Artificial Intelligence Radio Transceiver

The Artificial Intelligence Radio Transceiver (AIR-T) is a high-performance software-defined radio (SDR) seamlessly integrated with state-of-the-art processing and deep learning inference hardware. The incorporation of an embedded graphics processing unit (GPU) enables real-time wideband digital signal processing (DSP) algorithms to be executed in software, without requiring specialized field programmable gate array (FPGA) firmware development. The GPU is the most utilized processor for machine learning, therefore the AIR-T significantly reduces the barrier for engineers to create autonomous signal identification, interference mitigation, and many other machine learning applications. By granting the deep learning algorithm full control over the transceiver system, the AIR-T allows for fully autonomous software defined and cognitive radio.

Out of the box, the AIR-T is a fully functioning SDR that includes numerous examples and open source APIs. The system includes AirStack, the AIR-T's complete software package. AirStack consists of the Ubuntu operating system, drivers, FPGA firmware, and everything required for AIR-T operation.

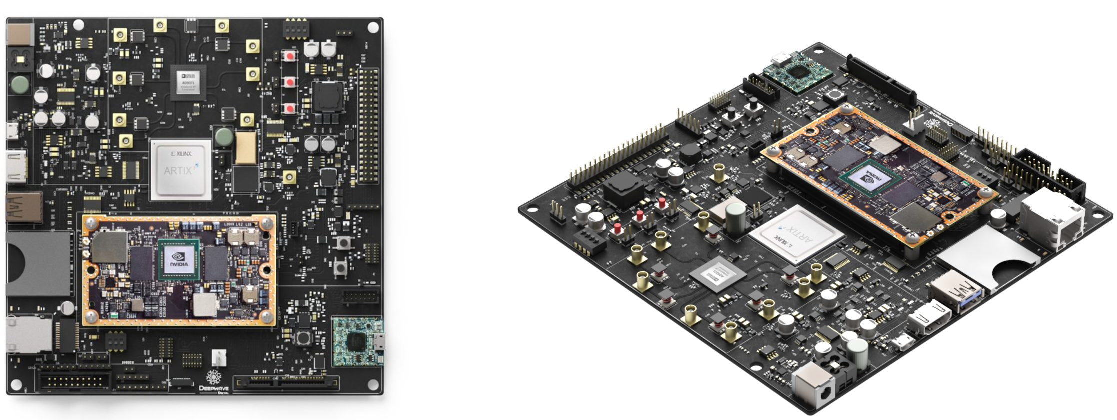

Figure 1: AIR-T Images

Document Overview

This document lists the specifications for the Artificial Intelligence Radio Transceiver (AIR-T). Specifications are subject to change without notice. For the most recent device specifications, refer to www.deepwavedigital.com.

