BeagleConnect Freedom

BeagleBoard.org Foundation
Nov 11, 2024

Table of contents

1 Introduction **3**
- 1.1 What is BeagleConnect™ Freedom? . . . 3
- 1.2 What makes BeagleConnect™ new and different? . . . 4
  - 1.2.1 Plug & Play approach . . . 4
  - 1.2.2 Reliable software update mechanism . . . 4
  - 1.2.3 Rapid prototyping without wiring . . . 5
  - 1.2.4 Long-range, low-power wireless . . . 5
  - 1.2.5 Fully customizable design . . . 5

2 Quick Start Guide **7**
- 2.1 What's included in the box? . . . 7
- 2.2 Attaching antenna . . . 7
- 2.3 Tethering to PC . . . 7
- 2.4 Wireless Connection . . . 7
- 2.5 Access Micropython . . . 9
- 2.6 Demos and Tutorials . . . 9

3 Design **11**
- 3.1 Detailed overview . . . 11
- 3.2 Detailed hardware design . . . 11
  - 3.2.1 LEDs . . . 11
  - 3.2.2 Buttons & Buzzer . . . 11
  - 3.2.3 Sensors . . . 11
  - 3.2.4 mikroBUS . . . 11
  - 3.2.5 USB-C port . . . 13
  - 3.2.6 Buck converter . . . 13
  - 3.2.7 LiPo battery charger . . . 13
  - 3.2.8 Battery input protection . . . 13
  - 3.2.9 MSP430F5503 . . . 13
  - 3.2.10 CC1352P7 . . . 13
  - 3.2.11 SPI Flash . . . 13
  - 3.2.12 Debug interface . . . 13
- 3.3 Mechanical . . . 13

4 Expansion **19**

5 Demos & tutorials **21**
- 5.1 Using Micropython . . . 21
  - 5.1.1 Flashed firmware . . . 21
  - 5.1.2 Examples . . . 21
  - 5.1.3 Updating . . . 24
  - 5.1.4 Contributing . . . 24
- 5.2 Using Zephyr . . . 24
  - 5.2.1 Equipment to begin development . . . 25
  - 5.2.2 Try demo applications . . . 26
- 5.3 Using BeagleConnect Greybus . . . 27
  - 5.3.1 BeagleConnect wireless user experience . . . 27
  - 5.3.2 Components . . . 29

5.3.3 What's different? . . . 30
5.4 Using Arduino Zephyr Template . . . 30
5.4.1 Pin Numbering For BeagleConnect Freedom . . . 30
5.4.2 Setup Arduino workspace . . . 32
5.4.3 Arduino blink code running on BeagleConnect Freedom . . . 33

6 Support **35**
6.1 Production board boot media . . . 35
6.2 Certifications and export control . . . 35
6.2.1 Export designations . . . 35
6.2.2 Size and weight . . . 35
6.3 Additional documentation . . . 35
6.3.1 Hardware docs . . . 35
6.3.2 Software docs . . . 36
6.3.3 Support forum . . . 36
6.3.4 Pictures . . . 36
6.4 Change History . . . 36
6.4.1 Board Changes . . . 36

BeagleConnect™ Freedom is an open-hardware wireless hardware platform developed by BeagleBoard.org and built around the TI CC1352P7 microcontroller, which supports both 2.4-GHz and long-range, low-power Sub-1 GHz wireless protocols. Rapidly prototyping of IoT applications is accelerated by hardware compatibility with over 1,000 mikroBUS add-on sensors, acutators, indicators and additional connectivity and storage options, and backed with software support utilizing the Zephyr scalable and modular real-time operating system, allowing developers to tailor the solution to their specific needs. BeagleConnect Freedom further includes MSP430F5503 for USB-to-UART functionality, temperature and humidity sensor, light sensor, SPI flash, battery charger, buzzer, LEDs, and JTAG connections to make it a comprehensive solution for IoT development and prototyping.

The TI CC1352P7 microcontroller (MCU) includes a 48-MHz Arm Cortex-M4F processor, 704KB Flash memory, 256KB ROM, 8KB Cache SRAM, 144KB of ultra-low leakage SRAM, and over-the-air upgrades (OTA) capability. This MCU provides flexible support for many different protocols and bands making it suitable for many different communication requirements.

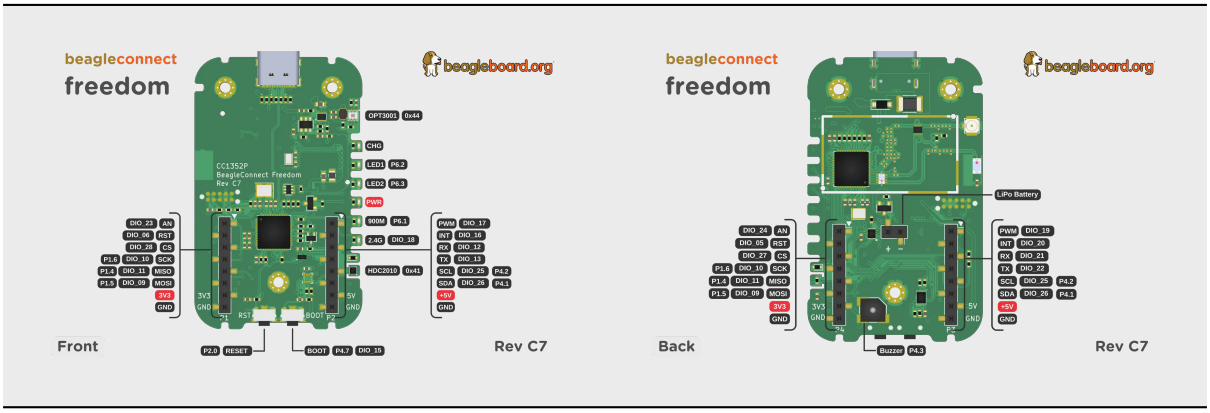
Chapter 1

Introduction

1.1 What is BeagleConnect™ Freedom?

BeagleConnect™ Freedom is based on a TI Arm Cortex-M4 wireless-enabled microcontroller and is the first available BeagleConnect™ solution. It features:

- BeagleConnect™ node device for Bluetooth Low-Energy (BLE) and Sub-GHz 802.15.4 long range wireless,
- Works with BeaglePlay® gateway,
- USB-based serial console and firmware updates,
- 2x mikroBUS sockets,
- On-board light and humidity/temperature sensors,
- Battery-charger circuit, and
- Buzzer, LEDs and buttons for user programming.



1.2 What makes BeagleConnect™ new and different?

1.2.1 Plug & Play approach

BeagleConnect™ uses the collaboratively developed Linux kernel to contain the intelligence required to speak to these devices (sensors, actuators, and indicators), rather than relying on writing code on a microcontroller specific to these devices. Some existing solutions rely on large libraries of microcontroller code, but the integration of communications, maintenance of the library with a limited set of developer resources and other constraints to be explained later make those other solutions less suitable for rapid prototyping than BeagleConnect™.

Linux presents these devices abstractly in ways that are self-descriptive. Add an accelerometer to the system and you are automatically fed a stream of force values in standard units. Add a temperature sensor and you get it back in standard units again. Same for sensing magnetism, proximity, color, light, frequency, orientation, or multitudes of other inputs. Indicators, such as LEDs and displays, are similarly abstracted with a few other kernel subsystems and more advanced actuators with and without feedback control are in the process of being developed and standardized. In places where proper Linux kernel drivers exist, no new specialized code needs to be created for the devices.

