MATLAB EVAL-AD4080ARDZ guide

This document is a guide for getting from the base MATLAB installation all the way to measuring samples from the EVAL-AD4080ARDZ board. The overall steps will be:

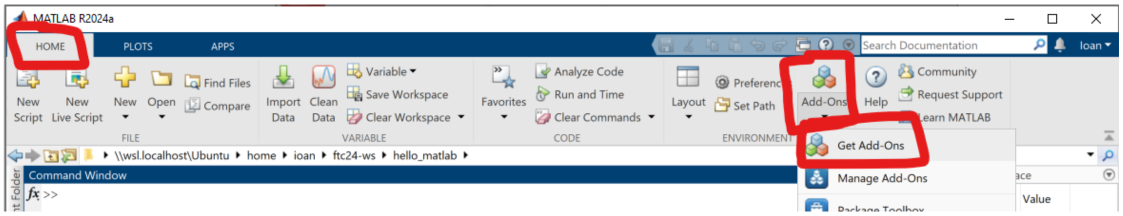
1. Installing the Precision Toolbox (and all of its dependencies)
2. Setting up iio context forwarding because the MATLAB bindings don't support serial contexts
3. A very short MATLAB program that connects to the board and reads samples

Install the Precision Toolbox

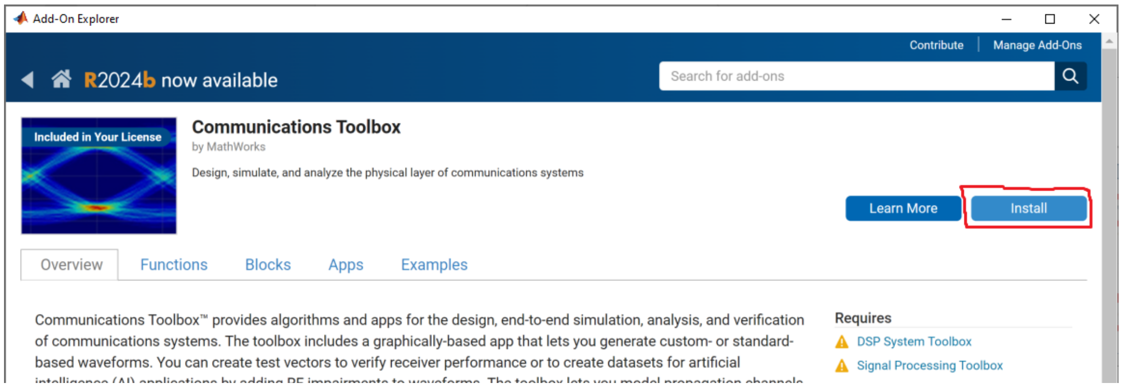
The Precision Toolbox depends on libiio bindings which further depend on the Communications toolbox. These dependencies are not automatically installed by MATLAB and must be installed manually, in reverse order, as follows.

1. Install Communications Toolbox

Open the MATLAB Add-on Explorer:



In the add-on explorer, find and install the Communications Toolbox:



Communications Toolbox
by MathWorks
Design, simulate, and analyze the physical layer of communications systems

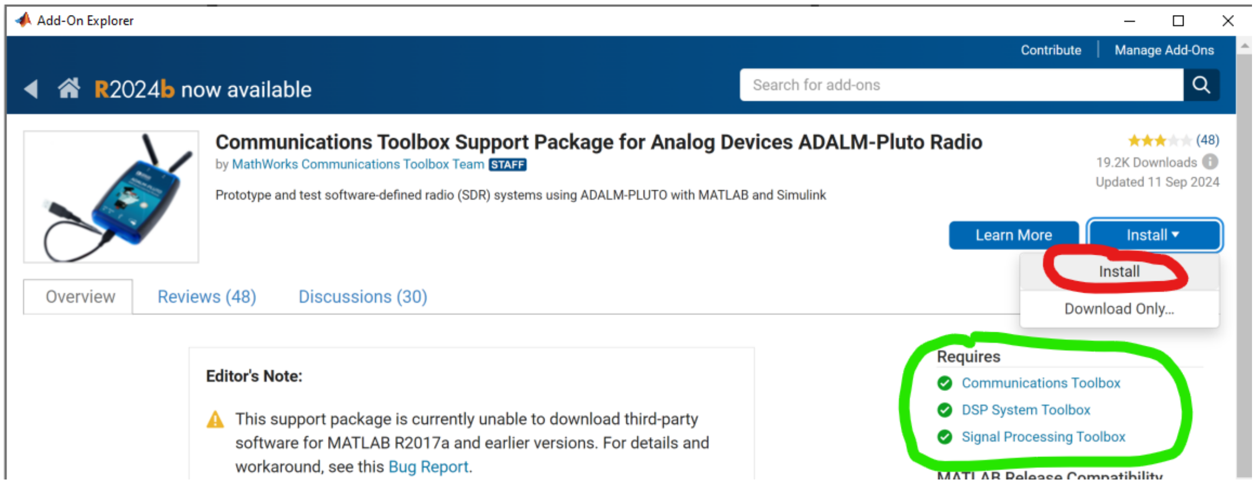
Communications Toolbox™ provides algorithms and apps for the design, end-to-end simulation, analysis, and verification of communications systems. The toolbox includes a graphically-based app that lets you generate custom- or standard-based waveforms. You can create test vectors to verify receiver performance or to create datasets for artificial

