BeagleY-AI

Release 1.0.20240517-wip

BeagleBoard.org Foundation
May 17, 2024

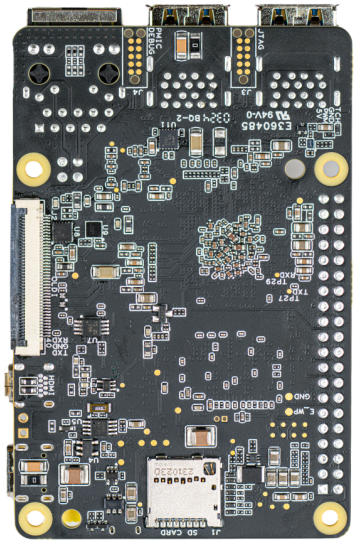
Table of contents

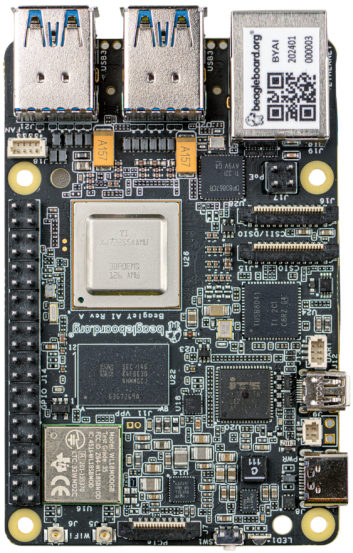
Important: This is a work in progress, for latest documentation please visit https://docs.beagleboard.org/latest/

BeagleY-AI is an open-source single board computer based on the Texas Instruments AM67A Arm-based vision processor.

License Terms

- This documentation is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License
- Design materials and license can be found in the git repository
- Use of the boards or design materials constitutes an agreement to the boards-terms-and-conditions
- Software images and purchase links are available on the board page
- For export, emissions and other compliance, see *Support*
- All support for BeagleY-AI design is through the BeagleBoard.org community at BeagleBoard.org forum.

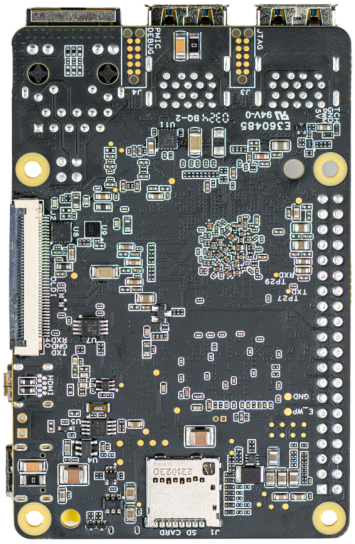


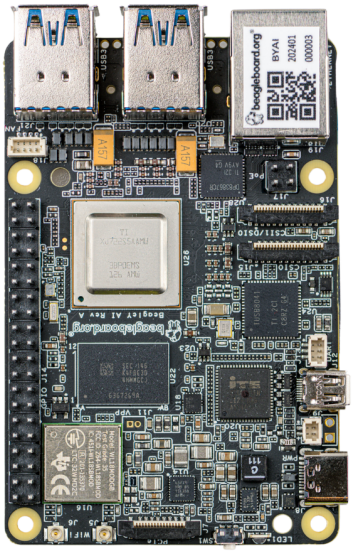


Chapter 1

Introduction

BeagleY-AI is an open-source single board computer designed for edge AI applications.





1.1 Detailed overview

BeagleY-AI is based on the Texas Instruments AM67A Arm-based vision processor. It features a quad-core 64-bit Arm®Cortex®-A53 CPU subsystem at 1.4GHz, Dual general-purpose C7x DSP with Matrix Multiply Accelerator (MMA) capable of 4 TOPs each, Arm Cortex-R5 subsystem for low-latency I/O and control, a 50 GFlop GPU, video and vision accelerators, and other specialized processing capability.

Table 1.1: BeagleY-AI features

Feature	Description
Processor	Texas Instruments AM67A, Quad 64-bit Arm® Cortex®-A53 @1.4 GHz, multiple cores including Arm/GPU processors, DSP, and vision/deep learning accelerators
RAM	4GB LPDDR4
Wi-Fi	Beagleboard BM3301, 802.11ax Wi-Fi
Bluetooth	Bluetooth Low Energy 5.4 (BLE)
USB Ports	4 x USB 3.0 TypeA ports supporting simultaneous 5Gbps operation, 1 x USB 2.0 TypeC, supports USB 2.0 device mode
Ethernet	Gigabit Ethernet, with PoE+ support (requires separate PoE HAT)
Camera/Display	2 x 4-lane MIPI camera connector (one connector muxed with DSI capability)
Display Output	1 x HDMI display, 1 x OLDI display, 1 x DSI MIPI Display
Real-time Clock (RTC)	Supports external coin-cell battery for power failure time retention
Debug UART	1 x 3-pin debug UART
Power	5V/3A DC power via USB-C
Power Button	On/Off included
PCIe Interface	PCI-Express® Gen3 x 1 interface for fast peripherals (requires separate M.2 HAT or other adapter)
Expansion Connector	40-pin header
Fan connector	1 x 4-pin fan connector, supports PWM control and fan speed measurement
Storage	microSD card slot with UHS-1 support
Tag Connect	1 x JTAG, 1 x External PMIC programming port

1.1.1 AM67A SoC

Todo: Add AM67A SoC details

1.1.2 Board components location

Front

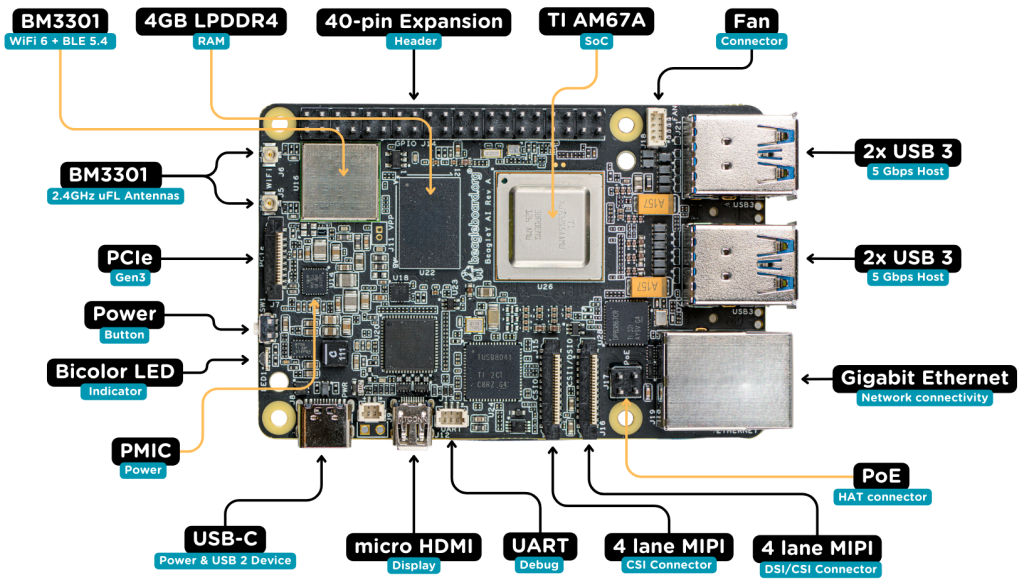


Table 1.2: BeagleY-AI board front components location

Feature	Description
WiFi/BLE	Beagleboard BM3301 with 802.11ax Wi-Fi & Bluetooth Low Energy 5.4 (BLE)
RAM	4GB LPDDR4
Expansion	40pin Expansion header compatible with HATs
SoC	TI AM67A Arm®Cortex®-A53 4 TOPS vision SoC with RGB-IR ISP for 4 cameras, machine vision, robotics, and smart HMI
Fan	4pin Fan connector
USB-A	4 x USB 3 TypeA ports supporting simultaneous 5Gbps operation host ports
Network Connectivity	Gigabit Ethernet
PoE	Power over Ethernet HAT connector
Camera/Display	1 x 4-lane MIPI camera/display transceivers, 1 x 4-lane MIPI camera
Debug UART	1 x 3-pin JST-SH 1.0mm debug UART port
Display Output	1 x HDMI display
USB-C	1 x Type-C port for power, and supports USB 2 device
PMIC	Power Management Integrated Circuit for 5V/5A DC power via USB-C with Power Delivery support
Bicolor LED	Indicator LED
Power button	ON/OFF button
PCIe	PCI-Express® Gen3 x 1 interface for fast peripherals (requires separate M.2 HAT or other adapter)