Important: BeagleConnect™ solves IoT in a different and better way than any previous solution. For hundreds of devices, users won't have to write a single line of code to add them their systems. The automation code they do write can be extremely simple, done with graphical tools or in any language they want. Maintenance of the code is centralized in a small reusable set of microcontroller firmware and the Linux kernel, which is highly peer reviewed under a highly-regarded governance model.

1.2.2 Reliable software update mechanism

Because there isn't code specific to any given network-of-devices configuration , we can all leverage the same software code base. This means that when someone fixes an issue in either BeagleConnect™ firmware or the Linux kernel, you benefit from the fixes. The source for BeagleConnect™ firmware is also submitted to the Zephyr Project upstream, further increasing the user base. Additionally, we will maintain stable branches of

the software and provide mechanisms for updating firmware on BeagleConnect™ hardware. With a single, relatively small firmware load, the potential for bugs is kept low. With large user base, the potential for discovering and resolving bugs is high.

1.2.3 Rapid prototyping without wiring

BeagleConnect™ utilizes the mikroBUS standard. The mikroBUS standard interface is flexible enough for almost any typical sensor or indicator with hundreds of devices available.

Note: Currently, we have support in the Linux kernel for a bit over 100 Click mikroBUS add-on boards from Mikroelektronika and are working with Mikroelektronika on a updated version of the specification for these boards to self-identify. Further, eventually the vast majority of over 800 currently available Click mikroBUS add-on boards will be supported as well as the hundreds of compliant boards developed every year.

1.2.4 Long-range, low-power wireless

BeagleConnect™ Freedom wireless hardware is built around a TI CC1352P7 multiprotocol and multi-band Sub-1 GHz and 2.4-GHz wireless microcontroller (MCU). CC1352P7 includes a 48-MHz Arm® Cortex®-M4F processor, 704KB Flash, 256KB ROM, 8KB Cache SRAM, 144KB of ultra-low leakage SRAM, and Over-the-Air upgrades (OTA).

1.2.5 Fully customizable design

BeagleConnect™ utilizes open source hardware and open source software, making it possible to optimize hardware and software implementations and sourcing to meet end-product requirements. BeagleConnect™ is meant to enable rapid-prototyping and not to necessarily satisfy any particular end-product's requirements, but with full considerations for go-to-market needs.

Each BeagleBoard.org BeagleConnect™ solution will be:

- Readily available for over 10 years,
- Built with fully open source software with submissions to mainline Linux and Zephyr repositories to aide in support and porting,
- Built with fully open source and non-restrictive hardware design including schematic, bill-of-materials, layout, and manufacturing files (with only the BeagleBoard.org logo removed due to licensing restrictions of our brand),
- Built with parts where at least a compatible part is available from worldwide distributors in any quantity,
- Built with design and manufacturing partners able to help scale derivative designs,
- Based on a security model using public/private keypairs that can be replaced to secure your own network, and
- Fully FCC/CE certified.

Chapter 2

Quick Start Guide

2.1 What's included in the box?

1. BeagleConnect Freedom board in enclosure
2. Antenna
3. USB cable
4. Quick-start card

Todo: Image with what's inside the box and a better description.

Tip: For board files, 3D model, and more, you can checkout BeagleConnect Freedom repository on OpenBeagle.

2.2 Attaching antenna

To connect the SubGHz antenna with SMA connector to the BeagleConnect Freedom you just have to align, place and rotate the antenna clockwise as shown in the image below. To detach the antenna just twist it anti-clockwise.

2.3 Tethering to PC

Todo: Describe how to get a serial connection.

2.4 Wireless Connection

Todo: Describe how to get an IEEE802.15.4g connection from BeaglePlay.

Fig. 2.1: https://youtu.be/bJYZ6PTiV9g

Fig. 2.2: Aattaching antenna to BeagleConnect Freedom

2.5 Access Micropython

Boards come pre-flashed with Micropython. Read *Using Micropython* for more details.

Todo: Describe how to get to a local console and websockets console.

