Getting Started

Requirement

• 1 x USB Type-C® cable with data transfer function (to connect your PC to the board's data port)
• 1 x 12~19V power supply*
• 1 x Monitor with HDMI cable
• 1 x Keyboard and Mouse set
* The power supply is purchased separately.
* The factory settings only includes a U-Boot bootloader and does not include a system

Before you begin the flashing procedure, please ensure of the following:
• The board is completely powered off, and the power cord and cables connecting the board to your computer are all disconnected.
• The boot mode switches are set to eMMC mode, please refer to the table and illustration below for the eMMC mode setting.

Boot mode	Switch 1	Switch 2	Switch 3	Switch 4
eMMC	ON	OFF	OFF	OFF

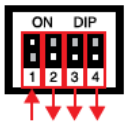


Image list
https://www.asus.com/AIoT-Industrial-Solutions/Tinker-Edge-T/HelpDesk_Download/

Flashing the Tinker Edge T

Initiating Fastboot mode:

I. Connect the USB Type-C® cable to the USB Type-C® ports on the Tinker Edge T and your host computer.
II. Power on the board, you should automatically be booted into Fastboot mode.
➢ Please note that you will only be booted into Fastboot mode when booting up the Tinker Edge T for the first time.
➢ Please refer to the readme file in the unzipped folder for details on other items such as re-flash or recovery.

Executing the flash script:

I. Download the OS image from the Tinker Edge T website, then unzip the image file.
II. Run the flash script or command file to start the flash process. The flash process should take a few minutes. Once the flash is completed, the Tinker Edge T will reboot and you should be booted to the terminal prompt.

➢ Please refer to the readme file in the unzipped folder for troubleshooting steps.

Customize Settings

- **Change Keyboard Layout**
  The keyboard layout is set to English (US) as default setting. Refer to the process below to change the language.

```
sudo dpkg-reconfigure keyboard-configuration
sudo reboot
```

- **Change Time Zone/Date/Time**
  Use Timedatectl built-in tool in OS to change Time.
  - **Time Zone**
    Print Time Zone list.

```
timedatectl list-timezones
```

    Once you identify which time zone is accurate to your location, run the following command as sudo user:

```
sudo timedatectl set-timezone your_time_zone
```

    For example, to change the system's timezone to `Europe/Ljubljana` you would run:

```
sudo timedatectl set-timezone Europe/Ljubljana
```

  - **Date/Time**
    Enable NTP

```
sudo timedatectl set-ntp yes
```

    Disable NTP

```
sudo timedatectl set-ntp no
```

    Set Time (need disable NTP)

```
sudo timedatectl set-time "2016-11-12"
sudo timedatectl set-time "18:10:40"
sudo timedatectl set-time "2016-11-12 18:10:40"
```

  - **Verify**
    Print to verify the change by issuing the `timedatectl` command:

```
timedatectl
```

    Example:

```
      Local time: Mon 2019-03-11 22:51:27 CET
  Universal time: Mon 2019-03-11 21:51:27 UTC
        RTC time: Mon 2019-03-11 21:51:26
       Time zone: Europe/Ljubljana (CET, +0100)
Network time on: yes
NTP synchronized: yes
RTC in local TZ: no
```

- **Check Screen's Resolution**

By default, the output is locked at a resolution of 1920 x 1080. Change this setting by editing file at `/etc/xdg/weston/weston.ini`. In the `[output]` section, edit the line `mode=1920x1080` to be a resolution of your choice.

- **Check Audio's Output Interface**
  - **Output Devices:**

Output Device	Description
imx-audio-hdmi	HDMI Audio

- **Check Internet connection**
  - **Ethernet**
    1. Connect an Ethernet cable to the board.
    2. Use the following command to check detailed connection information.

    ```
    Ifconfig eth0
    ```

  - **Wi-Fi**

    Select a Wi-Fi network by running the following command in the device shell:

    ```
    nmtui
    ```

    Then select `Activate a connection` and select a network from the list under `Wi-Fi (wlan0)`.

    Alternatively, use the following command to connect to a known network name:

    ```
    nmcli dev wifi connect <NETWORK_NAME> password <PASSWORD> ifname wlan0
    ```

    Verify your connection with this command:

    ```
    nmcli connection show
    ```

    You should see your selected network listed in the output. For example:

    ```
    NAME                UUID                                  TYPE             DEVICE
    MyNetworkName       61f5d6b2-5f52-4256-83ae-7f148546575a  802-11-wireless  wlan0
    ```

GPIO

Following table shows the header pinout, including the sysfs paths for each port, which is often the name required when using the periphery library. You can also see the header pinout from the command line by typing `pinout` on the Tinker Edge T.