Back

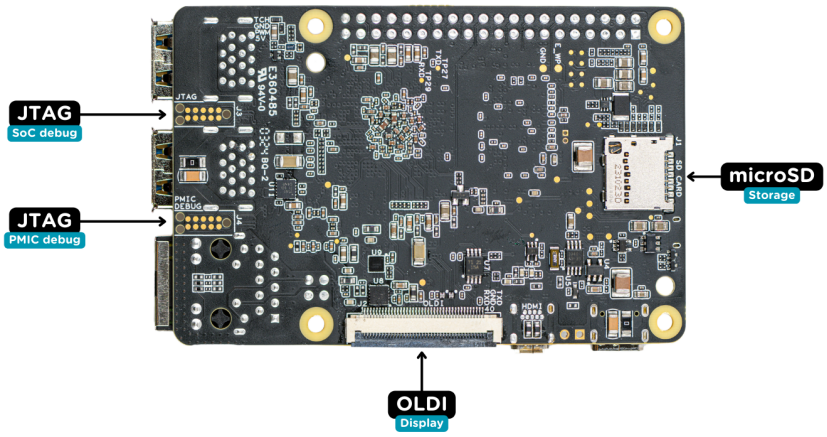


Table 1.3: BeagleY-AI board back components location

Feature	Description
Tag-Connect	1 x JTAG & 1 x Tag Connect for PMIC NVM Programming
Display output	1 x OLDI display
Storage	microSD card slot with support for high-speed SDR104 mode

Chapter 2

BeagleY-AI Quick Start

2.1 What's included in the box?

Todo: Update BeagleY-AI what's included in the box section as per production release.

When you purchase a BeagleY-AI, you'll get the following in the box:

1. BeagleY-AI
2. JST-SH cables
3. 2.4GHz antennas
4. Quick-start card

Tip: For board files, 3D model, and more, you can checkout the BeagleY-AI repository on OpenBeagle.

Todo: Attaching antennas instructions for BeagleY-AI

Todo: BeagleY-AI unboxing video

2.2 Getting started

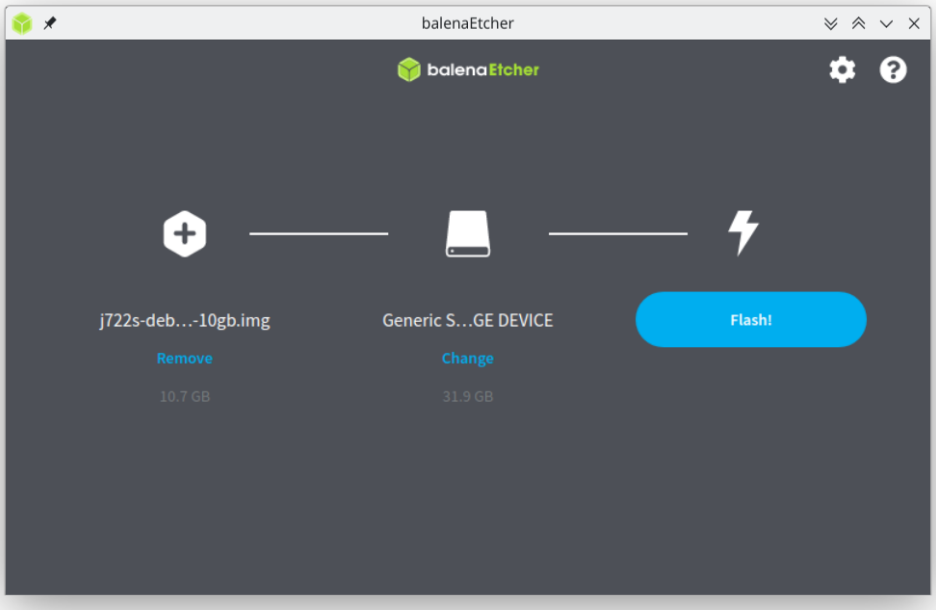
To get started you need the following:

1. USB type-A to type-C cable or type-C to type-C cable
2. 5V - 3A power supply
3. MicroSD Card
4. Boot media

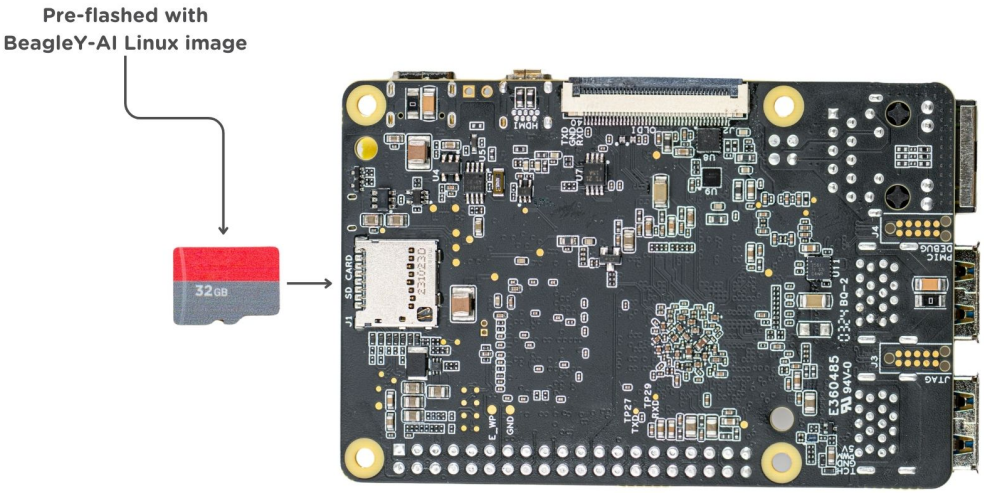
2.2.1 Boot Media

Download the boot media from https://www.beagleboard.org/distros/beagley-ai-debian-xfce-12-5-2024-03-25 and flash it on a micro SD Card using using Balena Etcher following these steps:

1. Select downloaded boot media
2. Select SD Card
3. Flash!



Once flashed, you can insert the SD card into your BeagleY-AI as shown in the image below:



2.2.2 Power Supply

To power the board you can either connect it to a dedicated power supply like a mobile charger or a wall adapter that can provide 5V ≥ 3A. Checkout the docs power supply page for power supply recommendations.

2.2.3 Board connection

There is only one USB type-C port on board, if you choose to use a dedicated power supply for first time setup, you may access the board via one of the following methods:

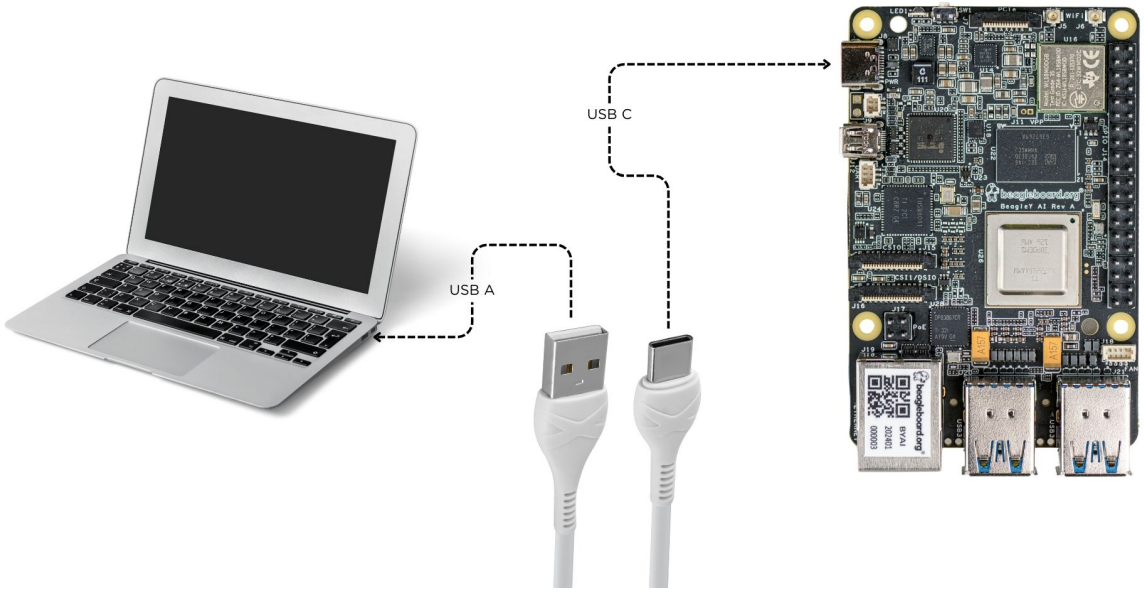
1. Connection to HDMI display, Keyboard and Mouse
2. UART using RPi debug probe or similar
3. Ethernet network connection

Another direct and easy option is to connect the board directly to your PC or Laptop using a USB type-C cable.