2.6 Demos and Tutorials

- *Using BeagleConnect Greybus*
- *Using Micropython*
- *Using Zephyr*

Chapter 3

Design

3.1 Detailed overview

3.2 Detailed hardware design

3.2.1 LEDs

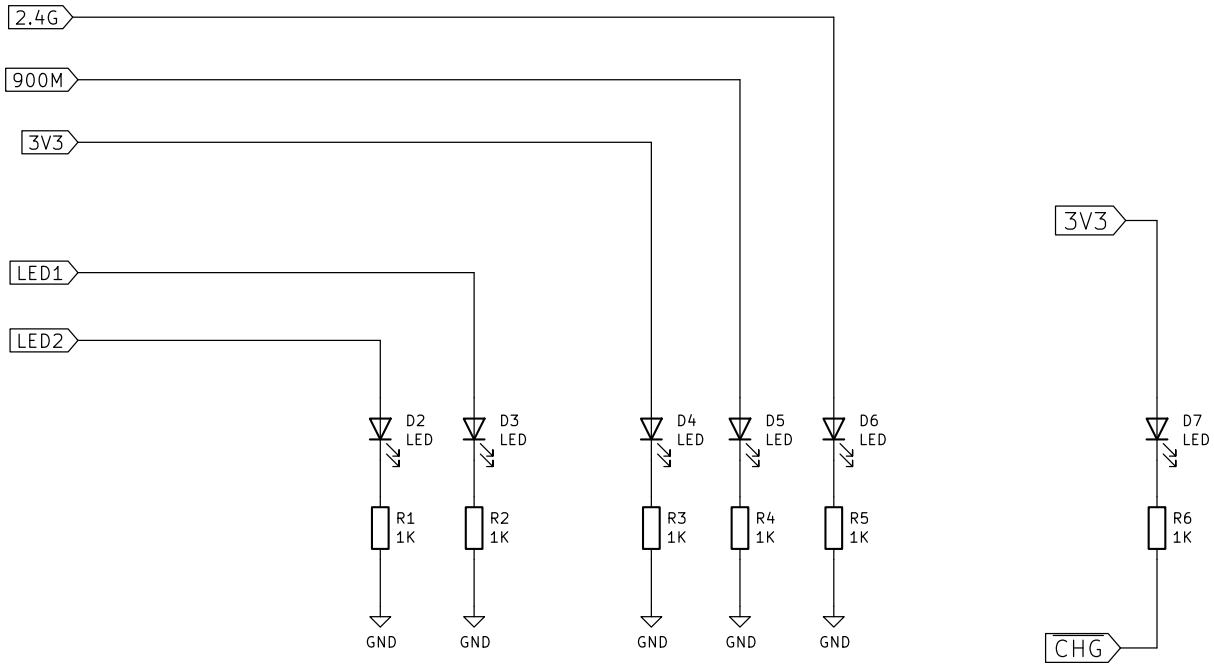


Fig. 3.1: BeagleConnect LEDs

3.2.2 Buttons & Buzzer

3.2.3 Sensors

3.2.4 mikroBUS

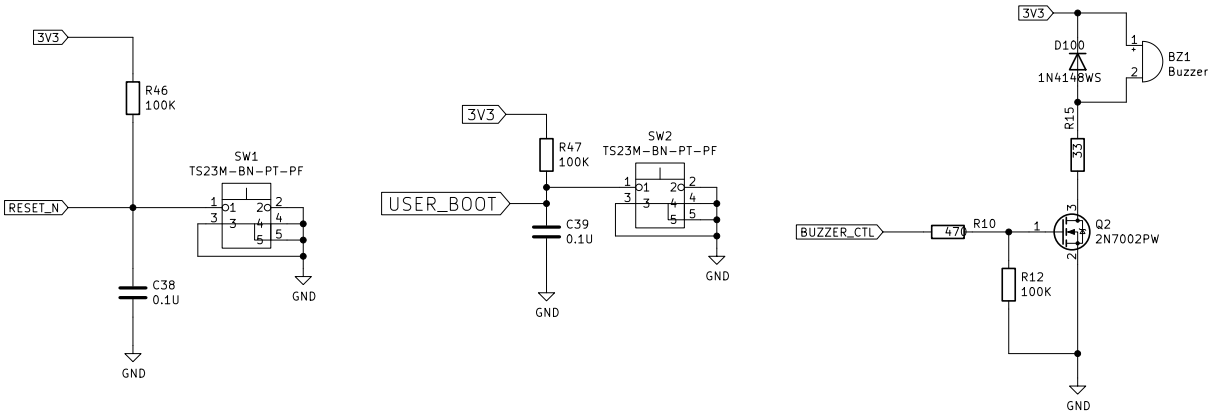


Fig. 3.2: User Input Output (Buttons & Buzzer)