Block Diagram

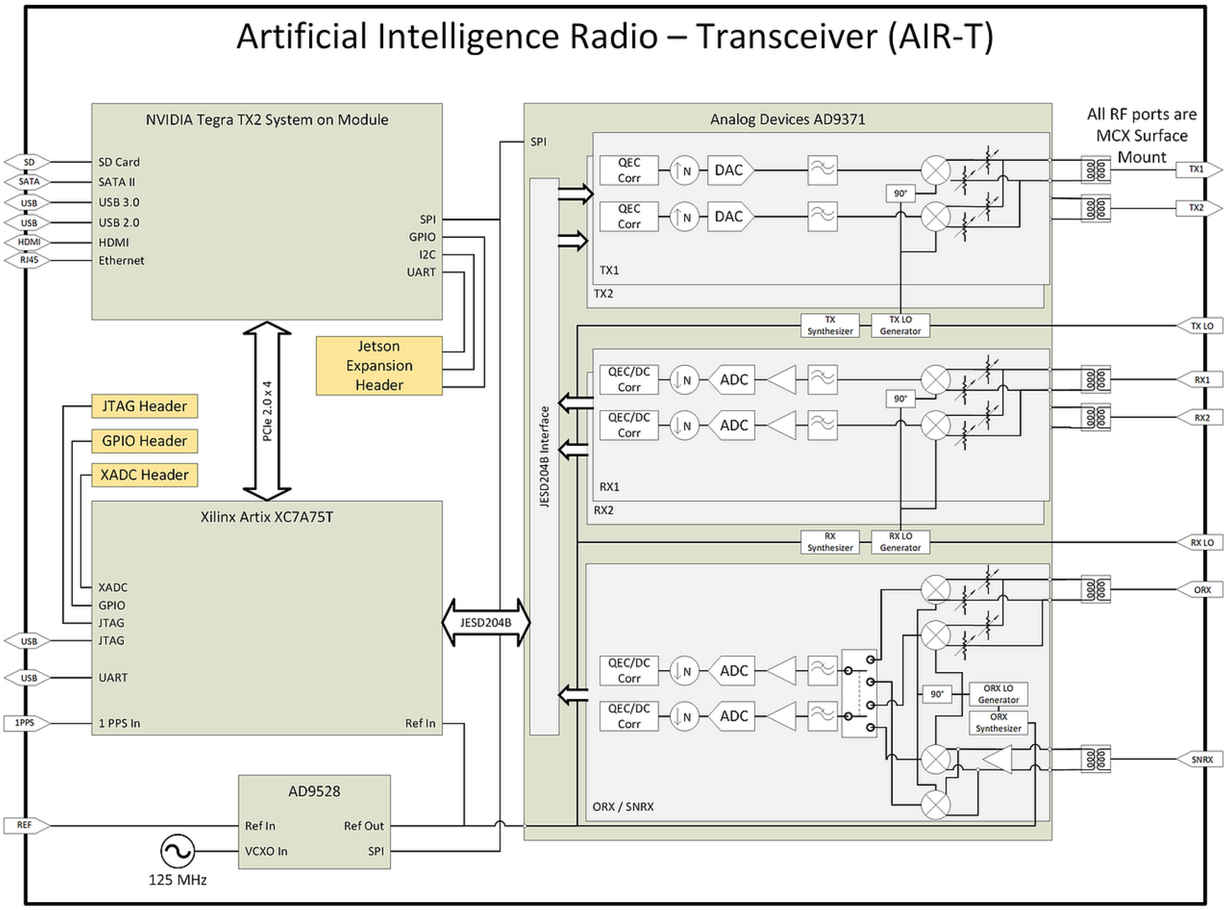

Figure 2: Functional Block Diagram

Processors

The AIR-T enables software defined radio for any signal processing application by utilizing three classes of tightly coupled processors:

- FPGA for strict real-time operations
- GPU for highly parallel processing and machine learning
- CPU for control, I/O, DSP, and software applications

General Purpose Processors

The AIR-T leverages the NVIDIA Jetson TX2 System On Module (SOM) as its General Purpose Processors (GPP). The Jetson TX2 SOM contains two ARM processors (6 cores total), an NVIDIA Pascal GPU (256 cores), and 8 GBytes of memory. The CPUs and GPU all share a common pool of memory, which (along with a unified memory architecture) allows for a zero-copy capability. As illustrated in Figure 3, zero-copy eliminates the host-to-devices (or device-to-host) memory transfer that is required by SDRs with discrete GPUs, such as an SDR connected to an external laptop or computer. Because of this, an SDR with a discrete GPU will have increased latency that is prohibitive for many applications. The AIR-T leverages zero-copy to remove this extra data transfer to enable a wide-range of SDR applications.