Note: All I/O pins have a 90k pull-down resistor inside the iMX8M SoC that is used by default during bootup, except for the I2C pins, which instead have a pull-up to 3.3V on the SoM.

Caution: Do not connect a device that draws more than ~ 82 mA of power or you will brownout the system.

sysfs path	Pin function	Pin		Pin function	sysfs path
	+3.3V Power	1	2	+5V Power	
/dev/i2c-1	I2C 2 (SDA)	3	4	+5V Power	
/dev/i2c-1	I2C 2 (SCL)	5	6	Ground	
/dev/ttymxc2	UART 3 (TXD)	7	8	UART 1 (TXD)	/dev/ttymxc0
	Ground	9	10	UART 1 (RXD)	/dev/ttymxc0
/dev/ttymxc2	UART 3 (RXD)	11	12	SAI 1 (TXC)	
/sys/class/gpio/gpio6	GPIO 6	13	14	Ground	
/sys/class/pwm/pwmchip2/pwm0	PWM 4	15	16	GPIO 73	/sys/class/gpio/gpio73
	+3.3V Power	17	18	GPIO 138	/sys/class/gpio/gpio138
/dev/spidev32766	SPI 1 (MOSI)	19	20	Ground	
/dev/spidev32766	SPI 1 (MISO)	21	22	GPIO 140	/sys/class/gpio/gpio140
/dev/spidev32766	SPI 1 (SCLK)	23	24	SPI 1 (SS0)	/dev/spidev32766.0
	Ground	25	26	SPI 1 (SS1)	/dev/spidev32766.1
/dev/i2c-2	I2C 3 (SDA)	27	28	I2C 3 (SCL)	/dev/i2c-2
/sys/class/gpio/gpio7	GPIO 7	29	30	Ground	
/sys/class/gpio/gpio8	GPIO 8	31	32	PWM 1	/sys/class/pwm/pwmchip0/pwm0
/sys/class/pwm/pwmchip1/pwm0	PWM 2	33	34	Ground	
	SAI 1 (TXFS)	35	36	GPIO 141	/sys/class/gpio/gpio141
/sys/class/gpio/gpio77	GPIO 77	37	38	SAI 1 (RXD)	
	Ground	39	40	SAI 1 (TXD)	

Warning: Use caution when handling the GPIO pins to avoid electrostatic discharge or contact with conductive materials (metals). Failure to properly handle the Tinker Edge T can result in a short circuit, electric shock, serious injury, death, fire, or damage to your board and other property.

Using the Periphery library

To access the header pins on the Tinker Edge T, you can use standard Linux sysfs interfaces. But if you'd like a Python API, we recommend you use the python-periphery library, which is built atop the sysfs interfaces.

You can install the library on your Dev Board as follows:

```
sudo apt-get update
sudo apt-get install python3-pip

sudo pip3 install python-periphery
```

Note:

- To access peripheral hardware resources on the Dev Board, you need to run your code with sudo privileges.
- You should install the Python 3 version of Periphery because that's the Python version required by the Edge TPU Python API.

The Periphery library allows you to select a GPIO or PWM pin with a pin number. Other interfaces, such as I2C and UART pins must be specified using the pin's device path. See the following examples.

GPIO

The following code shows how to instantiate each of the GPIO pins with Periphery:

```
gpio6 = GPIO(6, "in")
gpio7 = GPIO(7, "in")
gpio8 = GPIO(8, "in")
gpio138 = GPIO(138, "in")
gpio140 = GPIO(140, "in")
gpio141 = GPIO(141, "in")

gpio73 = GPIO(73, "out")
gpio77 = GPIO(77, "out")
```

Note: GPIO73 and GPIO77 currently support only the "out" direction.
For more examples, see the https://python-periphery.readthedocs.io/en/latest/gpio.html.

PWM

The following code shows how to instantiate each of the PWM pins with Periphery:

```
# PWM1 = pwmchip0, pwm0
pwm1 = PWM(0, 0)
# PWM2 = pwmchip1, pwm0
pwm2 = PWM(1, 0)
# PWM4 = pwmchip2, pwm0
pwm4 = PWM(2, 0)
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/pwm.html.

I2C

The following code shows how to instantiate each of the I2C ports with Periphery:

```
i2c2 = I2C("/dev/i2c-1")
i2c3 = I2C("/dev/i2c-2")
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/i2c.html.