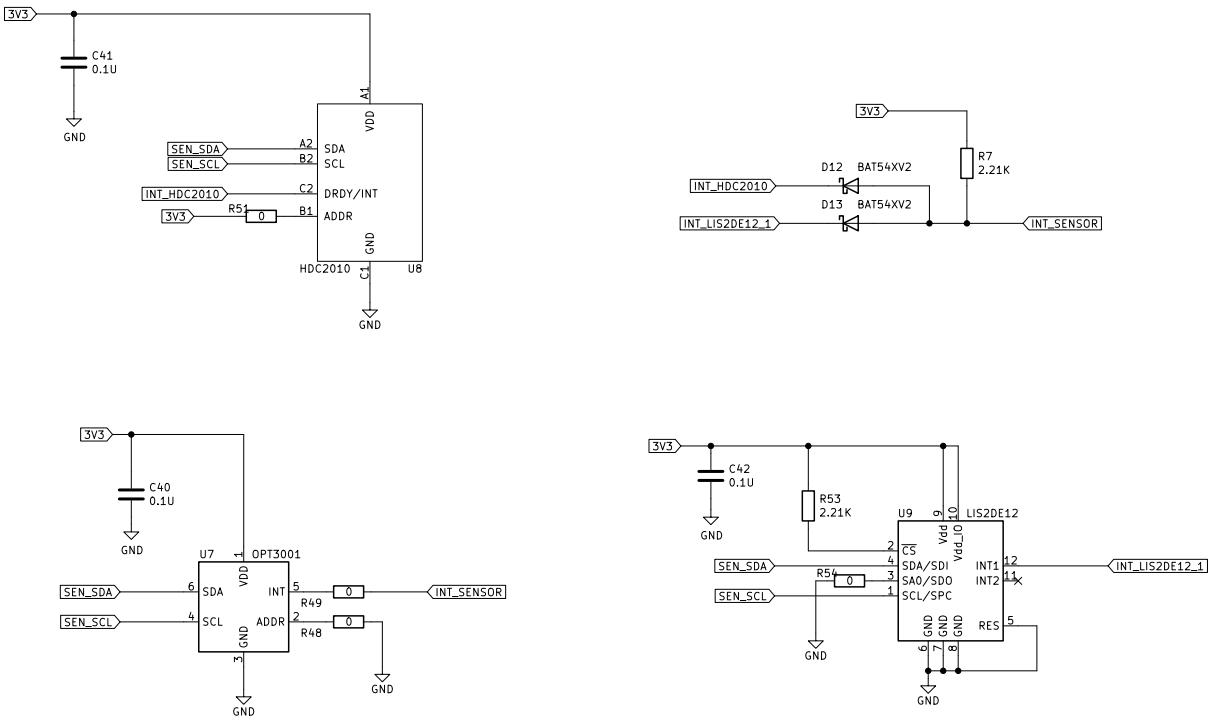


Fig. 3.3: On-board sesnsors

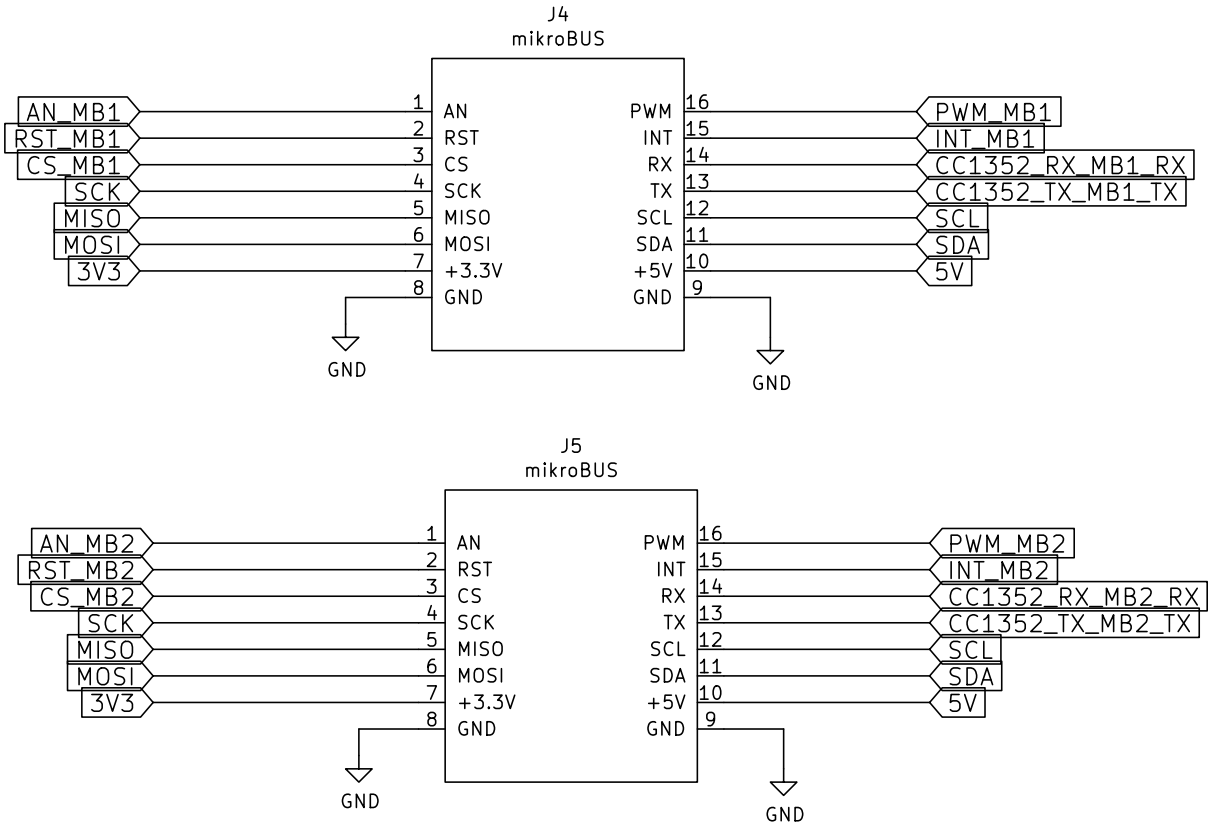


Fig. 3.4: mikroBUS ports

3.2.5 USB-C port

3.2.6 Buck converter

3.2.7 LiPo battery charger

3.2.8 Battery input protection

3.2.9 MSP430F5503

3.2.10 CC1352P7

Digital subsection

Analog subsection

Power subsection

RF subsection

3.2.11 SPI Flash

3.2.12 Debug interface

3.3 Mechanical

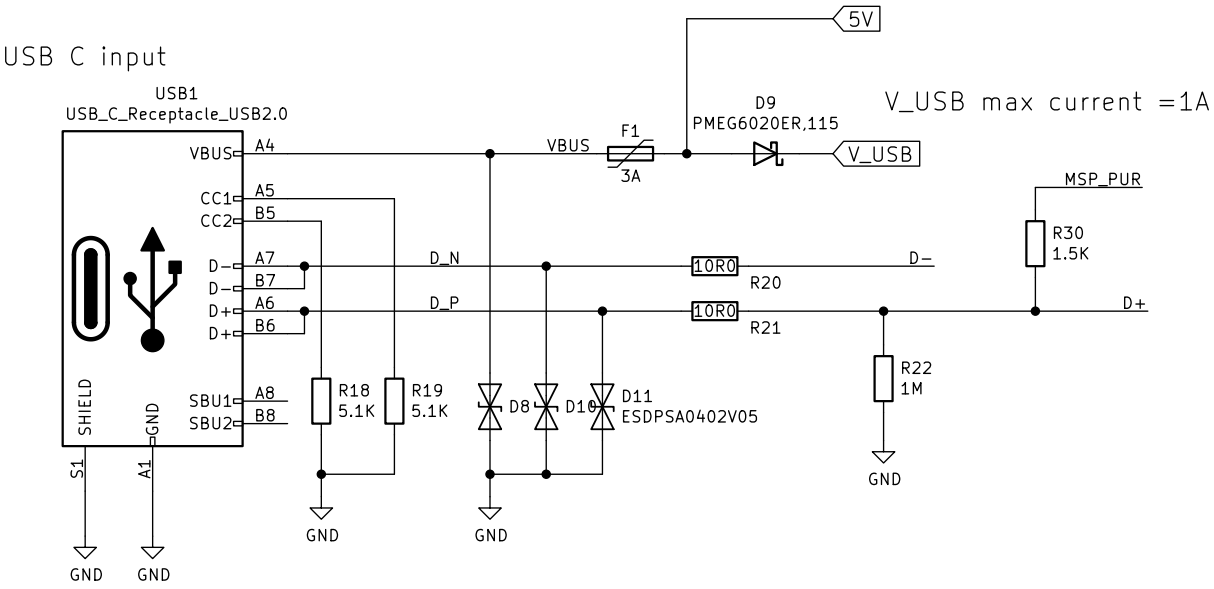


Fig. 3.5: USB-C for power & programming

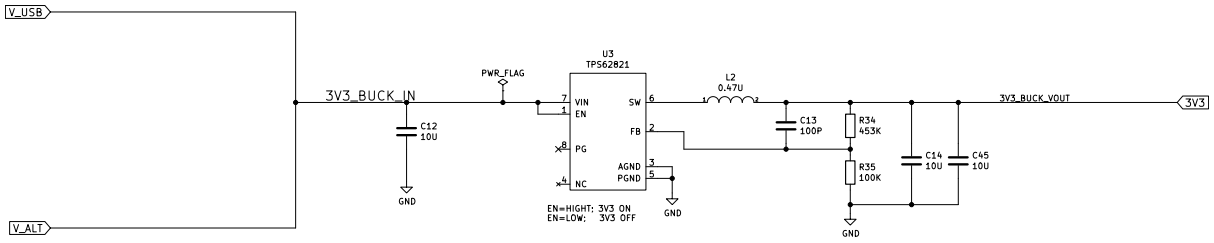


Fig. 3.6: BuckConverter (3.3V output)

