Motor Bridge Cape v1.0

Драйвер двигателей для одноплатного компьютера BeagleBone Green на базе TB6612FNG

https://www.chipdip.ru/product/motor-bridge-cape

Motor Bridge Cape - это плата для двунаправленного управления двигателями, основанная на базе двух двойных H-мостов TB6612FNG, таким образом, вы сможете управлять двумя шаговыми двигателями или четырьмя двигателями постоянного тока с напряжением питания 6 ~ 15В и потреблением тока 1А на двигатель. Плата обеспечивает 5В для SeeedStudio BeagleBone Green или BeagleBone Black с максимальным входным напряжением 15 В. На плате также имеется 6 интерфейсов для серво моторов и 6 контактов I/O. Эти функции обеспечивает сопроцессор STM32F0, который взаимодействует с BeagleBone через интерфейс I2C или UART.

Особенности

- Одновременное управление 4 двигателями постоянного тока или 2 шаговыми двигателями;
- Может управлять 6 сервоприводами;
- Mbed платформа;
- Сопроцессор STM32F0;

- Два чипа TB6612FNG;
- 6 расширений I/O;
- Взаимодействие с BBG с помощью интерфейса I2C или UART.

Спецификация
Напряжение питания модуля: 6 ~ 15В

Напряжение питания двигателей: 6 ~ 15В

Максимальная нагрузка по выходу 5V постоянного тока: 2А макс.

Максимальная нагрузка по выходу 3.3V постоянного тока: 350 мА макс.

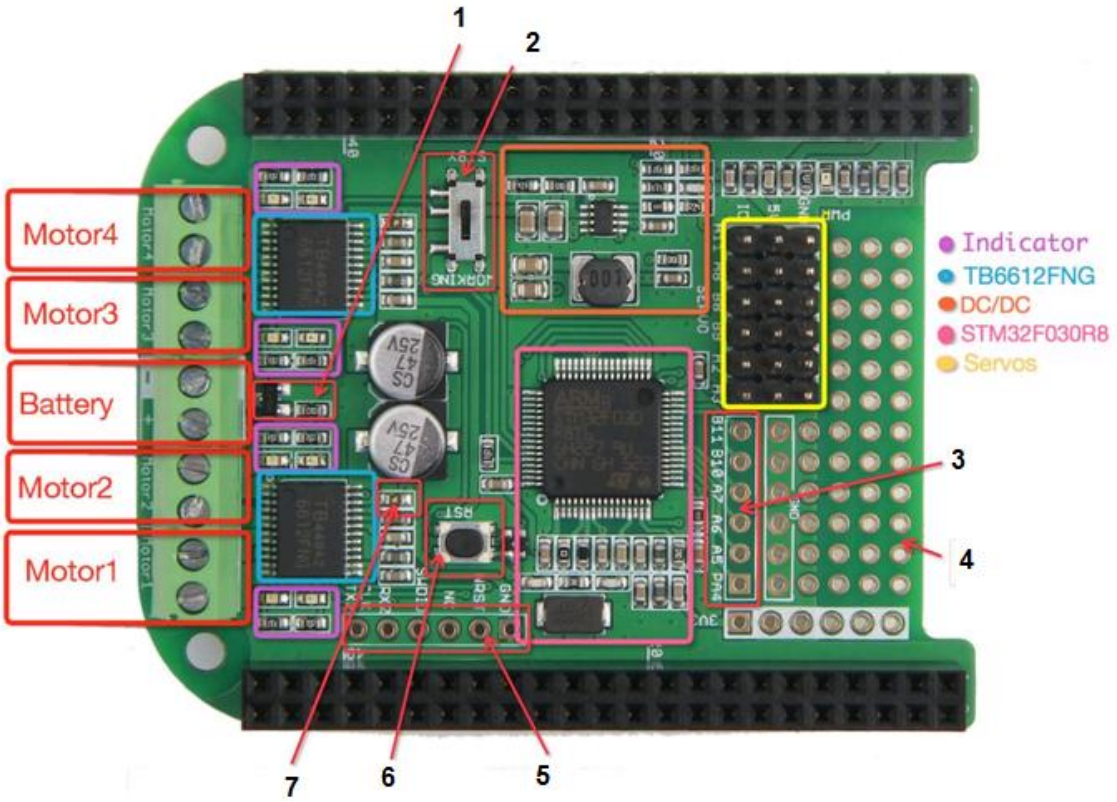
4 Н-моста, номинальный ток на каждом : не более 1.2А, пиковый ток: 3.2А

