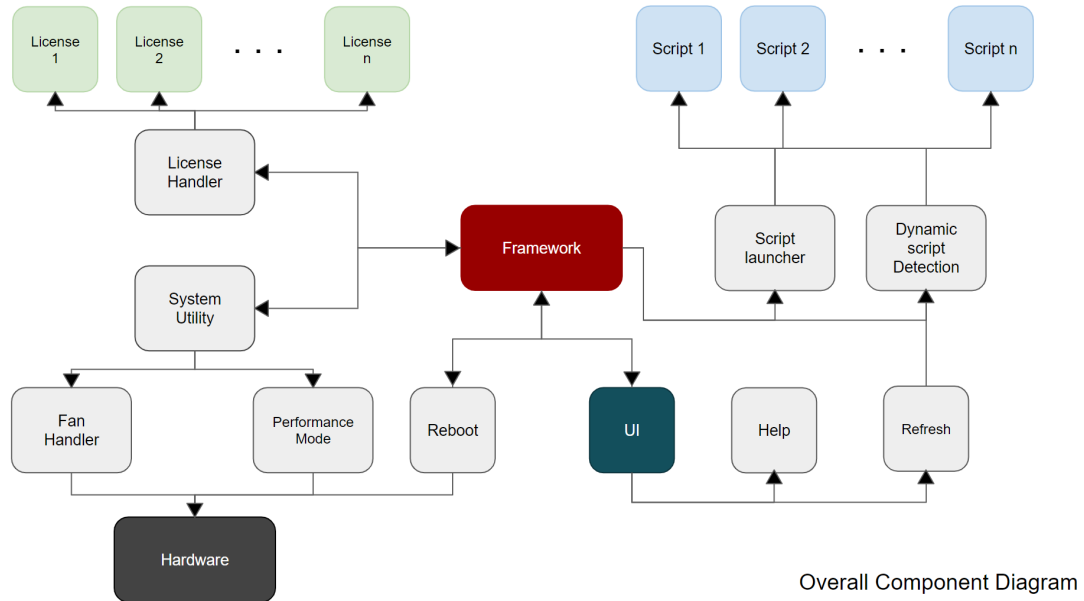


Figure 4: Overall Component Design

#### 4.1.2 Decomposition Description

JAI-TC is decomposed on the basis of cohesion into three general sections. These three sections are research, framework design and package installation script development. This approach to decomposition allows the development to focus on association of different requirements and allows for a faster and more efficient development process.

## 4.2 JAI-TC in Perspective

### 4.2.1 Concerns

There were several concerns that needed to be addressed through the development of this solution. The first concern was with regards to the complexity of using JAI-TC. JAI-TC was developed to ensure that the user can achieve the desired effect with least effort. Furthermore, JAI-TC makes it possible for the user to spend little time on going through the setup procedure. Please note that building a package can take several hours to complete. However, this time does not require the user's attention and could be done in the background while the user focuses on other aspects of their development or spike.

Another major concern with regards to JAI-TC was the maintenance of the tool in the long run. Within the development of the package installation scripts there were several updates. The first update broke several packages. By the last update, the installation scripts were set up in such a way that the updates were no longer affecting the installation script's success. This was achieved through trial and error. While a large part of this problem was addressed, it remains to be a threat to integrity of this tool and would require a routine maintenance in order to ensure its operation. This could only be achieved through a dedicated community and

a large user base.

One of the major concerns during the development of JAI-TC package installation scripts was the compatibility of the scripts with other variations of the Jetson hardware. So far the JAI-TC is tested on TX2 and Xavier. The packages that do not rely on CUDA are guaranteed to work on all variations of Jetson hardware and the few that do depend on CUDA, have custom scripts for different Jetson platforms in place.

The final concern with regards to JAI-TC is deployment and user acceptance. As mentioned earlier, the long term functionality of JAI-TC depends on an active community willing to assist in the maintenance of this tool chain. This was addressed by specifically target packages that have the highest demand in the community.

### 4.3 Component Design

This section provides a detailed description of each working component and connector of JAI-TC solution. JAI-TC contains 26 individual components and connectors. Each component and connector is designed specifically to address one of the major requirements of this tool chain found in the requirements section. For an overview of these components, please refer to figure 4.

#### 4.3.1 Performance Switch

The NVidia Jetson platform is designed to be a portable and self contained hardware that would depend on a power supply with limited battery life. In order to save power, Jetson provides several modes of operation. Since the installation of packages generally happen during the development phase, it is safe to assume that the hardware is generally plugged in to a more permanent power source such as a power adapter. Based on this assumption a decision was made to set the Jetson to high performance mode during compilation and building of packages in order to save time. In order to achieve this a component was designed to directly communicate with the hardware. This component is presented in the UML depicted in figure 5.