Note: If you are using the board with a fan or running a heavy task you should always power the board with a dedicated power supply that can supply 5V ≥ 3A.

2.2.4 USB Tethering

To initially test your board, you can connect the board directly to your computer using a type-A to type-C cable shown in the image below.



After connecting, you should see the power LED glow, and soon just like with other Beagles, you'll see a virtual wired connection on your computer. To access the board you can use SSH as shown below.

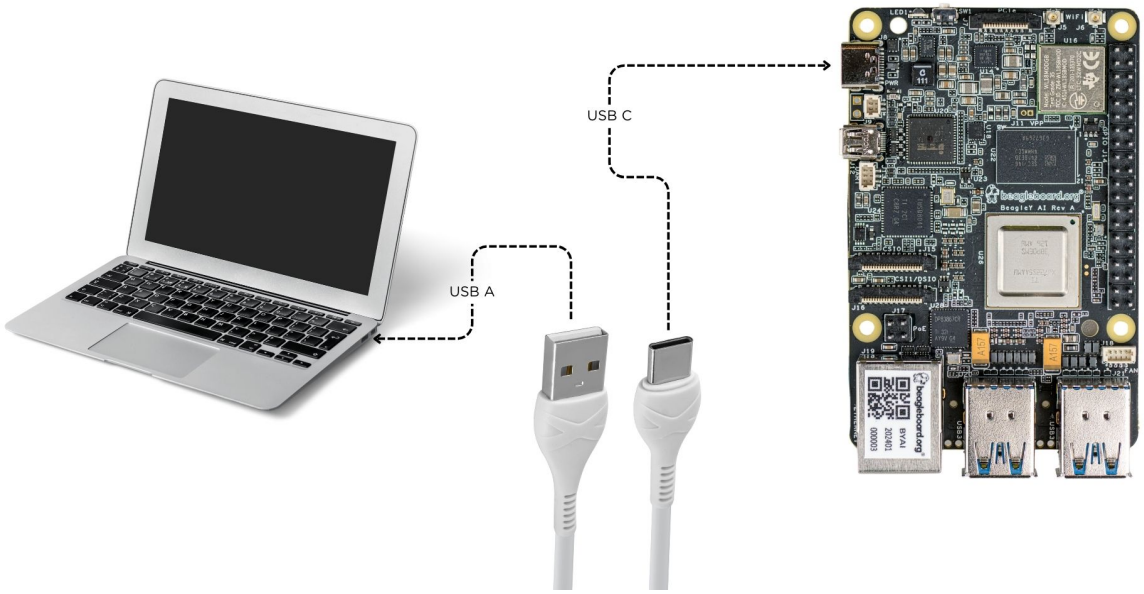
Note: Here you must update the default password to something safer.

```
[lorforlinux@fedora ~] $ ssh debian@192.168.7.2
Debian GNU/Linux 12

BeagleBoard.org Debian Bookworm Xfce Image 2024-03-25
Support: https://bbb.io/debian
default username is [debian] with a one time password of [temppwd]

debian@192.168.7.2's password:
You are required to change your password immediately (administrator enforced).
You are required to change your password immediately (administrator enforced).

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar 25 06:56:39 2024 from 192.168.7.1
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for debian.
Current password:
```

2.2.5 Using BeagleY-AI

To setup your BeagleY-AI for normal usage, connect the following:

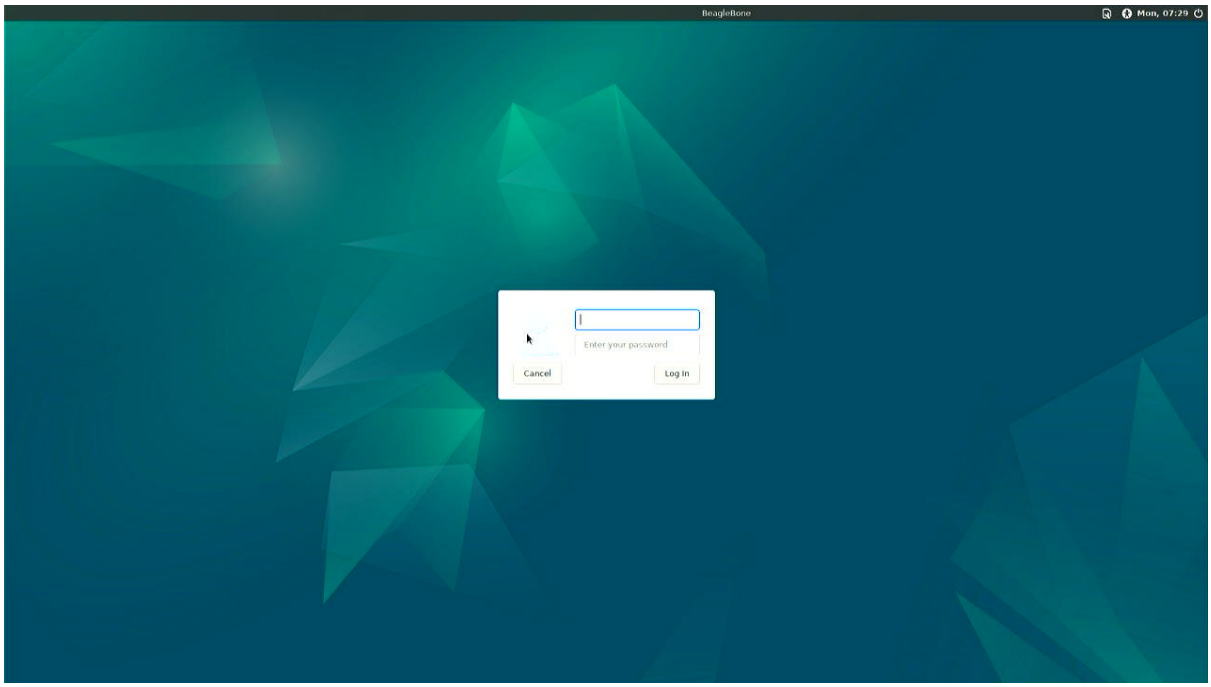
1. 5V ≥ 3A power supply
2. HDMI monitor using micro HDMI to full-size HDMI cable
3. Ethernet cable from the board to your router
4. Wireless or wired keyboard & mice