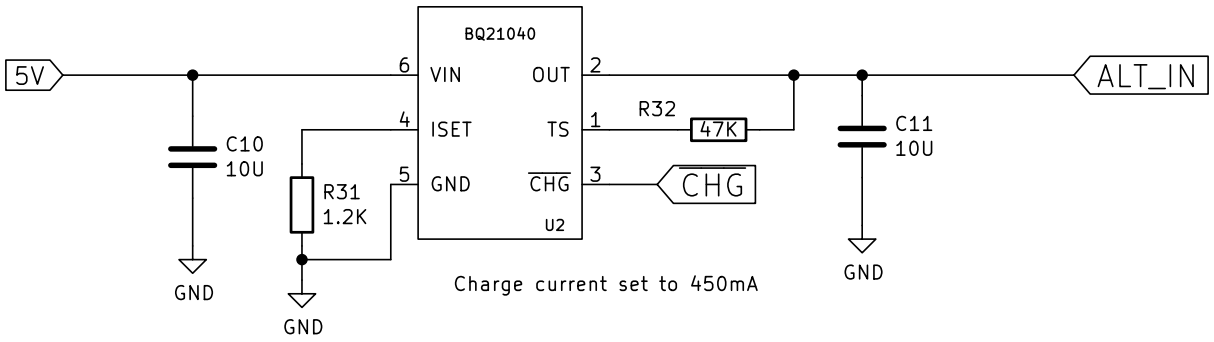


Fig. 3.7: 4.2V LiPo battery charger

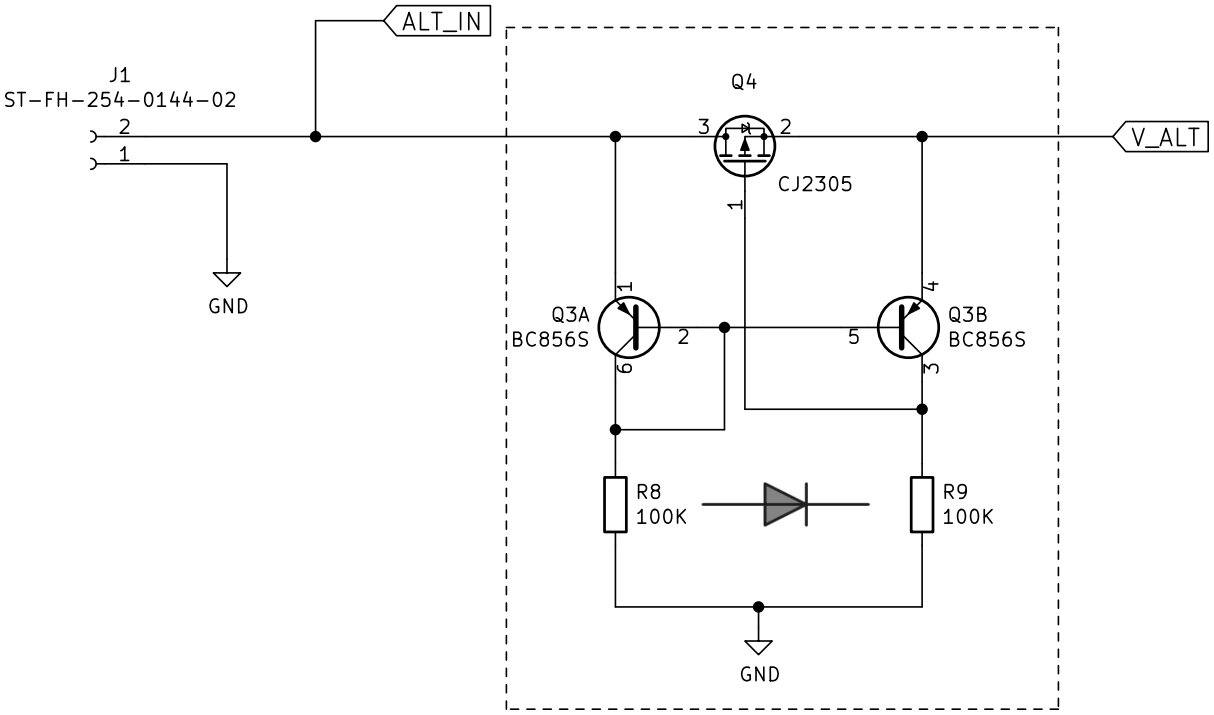


Fig. 3.8: LiPo battery input protection

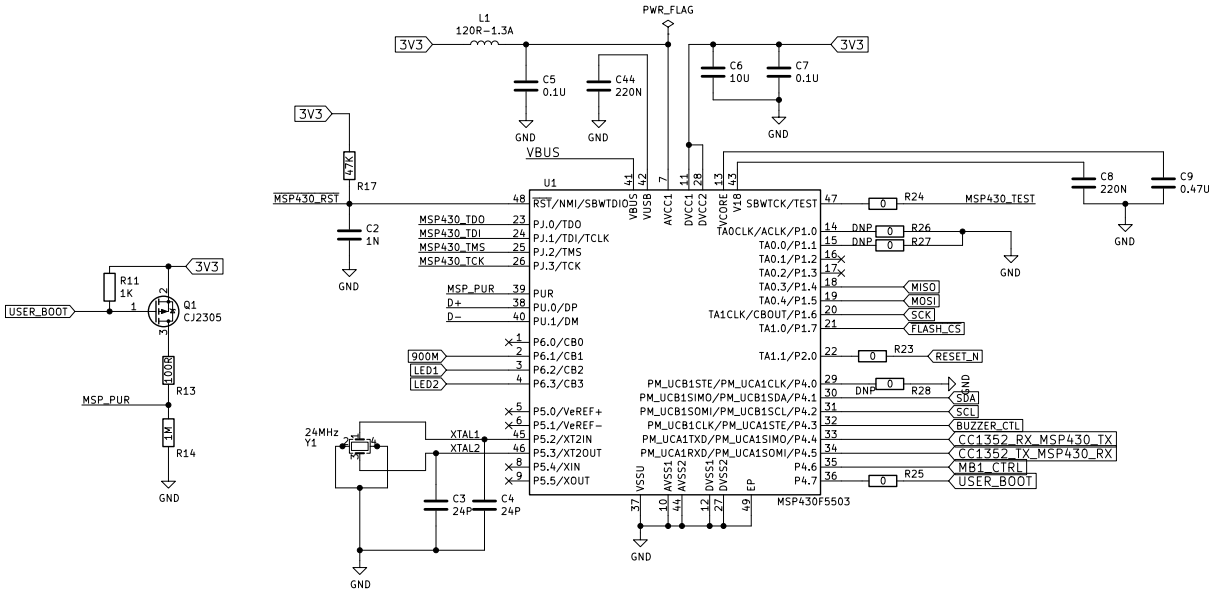


Fig. 3.9: MSP430F5503 (USB to UART & mikroBUS)