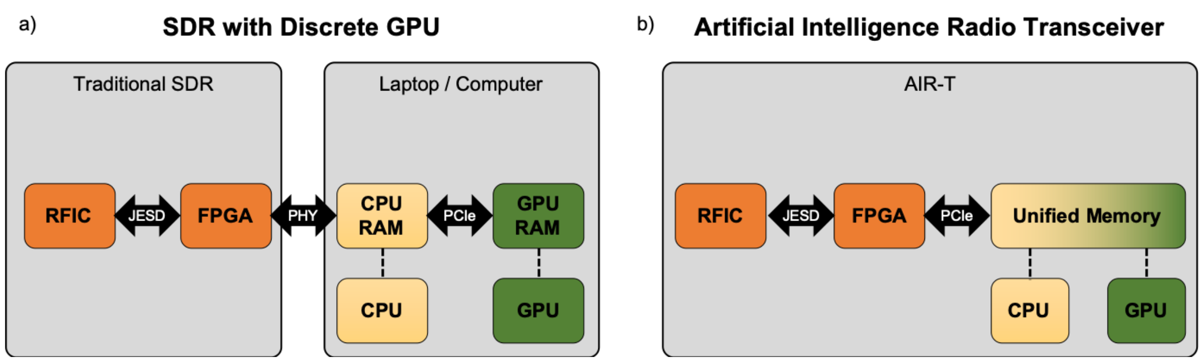

Figure 3: Comparison of a) traditional memory architecture with b) AIR-T unified memory architecture

For additional details on the NVIDIA Jetson TX2 please see that datasheet here: http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet. Some of the information from that datasheet is produced below.

Manufacturer	NVIDIA Corporation
Model	NVIDIA Jetson TX2 (GPU / CPU)
Packaging	System on Module (SOM)
GPU Type	NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
CPU Type	ARMv8 (64-bit) heterogeneous multi-processing CPU architecture with two CPU clusters (6 processor cores) connected by a coherent

	interconnect fabric. ARM® Cortex® -A57 MPCore (Quad-Core) Processor: ● 2.0 GHz ● L1 Cache: 48KB L1 instruction cache (I-cache) per core, 32KB L1 data cache (D-cache) per core ● L2 Unified Cache: 2MB NVIDIA Denver 2 (Dual-Core) Processor: ● 2.0 GHz ● L1 Cache: 128KB L1 instruction cache (I-cache) per core, 64KB L1 data cache (D-cache) per core ● L2 Unified Cache: 2MB
Unified Memory (GPU/CPU shared)	Capacity: 8 GB Type: 128-bit (4ch x 32-bit) LPDDR4 Memory Bus Frequency: 1866 MHz (59.7 GB/s)
Storage Capacity	32GB eMMC 5.1 Flash Storage

Reconfigurable FPGA

The FPGA on the AIR-T comes preloaded with the AirStack firmware to support transmit and receive functionality. Customers may choose to load custom firmware if needed by their application. Details regarding the FPGA on the AIR-T are shown below.

Manufacturer	Xilinx
Family	Artix-7
Model	XC7A75T-2FGG676C
LUTs	47,200
DSP48E1 Slices	180
Embedded Block RAM	3.78 kbits
Default Timebase	62.5 MHz or 125 MHz
Flash Memory (non-volatile)	256Mb

Networking

Ethernet	• 10/100/1000 BASE-T, RJ-45 connector
WLAN	• IEEE 802.11a/b/g/n/ac dual-band 2x2 MIMO • Maximum transfer rate 866.7Mbps
Bluetooth	• Version 4.1 •

External Display

HDMI 2.0a/b	Up to 3840x2160 at 60Hz (4k)

Peripheral Interfaces

NVIDIA Jetson TX2

SATA	Version 3.1
SD Card	SD 3.0 or SD-XC cards up to 2 TB
USB	USB 3.0 Super Speed mode (up to 5Gb/s) USB 2.0 High Speed mode (up to 480Mb/s), USB On-The-Go
UART	See NVIDIA Jetson TX2 datasheet for information
GPIO	See NVIDIA Jetson TX2 datasheet for information
SPI	See NVIDIA Jetson TX2 datasheet for information
I2C	See NVIDIA Jetson TX2 datasheet for information
Audio Input / Output	I2S or Digital. See NVIDIA Jetson TX2 datasheet for information

Xilinx FPGA

JTAG	Programmable via JTAG or Digilent USB to JTAG converter
XADC	Integrated Analog with Digital Customization for the FPGA
Digital I/O	GPIO, SPI
UART	USB to UART bridge

Analog Devices 9371

GPIO	System monitoring and external attenuator control

Transceiver

Radio Frequency Integrated Circuit

The Radio Frequency Integrated Circuit (RFIC) on the AIR-T is the Analog Devices AD9371. For additional details on the AD9371 please see that datasheet here: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9371.pdf