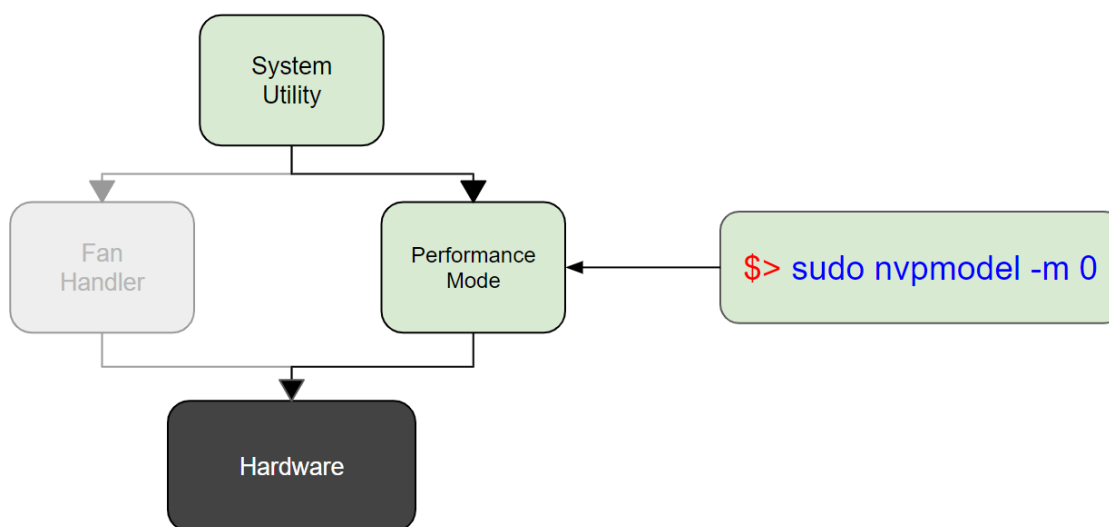


Figure 5: System Utility: Performance Switch Component Design

### 4.3.2 Fan speed adjustment

Once the Jetson is set to high performance mode, the core temperature of the SoC starts rising. This presents several issues. If the temperature rises higher than a certain threshold, it risks damaging the SoC. In order to prevent this there are built in measures that would limit the performance of the hardware to prevent the temperature from going any higher. These measures include thermal throttling and system shutdowns. In order to prevent these measures from kicking in, the on-board fan needs to be turned on. Initially there was an included bash script with the Jetpack that took care of the fan handling. This script is no longer shipped with Jetpack and now is deprecated. So a component was designed to handle the fan controls for the system. This task is done by setting the pulse with modulation pin that the fan is connected to, to the highest possible value to turn the fan on. This is done by the following command:

```
1 # Turning the fan on by setting the pin high
2 > sudo su -c 'echo 255 > /sys/devices/pwm-fan/target_pwm'
```

Once JAI-TC exits, this fan needs to be turned back off. This could be done by setting the same pin to its lowest possible value:

```
1 # Turning the fan off by setting the pin low
2 > sudo su -c 'echo 0 > /sys/devices/pwm-fan/target_pwm'
```

This component's UML is provided in figure 6.

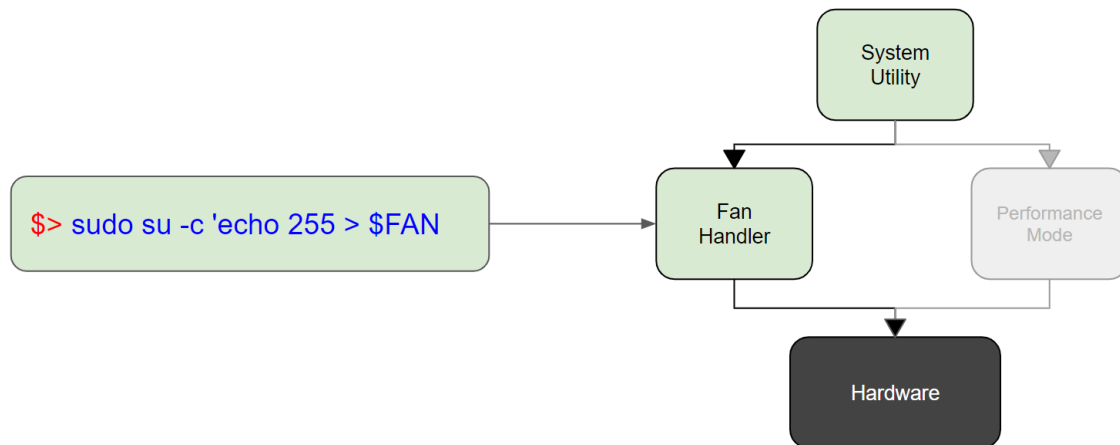


Figure 6: System Utility: Fan Handler Component Design

### 4.3.3 Internet Connection Check

JAI-TC requires internet connection in order to operate. Therefore, it is important to check the internet connectivity in the very beginning of the operation. The internet connection could be checked from bash. In order to do that a system utility script was developed. The system utility pings a website and if it gets any responses, it means the internet connection is established. If it receives a timeout on the other hand, it means the connection is interrupted. The program will close if there is no internet connection. The flowchart in figure 7 describes the Internet Connection check system utility.