6 Сервопривод, рабочее напряжение: 5В, общий ток не более 1,5А

Защита от обратного подключения входа

Защита от перегрузки по току: предохранитель 3А

Обзор оборудования



1. Входная обратная защита: защитная цепь

2. Переключатель рабочего режима: ждущий режим или рабочий режим

3. GPIO: возможность подключения устройств с протоколом **GPIO**

4. Область для макетирования: позволяет собирать на свободном макетном поле ваши разработки

5. Интерфейс SWD: интерфейс отладки

6. Кнопка сброса: служит для перезагрузки сопроцессора

7. 3A Предохранитель: Защита от перегрузки по току.

Сервоприводы: интерфейс серво двигателя

STM32F030R8: сопроцессор

DC / DC: схема преобразования напряжения 5В в 3.3В

TB6612FNG: H-мост IC

Индикатор: индикаторная лампа для указания направления двигателя

Motor4 / Motor3: привод 2 моторов постоянного тока или 1 шаговый мотор

Motor2 / Motor1: привод 2 моторов постоянного тока или 1 шаговый мотор

Батарея: питание для двигателя

Начало работы

Рассмотрим как использовать **Motor Bridge Cape** на BBG. Прежде чем приступить к работе, пожалуйста, загрузите библиотеку кода **Motor Bridge Cape** из Github.

Чтобы использовать библиотеку **Motor Bridge Cape**, просто добавьте файл MotorBridge.py в свой проект. И импортируйте файл python в свой проект и создайте объект **Motor Bridge Cape**.

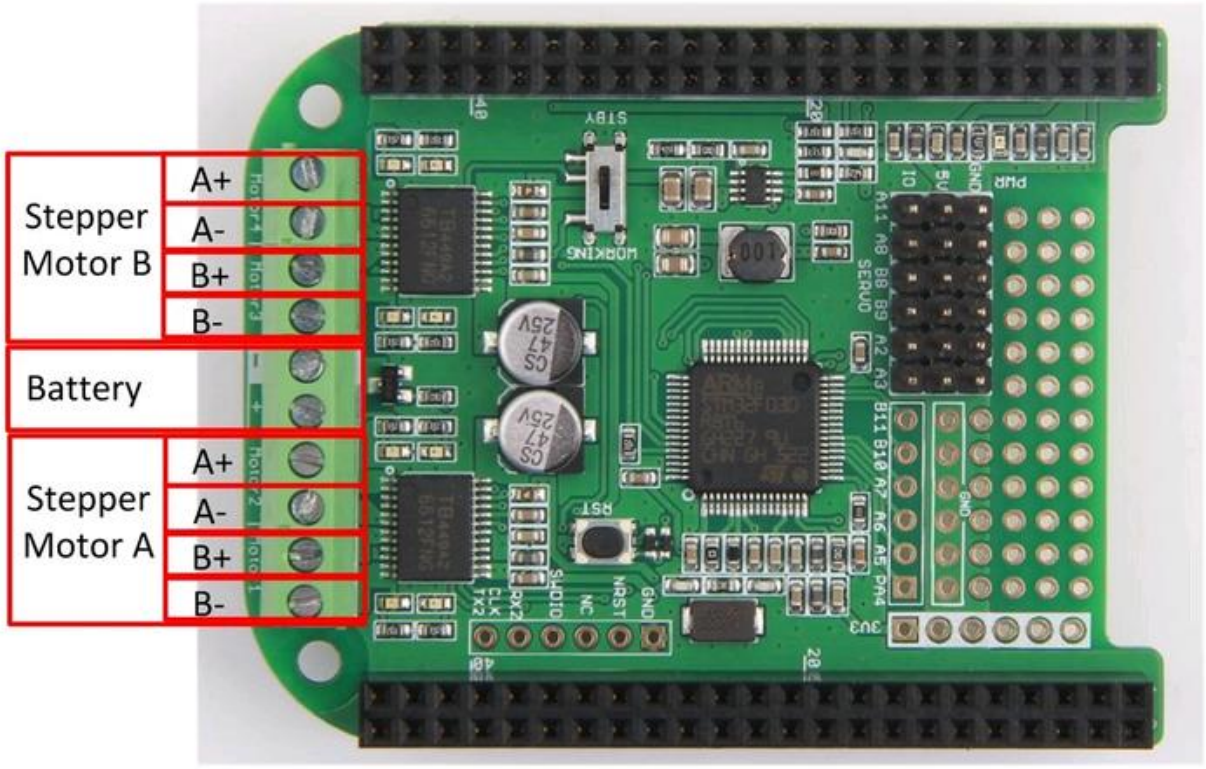
```
import MotorBridge
motor = MotorBridge.MotorBridgeCape()
```