Manufacturer	Analog Devices
Model	AD9371
Frequency Conversion Type	Direct conversion

Receiver Specifications

Number of Channels	2 (LO shared)
Sample Rates	• 125 MSPS • 62.5 MSPS • 31.25 MSPS • 15.625 MSPS

	• 7.8125 MSPS
Maximum Bandwidth	112.5 MHz
Frequency Tuning Range	300 MHz to 6 GHz (no daughter card)
Power Level Control	• Automatic Gain Control (AGC) up to 30 dB attenuation • Manual gain control 0 to 30 dB attenuation in 0.5 dB increments
Maximum Input Power	• -15 dBm (no AGC) • 15 dBm (w/ AGC)
ADC Resolution	16 bits
Built-in Calibrations	• Quadrature Error Correction • DC offset correction
Local Oscillator	Internal (built-in) or external
Auxiliary Channels[1]	Sniffer, Observation

Transmitter Specification

Transmit Channels	2 (LO shared)
Sample Rates	• 125 MSPS • 62.5 MSPS • 31.25 MSPS • 15.625 MSPS • 7.8125 MSPS
Maximum Bandwidth	112.5 MHz
Frequency Tuning Range	300 MHz to 6 GHz (no daughter card)
Power Level Control	• Transmit Power Control (TPC) up to 42 dB attenuation • Manual gain control
Maximum Output Power	+6 dBm
DAC Resolution	14 bits
Built-in Calibrations	• Quadrature Error Correction

[1] The AIR-T supports up to two transmit and two receive channels simultaneously.

	• LO leakage correction
Local Oscillator	Internal (built-in) or external

Internal Reference Clock

Clock distribution part number	AD9528
Oscillator Type	VCXO
Oscillator Model	Crystek Corporation CVHD-950-125M
Oscillator Frequency	125 MHz
Frequency Pull Range	± 20 ppm

External Reference Clock

The AIR-T will phase lock to an external 10 MHz reference signal.

Number of Channels	1
Connector Type	• MCX (Board) • SMA (Enclosure)
Frequency	10 MHz
Input Impedance	50 Ohms
Input Voltage Rating	3.3V CMOS
Absolute Maximum Voltage	3.45 Volts

Power

Input Voltage Range	5-15 VDC
Typical Standby Power Consumption	9.3 W
Recommended Power Supply	80 W, 12 VDC

Physical

Form Factor (no enclosure)	Mini-ITX
Dimensions (no enclosure)	170 × 170 x 35 mm (6.7" × 6.7" x 1.4")
Weight (no enclosure)	0.35 kg (0.8 lbs)

Proper Handling

The AIR-T is a printed circuit board with many exposed conductors. It is essential that no conductive material be left near or in contact with the system.

Best practices include using an anti-static mat and other ESD procedures when handling sensitive electronic equipment, including low humidity and not exposing the radio to liquids.

During the calibration procedure that is run when the radio is initialized, calibration signals may be emitted from both the TX and RX ports. To ensure that connected equipment is not damaged, it is recommended to disconnect the AIR-T from any equipment and terminate with 50 Ohms during the radio initialization.

Software

The AIR-T comes preloaded with a full software stack, called AirStack. AirStack includes all the components necessary to utilize the AIR-T, such as an Ubuntu based operating system, AIR-T specific device drivers, and the FPGA firmware. The operating system is based off of the NVIDIA Jetpack and is upgraded periodically. Please check for the latest software at www.deepwavedigital.com

Application Programming Interfaces

Applications for the AIR-T may be developed using almost any software language, but C/C++ and Python are the primary supported languages. Various Application Programming Interfaces (APIs) are supported by AirStack and a few of the most common APIs are described below.