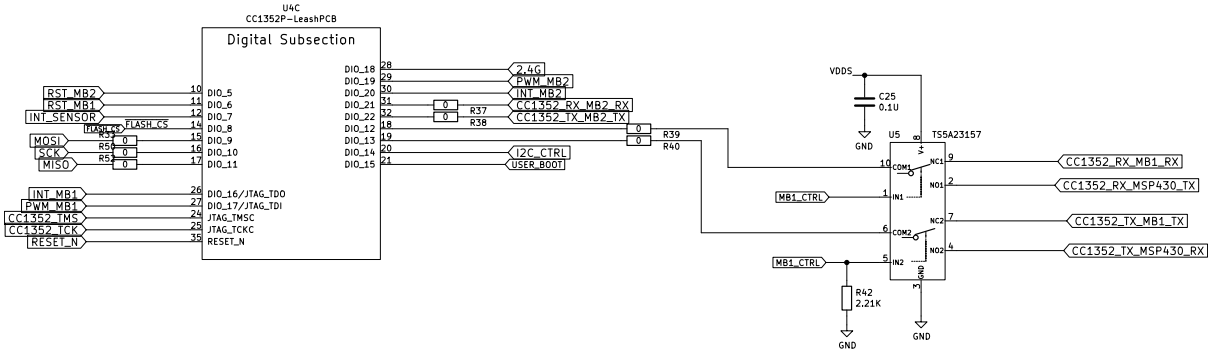


Fig. 3.10: CC1352P7 Digital subsection

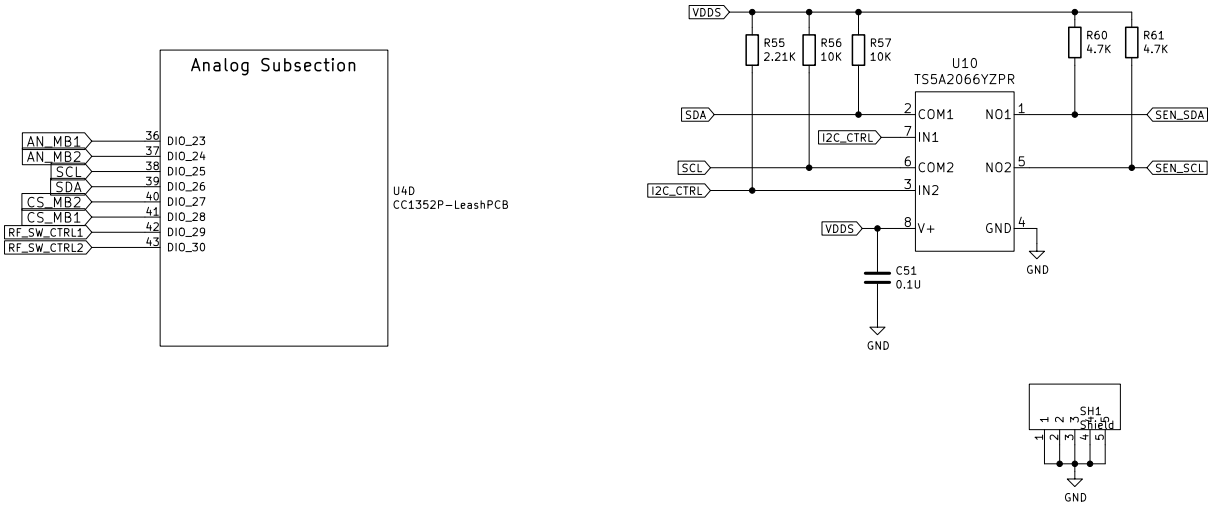


Fig. 3.11: CC1352P7 Analog subsection

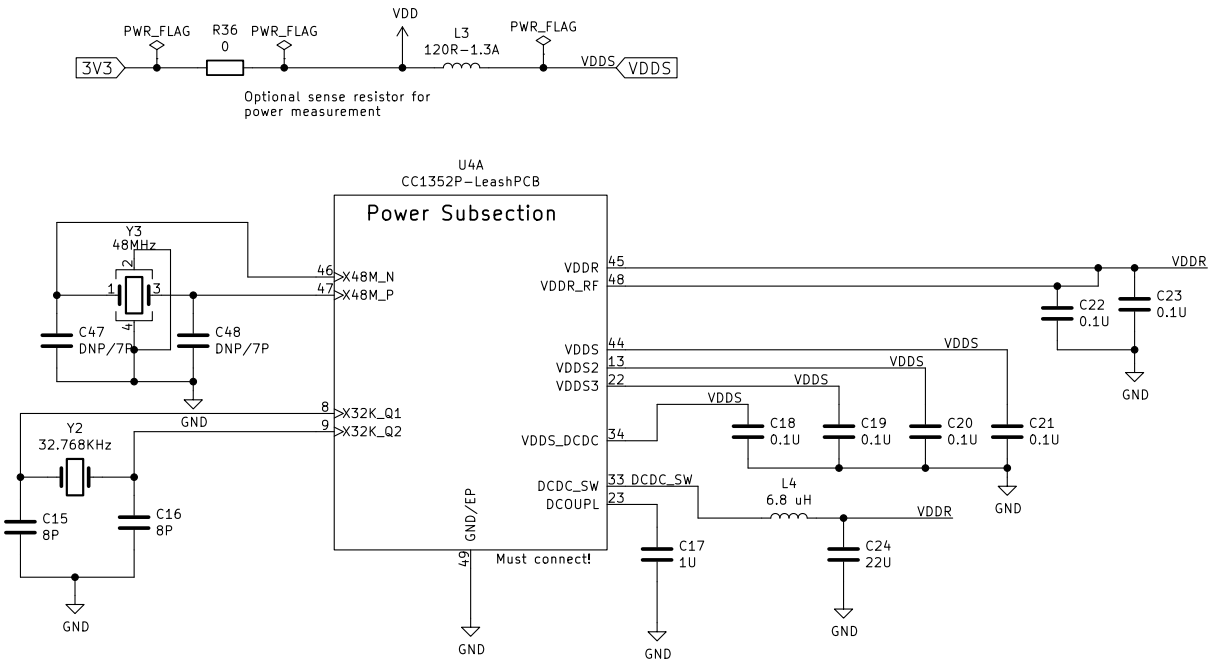


Fig. 3.12: CC1352P7 Power subsection

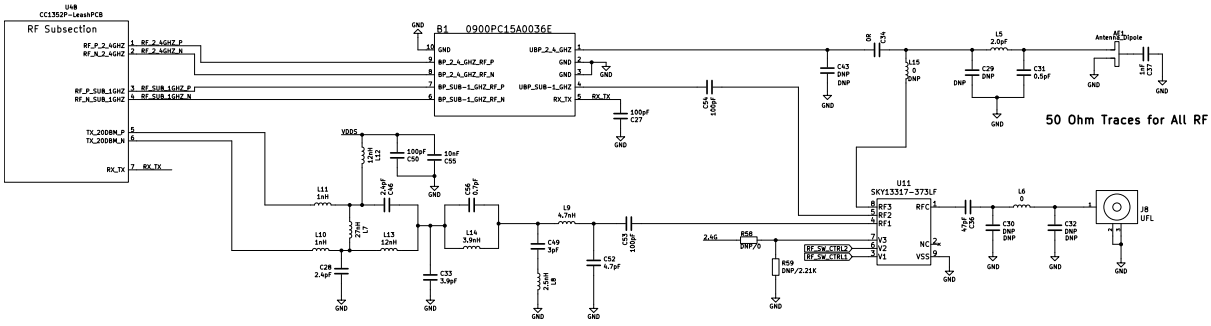


Fig. 3.13: CC1352P7 RF subsection

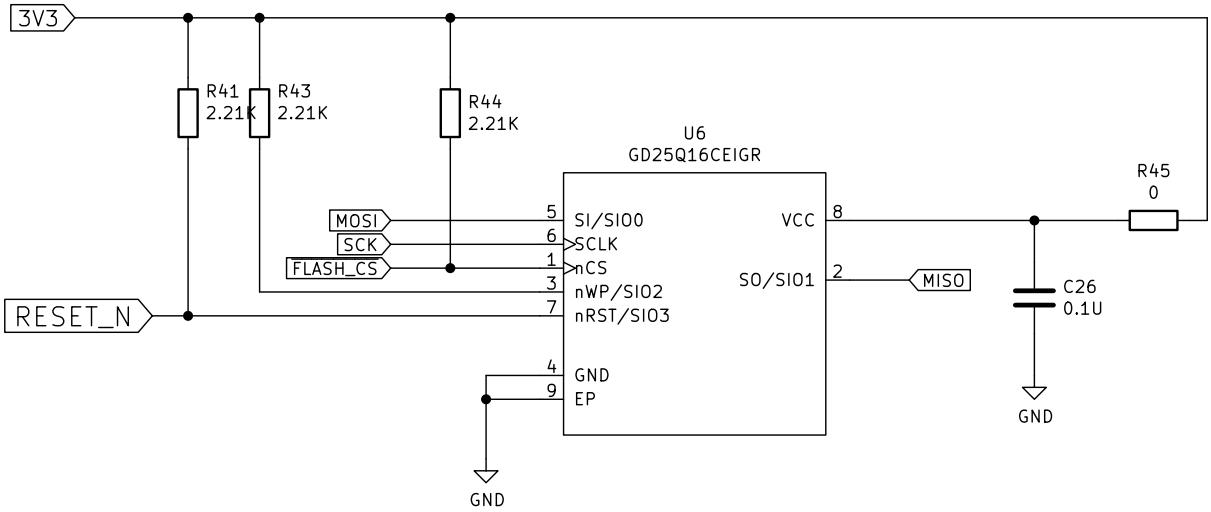


Fig. 3.14: SPIFlash

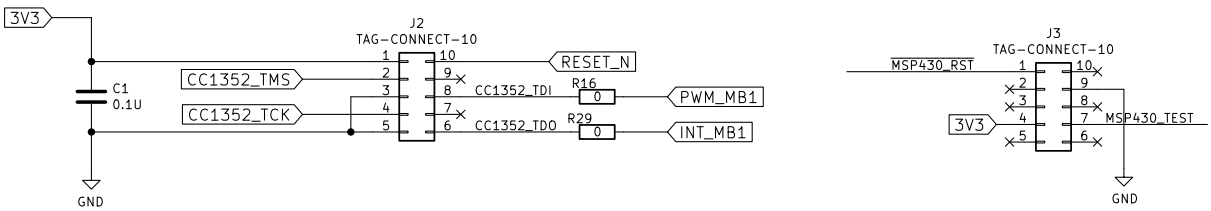


Fig. 3.15: CC1352P7 & MSP430F5503 TagConnect

Chapter 4

Expansion

Todo: Add BeagleConnect Freedom expansion chapter content.

