Pumpkin i350 EVK: Getting Started with AWS IoT Greengrass V2 Guide

Table of Contents

1 Document Information

1.1 Glossary

EVK	Evaluation Kit
SoC	System on Chip
IoT	Internet of Things
APU	AI Processing Unit
DSP	Digital Signal Processor

1.2 Revision History (Version, Date, Description of change)

Version	Date	Description of Change
1.0	09/12/2022	Initial version of documentation

2 Overview

The Pumpkin i350 EVK is designed by OLogic and powered by MediaTek Genio 350 (MT8365 SoC).

The Pumpkin i350 EVK is an Edge AI platform designed for mainstream AI + IoT applications that require vision and voice edge processing, such as facial, object, gesture, motion recognition, LPR, voice activation and speed recognition, sound isolation, biotech and biometric measurements, and more.

Built on the success of our high-end Pumpkin i500, the new Pumpkin i350 is a lower cost, mid-range performance platform that enables engineering teams to design for wider market adoption use cases such as door entry systems, personal gym trainers, child sleep companion robots, and much more.

Built on an ultra-efficient 14nm process, the Pumpkin i350 is a highly integrated Edge AI platform incorporates a dedicated APU (AI processor) and DSP to enable vision and voice edge AI, with considerably greater performance and power efficiency in common applications.

With the advent of higher-performing AI-integrated chips, such as the MediaTek Genio 350 SoC and development board designed by OLogic, use the Pumpkin i350 to accelerate your engineering, shorten your development cycle, helping you to launch your new product faster at a lower development cost and faster ROI.

2.1 About AWS IoT GreengrassV2

To learn more about AWS IoT GreengrassV2, see how it works and what's new.

3 Hardware Description

3.1 DataSheet

The board reference guide for Pumpkin i350 EVK can be found here.
Genio 350 (MT8365) AIoT Application Processor Datasheet can be found here.

3.2 Standard Kit Contents

Hardware package:

- *The hardware package contains the Pumpkin i350 EVK preinstalled with a Yocto image.*

Note: No cables are provided.

Cables needed:

- *1x USB-C cable (data transfer capable)*
- *1x micro-USB (UART debug connection)*
- *1x USB-C cable (5V 3A Power supply)*

Additional bring-up information can be found here.

3.3 Other purchasable items

Other purchasable products include:

- *Raw camera adapter with AR0330 camera module*
- *Pumpkin ISP Camera Adapter with AR0330 camera module*
- *AR0330 camera module*
- *AR0144 IAS Module*

4 Set up your Development Environment

4.1 Tools Installation (IDEs, Toolchains, SDKs)

To set up your development environment, you need to:
1) Setup Gitlab account.
2) Setup Build Environment (Linux).
3) Setup Tool Environment (Windows or Linux).

Information pertaining to the above-mentioned setups can be found here.

These tools will help you create a Yocto (custom Linux) image.

4.2 Additional Software References

The complete developer guide can be found here.

5 Set up your hardware

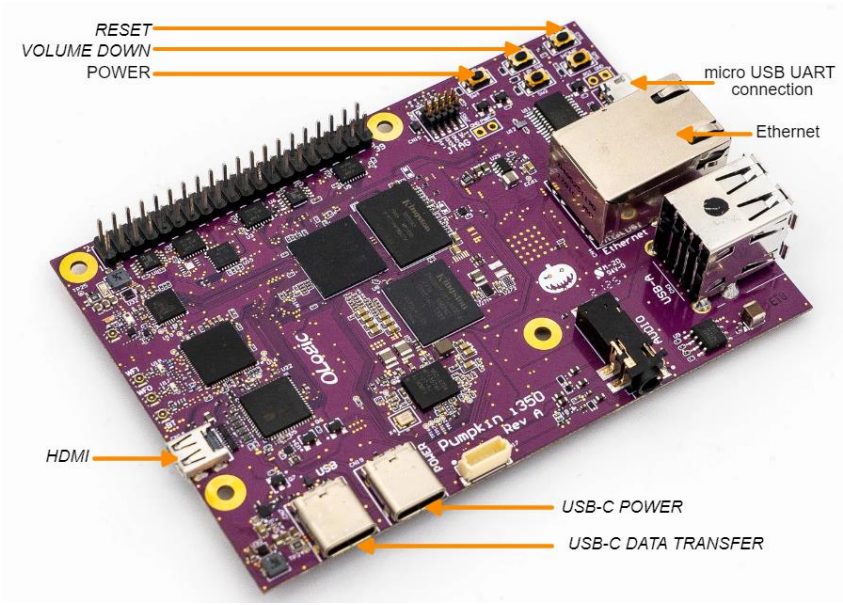

Figure 1: Pumpkin i350 EVK

5.1 Power:

Connect a USB-C cable to the "USB-C POWER" port using a 5V 3A USB Charger.
To power on the EVK,

- ➔ *Long press the "POWER"*
- ➔ *Or connect a data-transfer capable USB-C cable to the "USB-C DATA TRANSFER" port.*

You should see a blue light once the EVK has been powered up.