Hardware Control

SoapyAIRT

SoapySDR is the primary API for interfacing with the AIR-T via the SoapyAIRT driver. SoapySDR is an open-source API and runtime library for interfacing with various SDR devices. The AirStack environment includes the SoapySDR and the SoapyAIRT driver to enable communication with the radio interfaces using Python or C++. The Python code below provides an operational example of how to leverage the SoapyAIRT for SDR applications.

```
#!/usr/bin/env python3
from SoapySDR import Device, SOAPY_SDR_RX, SOAPY_SDR_CS16
import numpy as np
sdr = Device(dict(driver="SoapyAIRT"))          # Create AIR-T instance
sdr.setSampleRate(SOAPY_SDR_RX, 0, 125e6)       # Set sample rate on channel 0
sdr.setGainMode(SOAPY_SDR_RX, 0, True)          # Use AGC on channel 0
sdr.setFrequency(SOAPY_SDR_RX, 0, 2.4e9)        # Set tune frequency on channel 0
buff = np.empty(2 * 16384, np.int16)            # Create memory buffer for data stream
stream = sdr.setupStream(SOAPY_SDR_RX,
                         SOAPY_SDR_CS16, [0])   # Setup data stream
sdr.activateStream(stream)                      # Turn on the radio
for i in range(10):                             # Receive 10x16384 windows of signal
    sr = sdr.readStream(stream, [buff], 16384)  # Read 16384 samples
    rc = sr.ret                                 # Number of samples read or error code
    assert rc == 16384, 'Error code = %d!' % rc # Make sure no errors
    s0 = buff.astype(float) / np.power(2.0, 15) # Interleaved signal data bw (-1, 1)
    s = s0[::2] + 1j*s0[1::2]                   # Complex signal data
    # <Insert code here that operates on s>
sdr.deactivateStream(stream)                    # Stop streaming samples
sdr.closeStream(stream)                         # Turn off radio
```

UHD

A key feature of SoapySDR is its ability to translate to/from other popular SDR APIs, such as UHD. The SoapyUHD plugin is included with AirStack and enables developers to create applications using UHD or execute existing UHD-based applications on the AIR-T. This interface is described in Figure 4.

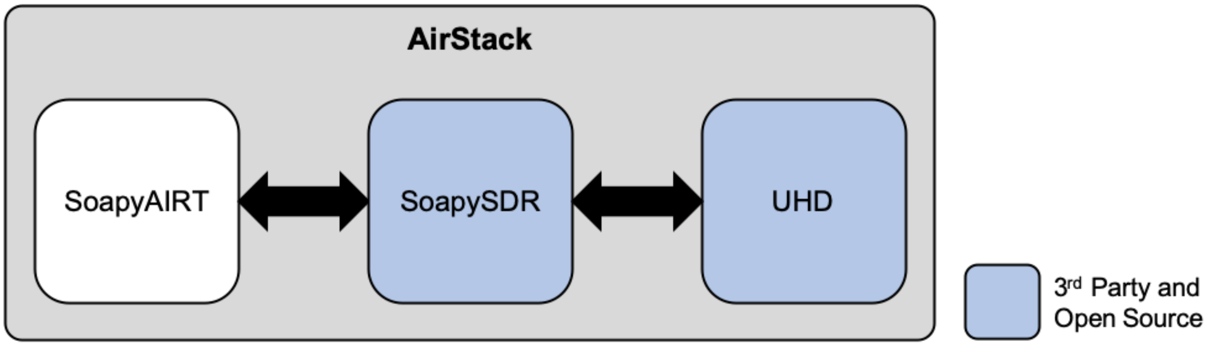

Figure 4: UHD Support Overview

Signal Processing

Python Interfaces

Figure 5 illustrates supported Python APIs that can be used to develop signal processing applications on both the CPU and GPU of the AIR-T. In general, these have been selected because they have modest overhead compared to native code and are well suited to rapid prototyping. In addition, C++ interfaces are provided for many control and processing interfaces to the AIR-T for use in performance-critical applications.

The table below outlines the common data processing APIs that are natively supported by AirStack, along with the supported GPP for each API. Some of these are included with AirStack, while some are available via the associated URL.