Установка зависимостей

```
sudo apt-get update
sudo apt-get install build-essential python-pip python-dev python-smbus git
sudo pip install Adafruit-GPIO
sudo apt-get install build-essential python-dev python-pip -y
sudo pip install Adafruit_BBIO
```

Шаговый мотор

Интерфейс шагового двигателя **Motor Bridge Cape**, устроен как показано на рисунке ниже.



Функции шагового двигателя

Вот краткое описание функции шагового двигателя

StepperMotorAInit ()

Описание: Инициируйте порт шагового двигателя A.

StepperMotorAMove (MoveSteps, StepDelayTime)

Описание: Привод шагового двигателя A.

MoveSteps: Сколько шагов шаговый будет выполнять. Положительное значение означает направление по часовой стрелке. Отрицательное значение означает против часовой стрелки.

StepDelayTime: время замирания для каждого шага. единица: us.

StepperMotorBInit ()

Описание: подключите порт шагового двигателя B.

StepperMotorBMove (MoveSteps, StepDelayTime)

Описание: Начало движения шагового двигателя B.

MoveSteps: На сколько шагов шаговый двигатель повернет свой вал. Положительное значение означает направление по часовой стрелке. Отрицательное значение означает против часовой стрелки.

StepDelayTime: время замирания для каждого шага. единица: us.