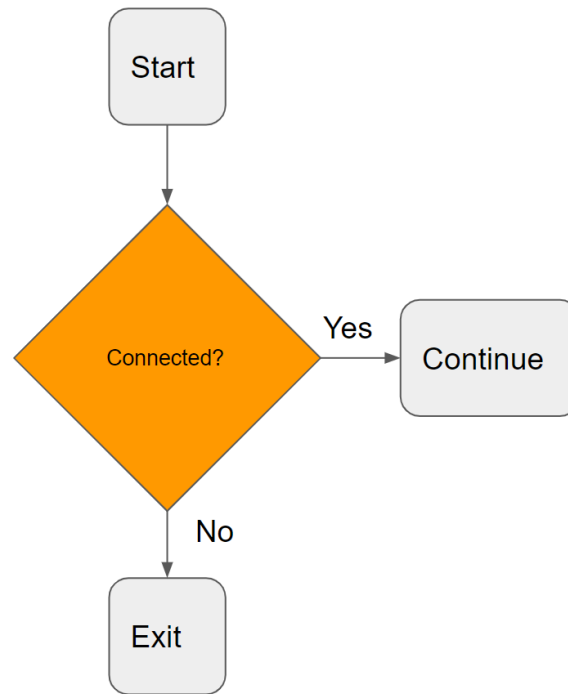


Figure 7: System Utility: Internet Connection Check Flowchart

#### 4.3.4 Display JAI-TC Licence

This component is in charge of displaying both JAI-TC and the package licenses. The JAI-TC License is displayed after internet check and the user has to agree with it in order to proceed. If the user refuses to agree with the license, JAI-TC exits. If they do agree, they have to enter their name and this information is stored until the next system reboot. When the system starts up after a reboot, the user is prompted with the same license agreement again. The package licenses are displayed before the tool begins to install. It mentions that by installing the package the user is agreeing to the terms and conditions of the license provided by the package, installation script and the patch developers. The user can also choose to view the online license. This is done by the question user is prompted with. If the user chooses to not view the online license, the installation begins. If they do choose to view the online license, then they have the option to scroll down by hitting space or quit viewing it by hitting the 'Q' key on their keyboard. As it is shown in figure 8, there are two variants of license files to be managed by the license handler component. The first group is a simple text file. This license file is also referred to as the offline license and is represented by "`;;Name;.license`". This file does not depend on any websites or internet to be displayed. Furthermore, this license is displayed to the user immediately after selecting a package and cannot be skipped. The second type of licenses are the executable license files. These license files are denoted as "`Name.sh`" and are referred to as the online licenses. These licenses are directly downloaded from the package developer's website or are loaded from a file if they follow one or more of the general licenses such as MIT, BSD or GPL.



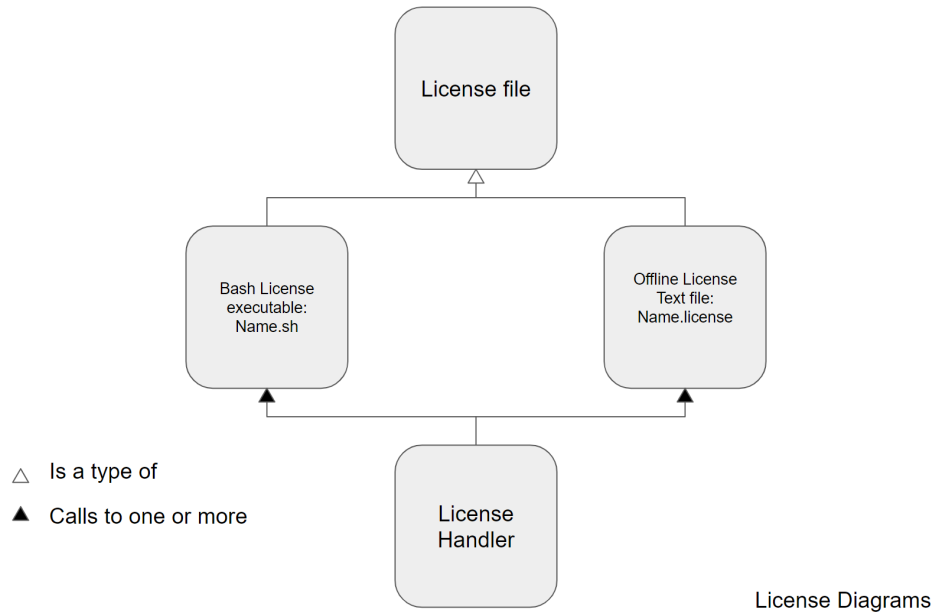


Figure 8: License Handler Component

#### 4.3.5 Draw Menu

JAI-TC users can use the text-based UI provided by the tool in order to interact with this tool chain. This is implemented by a bash while loop that prints each element in the menu from a list. At the end, the list is appended with the basic functions of the UI, such as Help, Refresh and Quit. The list is enumerated based on alphabetical order of each item. The menu, receives the user input on which item to process. After the script is finished processing, the same menu is displayed again. As it is shown in figure 9, UI and framework need to communicate since the UI is telling the framework which scripts to run and the framework needs to re-display the UI menu to the user.

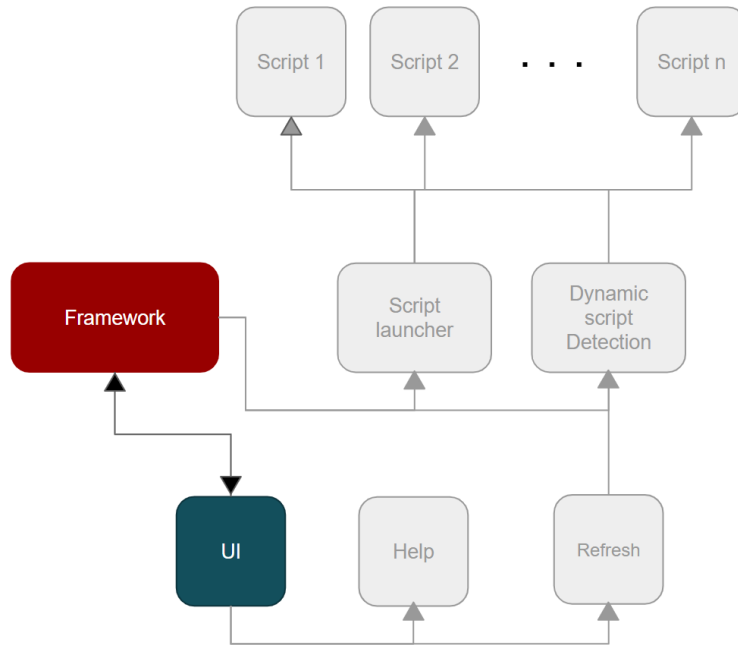


Figure 9: User Interface Component

#### 4.3.6 Dynamic Menu Updater and Script Detection

These two components are responsible with ensuring that when a new script is added, the user does not have to restart the tool chain in order to view them. Once these components are called, the script detector will look for new scripts that were added to the scripts folder. Then it will create a new list of all the available scripts and pass them to the menu. The menu updater will ensure that the new list is displayed to the user. From this point on, the menu draw component will include the new script. The same component works when a script is removed as well. Please note that this component is not responsible for making sure that the script will run within JAI-TC since that depends on other elements such as whether the offline license is present or not.