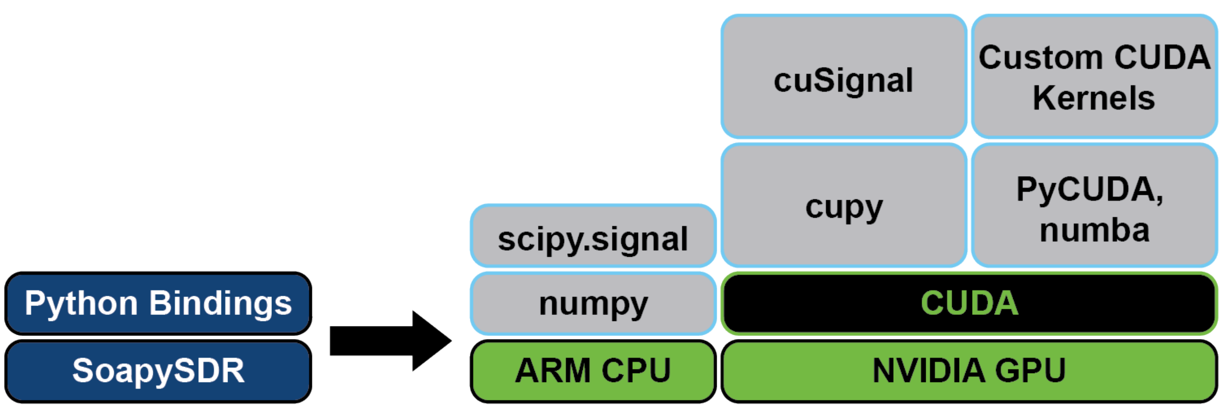

Figure 5: Python Software Suite for DSP on the AIR-T

API	GPP	Description
numpy	CPU	numpy is one a common data analysis and processing Python module. URL: https://numpy.org/
scipy.signal	CPU	SciPy is a scientific computing library for Python that contains a signal processing library, scipy.signal. URL: https://docs.scipy.org/doc/scipy/reference/signal.html
cupy	GPU	Open-source matrix library accelerated with NVIDIA CUDA that is semantically compatible with numpy. URL: https://cupy.chainer.org/
cuSignal	GPU	Open-source signal processing library accelerated with NVIDIA CUDA based on scipy.signal. URL: https://github.com/rapidsai/cusignal
PyCUDA / numba	GPU	Python access to the full power of NVIDIA's CUDA API URL: https://documen.tician.de/pycuda/
Custom CUDA Kernels	GPU	Custom CUDA kernels may be developed and executed on the AIR-T

GNU Radio

The AIR-T also supports GNU Radio, one of the most widely used open-source toolkits for signal processing and SDR. Included with AirStack, the toolkit provides modules for the instantiation of bidirectional data streams with the AIR-T's transceiver (transmit and receive) and multiple DSP modules in a single framework. GNU Radio Companion may also be leveraged for a graphical programming interface, as shown in Figure 6. GNU Radio is written in C++ and has Python bindings.

Like the majority of SDR applications, most functions in GNU Radio rely on CPU processing. Since many DSP engineers are already familiar with GNU Radio, two free and open source modules have been created for AirStack to provide GPU acceleration on the AIR-T from within GNU Radio. Gr-cuda and gr-wavelearner, along with the primary GNU Radio modules for sending and receiving samples to and from the AIR-T, are shown in the table below and included with AirStack.

gr-cuda	A detailed tutorial for incorporating CUDA kernels into GNU Radio. URL: https://github.com/deepwavedigital/gr-cuda
gr-wavelearner	A framework for running both GPU-based FFTs and neural network inference in GNU Radio. URL: https://github.com/deepwavedigital/gr-wavelearner
gr-uhd	The GNU Radio module for supporting UHD devices. URL: https://github.com/gnuradio/gnuradio/tree/master/gr-uhd
gr-soapy	Vendor neutral set of source/sink blocks for GNU Radio. URL: https://gitlab.com/librespacefoundation/gr-soapy