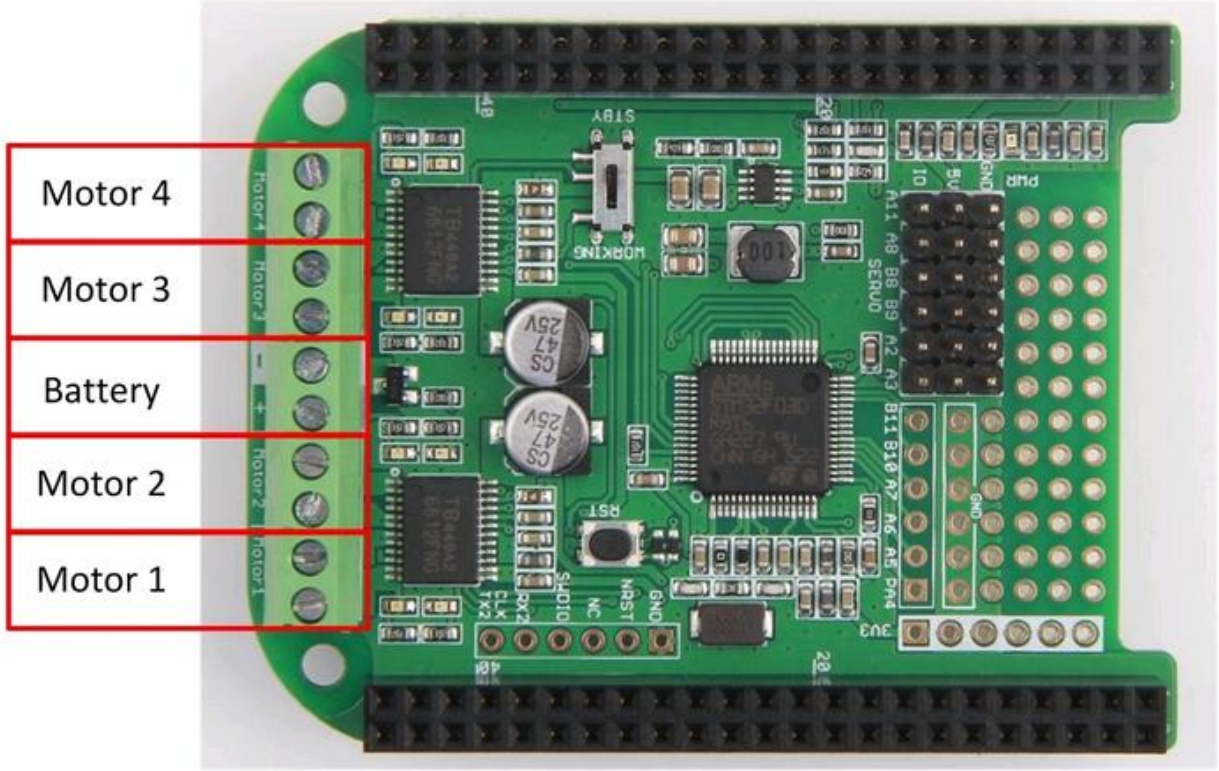
Пример шагового двигателя

Скопируйте следующий код в свой проект и сохраните его как файл python.

```
import MotorBridge
import time

def StepperMotorATest():
    print 'Hello From MotorBridge'
    motor.StepperMotorAInit()
    motor.StepperMotorAMove(1000,1000) # 1000 steppers  1000us every step
    time.sleep(1)
    motor.StepperMotorAMove(-1000,1000) #1000 steppers  1000us every step
    time.sleep(1)

def StepperMotorBTest():
    print 'Hello From MotorBridge'
    motor.StepperMotorBInit()
    motor.StepperMotorBMove(1000,1000) # 1000steppers  1000us every step
    time.sleep(1)
    motor.StepperMotorBMove(-1000,1000) # 1000 steppers  1000us every step
    time.sleep(1)


if __name__=="__main__":
    motor = MotorBridge.MotorBridgeCape()
    StepperMotorATest()
    StepperMotorBTest()
```

Мотор постоянного тока

Интерфейс двигателя постоянного тока **Motor Bridge Cape**, как показано на рисунке ниже.



Функции DC Motor

Вот краткое описание функций двигателя постоянного тока.

DCMotorInit (MotorName, частота)

Описание: Подключите мотор постоянного тока и задайте частоту вращения его вала..

MotorName: 1 ~ 4 обозначает Motor1 ~ Motor4.

Частота: Установите частоту двигателя постоянного тока.

Примечание

Если вы измените частоту двигателя постоянного тока, изменится и частота других моторов постоянного тока.

DCMotorMove (MotorName, Direction, PWMDuty)

Описание: Включение двигателя постоянного тока. Выберете направление и скорость вращения..

MotorName: 1 ~ 4 обозначает Motor1 ~ Motor4.

Направление: 1 означает направление по часовой стрелке. 2 - против часовой стрелки. 3 - Остановите двигатель.

Скорость вращения : 0 ~ 100 означает 0% ~ 100% от максимальной скорости .

motor.DCMotorStop (MotorName)

Описание: Остановка мотора постоянного тока.