#### 4.3.7 Quit with Reboot Option

Some packages install drivers that would require a reboot before they could be utilized. The same could be said about certain driver patches as well. The Bash environment allows JAI-TC to use the "reboot" command to make this happen. This happens after user attempt to exit JAI-TC. The flowchart displayed at figure 10 shows the sequence of events that take place when the user attempts to exit the program.

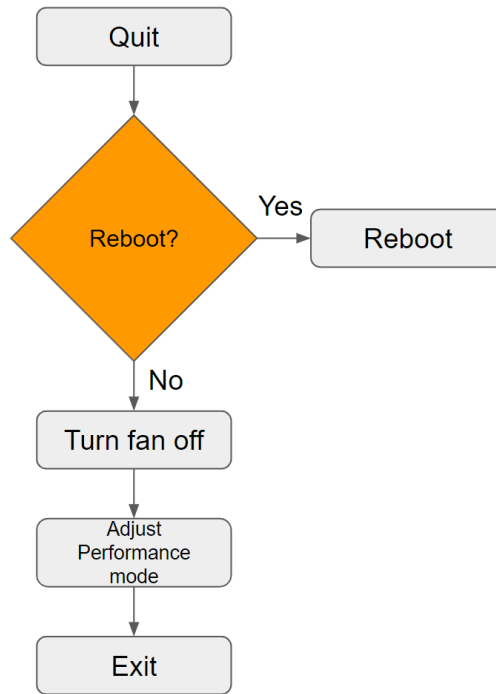


Figure 10: Quit with Reboot Option Flowchart

When the user tries to exit, they are asked if they would like to reboot their Jetson hardware. If they chose no, the program exits. If the user chooses to reboot, then the quit component attempt to reboot Jetson hardware after 5 seconds.

#### 4.3.8 Package handler

The package handler is in charge of running the package installation scripts once the framework calls them with the appropriate package index. The package handler is responsible for ensuring that the script runs and the user is informed whether the process was successful or not. As it is shown in figure 11, there are 3 types of scripts that are managed by this component. The first type is the scripts that require a standardized package manager such as Aptitude or python's Pip to install a certain package. The second group are the installation scripts that depend on building the package from the source code. The final type, are the packages that need to be installed by extracting a pre compiled binary into its appropriate destination.

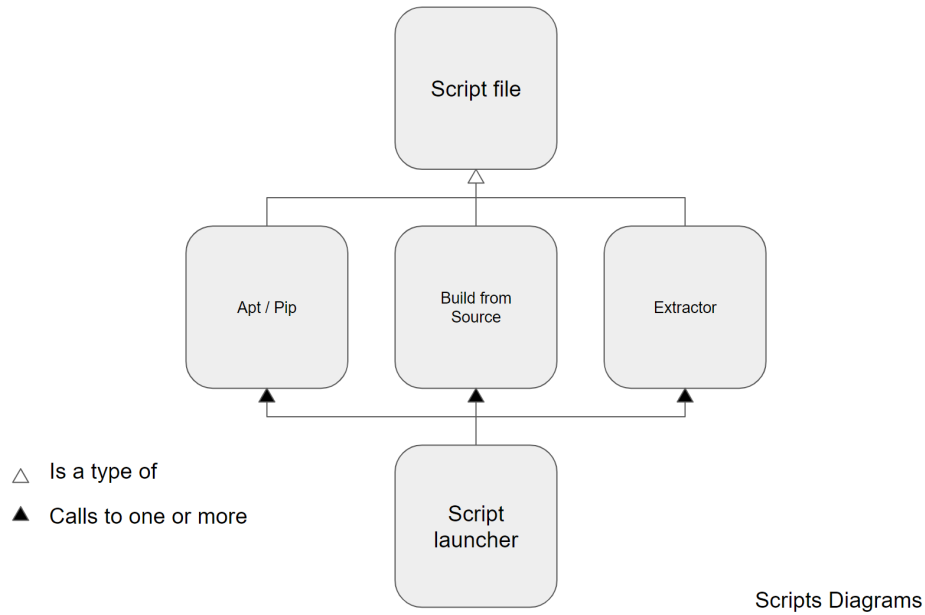


Figure 11: Package handler Component Diagram

#### 4.3.9 Package: Essentials

This package is perhaps the most important package JAI-TC provides. This is because, the essentials package includes all of the necessary development tools that other packages would require to be built and installed. Furthermore, it provides the necessary tools that a developer might need in order to start their development. Because of its importance, it is beneficial to have a more in-depth look at this particular package. This script is one of the Apt/pip type of packages from the package handler perspective. (View figure 11). The following packages are installed if the user chooses the essentials from the enumerated menu:

- Aptitude: screen, gcc, g++, cmake, gnome-tweaks, gnome-tweak-tool, python2.7, python3.7, python-pip, python3-pip, libffi-dev, python-dev, build-essential, -y libtiff5-dev, libjpeg8-dev, zlib1g-dev, libfreetype6-dev, liblcms2-dev, libwebp-dev, tcl8.6-dev, tk8.6-dev, python-tk, python-numpy, python-scipy, python-matplotlib, python3-numpy, python3-scipy, python3-matplotlib, clang, python, python-dev, libzmq, libzmq-dev, libpng-dev and pkg-config.
- Pip: setuptools, python, numpy, scipy and matplotlib.

This is done by running a command such as:

---

```
1 > sudo apt install <Package Name> -y
```

---

and

---

```
1 > pip install --user <Package Name>
```

---

#### *4.3.10 Other Packages*

Each package will be installed, build or extracted based on the instructions provided in the script. For more information on how some of these components were implemented please visit the source code repository of JAI-TC. The "scripts" folder contains the instructions for installation of each one of the packages. Furthermore, more packages could be added by the user. The procedure for creating a new script for JAI-TC is covered in the user manual section of this documentation.