Chapter 5

Demos & tutorials

5.1 Using Micropython

Important: Currently under development

Micropython is a great way to get started developing with BeagleConnect Freedom quickly.

5.1.1 Flashed firmware

BeagleConnect Freedom initial production firmware is release 0.0.3 of our own fork of Micropython.

https://git.beagleboard.org/beagleconnect/zephyr/micropython/-/releases/0.0.3

You can verify this version by using `mcumgr` over a UDP connection or `mcuboot` over the serial console shell.

Latest releases are part of our Zephyr SDK releases.

https://git.beagleboard.org/beagleconnect/zephyr/zephyr/-/releases

5.1.2 Examples

0.0.3

The first boards were flashed with this firmware.

```
debian@BeaglePlay:~$ sudo systemd-resolve --set-mdns=yes --interface=lowpan0
debian@BeaglePlay:~$ avahi-browse -r -t _zephyr._tcp
+ lowpan0 IPv6 zephyr                                        _zephyr._tcp   ␣
↪     local
= lowpan0 IPv6 zephyr                                        _zephyr._tcp   ␣
↪     local
   hostname = [zephyr.local]
   address = [fe80::3265:842a:4b:1200]
   port = [12345]
   txt = []
debian@BeaglePlay:~$ avahi-resolve -6 -n zephyr.local
zephyr.local    fe80::ec0f:7a22:4b:1200
debian@BeaglePlay:~$ mcumgr conn add bcf0 type="udp" connstring=
↪"[fe80::3265:842a:4b:1200%lowpan0]:1337"
Connection profile bcf0 successfully added
debian@BeaglePlay:~$ mcumgr -c bcf0 image list
```

(continues on next page)

(continued from previous page)

```
Images:
 image=0 slot=0
    version: hu.hu.hu
    bootable: true
    flags: active confirmed
    hash: 3697bcef05a6becda7dc14150d46c05dbed5fa78633657b20cf34e1418affee9
Split status: N/A (0)
debian@BeaglePlay:~$ mcumgr -c bcf0 shell exec "device list"
status=0

devices:
- GPIO_0 (READY)
- random@40028000 (READY)
- UART_1 (READY)
- UART_0 (READY)
- i2c@40002000 (READY)
- I2C_0S (READY)
  requires: GPIO_0
  requires: i2c@40002000
- flash-controller@40030000 (READY)
- spi@40000000 (READY)
  requires: GPIO_0
- ieee802154g (READY)
- gd25q16c@0 (READY)
  requires: spi@40000000
- leds (READY)
- HDC2010-HUMIDITY (READY)
  requires: I2C_0S
-
debian@BeaglePlay:~$ mcumgr -c bcf0 shell exec "net iface"
status=0

Hostname: zephyr


Interface 0x20002de4 (IEEE 802.15.4) [1]
========================================
Link addr : 30:65:84:2A:00:4B:12:00
MTU       : 125
Flags     : AUTO_START,IPv6
IPv6 unicast addresses (max 3):
        fe80::3265:842a:4b:1200 autoconf preferred infinite
        2001:db8::1 manual preferred infinite
IPv6 multicast addresses (max 4):
        ff02::1
        ff02::1:ff4b:1200
        ff02::1:ff00:1
debian@BeaglePlay:~$ tio /dev/ttyACM0
[tio 07:32:17] tio v1.32
[tio 07:32:17] Press ctrl-t q to quit
[tio 07:32:17] Connected
gd25q16c@0: SFDP v 1.0 AP ff with 2 PH
I: PH0: ff00 rev 1.0: 9 DW @ 30
I: gd25q16c@0: 2 MiBy flash
I: PH1: ffc8 rev 1.0: 3 DW @ 60
*** Booting Zephyr OS build zephyr-v3.2.0-3470-g14e193081b1f ***
I: Starting bootloader
I: Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Boot source: primary slot
I: Swap type: test
```

(continues on next page)

(continued from previous page)

```
I: Bootloader chainload address offset: 0x20000
I: Jumping to the first image slot


[00:00:00.001,647] <inf> spi_nor: gd25q16c@0: SFDP v 1.0 AP ff with 2 PH
[00:00:00.001,647] <inf> spi_nor: PH0: ff00 rev 1.0: 9 DW @ 30
[00:00:00.001,983] <in
>>>
```

Press reset

```
I: gd25q16c@0: SFDP v 1.0 AP ff with 2 PH
I: PH0: ff00 rev 1.0: 9 DW @ 30
I: gd25q16c@0: 2 MiBy flash
I: PH1: ffc8 rev 1.0: 3 DW @ 60
*** Booting Zephyr OS build zephyr-v3.2.0-3470-g14e193081b1f ***
I: Starting bootloader
I: Primary image: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Scratch: magic=unset, swap_type=0x1, copy_done=0x3, image_ok=0x3
I: Boot source: primary slot
I: Swap type: test
I: Bootloader chainload address offset: 0x20000
I: Jumping to the first image slot


[00:00:00.001,495] <inf> spi_nor: gd25q16c@0: SFDP v 1.0 AP ff with 2 PH
[00:00:00.001,525] <inf> spi_nor: PH0: ff00 rev 1.0: 9 DW @ 30
[00:00:00.001,800] <inf> spi_nor: gd25q16c@0: 2 MiBy flash
[00:00:00.001,831] <inf> spi_nor: PH1: ffc8 rev 1.0: 3 DW @ 60
uart:~$ build time: Feb 22 2023 07:13:09MicroPython v1.19.1 on 2023-02-22;␣
 ↪zephyr-beagleconnect_freedom with unknown-cpu
Type "help()" for more information.
>>> help()
Welcome to MicroPython!

Control commands:
  CTRL-A        -- on a blank line, enter raw REPL mode
  CTRL-B        -- on a blank line, enter normal REPL mode
  CTRL-C        -- interrupt a running program
  CTRL-D        -- on a blank line, do a soft reset of the board
  CTRL-E        -- on a blank line, enter paste mode

For further help on a specific object, type help(obj)

See https://beagleconnect.org/micropython for examples.
>>> import zsensor
>>> light=zsensor.Sensor("OPT3001-LIGHT")
>>> humidity=zsensor.Sensor("HDC2010-HUMIDITY")
>>> light.measure()
>>> light.get_float(zsensor.LIGHT)
35.94
>>> humidity.measure()
>>> humidity.get_float(zsensor.HUMIDITY)
24.32861
>>> humidity.get_float(zsensor.AMBIENT_TEMP)
22.37704
>>> dir(zsensor)
['__name__', 'ACCEL_X', 'ACCEL_Y', 'ACCEL_Z', 'ALTITUDE', 'AMBIENT_TEMP',
 ↪'BLUE', 'CO2', 'DIE_TEMP', 'DISTANCE', 'GAS_RES', 'GREEN', 'GYRO_X', 'GYRO_
 ↪Y', 'GYRO_Z', 'HUMIDITY', 'IR', 'LIGHT', 'MAGN_X', 'MAGN_Y', 'MAGN_Z', 'PM_
 ↪10', 'PM_1_0', 'PM_2_5', 'PRESS', 'PROX', 'RED', 'Sensor', 'VOC', 'VOLTAGE
```