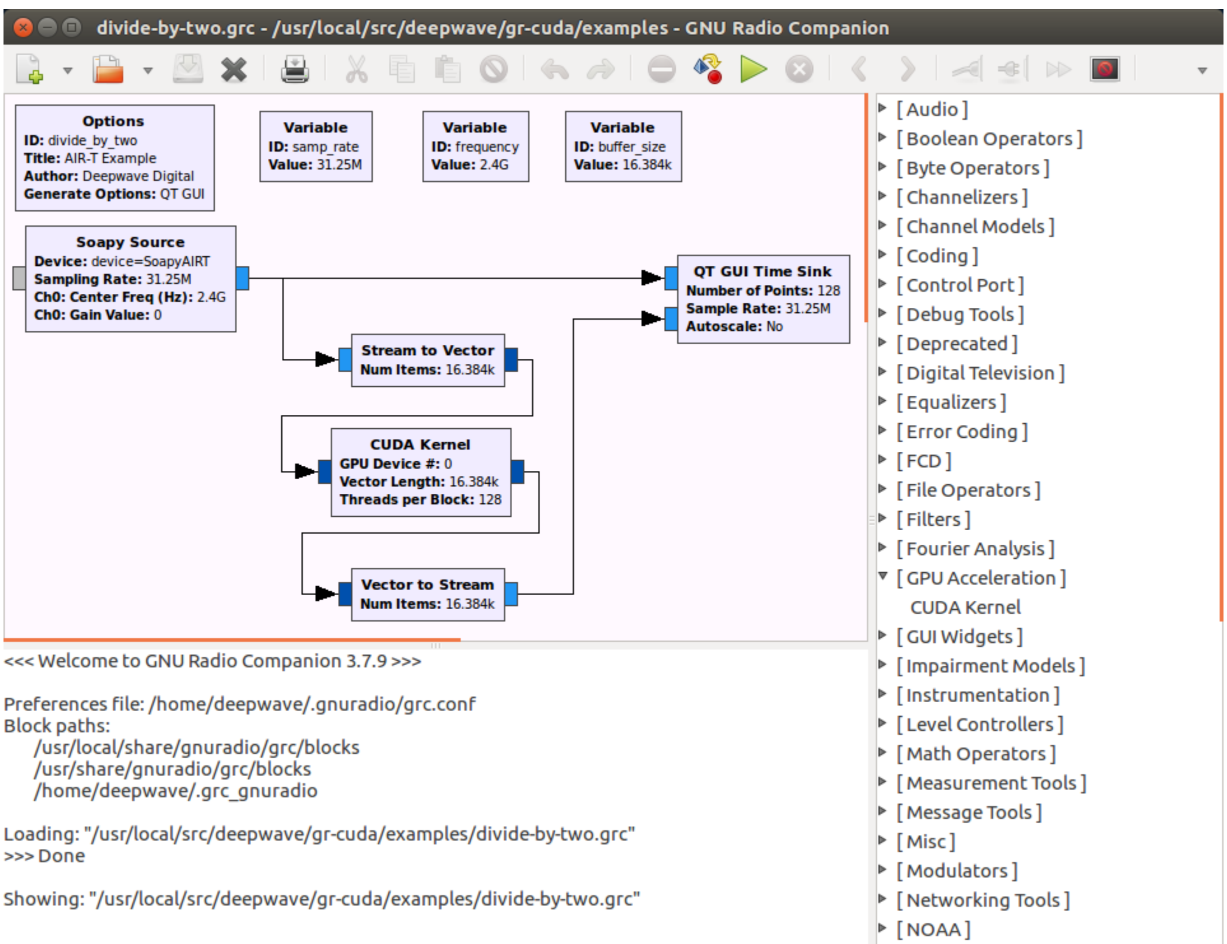

Figure 6: GNU Radio Companion GUI executing a CUDA kernel

Deep Learning

The workflow for creating a deep learning application for the AIR-T consists of three phases: training, optimization, and deployment. These steps are illustrated in Figure 7 and covered in the sections below.

AirPack is an add-on software package (not included with the AIR-T) that provides source code for the complete training-to-deployment workflow described in this section. More information about AirPack may be found here: https://deepwavedigital.com/airpack/.