SPI

The following code shows how to instantiate each of the SPI ports with Periphery:

```
# SPI1, SS0, Mode 0, 10MHz
spi1_0 = SPI("/dev/spidev32766.0", 0, 10000000)
# SPI1, SS1, Mode 0, 10MHz
spi1_1 = SPI("/dev/spidev32766.1", 0, 10000000)
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/spi.html.

UART

The following code shows how to instantiate each of the UART ports with Periphery:

```
# UART1, 115200 baud
uart1 = Serial("/dev/ttymxc0", 115200)
# UART3, 9600 baud
uart3 = Serial("/dev/ttymxc2", 9600)
```

Caution: UART1 is shared with the Linux serial console. To use the UART1 port in your application, you must disable the serial console with the following command:

```
systemctl stop serial-getty@ttymxc0.service
```

Note: UART3 is not enabled by default in Mendel 3.0 (Chef), but we've released a package update to enable it. First check to see if it's available on your board:

```
ls /dev | grep ttymxc
```

If you see `ttymxc2` listed, then you're all good. If not, then you need to update as follows:

```
sudo apt-get update

sudo apt-get dist-upgrade

sudo reboot now
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/serial.html.

Sample Code

blink.py

```
from periphery import GPIO
import time

LED_Pin = 73 #Physical Pin-16 is GPIO 73

# Open GPIO /sys/class/gpio/gpio73 with output direction
LED_GPIO = GPIO(73, "out")

while True:
    try: #Blink the LED
        LED_GPIO.write(True)
        # Send HIGH to switch on LED
        print("LED ON!")
        time.sleep(0.5)

        LED_GPIO.write(False)
        # Send LOW to switch off LED
        print("LED OFF!")
        time.sleep(0.5)
    except KeyboardInterrupt:
        # Turn LED off before stopping
        LED_GPIO.write(False)
        break
    except IOError:
        print ("Error")

LED_GPIO.close()
```

Example (Run)

```
sudo python3 blink.py
```

How to check current hardware information

Current CPU frequency

To read current real-time CPU frequency:

```
sudo cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
```

Current GPU frequency

To read current GPU frequency

```
sudo cat /sys/kernel/debug/gc/clk | grep sh | awk '{print $4}'
```

Current CPU & GPU Temperature

To monitor real-time SoC temperature

```
watch -n 1 sudo cat /sys/class/thermal/thermal_zone0/temp
```

Advanced Script

save below text as hwinfo_moniter.sh

```
#!/bin/bash

soc_temp=$(sudo cat /sys/class/thermal/thermal_zone0/temp | awk '{printf "%.2f", $0 / 1000}')

cpu_freq=$(sudo cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq | awk '{printf "%.2f", $0 / 1000000}')

gpu_freq=$(sudo cat /sys/kernel/debug/gc/clk | grep sh | awk '{printf "%.2f", $4 / 1000000}')

echo "SoC Temp=> $soc_temp degree C"

echo "CPU Freq=> $cpu_freq GHz"

echo "GPU Freq=> $gpu_freq MHz"
```

example:

```
$ sudo chmod +x hwinfo_moniter.sh
$ ./hwinfo_moniter.sh
SoC => 55.00°C
```

also can watch it every seconds by:

```
$ watch -t -p -n 1 ./hwinfo_moniter.sh
```