### **4.4 User Interface Design**

#### *4.4.1 Overview of User Interface*

From the user's perspective the UI contains a numbered list of packages to be installed. This list also includes 3 constant items that do not change even if all scripts are removed. These 3 elements are Refresh, Help and Quit. The decision was made to implement a text-based UI due to its simplicity. Text-based UIs are faster to develop and much easier to maintain. Finally it is easier to ensure that a text-based UI is compatible with different platforms by not requiring 3<sup>rd</sup> party packages.

#### *4.4.2 Screen Objects and Actions*

This portion of the documentation covers different UI functions and their expected output. When JAI-TC is launched, the first thing that user experiences is the JAI-TC License page. This block of text provides JAI-TC's license, reminds them that by installing each package they are agreeing to all individual licenses of those packages and then asks for user's signature. Please note that this signature is only valid until the next time the hardware is shut down.

After user agreed to JAI-TC license, the program starts installing necessary packages for JAI-TC and then bring the user to the installation scripts menu. This menu is shown in figure 12. Please note that the package list could vary based on various package installers the user has.

```

Path is /home/behnamsaedi/Desktop/DM/JAI-TC.
Please select one of the following packages to install:
1: Caffe2+CUDA
2: Essentials
3: GLM
4: Kinect
5: Opencv_Contrib+CUDA
6: OpenCV+CUDA
7: OpenGL
8: PyTorch
9: Realsense
10: ROS-by-Package
11: ROS-Desktop-Full
12: ROS
13: Scikit-learn
h: Help
c: Credits
q: Quit
Enter Package number: 

```

Figure 12: JAI-TC installation script menu

When the user selects a package from this menu, the license handler takes over and displays the appropriate license as seen in figure 13. The license handler presents the user with the license information in a small bite size chunks to not overwhelm the user with information.

```

13: Scikit-learn
h: Help
c: Credits
q: Quit
Enter Package number: 2
Installing Essentials...
By installing this, you are agreeing to all the following terms and licenses:
Package category: Essentials
By installing this package you are agreeing to all the licenses of the following packages:
- screen
- GCC and G++
- gnome-tweaks
- gnome-tweak-tools
- python2.7
- python3.7
- pip
- numpy
- scipy
- matplotlib
- ipython
- ipython-notebook
- python-panda
- clang
- python
- python-dev
- fftw
- libzmq
- libzmq-dev
- freetype
- freetype-dev
- libpng
- libpng-dev
- pkg-config

MIT License
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

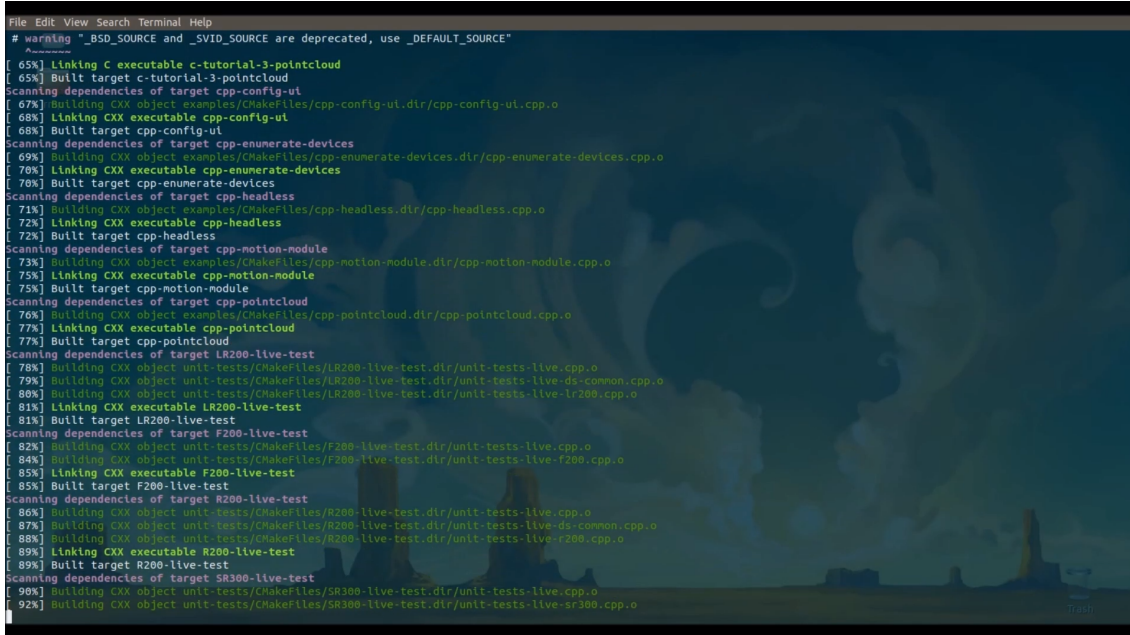
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Do you wish view these licenses?

```

Figure 13: JAI-TC installation script license handler

After this step the installation begins. Figure 14 shows an example of the build procedure as it is displayed to the user. This organization looks familiar with users that have prior experience of building packages from