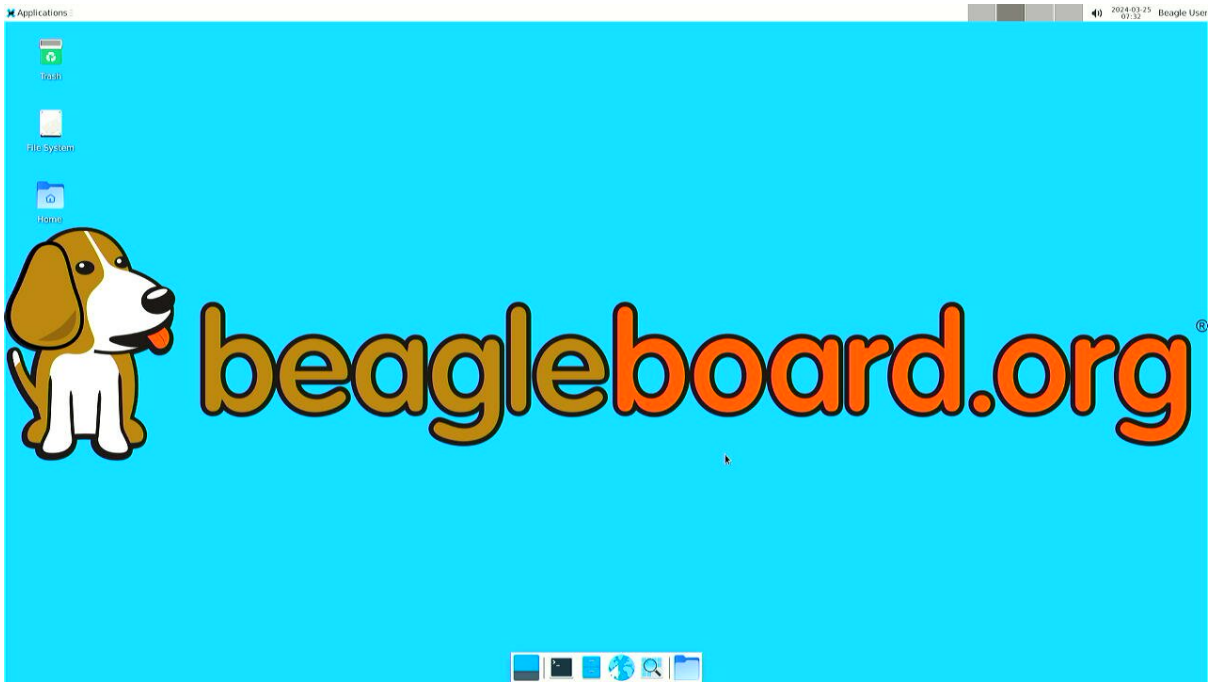
If everything is connected properly you should see four penguins on your monitor.

When prompted, log in using the updated login credentials you updated during the USB tethering step.

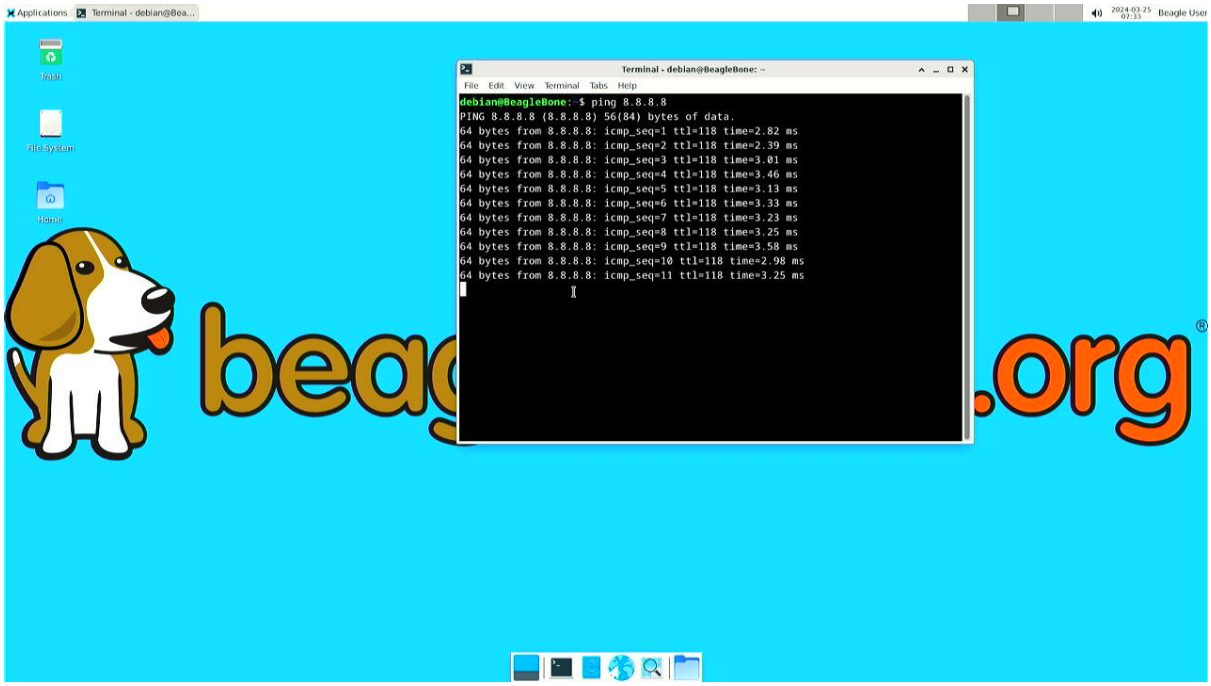
Note: You can not update login credentials at this step, you must update them during USB tethering step!



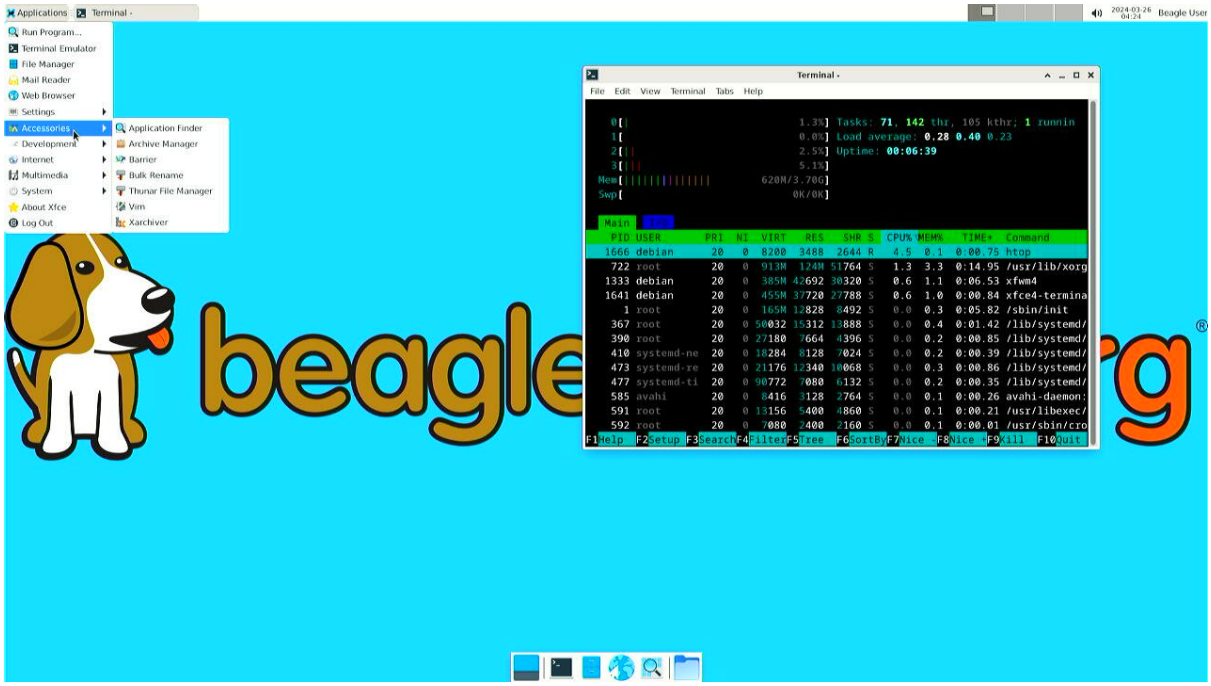
Once logged in you should see the splash screen shown in the image below:



Test network connection by running ping 8.8.8.8



Explore and build with your new BeagleY-AI board!



2.2.6 Connecting to WiFi

Connect 2x antennas to your BeagleY-AI board if not pre-attached.