Requires
DSP System Toolbox
Signal Processing Toolbox

This will also install its dependencies: **DSP System Toolbox**, **Signal Processing Toolbox**, if they are not already installed. As of October 2024, the download size is about 450MB, and the install size is around 1.3GB.

2. Install libiio bindings

The libiio MATLAB bindings can be installed through the ADALM-Pluto support package or the Xilinx Zynq-Based Radio support package, even though we will not be using the Pluto or Xynq-based radios themselves. The following screenshots use the Pluto package, but the process should basically be the same, shall you choose the other variant.



Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio
by MathWorks Communications Toolbox Team STAFF

(48)
19.2K Downloads
Updated 11 Sep 2024

Prototype and test software-defined radio (SDR) systems using ADALM-PLUTO with MATLAB and Simulink

Editor's Note:

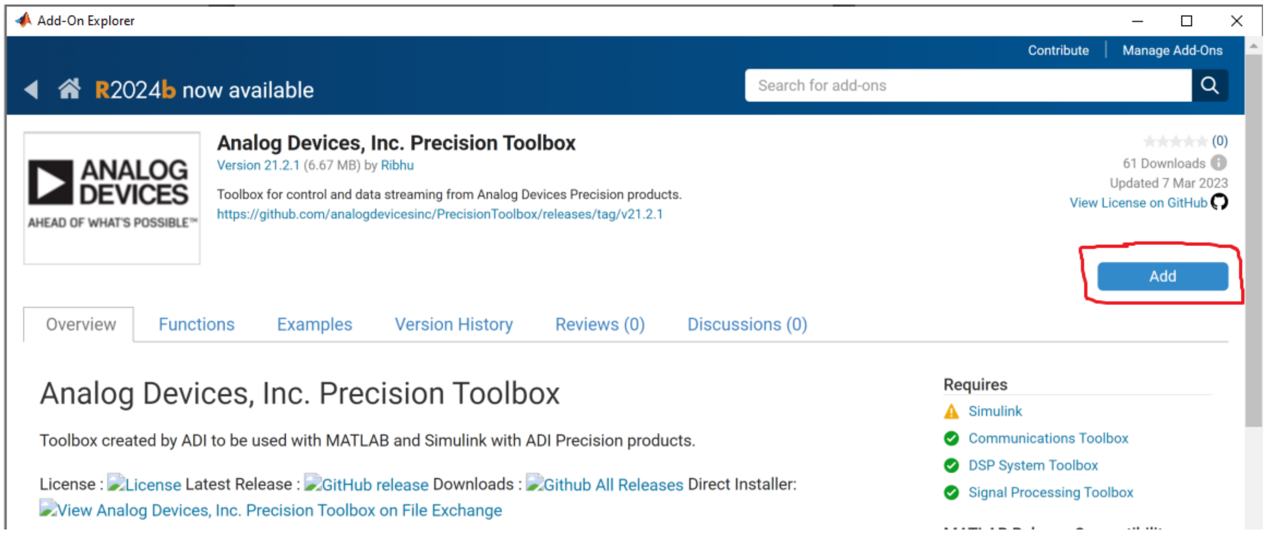
This support package is currently unable to download third-party software for MATLAB R2017a and earlier versions. For details and workaround, see this Bug Report.