Выбор мотора: 1 ~ 4 обозначает Motor1 ~ Motor4.

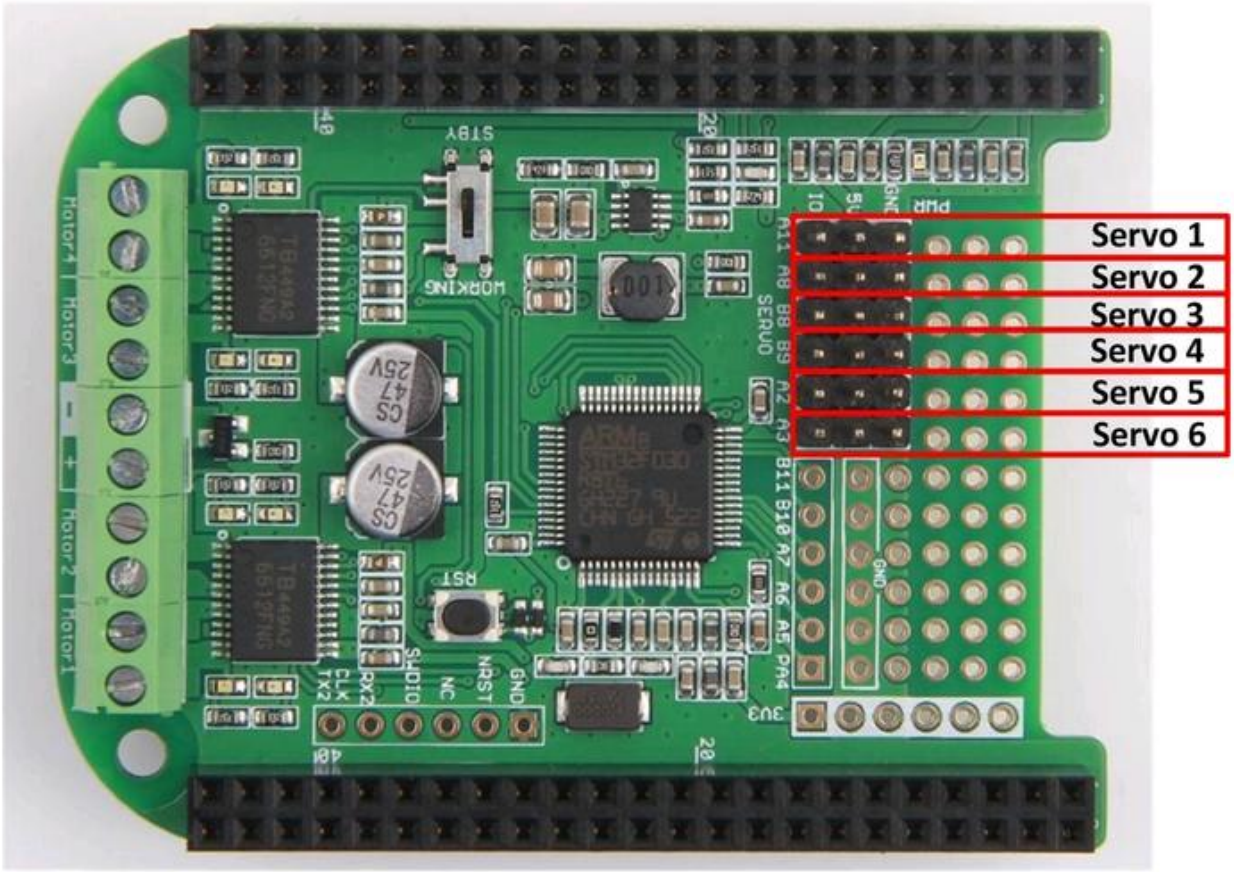
Пример для двигателя постоянного тока

Скопируйте следующий код в свой проект и сохраните его как файл python.

```
import MotorBridge
import time

MotorName        = 1
ClockWise        = 1
CounterClockWise = 2
PwmDuty          = 90
Frequency        = 1000

if __name__=="__main__":
    motor = MotorBridge.MotorBridgeCape()
    motor.DCMotorInit(MotorName,Frequency)
    while True:
        motor.DCMotorMove(MotorName,ClockWise,PwmDuty)
        time.sleep(2)
        motor.DCMotorMove(MotorName,CounterClockWise,PwmDuty)
        time.sleep(2)
        print "hello"
        motor.DCMotorStop(MotorName)
        time.sleep(2)
```

Servo

Интерфейс **Motor Bridge Cape для серводвигателя**, как показано на рисунке ниже.



Функции Servo

Вот краткое описание функций Servo.

Включение сервопривода (Номер, частота)