After successfully attaching the antenna, power up the board. Once booted you can follow the commands below to connect to any WiFi access point,

- To list the wireless devices attached, (you should see wlan0 listed)

```
iwctl device list
```

- Scan WiFi using,

```
iwctl station wlan0 scan
```

- Get networks using,

```
iwctl station wlan0 get-networks
```

- Connect to your wifi network using,

```
iwctl --passphrase "<wifi-pass>" station wlan0 connect "<wifi-name>"
```

- Check wlan0 status with,

```
iwctl station wlan0 show
```

- To list the networks with connected WiFi marked you can again use,

```
iwctl station wlan0 get-networks
```

- Test connection with ping command,

```
ping 8.8.8.8
```

2.3 Demos and Tutorials

- beagley-ai-expansion-nvme

Chapter 3

Design and specifications

Chapter 4

Expansion

Todo: Describe how to build expansion hardware for BeagleY-AI

4.1 PCIe

For software reference, you can see how PCIe is used on NVMe HATs.

- beagley-ai-expansion-nvme
- beagley-ai-imx219-csi-cameras
- *Using the on-board Real Time Clock (RTC)*

Chapter 5

Demos and tutorials

5.1 Using the on-board Real Time Clock (RTC)

Todo: Add specific actions rather than notes that this is a work-in-progress.

Real Time Clocks (RTCs) provide precise and reliable timekeeping capabilities, which are beneficial for applications ranging from simple timekeeping to complex scheduling and secure operations.

Without an RTC, a computer must rely on something called Network Time Protocol (NTP) to obtain the current time from a network source. There are many cases however where an SBC such as BeagleY-AI may not have a constant or reliable network connection. In situations such as these, an RTC allows the board to keep time even if the network connection is severed or the board loses power for an extended period of time.

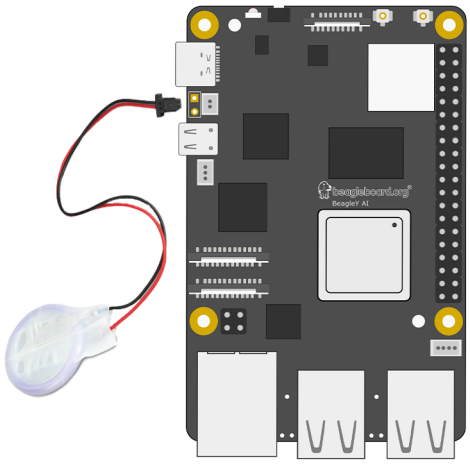
Fortunately, BeagleY-AI comes with a built-in DS1340 RTC for all your fancy time keeping needs!

5.1.1 Required Hardware

«««< HEAD BeagleY provides a **1.00 mm pitch, 2-pin JST SH connector** for a coin cell battery to enable the RTC to keep time even if power is lost to the board. ======= BeagleY-AI provides a **1.25 mm pitch, 2-pin JST GH connector** for a coin cell battery to enable the RTC to keep time even if power is lost to the board. »»»> 12aa969 (Using RTC review items)

These batteries are available from several vendors:

- Raspberry Pi 5 RTC Battery via Adafruit
- Raspberry Pi 5 RTC Battery via DigiKey
- CR2023 battery holder for Pi 5 via Amazon



5.1.2 Uses for an RTC

1. **Maintaining Accurate Time:** RTCs provide an accurate clock that continues to run even when the SBC is powered down. This is crucial for maintaining the correct time and date across reboots.
2. **Timestamping:** Many applications need to know the current time for timestamping data, logs, or events. For example, IoT devices may need to log sensor data with precise timestamps.
3. **Scheduling Tasks:** In some applications, tasks need to be scheduled at specific times. An RTC allows the SBC to keep track of time accurately, ensuring that tasks are performed at the correct times.
4. **Network Synchronization:** If the SBC is part of a larger network, having an accurate time helps with synchronizing data and events across the network.
5. **Standby Power Efficiency:** Many RTCs operate with a very low power requirement and can keep time even when the rest of the board is in a low-power or sleep mode. This helps in reducing overall power consumption.

5.1.3 Reading time

Note: If you have not connected your BeagleY-AI to a network so it can get time from an NTP server, you must set the time before being able to read it. If you don't do this first, you'll see errors.

Reading the current time on the RTC is achieved using the **hwclock** command.

```
debian@BeagleY:~$ sudo hwclock
2024-05-10 00:00:02.224187-05:00
```

5.1.4 Setting time

You can set time manually by running the following command:

```
hwclock --set --date "10/05/2024 21:01:05"
```

5.1.5 Diving Deeper

There are actually two different "times" that your Linux system keeps track of.

- System time, which can be read using the **date** or **timedatectl** commands
- RTC (hardware) time which can be read using the **hwclock** command shown above.

Comparing the time, we see something interesting, they're different!

You can just type "date" but the format will be different, so we add some extra instructions to match the format.

```
debian@BeagleBone:~$ date +%Y-%m-%d' '%H:%M:%S.%N%:z
2024-05-10 21:06:50.058595373+00:00

debian@BeagleBone:~$ sudo hwclock
2024-05-10 21:06:56.692874+00:00
```

But why? We see here that our system and hardware clock are over 9 seconds apart!

Ok, in this particular case we set the HW clock slightly ahead to illustrate the point, but in real life "drift" is a real problem that has to be dealt with. Environmental conditions like temperature or stray cosmic rays can cause electronics to become ever so slightly out of sync, and these effects only grow over time unless corrected. It's why RTCs and other fancier time keeping instruments implement various methods to help account for this such as temperature compensated oscillators.

Let's fix our hardware clock. We assume here that the system clock is freshly synced over NTP so it's going to be our true time "source".

```
debian@BeagleBone:~$ sudo hwclock --systohc
```

Let's write a simple script to get the two times, we'll call it **getTime.sh**:

```
HWTIME=$(sudo hwclock)
echo "RTC - ${HWTIME} "

SYSTIME=$(date +%Y-%m-%d' '%H:%M:%S.%N%:z)
echo "SYS - ${SYSTIME} "
```

Now let's run it!

```
debian@BeagleBone:~$ sudo chmod +x getTime.sh
debian@BeagleBone:~$ ./getTime.sh

RTC - 2024-05-10 21:52:58.374954+00:00
SYS - 2024-05-10 21:52:59.048442940+00:00
```

As we can see, we're still about a second off, but this is because it takes a bit of time to query the RTC via I2C.

If you want to learn more, the **Going Further** at the end of this article is a good starting point!

5.1.6 Troubleshooting

The most common error results from not having initialized the RTC at all. This usually happens if the system is powered on without an RTC battery and without a network connection.

In such cases, you should be able to read the time after setting the time as follows:

```
debian@BeagleBone:~$ sudo hwclock --systohc

debian@BeagleBone:~$ sudo hwclock
2024-05-10 21:06:56.692874+00:00
```

5.1.7 Going Further

Consider learning about topics such as time keeping over GPS and Atomic Clocks!