5.2 Serial (Debug) connection:

Connect a micro-USB cable to the "micro-USB UART connection" port.
After installing the necessary UART drivers from the Setup Tool Environment section (4.1), establish the serial connection using:

- *Linux:*
  picocom -b 921600 /dev/ttyUSB0

- *Windows:*
  Check connected COM port from "Device Manager" on Windows.

Set the baud rate to '921600'.

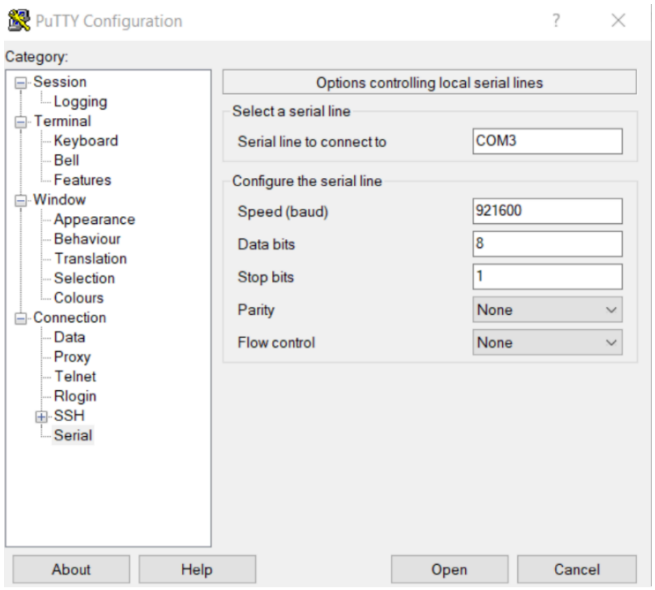

Figure 2: PUTTY setup

6 Setup your AWS account and Permissions

Refer to the instructions at Set up your AWS Account. Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account and
- Create a user and grant permissions.
- Open the AWS IoT console

Pay special attention to the Notes.

7 Create Resources in AWS IoT

Refer to the instructions at Create AWS IoT Resources. Follow the steps outlined in these sections to provision resources for your device:

- Create an AWS IoT Policy
- Create a thing object

Pay special attention to the Notes.

8 Install the AWS Command Line Interface

To install the AWS CLI on your host machine, refer to the instructions at Installing the AWS CLI v2. Installing the CLI is needed to complete the instructions in this guide.

Once you have installed AWS CLI, configure it as per the instructions in this online guide. Set the appropriate values for Access key ID, Secret access key, and AWS Region. You can set Output format to "json" if you prefer.

9 Build a linux image with Greengrass pre-requisites

9.1 Build the AIoT Yocto default image:

Follow the instructions from here to build an AIoT Yocto Linux image.
NOTE: Use "MACHINE=i350-pumpkin" for Pumpkin i350 EVK while building the image.

9.2 Add meta-aws, greengrass v2 and other useful packages.

Adding meta-aws layer:

```
cd aiot-yocto/src
git clone https://github.com/aws4embeddedlinux/meta-aws.git -b dunfell (or th
e version of your yocto setup)
cd ~/aiot-yocto
source src/poky/oe-init-build-env
bitbake-layers add-layer "/home/aiot-yocto/src/meta-aws" (absolute path to me
ta-aws layer)
bitbake-layers show-layers (confirm that you see meta-aws)
```

Add packages to ~/aiot-yocto/build/conf/local.conf

Add the following text to the end of local.conf:

```
IMAGE_INSTALL:append = " nano git packagegroup-core-buildessential cmake make
gcc g++ "
########################################################
# Add time sync features for greengrass
DISTRO_FEATURES:append = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED += "sysvinit"
VIRTUAL-RUNTIME_init_manager += "systemd"
VIRTUAL-RUNTIME_initscripts = ""
########################################################
IMAGE_INSTALL:append = " greengrass-bin-demo "
```

9.3 Verify that Java is available

Insert the SD card into the device.
Power on
Once the system has booted, verify that java is available using the command:

```
java --version
```