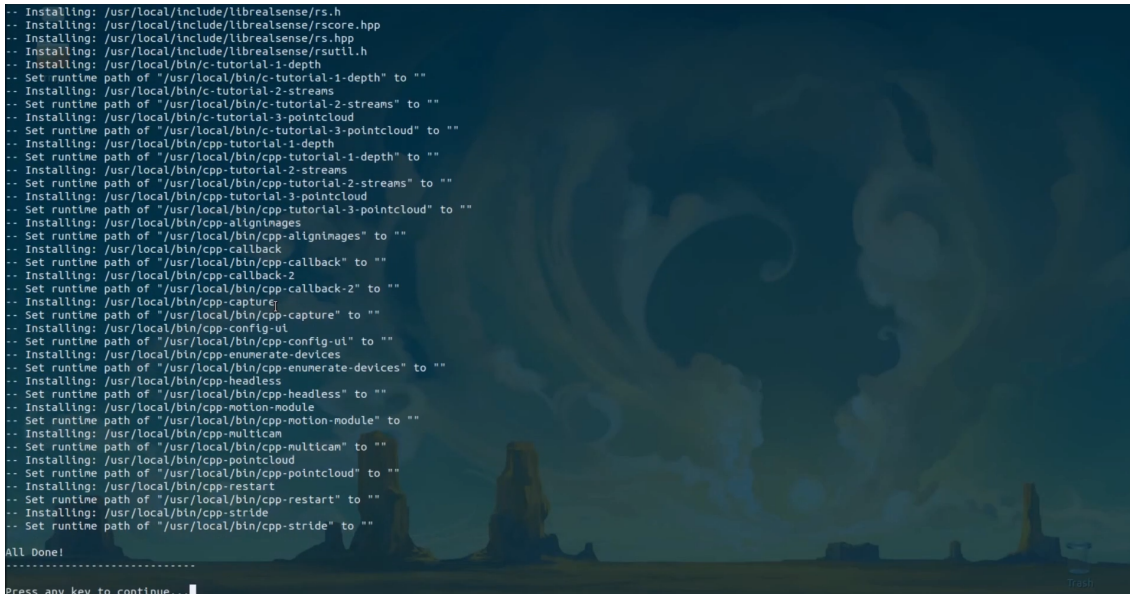
source code. This is important because it simplifies the crash handling and also gives the user a good idea on what went wrong in case of any issues.



```
File Edit View Search Terminal Help
# warning "_BSD_SOURCE and _SVID_SOURCE are deprecated, use _DEFAULT_SOURCE"
[ 65%] Linking C executable c-tutorial-3-pointcloud
[ 65%] Built target c-tutorial-3-pointcloud
Scanning dependencies of target cpp-config-ut
[ 67%] Building CXX object examples/CMakeFiles/cpp-config-ut.dir/cpp-config-ut.cpp.o
[ 68%] Linking CXX executable cpp-config-ut
[ 68%] Built target cpp-config-ut
Scanning dependencies of target cpp-enumerate-devices
[ 69%] Building CXX object examples/CMakeFiles/cpp-enumerate-devices.dir/cpp-enumerate-devices.cpp.o
[ 70%] Linking CXX executable cpp-enumerate-devices
[ 70%] Built target cpp-enumerate-devices
Scanning dependencies of target cpp-headless
[ 71%] Building CXX object examples/CMakeFiles/cpp-headless.dir/cpp-headless.cpp.o
[ 72%] Linking CXX executable cpp-headless
[ 72%] Built target cpp-headless
Scanning dependencies of target cpp-motion-module
[ 73%] Building CXX object examples/CMakeFiles/cpp-motion-module.dir/cpp-motion-module.cpp.o
[ 75%] Linking CXX executable cpp-motion-module
[ 75%] Built target cpp-motion-module
Scanning dependencies of target cpp-pointcloud
[ 76%] Building CXX object examples/CMakeFiles/cpp-pointcloud.dir/cpp-pointcloud.cpp.o
[ 77%] Linking CXX executable cpp-pointcloud
[ 77%] Built target cpp-pointcloud
Scanning dependencies of target LR200-live-test
[ 78%] Building CXX object unit-tests/CMakeFiles/LR200-live-test.dir/unit-tests-live.cpp.o
[ 79%] Building CXX object unit-tests/CMakeFiles/LR200-live-test.dir/unit-tests-live-ds-common.cpp.o
[ 80%] Building CXX object unit-tests/CMakeFiles/LR200-live-test.dir/unit-tests-live-lr200.cpp.o
[ 81%] Linking CXX executable LR200-live-test
[ 81%] Built target LR200-live-test
Scanning dependencies of target F200-live-test
[ 82%] Building CXX object unit-tests/CMakeFiles/F200-live-test.dir/unit-tests-live.cpp.o
[ 83%] Building CXX object unit-tests/CMakeFiles/F200-live-test.dir/unit-tests-live-f200.cpp.o
[ 85%] Linking CXX executable F200-live-test
[ 85%] Built target F200-live-test
Scanning dependencies of target R200-live-test
[ 86%] Building CXX object unit-tests/CMakeFiles/R200-live-test.dir/unit-tests-live.cpp.o
[ 88%] Building CXX object unit-tests/CMakeFiles/R200-live-test.dir/unit-tests-live-r200.cpp.o
[ 89%] Linking CXX executable R200-live-test
[ 89%] Built target R200-live-test
Scanning dependencies of target SR300-live-test
[ 90%] Building CXX object unit-tests/CMakeFiles/SR300-live-test.dir/unit-tests-live.cpp.o
[ 92%] Building CXX object unit-tests/CMakeFiles/SR300-live-test.dir/unit-tests-live-sr300.cpp.o
```