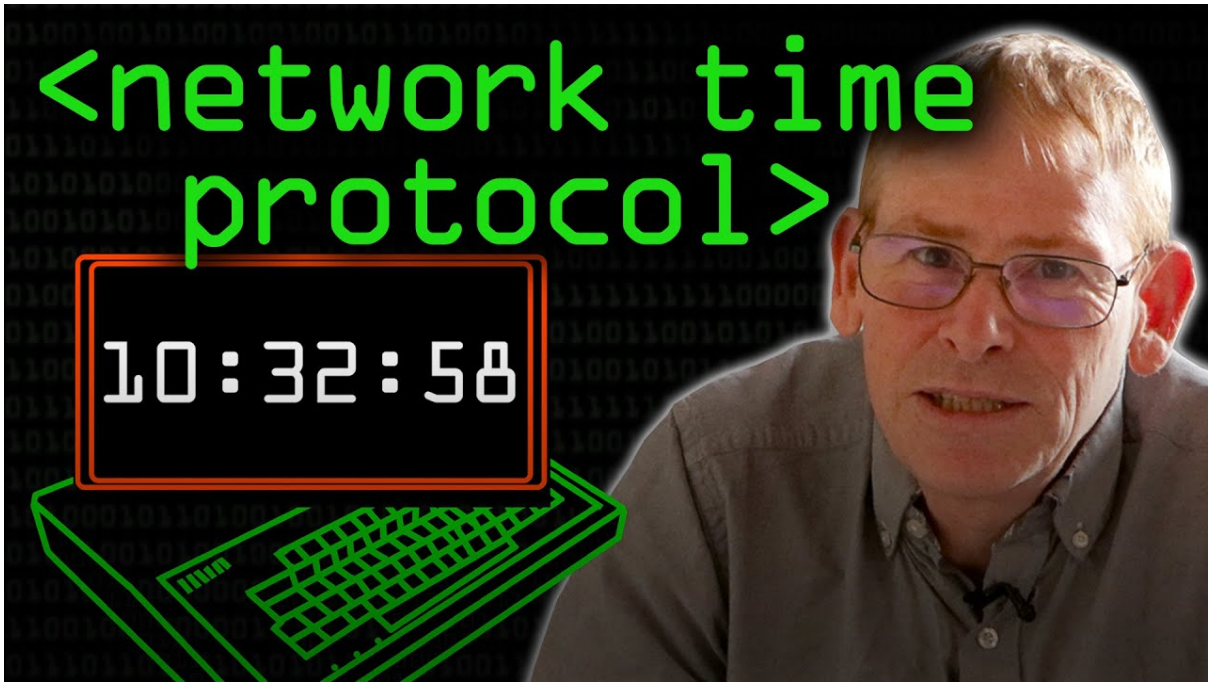


Fig. 5.1: https://youtu.be/BAo5C2qbLq8

There are some good YouTube videos below to provide sources for inspiration.

Network Time Protocol - Computerphile

Nanosecond Clock Sync - Jeff Geerling

Using GPS with PPS to synchronize clocks over the network

Note: This page is a work in progress. Further drive testing and images will be added soon

5.2 Using PCA9685 Motor Drivers

There are several such “Motor and Servo Driver HATs” available on Amazon, Adafruit and other marketplaces. While different manufacturers implement them slightly differently, the operating principle remains the same.

This guide aims to show you examples for two, namely the Xicoolee and Adafruit variants and how you can modify the example Python userspace library for other variations.

5.2.1 Operating Principle

The NXP PCA9685 is a simple 16-channel, 12-bit PWM controller that communicates over I2C.

While originally designed as an LED driver, it’s ability to output PWM also makes it suitable as a Servo Motor driver.

In addition, to add the ability to drive DC motors, some board designers add one or two Toshiba TB6612FNG dual motor drivers as shown in the schematic below.

Fig. 5.2: https://youtu.be/RvnG-ywF6_s

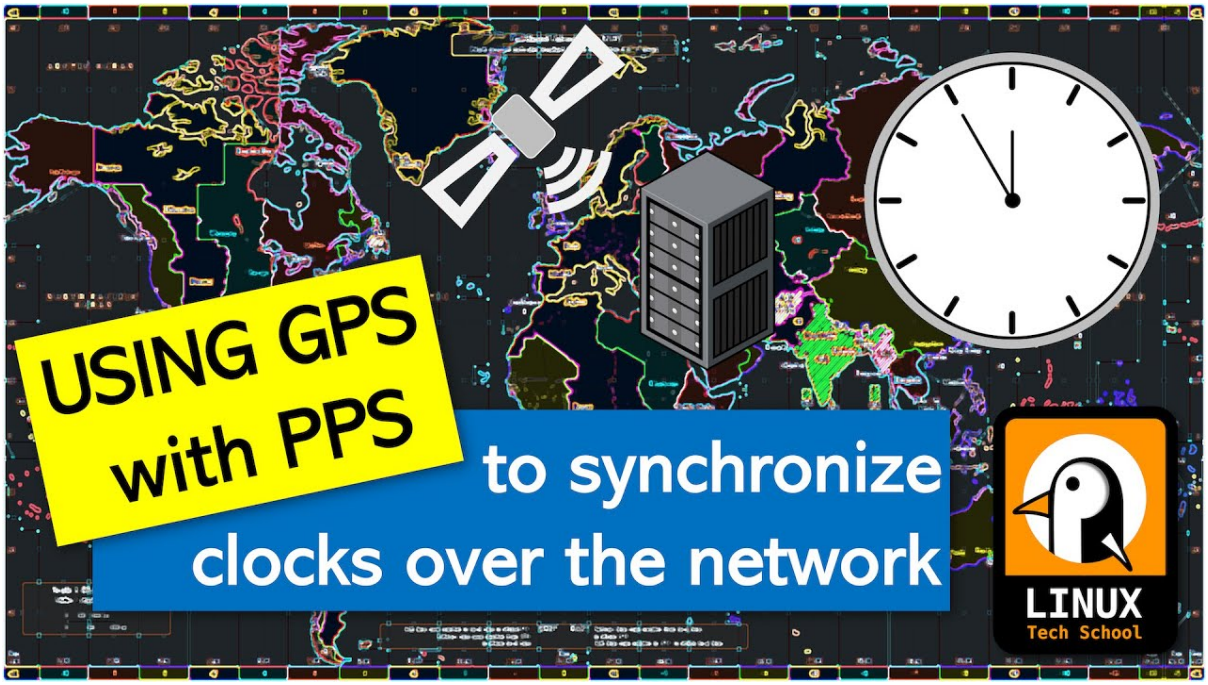
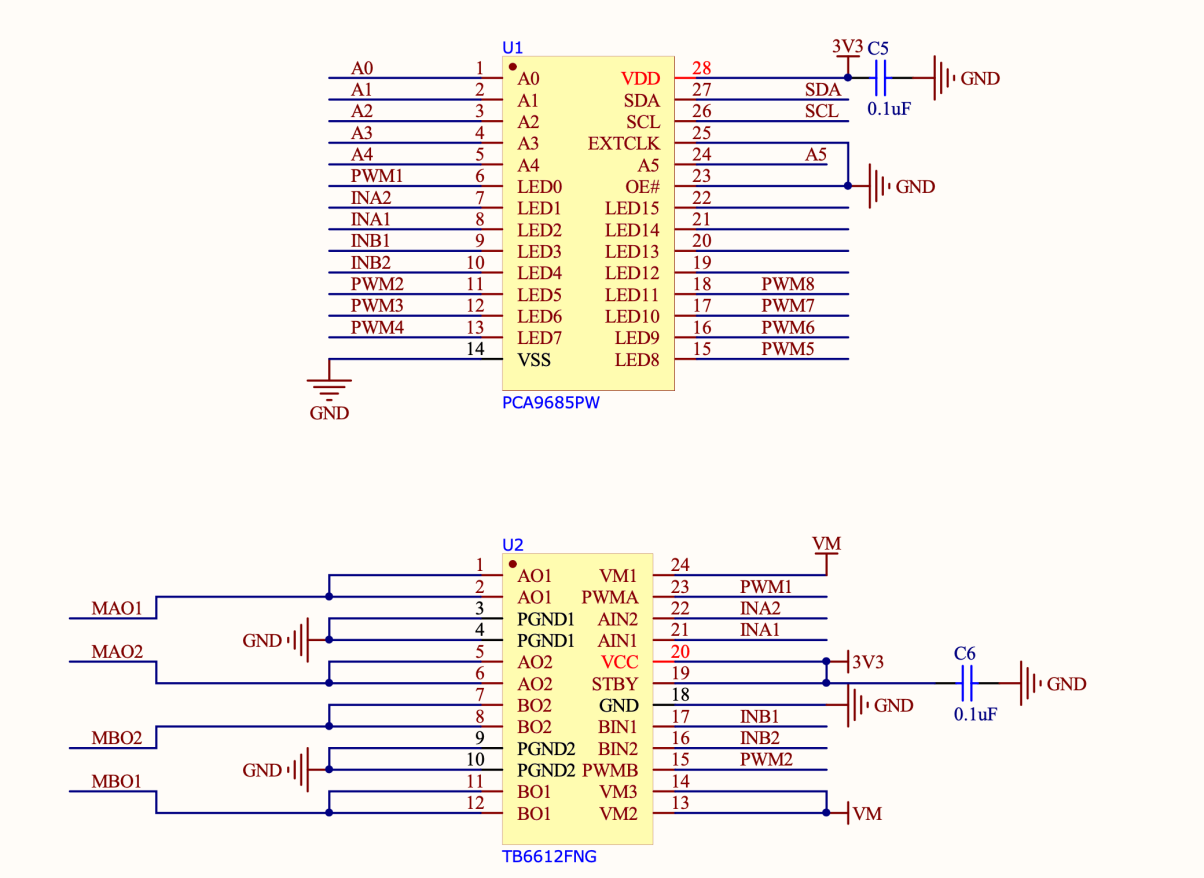


Fig. 5.3: https://youtu.be/7aTZ66ZL6Dk



If we look at the Xicoolee board and compare it to the schematic, we see that indeed Servo Channels 3-8 on the PCB Silkscreen match pins 12 through 18 of the PWM Driver, while PWM1, PWM2, INA1/2 and INB1/2 are used in conjunction with the TB6612FNG.