10 Install AWS IoT Greengrass

10.1 Download the AWS IoT Greengrass Core software

If Greengrass has not been included in the SD card image, you can download the latest greengrass core software as follows:

```
wget https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip
```

10.2 Install the AWS IoT Greengrass Core software

Unzip the AWS IoT Greengrass Core software to a folder on your device. Replace GGCoreInstall with the folder that you want to use

```
unzip greengrass-nucleus-latest.zip -d GGCoreInstall
rm greengrass-nucleus-latest.zip
```

Verify the version of the AWS IoT Greengrass Core software:

```
java -jar ./GGCoreInstall/lib/Greengrass.jar --version
```

You will see the Greengrass version displayed - similar to:

AWS Greengrass v2.4.0

10.2.1 Provide your credentials

Run the following commands to provide the credentials to the AWS IoT Greengrass Core software.

```
export AWS_ACCESS_KEY_ID=<the access key id for your account>
export AWS_SECRET_ACCESS_KEY=<the secret access key for your account>
```

10.2.2 Run the installer

Run the installer as shown below. Modify the values as per your region, install directory and thing name.

Use the **--provision true** option to have the installer set up the "thing" and required policies for you. If you prefer to configure Greengrass manually, see the online guide.

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
-jar ./GGCoreInstall/lib/Greengrass.jar \
--aws-region us-west-2 \
--thing-name thing-name \
--tes-role-name GreengrassV2TokenExchangeRole \
--tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
--component-default-user ggc_user:ggc_group \
--provision true \
--setup-system-service true \
--deploy-dev-tools true
```

If all goes well, you will see the following output on the device console:

Successfully configured Nucleus with provisioned resource details!
Configured Nucleus to deploy aws.greengrass.Cli component
Successfully set up Nucleus as a system service

The local development tools (specified by the --deploy-dev-tools option) take some time to deploy. The following command can be used to check the status of this deployment:

```
aws greengrassv2 list-effective-deployments --core-device-thing-name thing-name
```

When the status is SUCCEEDED, run the following command to verify that the Greengrass CLI is installed and runs on your device. Replace /greengrass/v2 with the path to the base folder on your device as needed.

```
/greengrass/v2/bin/greengrass-cli help
```

11 Create a Hello World component

In Greengrass v2, components can be created on the edge device and uploaded to the cloud, or vice versa.

11.1 Create the component on your edge device

Follow the instructions online under the section To create a Hello World component to create, deploy, test, update and manage a simple component on your device.

11.2 Upload the Hello World component

Follow the instructions online at Upload your component to upload your component to the cloud, where it can be deployed to other devices as needed.

12 Debugging

12.1 Debug via Serial connection:

Follow the instructions from section 5.2 to establish a serial connection.

12.1.1 Baud rate 921600

Select Baud rate as 921600 to debug the EVK's normal environment. The boot up log will look like this:

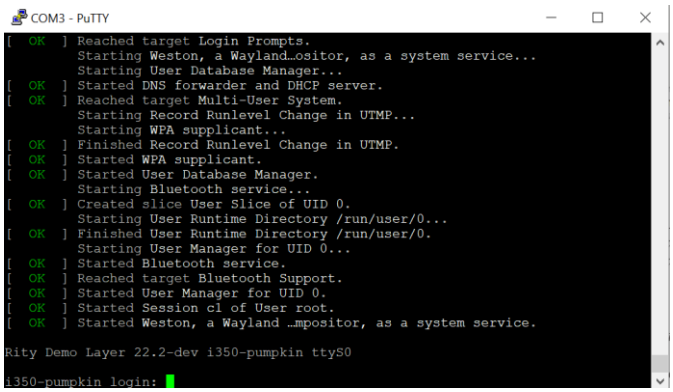

Figure 3: serial connection

12.1.2 Baud rate 115200

Select Baud rate as 115200 to debug the EVK's boot up environment.

12.2 Kernel debugging

12.2.1 dmesg

Use the command 'dmesg' to debug the linux kernel logs.
Set the kernel debug level using "-n". For example, for maximum verbosity, use "dmesg -n 8".

12.2.2 printk

To enable printk logs,

1) *Enable* `CONFIG_DYNAMIC_DEBUG` *in the linux kernel.*

```
devtool menuconfig linux-mtk
```

More information pertaining to menuconfig can be found here.

2) *Set printk verbosity level:*
   "echo 8 > /proc/sys/kernel/printk"

3) *Select files where printk messages need to be shown:*
   "echo 'file <driver file name> +p'>/sys/kernel/debug/dynamic_debug/control"

The printk logs will then be visible in "dmesg" output.

13 Troubleshooting

Troubleshooting tips for resolving common or potential problems can be found here.