Figure 14: JAI-TC installation script building from source code

Once the installation is done, the user is notified that the installation is completed and prompted to go back to the main menu. This prompt is shown in figure 15. By pressing any key user is returned.



```
-- Installing: /usr/local/include/librealsense/rs.h
-- Installing: /usr/local/include/librealsense/rscore.hpp
-- Installing: /usr/local/include/librealsense/rs.hpp
-- Installing: /usr/local/include/librealsense/rsutil.h
-- Installing: /usr/local/bin/c-tutorial-1-depth
-- Set runtime path of "/usr/local/bin/c-tutorial-1-depth" to ""
-- Set runtime path of "/usr/local/bin/c-tutorial-2-streams" to ""
-- Installing: /usr/local/bin/c-tutorial-3-pointcloud
-- Set runtime path of "/usr/local/bin/c-tutorial-3-pointcloud" to ""
-- Installing: /usr/local/bin/cpp-tutorial-1-depth
-- Set runtime path of "/usr/local/bin/cpp-tutorial-1-depth" to ""
-- Installing: /usr/local/bin/cpp-tutorial-2-streams
-- Set runtime path of "/usr/local/bin/cpp-tutorial-2-streams" to ""
-- Installing: /usr/local/bin/cpp-tutorial-3-pointcloud
-- Set runtime path of "/usr/local/bin/cpp-tutorial-3-pointcloud" to ""
-- Installing: /usr/local/bin/cpp-alignnages
-- Set runtime path of "/usr/local/bin/cpp-alignnages" to ""
-- Installing: /usr/local/bin/cpp-callback
-- Set runtime path of "/usr/local/bin/cpp-callback" to ""
-- Installing: /usr/local/bin/cpp-callback-2
-- Set runtime path of "/usr/local/bin/cpp-callback-2" to ""
-- Installing: /usr/local/bin/cpp-capture
-- Set runtime path of "/usr/local/bin/cpp-capture" to ""
-- Installing: /usr/local/bin/cpp-config-ut
-- Set runtime path of "/usr/local/bin/cpp-config-ut" to ""
-- Installing: /usr/local/bin/cpp-enumerate-devices
-- Set runtime path of "/usr/local/bin/cpp-enumerate-devices" to ""
-- Installing: /usr/local/bin/cpp-headless
-- Set runtime path of "/usr/local/bin/cpp-headless" to ""
-- Installing: /usr/local/bin/cpp-motion-module
-- Set runtime path of "/usr/local/bin/cpp-motion-module" to ""
-- Installing: /usr/local/bin/cpp-multicam
-- Set runtime path of "/usr/local/bin/cpp-multicam" to ""
-- Installing: /usr/local/bin/cpp-pointcloud
-- Set runtime path of "/usr/local/bin/cpp-pointcloud" to ""
-- Installing: /usr/local/bin/cpp-restart
-- Set runtime path of "/usr/local/bin/cpp-restart" to ""
-- Installing: /usr/local/bin/cpp-stride
-- Set runtime path of "/usr/local/bin/cpp-stride" to ""

All Done!
.....
Press any key to continue...
```

Figure 15: JAI-TC installation script done and ready to go back to main menu

## 5 USER MANUAL

This section covers the user's operations manual for Jetson Artificial Intelligence - Tool chain (JAI-TC). This section's purpose is to be used as a reference for users and help them understand how the tool is intended to be used. This section will provide an in depth procedure of how to perform various tasks in JAI-TC.

### 5.1 Installation

This subsection focuses on downloading, setting up and running JAI-TC.

#### 5.1.1 Downloading JAI-TC

JAI-TC is available through JAI-TC's GitHub Repository. This repository could be accessed at: "<https://github.com/BeNsAeI/JAI-TC>". From here, users have 2 options for downloading JAI-TC:

- 1) **Cloning:** Cloning this tool will require Git. If the Jetson hardware already has Git installed, this is the preferred method of acquiring JAI-TC. This could be done by entering the following command in Jetson terminal:

---

```
1  # Navigate to the desired destination folder
2  > cd path/to/JAI-TC
3  # Clone from repository
4  > git clone https://github.com/BeNsAeI/JAI-TC.git
```

---

This will produce a new folder titled "JAI-TC". Inside the that folder you can find all the necessary files to start using this project.

- 2) **Downloading as a zip:** If the Jetson hardware does not already have Git installed, you have the option of downloading the file from GitHub. Please move the file in the desired destination and extract the package. The operation procedure for this option is no different than the first option.

One thing to remember in both cases is that some of the packages require upwards 16 GB of storage in order to build. Please make sure JAI-TC is located on a storage with enough free space to build your desired packages. Furthermore, the storage that is used as JAI-TC's installation target needs to allow symbolic links. This means storage that are formatted as FAT, are not supported. This could cause the installation procedure to fail.

#### 5.1.2 Setting Up JAI-TC

In order to setup JAI-TC, enter the JAI-TC folder and open a new terminal in that location. Alternatively, you can open a terminal and navigate to JAI-TC's folder:

---

```
1 > cd path/to/JAI-TC
```

---

Once in the destination folder, just simply run JAI-TC's main install script with root privileges. This will setup all the necessary packages and start the scripts menu:

---

```
1 > sudo ./install.sh
```

---



Once this command is entered, you are prompted to enter the root password. Generally on fresh Jetpack installation, this password is just "nvidia". Once the program has root access, you can see that Jetson's on-board fan will start spinning. This indicates that the Jetson is set to high performance mode. User is prompted with JAI-TC's end user license agreement. Please read this carefully and express whether you agree with those terms and conditions or not by following the on-screen procedure. Please note, the JAI-TC's license only lasts until the system is shut down. This means every time the system boots up, you are prompted with a new license agreement and your signature will be required again.

## 5.2 JAI-TC Operations

By this point, the JAI-TC should be downloaded, set up and ready to be used.

### 5.2.1 Package Selection

The menu is organized in a specific format:

---

```
1 1: Name of the 1st package
2 2: Name of the 2nd package
3 .
4 .
5 .
6 #: Name of the nth package
7 r: Refresh
8 h: Help
9 c: Credits
10 q: Quit
11 Enter Package number:
```

---

By entering the number, that package will be queued for installation. Please read the end user license agreement carefully and follow the on screen process to install your package. Please note, by installing these packages you are agreeing to ALL user license agreements that these tools have whether they are displayed or not. The user still is responsible to make sure they agree with all terms and conditions of the package, installation scripts and patches they are using. Once the installation is done, user is prompted to return to the main menu by pressing any key.

### 5.2.2 License handling

Please make sure to read the licenses these tools have. There is a possibility that a specific license is missing from license files. In that case please contact the community and providers of that installation script as soon as possible. User can choose to either view or skip the displaying of the online license. Please note, by installing the package, you are agreeing to all terms and conditions of licenses on the package, installation scripts and patches used even if you choose to skip viewing the online license. If you do not agree to the terms and conditions presented by the script, feel free to close the program by hitting "Ctrl + C".