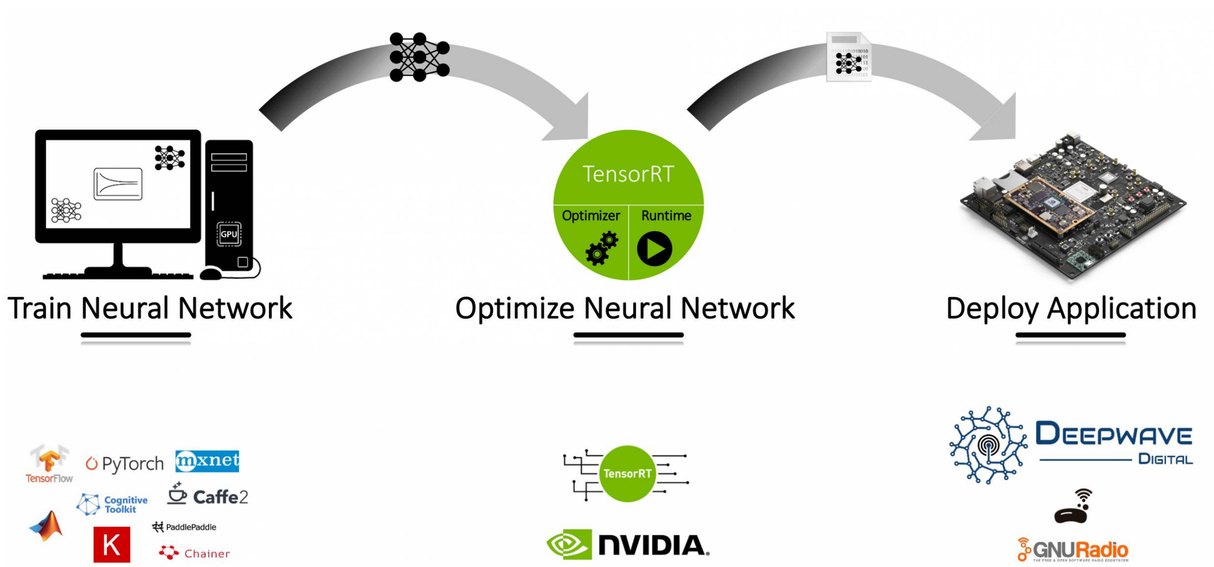

Figure 7: Deep learning training-to-deployment workflow for the AIR-T

Training Frameworks

The primary inference library used on the AIR-T is NVIDIA's TensorRT. TensorRT allows for optimized interference to run on the AIR-T's GPU. TensorRT is compatible with models trained using a wide variety of frameworks as shown below.

Deep Learning Framework	Description	TensorRT Support	Programming Languages
TensorFlow	Google's deep learning framework	UFF,	Python, C++,

	URL: www.tensorflow.org/	ONXX	Java
PyTorch	Open source deep learning framework maintained by Facebook URL: www.pytorch.org/	ONNX	Python, C++
MATLAB	MATLAB has a Statistics and Machine Learning Toolbox and a Deep Learning Toolbox URL:www.mathworks.com/solutions/deep-learning.html	ONNX	MATLAB
CNTK	Microsoft's open source Cognitive Toolkit. URL: docs.microsoft.com/cognitive-toolkit/	ONNX	Python, C#, C++

When training a neural network for execution on the AIR-T, make sure that the layers being used are supported by your version of TensorRT. To determine what version of TensorRT is installed on your AIR-T, open a terminal and run:

```
$ dpkg -l | grep TensorRT
```

The supported layers may be found in the TensorRT SDK Documentation under the TensorRT Support Matrix section.

Optimization Frameworks

Once a model is trained (and saved in the file formats listed in the table above), it must be optimized to run efficiently on the AIR-T. The primary function of the AIR-T is to be an edge-compute inference engine for the real-time execution of deep learning applications. This section discusses the supported framework(s) for optimizing a DNN for deployment on the AIR-T.

TensorRT

The primary method for executing a deep learning algorithm on the AIR-T's GPU is to use NVIDIA's TensorRT inference accelerator software. This software will convert a trained neural network into a series of GPU operations, known as an inference engine. The engine can then be saved and used for repeated inference operations.

Deployment on the AIR-T

The procedure for deploying a trained neural network is outlined in Figure 7, where a deep neural network (DNN) is used. After the neural network has been optimized, the resulting inference engine is loaded into a user's inference application using either the C++ or Python