Описание: Запустите сервопривод и установите частоту.

Номер : 1 ~ 6 означает Servo1 ~ Servo6.

Часто та: установите рабочую частоту сервопривода, значение по умолчанию - 50 Гц.

Угол поворота сервопривода(номер, угол)

Описание: выбор сервопривода. Установите угол сервопривода.

Номер сервопривода: 1 ~ 6 означает Servo1 ~ Servo6.

Угол: 0 ~ 180 обозначает 0 градусов до 180 градусов.

Пример Servo

Скопируйте следующий код в свой проект и сохраните его как файл python.

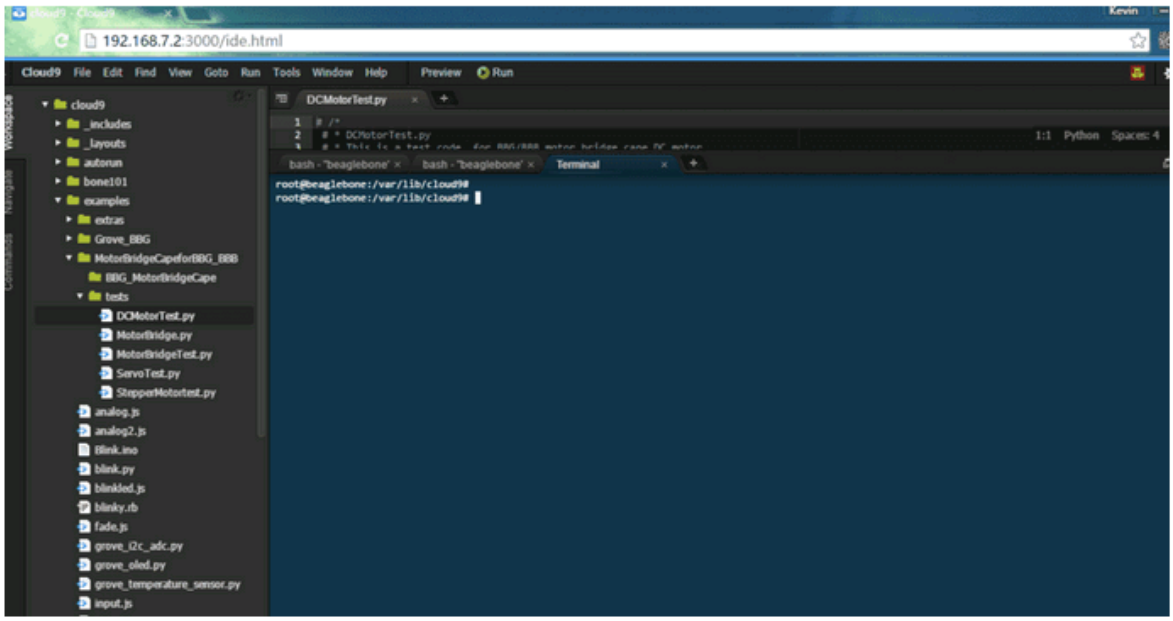
```
import MotorBridge
import time
ServoName   =  2
Frequency   =  50
Angle1      =  20
Angle2      =  160

if __name__=="__main__":
    motor = MotorBridge.MotorBridgeCape()
    motor.ServoInit(ServoName,Frequency)

    while True:
        print 'Servo Test'
        motor.ServoMoveAngle(ServoName,Angle1)
        time.sleep(2)
        motor.ServoMoveAngle(ServoName,Angle2)
        time.sleep(2)
```