### 5.2.3 Help Menu

The help menu is accessed by entering h or 'H' as input from the main menu. In this menu, you can use space to scroll down or q to exit. once done, you will be asked to press any key to return to the menu or you can go back by pressing 'q' at any time.

### 5.2.4 Refreshing the Package List

This feature is very simple to use. By entering 'r' or 'R' as input in main menu, the list gets refreshed. This can help users to not have to restart JAI-TC every time they add or remove a package installation script.

### 5.2.5 Exiting JAI-TC

Entering 'q' or 'Q' as input in the main menu allows the user to exit JAI-TC. User is asked if hardware needs to be restarted or not. It is generally recommended to restart the hardware if the package installs any drivers such as Kinect or RealSense™ technology drivers. If user chooses to reboot, the hardware gives them a 5 seconds grace period before rebooting. This can be cancelled by hitting "Ctrl + C". If the user does not wish to reboot, they can choose 'n', 'N' or "No" when they are asked.

## 5.3 Customizing The Software

This section is about developing custom package installation scripts for JAI-TC. More advanced users can use this feature to customize their experience with this tool.

### 5.3.1 Installation Scripts

Please follow the following steps in order to produce a custom installation script:

- 1) Create a new file "PackageName.sh" in the "scripts/" folder. Please note not to use any spaces in the file name.
- 2) Inside the script file "PackageName.sh", start the first line with `#!/bin/bash`. This is a bash script file.

---

```
1  #!/bin/bash
```

---

- 3) Enter the bash scripts that will install the tool.
- 4) Save your work
- 5) Give "PackageName.sh" execution permissions:

---

```
1  chmod u+x PackageName.sh
```

---

- 6) The item should now show up in the menu after refreshing the list

Please note that this will not work until the offline license is added.

### 5.3.2 Offline License File

Please follow the following steps closely to create the offline license file. This file needs to be present in order for the installation script to actually be executed by JAI-TC.

- 1) Create a new file "PackageName.licenses" in the "licenses/" folder. Please note that this name should match the co-responding script from "scripts/" folder that you created in last subsection.
- 2) Enter the license of the tool in plain text form.
- 3) Save your work.

The program can now be called from the JAI-TC menu.

### 5.3.3 Online License File

Online license is responsible for grabbing the most recent license text from the package provider's website. In order to create this license file please follow the following steps:

- 1) Create a new file "PackageName.sh" in the "licenses/" folder. Please note that this name should match the co-responding script from "scripts/" and license file from "licenses/" folder that you created in last subsections.
- 2) Inside the script file "PackageName.sh", start the first line with `#!/bin/bash`. This is a bash script file.

---

```
1  #!/bin/bash
```

---

- 3) Enter the bash scripts that display the appropriate licenses. JAI-TC recommends one of the two following approaches for consistency:
  - For file licenses that are present in the folder like BSD, GPL or MIT licenses you can use the following commands:

---

```
1  cat <License file address> | more -d
```

---

- For websites and web links with the license you can use the following commands:

---

```
1  lynx -dump <License web address>
```

---

- 4) Save your work
- 5) Give "PackageName.sh" execution permissions:

---

```
1  chmod u+x PackageName.sh
```

---

Note: this is only displayed if the user chooses the yes option on the question prompting them to view the online license. This format by no means replaces the "PackageName.license" file. It can be used as a supplement. If the "PackageName.license" does not match the online license file, please contact the script's developer as soon as possible to fix this error.

## 5.4 Repairing Missing Licences

If the licenses of a package are missing the JAI-TC will display an error and will not proceed with the installation of the package. When this happens users have 2 options for standard packages:

- 1) **Option 1:** re-clone the repository from GitHub. Please note that this will remove all of your custom scripts. So make sure to back up your custom packages before doing this.
- 2) **Option 2:** From repository, download the missing "PackageName.license" file and add it to the "licenses/" folder. It is important to respect the developer licenses and agreements. JAI-TC accepts no responsibility with this regards.

For custom packages, please contact the script developers in order to address this issue.

## REFERENCES

- [1] "merriam webster," <https://www.merriam-webster.com/dictionary/artificial%20intelligence>, accessed: 2019-05-31.
- [2] "Bash reference manual," [https://tiswww.case.edu/php/chet/bash/bashref.html#What-is-Bash\\_003f](https://tiswww.case.edu/php/chet/bash/bashref.html#What-is-Bash_003f), accessed: 2019-05-31.
- [3] "Caffe," <https://caffe.berkeleyvision.org/>, accessed: 2019-05-31.
- [4] "A gentle introduction to computer vision," <https://machinelearningmastery.com/what-is-computer-vision/>, accessed: 2019-05-31.
- [5] "Gcc, the gnu compiler collection," <https://gcc.gnu.org/>, accessed: 2019-05-31.
- [6] "Opengl mathematics," <https://glm.g-truc.net/0.9.9/index.html>, accessed: 2019-05-31.
- [7] "Learningopengl - shaders," <https://learnopengl.com/Getting-started/Shaders>, accessed: 2019-05-31.
- [8] "Nvidia jetson nano developer kit," <https://developer.nvidia.com/embedded-computing>, accessed: 2019-05-31.
- [9] "Kinect for windows," <https://developer.microsoft.com/en-us/windows/kinect>, accessed: 2019-05-31.
- [10] "<http://freesoftwaremagazine.com/>," , accessed: 2019-05-31.
- [11] "Linux man pages," <https://linux.die.net/man/>, accessed: 2019-05-31.
- [12] "Intel@realsense™technologies," <https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html>, accessed: 2019-05-31.
- [13] "Ros.org - about ros," <https://www.ros.org/about-ros/>, accessed: 2019-05-31.