Looking at the TB6612FNG Datasheet, we can see that the IN pins for Channels A and B (INAx, INBx) are used to control the direction or “mode” of the DC motor, while the PWM signal controls the rotation speed for that particular channel.

TOSHIBA TB6612FNG

H-SW Control Function

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Thus, we can use the decoder table above to infer that to drive motor channel A at 50% speed clockwise, we

would set the PCA9685 to output INA1 High, INA2 Low and PWM1 at a 50% duty cycle.

If we wanted to go counter-clockwise, we would simply swap things around so INA1 was Low, INA2 was High and assuming we want to keep the same rotation speed, PWM1 at a 50% duty cycle.

Lastly, we have the option for a "Short Brake" for the motors but please note that it is not recommended to keep motors in this state as that shorts the coils internally and will cause them to heat up over time. If you want to stop your motor, you should issue a "Short brake" state followed by a short delay to allow the motor to physically stop rotating and then leave the motor in the "Stop" state (which de-energizes the coils) by setting IN1 and IN2 to LOW.

But enough theory, let's use some actual code to make things spin...

5.2.2 Using Adafruit ServoKit

If you are looking to drive Servo motors accurately and not particularly interested in driving DC motors, you may consider using the Adafruit ServoKit library which simplifies this type of use case. As with all python modules, make sure you do so inside a virtual environment as shown below!

```
mkdir project-name && cd project-name
python3 -m venv .venv
source .venv/bin/activate
sudo pip3 install --upgrade setuptools
sudo pip install --upgrade adafruit-python-shell
wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-
↪Scripts/master/raspi-blinka.py
sudo python raspi-blinka.py
pip3 install adafruit-circuitpython-servokit adafruit-circuitpython-
↪busdevice adafruit-circuitpython-register
```

From here, you should be able to run some example code such as the following:

```
import time
from adafruit_servokit import ServoKit

# Set channels to the number of servo channels on your kit.
# 8 for FeatherWing, 16 for Shield/HAT/Bonnet.
kit = ServoKit(channels=16)

kit.servo[0].angle = 180
kit.continuous_servo[1].throttle = 1
time.sleep(1)
kit.continuous_servo[1].throttle = -1
time.sleep(1)
kit.servo[0].angle = 0
kit.continuous_servo[1].throttle = 0
```

To explore ServoKit further, check out the ServoKit Github Page and Examples

5.2.3 Python User-space Driver

As mentioned before, the PCA9685 is a rather simple I2C device, so the driver for it is equally simple: PCA9685.py

Simply download this to the root of your project and you are most of the way there.

From there, you simply need an import statement and to define the driver instance:

```
from PCA9685 import PCA9685

pwm = PCA9685(0x60, debug=False) #Default I2C Address for the shield is 0x60
pwm.setPWMFreq(50) #Most Servo Motors use a PWM Frequency of 50Hz
```

You can now drive LEDs or servo motors by issuing the following command (replacing pin and dutyCycle with your particular values):

```
pwm.setDutycycle(pin, dutyCycle)
```

5.2.4 WaveShare Motor and Servo Driver HAT

Waveshare writes some of the better documentation for these types of Motor Driver HATs

Todo: Add more information on Waveshare motor & servo driver HAT.

5.2.5 XICOOLEE Motor and Servo Driver HAT

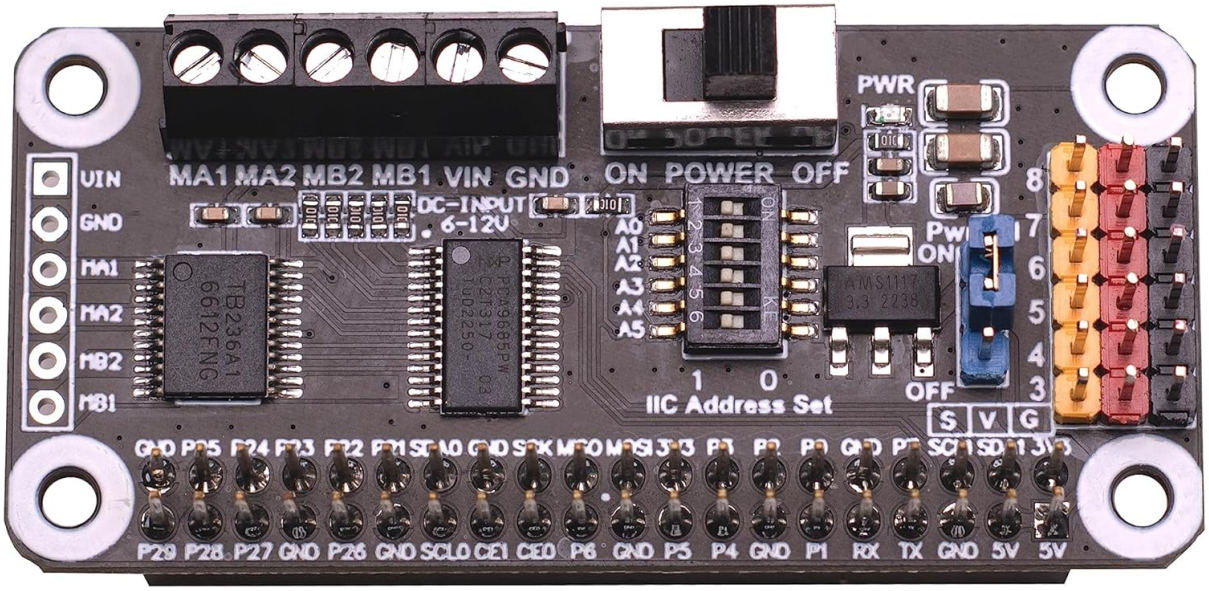


Photo Credit - Xicoolee

Looking at the schematic for the Xicoolee HAT, we see that we need to define our DC motor pins as follows:

```
#Xicoolee TB6612FNG

self.PWMA = 0
self.AIN1 = 2
self.AIN2 = 1
self.PWMB = 5
self.BIN1 = 3
self.BIN2 = 4
```

We can then run some simple example code as shown below:

```
#!/usr/bin/python

from PCA9685 import PCA9685
import time

Dir = [
    'forward',
    'backward',
]
```

(continues on next page)

(continued from previous page)

```
pwm = PCA9685(0x40, debug=False)
pwm.setPWMFreq(50)

class MotorDriver():
    def __init__(self):
        # Match these to your particular HAT!
        self.PWMA = 0
        self.AIN1 = 2
        self.AIN2 = 1
        self.PWMB = 5
        self.BIN1 = 3
        self.BIN2 = 4

    def MotorRun(self, motor, index, speed):
        if speed > 100:
            return
        if(motor == 0):
            pwm.setDutycycle(self.PWMA, speed)
            if(index == Dir[0]):
                print ("1")
                pwm.setLevel(self.AIN1, 0)
                pwm.setLevel(self.AIN2, 1)
            else:
                print ("2")
                pwm.setLevel(self.AIN1, 1)
                pwm.setLevel(self.AIN2, 0)
        else:
            pwm.setDutycycle(self.PWMB, speed)
            if(index == Dir[0]):
                print ("3")
                pwm.setLevel(self.BIN1, 0)
                pwm.setLevel(self.BIN2, 1)
            else:
                print ("4")
                pwm.setLevel(self.BIN1, 1)
                pwm.setLevel(self.BIN2, 0)

    def MotorStop(self, motor):
        if (motor == 0):
            pwm.setDutycycle(self.PWMA, 0)
        else:
            pwm.setDutycycle(self.PWMB, 0)

print("this is a motor driver test code")
Motor = MotorDriver()

print("forward 2 s")
Motor.MotorRun(0, 'forward', 100)
Motor.MotorRun(1, 'forward', 100)
time.sleep(2)

print("backward 2 s")
Motor.MotorRun(0, 'backward', 100)
Motor.MotorRun(1, 'backward', 100)
time.sleep(2)

print("stop")
Motor.MotorStop(0)
Motor.MotorStop(1)
```