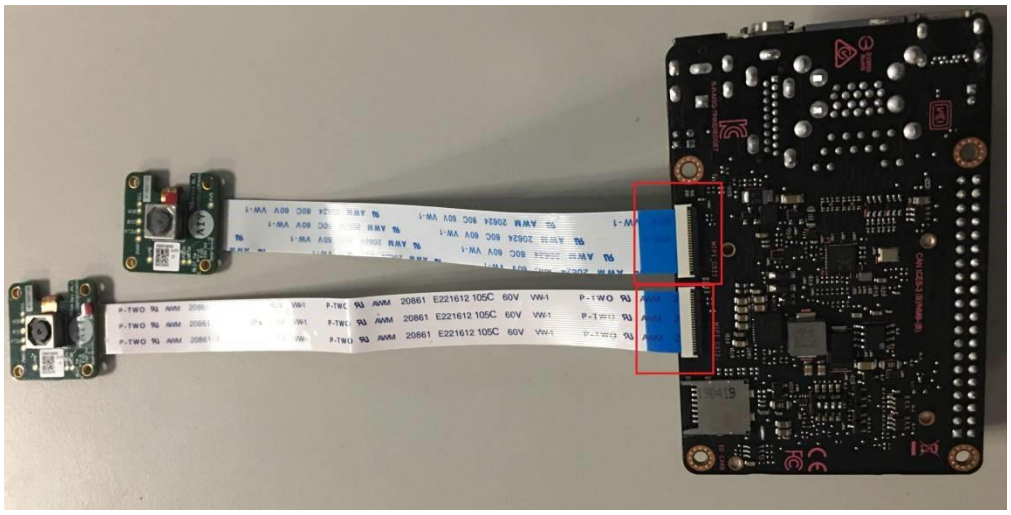
Get started with the Two Camera Demo

Requirements

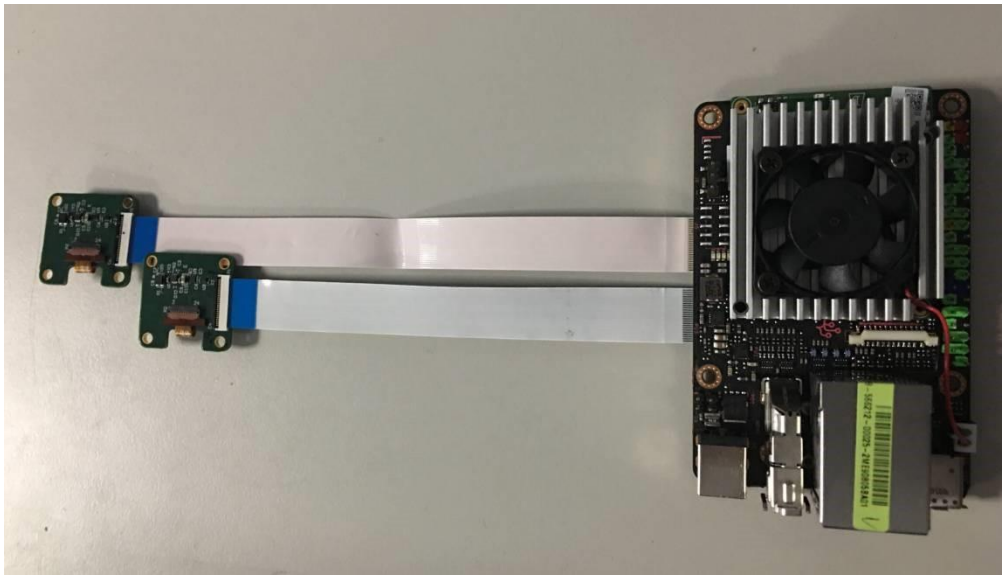
1. Tinker Edge T
2. Image version >= TINKER_EDGE_T-MENDEL-0.0.14-20190904
3. 2 x Coral or AIY's MIPI Camera