Обновление прошивки

Если что-то не так с **Motor Bridge Cape**, попробуйте обновить программу. В этом разделе показано, как обновить прошивку **Motor Bridge Cape** с помощью BeagleBone Green. Он также работает на BBGW и BBB.

1. Вставьте **Motor Bridge Cape** в BBG / BBGW / BBB и подключите BBG к компьютеру через USB-кабель.

2. Подключите BBG к Интернету и получите доступ к нему через SSH, как показано на рисунке ниже.



3.Загрузите код из Github, выполнив следующие команды

```
git clone https://github.com/Seeed-Studio/MotorBridgeCapeFirmware
```

```
root@beaglebone:/var/lib/cloud9#
root@beaglebone:/var/lib/cloud9# git clone https://github.com/Seeed-Studio/MotorBridgeCapeFirmware
Cloning into 'MotorBridgeCapeFirmware'...
remote: Counting objects: 21, done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 21 (delta 7), reused 11 (delta 2), pack-reused 0
Unpacking objects: 100% (21/21), done.
root@beaglebone:/var/lib/cloud9#
```

4. Перейдите к «MotorBridgeCapeFirmware» и выполните команду «make flash» для прошивки.

```
cd MotorBridgeCapeFirmware / && make flash
```

Через несколько секунд вы можете увидеть информацию «Verification OK» с терминала

5. Проверьте прошивку, выполнив следующую команду

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

```
root@beaglebone:/var/lib/cloud9/MotorBridgeCapeFirmware# i2cdetect -y -r 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- 4b -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@beaglebone:/var/lib/cloud9/MotorBridgeCapeFirmware#
```

Если вы найдете I2C адрес 0x4b, это значит, что вы уже обновили прошивку.

Часто задаваемые вопросы

Не удается найти адрес I2C.

В: Я уже обновил прошивку, но не могу определить адрес i2c?

О: Убедитесь, что контакт P9_23 High, так как P9_23 подключается к выводу сброса STM32, я устанавливаю вывод P9_23 на высокий уровень High в функции инициализации класса MotorBridgeCape.

Не удалось обновить прошивку

В: Когда я обновляю прошивку, информация об ошибке говорит, что не удается найти UART2?

В: Вы должны включить BB-UART2, так как BBG начнет прошивку микропрограммы к Motor Bridge Cape по средством UART2 .

vi /boot/uEnv.txt

Затем раскомментируйте «cape_enable = capemgr.enable_partno = BB-UART2». Сохраните и выйдите из редактора, и, наконец, перезагрузите плату.

Motor Bridge Cape все еще не работает ...

Вопрос: Я уже успешно обновил прошивку и могу определить адрес I2C, но почему я все еще не работает?

О: Пожалуйста, отметьте, что на плате установлен переключатель рабочего режима, убедитесь, что переключатель переключен на рабочее состояние. Если у вас остались другие вопросы, пожалуйста, заходите на наш форум.

Не запускается make flash

В: Не удается запустить make flash с кодом ошибки «Не удается инициализировать. Убедитесь, что BOOT0 = 1 BOOT1 = 0 и сброс устройства»

О: Для решения проблемы выполните следующие шаги.

Шаг 1. Запустите sudo nano /boot/uEnv.txt, а затем добавьте ниже 2 строки внизу файла uEnv.txt.

uboot_overlay_addr0=/lib/firmware/BB-UART2-00A0.dtbo
enable_uboot_cape_universal=1

Шаг 2. Перезапустите BBGW

Шаг 3. Запустите Sudo снова.