5.2.6 Adafruit DC & Stepper Motor HAT

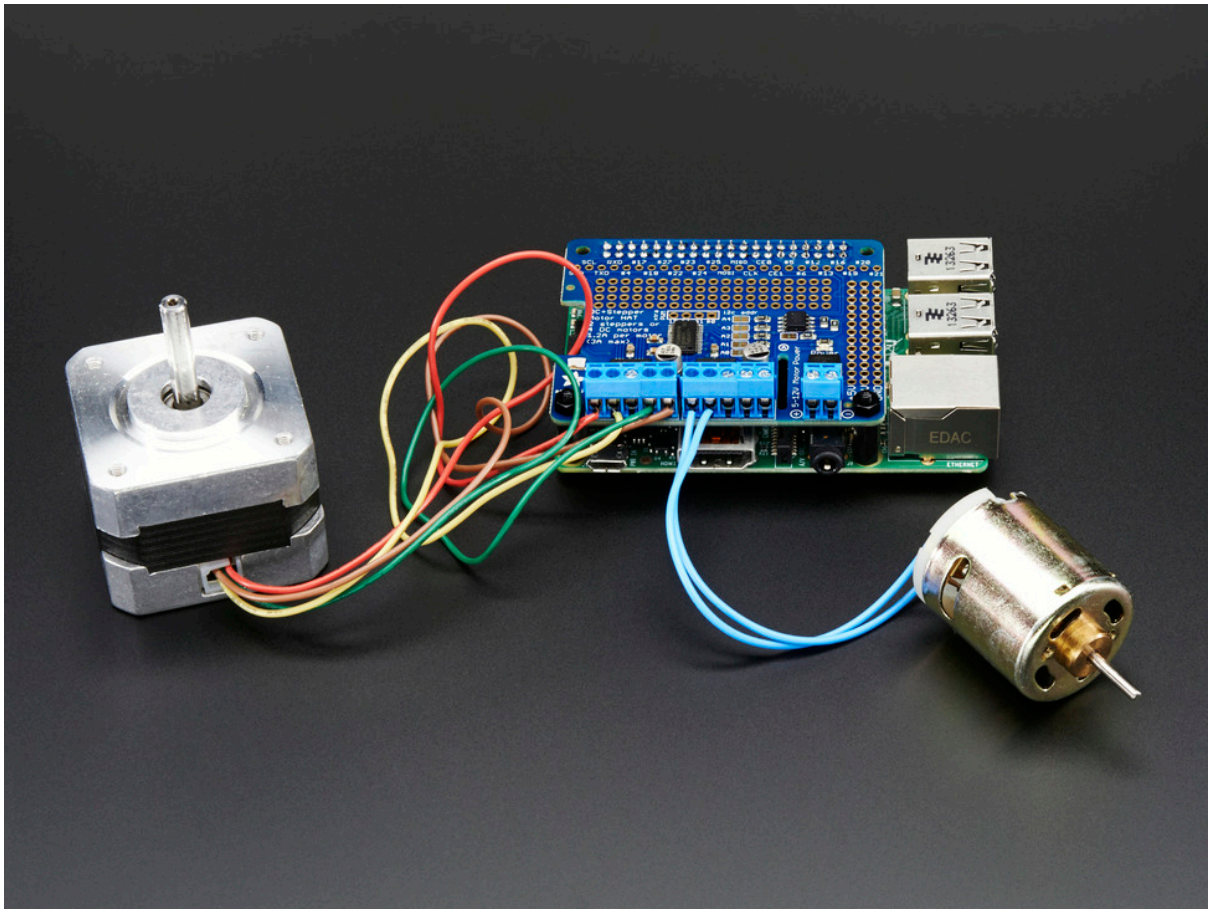


Photo Credit - Adafruit

Looking at the schematic for the Adafruit HAT, we see that we need to define our DC motor pins as follows:

```
#Adafruit TB6612FNG #1

self.PWMA = 8
self.AIN1 = 10
self.AIN2 = 9
self.PWMB = 13
self.BIN1 = 11
self.BIN2 = 12

#Adafruit TB6612FNG #2

self.PWMA_2 = 2
self.AIN1_2 = 4
self.AIN2_2 = 3
self.PWMB_2 = 7
self.BIN1_2 = 5
self.BIN2_2 = 6
```

Todo: Expand on running 2 DC motor objects

Note: This page is a work in progress. Further drive testing and images will be added soon

5.3 Using the Arducam Dual V3Link Camera Kit

The Arducam Dual V3Link Camera Kit is an IMX219 based kit that leverages Texas Instruments' FPDLink technology to enable using two CSI cameras over a single port up to 15 meters away using twisted pair cables.

Note: Unlike the larger quad-camera kit, the dual camera kit aims to simplify the software stack and improve interoperability with the Raspberry Pi and other non-TI SBCs by forgoing the ability to support multi-stream CSI inputs. This means that it is limited to "switching" between the two FPDLink inputs but has the benefit of not requiring additional drivers beyond support for the base CSI camera driver (IMX219 in this case)

5.3.1 Initial Hardware Connection

Simply plug in the HAT into the BeagleY GPIO header and connect the CSI header as shown below.

Either CSI header may be connected but make sure you use the corresponding CSI port DTS when enabling your "camera".

Todo: ADD CSI 0/1 Header Location photo.

5.3.2 Verify that the HAT is connected

The Arducam HAT should present itself as an I2C device on Bus 1.

To check that the I2C Bus looks like we expect:

```
sudo i2cdetect -r -y 1
```

To verify actual communication with the FPDlink device, we issue the following command:

```
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x01 r1
```

5.3.3 Switching CSI Channels

The channel numbering for FPDLink goes from 1 to 2 (as opposed to counting from 0 as is the case for CSI)

Thus, to select video output from channel 1:

```
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x01
```

To switch to channel 2:

```
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x02
```

5.3.4 Troubleshooting

For additional documentation and support, see the Arducam Docs.

Chapter 6

Support

All support for BeagleY-AI design is through BeagleBoard.org community at BeagleBoard.org forum.

6.1 Production board boot media

Todo: Add production boot media link in `_static/epilog/production.image` and reference it here.

6.2 Certifications and export control

6.2.1 Export designations

- HS: 8471504090
- US HS: 8543708800
- UPC: 640265311062
- EU HS: 8471707000
- COO: CHINA

6.2.2 Size and weight

- Bare board dimensions: **TBD**
- Bare board weight: **TBD**
- Full package dimensions: **TBD**
- Full package weight: **TBD**

6.3 Additional documentation

6.3.1 Hardware docs

For any hardware document like schematic diagram PDF, EDA files, issue tracker, and more you can checkout the BeagleY-AI design repository.

6.3.2 Software docs

For BeagleY-AI specific software projects you can checkout all the BeagleY-AI project repositories group.

6.3.3 Support forum

For any additional support you can submit your queries on our forum, https://forum.beagleboard.org/tag/beagley-ai

6.3.4 Pictures

6.4 Change History

Note: This section describes the change history of this document and board. Document changes are not always a result of a board change. A board change will always result in a document change.

6.4.1 Board Changes

For all changes, see https://openbeagle.org/beagley-ai/beagley-ai. Versions released into production are noted below.

Table 6.1: BeagleY-AI board change history

Rev	Changes	Date	By