Set up on Tinker Edge T and MIPI Camera

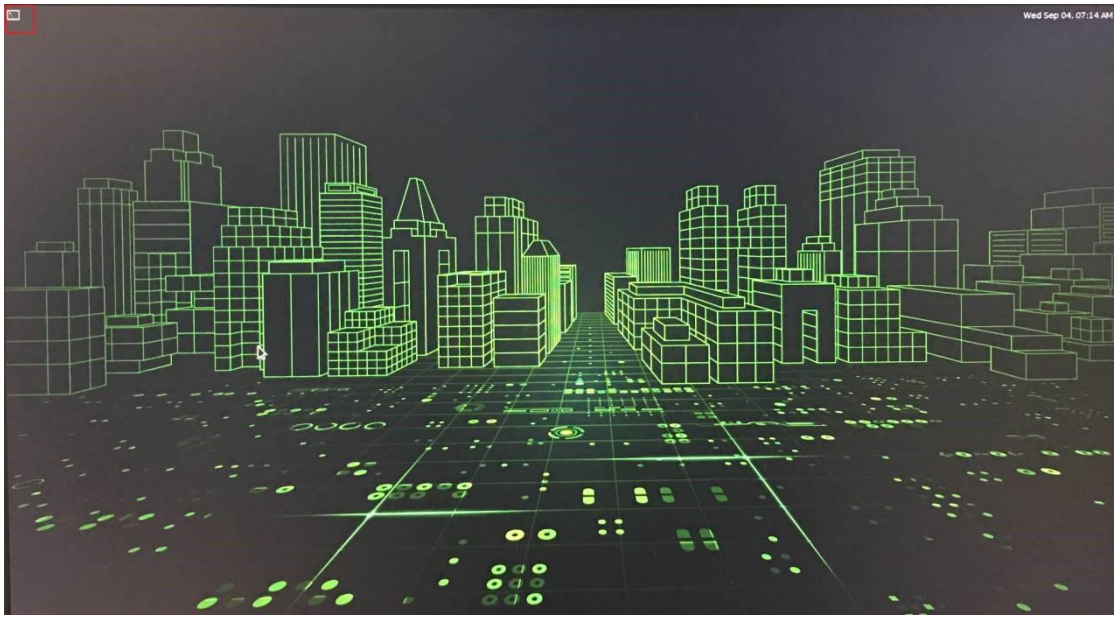
4. Insert the two MIPI camera to the CSI ports (Red part) of Tinker Edge T:
    - Back:



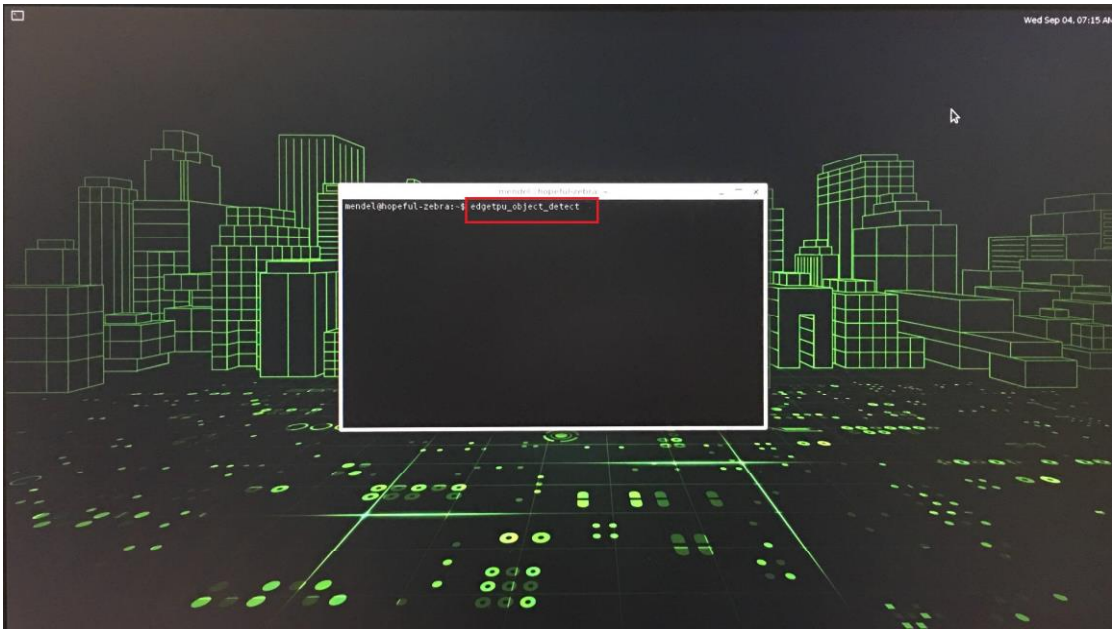
    - Front:



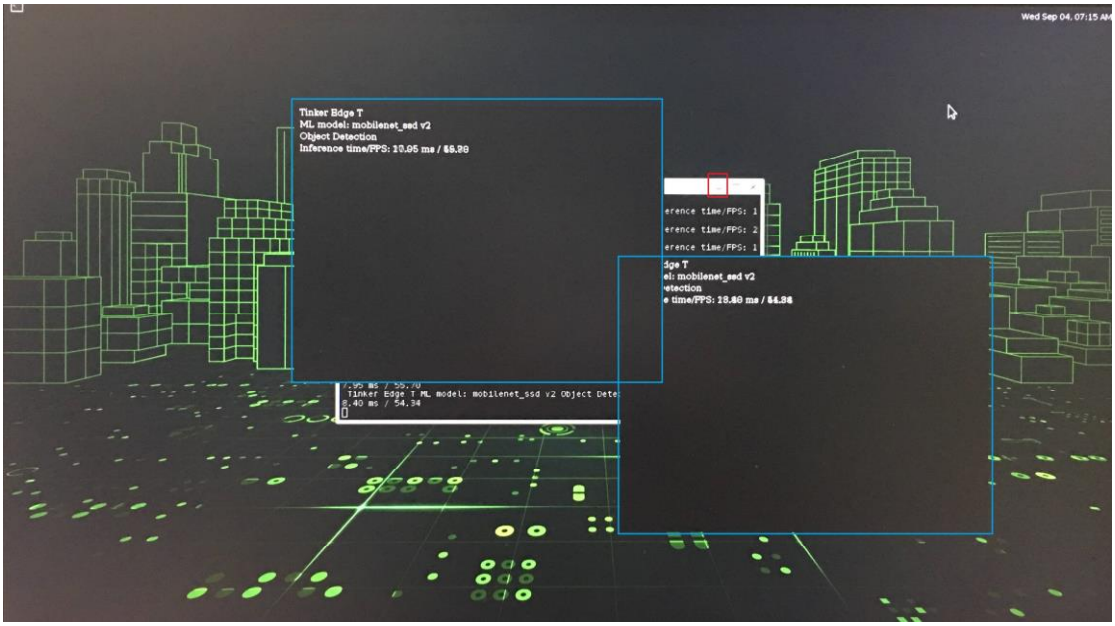
5. Power on Tinker Edge T and open the terminal by mouse (Red part):



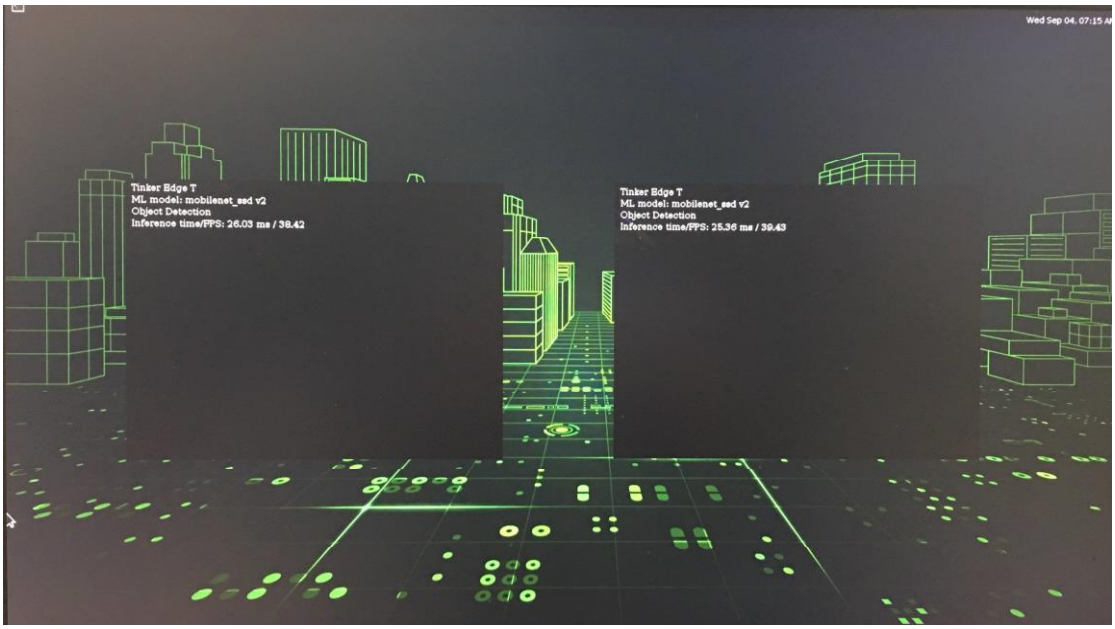
6. Enter the name of demo app in the terminal (`edgetpu_object_detect` or `edgetpu_face_detect`) (Red part):
   - `edgetpu_object_detect` : Object Detection by MobileNet-SSD v2
   - `edgetpu_face_detect`: Face Detection by MobileNet-SSD v2



7. Minimized the terminal by clicking the **red part** and move the camera windows by **hold the windows key + mouse left key**:



8. You should see results like this:



Congrats! You've just performed an inference on Tinker Edge T.