Requires
- Communications Toolbox
- DSP System Toolbox
- Signal Processing Toolbox

Notice how all three required toolboxes are checked.

3. Install Precision Toolbox

In the MATLAB Add-on Explorer, look up and install the **Analog Devices, Inc. Precision Toolbox**:



Analog Devices, Inc. Precision Toolbox
Version 21.2.1 (6.67 MB) by Ribhu

(0)
61 Downloads
Updated 7 Mar 2023
View License on GitHub

Toolbox for control and data streaming from Analog Devices Precision products.
https://github.com/analogdevicesinc/PrecisionToolbox/releases/tag/v21.2.1

Analog Devices, Inc. Precision Toolbox

Toolbox created by ADI to be used with MATLAB and Simulink with ADI Precision products.

License : License Latest Release : GitHub release Downloads : Github All Releases Direct Installer:
View Analog Devices, Inc. Precision Toolbox on File Exchange

Requires
- Simulink
- Communications Toolbox
- DSP System Toolbox
- Signal Processing Toolbox

As of october 2024, installation takes less than 10 seconds.

Forward the iio serial context

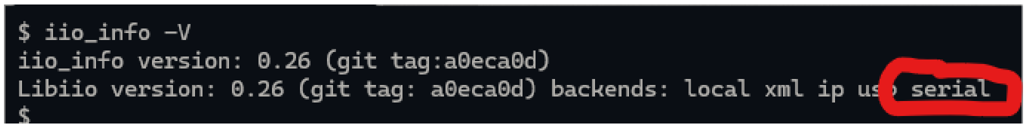
Because the MATLAB libiio bindings do not yet support the serial backend, we must use iiod as a middleman. This will require using a linux-based system, which may be: a Raspberry Pi, a Windows Subsystem for Linux instance, a Virtual Machine running a Linux distro, etc.

Install libiio

Download and install the latest libiio build from https://github.com/analogdevicesinc/libiio/releases

For Ubuntu / Debian-based distributions, after downloading the `` `libiio-....deb` `` file, double click it to install using a GUI or run the following command: `` `sudo dpkg -i libiio-....deb` ``

After installing, run `` `iio_info -V` `` and check that the installed libiio supports the serial backend:



```
$ iio_info -V
iio_info version: 0.26 (git tag:a0eca0d)
Libiio version: 0.26 (git tag: a0eca0d) backends: local xml ip usb serial
$
```

Run iiod to forward the serial context over ip

Run the following command (or a variation thereof) to use iiod to forward the `` `serial:/dev/ttyACM0,230400,8n1` `` to port 50905:

```
$ iiod -u serial:/dev/ttyACM0,230400,8n1 -p 50905
```

If you need to forward a different context (e.g. in case the device shows up as something different from `` `/dev/ttyACM0` ``, or if you want to forward another device), just change the `` `-u ...` `` parameter.

On WSL, it might complain about avahi, but that's safe to ignore. Leave this process running.

Get board class for the EVAL-AD4080ARDZ

Many Analog Devices precision ADC boards are already supported by the Precision Toolbox, but the EVAL-AD4080ARDZ isn't. We provide a **minimal** implementation of the `` `adi.AD4080.Rx` `` class at: https://github.com/cristina-suteu/ftc24-ws/tree/main/hello_matlab/+adi. Download the `` `+adi` `` folder and copy it to your working folder (or just work within this repository).

Your directory structure should look like:

```
matlab_current_folder
├─ +adi
|   ├─ +AD4080
|       ├─ Rx.m
├─ your_script.m
├─ ...
```

Connect to the board and read samples using MATLAB

Finally! The following code should connect to the EVAL-AD4080ARDZ (via the forwarded context), read a 16k window of samples, convert them to volts, and plot:

```
rx = adi.AD4080.Rx;
rx.uri = 'ip:172.18.228.187:50905'; % Change this to the IP address on
                                    % which iiod is listening
rx.SamplesPerFrame = 16384;
rx.EnabledChannels = [1];

data = rx();
data = data * rx.Scale / 1e3; % Scale is in mV

plot(data);

% Delete the system object
release(rx);
```

And there you have it! Once measured, `` `data` `` is just an array of samples, which you can now process whichever way you like!