(continues on next page)

(continued from previous page)

```
↪']
>>> import os
>>> with open('/flash/test.txt', 'w') as f:
...     f.write("My test.txt\n")
...     ^H
12
>>> print(open('/flash/test.txt').read())
My test.txt

>>> import socket
>>> sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
>>> sock.bind(('ff02::1', 9999))
>>> for i in range(3):
...     data, sender = sock.recvfrom(1024)
...     print(str(sender) + '  ' + repr(data))
...     ^H
('fe80::ec0f:7a22:4b:1200', <>, 0, 7)  b'4h:32.71;4t:17.29;'
('fe80::ec0f:7a22:4b:1200', <>, 0, 7)  b'2l:0.35;'
('fe80::ec0f:7a22:4b:1200', <>, 0, 7)  b'4h:32.71;4t:17.29;'
>>> import machine
>>> AN=machine.Pin(("GPIO_0", 23), machine.Pin.OUT)
>>> AN.init(machine.Pin.OUT, machine.Pin.PULL_UP, value=1)
>>> LNK_LED=machine.Pin(("GPIO_0", 18), machine.Pin.OUT)
>>> LNK_LED.init(machine.Pin.OUT, machine.Pin.PULL_UP, value=1)
>>> LNK_LED.off()
>>> LNK_LED.on()
>>>
^Tq
[tio 07:40:16] Disconnected
debian@BeaglePlay:~$
```

0.2.2

Todo: Need to describe functionality of 0.2.2

5.1.3 Updating

Look for the latest firmware release on https://www.beagleboard.org/distros or on https://beagleconnect.org.

Download, unzip and flash the `micropython-w-boot` image.

```
wget https://files.beagle.cc/file/beagleboard-public-2021/images/zephyr-
↪beagle-cc1352-0.2.2.zip
unzip zephyr-beagle-cc1352-0.2.2.zip
./build/freedom/cc2538-bsl.py build/freedom/micropython-w-boot
```

5.1.4 Contributing

Repository: https://git.beagleboard.org/beagleconnect/zephyr/micropython

5.2 Using Zephyr

Developing directly in Zephyr will not be ultimately required for end-users who won't touch the firmware running on BeagleConnect™ Freedom and will instead use the BeagleConnect™ Greybus functionality, but is important

for early adopters as well as people looking to extend the functionality of the open source design. If you are one of those people, this is a good place to get started.

5.2.1 Equipment to begin development

There are many options, but using BeaglePlay gives a reasonable common environment. Please adjust as you see fit.

Required

- BeaglePlay with provided antennas
- BeagleConnect Freedom with provided USB cable
- 2x 5V/3A USB power adapters
- USB Type-C cable for use with BeaglePlay

Recommended

- Ethernet cable and Internet connection

Install the SDK on BeaglePlay

You can use BeaglePlay as the primary development board for setting up the Zephyr environment for BeagleConnect Freedom. To setup Zephyr on BeaglePlay, see beagleplay-zephyr-development-setup.

The instructions linked above setup the environment for targeting BeaglePlay's on CC1352. We need to change it to target BeagleConnect Freedom. You can directly go on to Step 5 under *Zephyr setup for BeagleConnect Freedom*.

Todo: note the tested version of software for BeaglePlay

Zephyr setup for BeagleConnect Freedom

To setup Zephyr on your machine for running modules on BeagleConnect Freedom, follow the given instructions :

1. First, install the Zephyr SDK bundle:

```
cd ~
wget https://github.com/zephyrproject-rtos/sdk-ng/releases/
↪download/v0.15.1/zephyr-sdk-0.16.8_linux-aarch64_minimal.tar.gz
wget -O - https://github.com/zephyrproject-rtos/sdk-ng/releases/
↪download/v0.16.8/sha256.sum | shasum --check --ignore-missing
```

Tip: Find the all releases on Zephyr SDK Tags page, change to the version of the sdk according to your requirement. Also, you may replace `aarach64` with the host system you are currently working with.

---

1. Extract the Zephyr SDK bundle:

```
tar xf zephyr-sdk-0.16.8_linux-aarch64_minimal.tar.gz
```

Important: It is recommended to extract the Zephyr SDK bundle at one of the following locations:

```
- $HOME
- $HOME/.local
- $HOME/.local/opt
- $HOME/bin
- /opt
- /usr/local
```

The Zephyr SDK bundle archive contains the `zephyr-sdk-0.16.8` directory and, when extracted under `$HOME` directory, the resulting installation path will be `<$HOME/zephyr-sdk-<version>`.

3. Run the Zephyr SDK bundle setup script:

```
cd zephyr-sdk-0.16.8
./setup.sh
```

4. Further, go on to setup the BeagleConnect Freedom (BCF)'s SDK.

```
west init -m https://github.com/zephyrproject-rtos/zephyr --mr␣
↪sdk zephyr-beagle-cc1352-sdk
cd $HOME/zephyr-beagle-cc1352-sdk
python3 -m venv zephyr-beagle-cc1352-env
```

5. Export the required variables as given below:

```
echo "export ZEPHYR_TOOLCHAIN_VARIANT=zephyr" >> $HOME/zephyr-
↪beagle-cc1352-sdk/zephyr-beagle-cc1352-env/bin/activate
echo "export ZEPHYR_SDK_INSTALL_DIR=$HOME/zephyr-sdk-0.16.8" >>
↪$HOME/zephyr-beagle-cc1352-sdk/zephyr-beagle-cc1352-env/bin/
↪activate
echo "export ZEPHYR_BASE=$HOME/zephyr-beagle-cc1352-sdk/zephyr" >
↪> $HOME/zephyr-beagle-cc1352-sdk/zephyr-beagle-cc1352-env/bin/
↪activate
echo 'export PATH=$HOME/zephyr-beagle-cc1352-sdk/zephyr/scripts:
↪$PATH' >> $HOME/zephyr-beagle-cc1352-sdk/zephyr-beagle-cc1352-
↪env/bin/activate
echo "export BOARD=beagleplay" >> $HOME/zephyr-beagle-cc1352-sdk/
↪zephyr-beagle-cc1352-env/bin/activate
source $HOME/zephyr-beagle-cc1352-sdk/zephyr-beagle-cc1352-env/
↪bin/activate
west update
west zephyr-export
pip3 install -r zephyr/scripts/requirements-base.txt
```

Note: You might need to change the version of the Zephyr-SDK based on the SDK setup done in Step 1.

5.2.2 Try demo applications

Now you can build various Zephyr applications

Build and flash Blinky

Note: Before building any example, ensure to run this command:

```
source $HOME/zephyr-beagle-cc1352-sdk/zephyr-beagle-cc1352-env/bin/activate
```

Run the build and flash commands. Make sure to connect the BeagleConnect Freedom to your computer before flashing.

```
cd $ZEPHYR_BASE
west build zephyr/samples/basic/blinky
west flash
```

Zephyr Documentation

You can refer to Zephyr Getting Started for further development!

Debug applications over the serial terminal

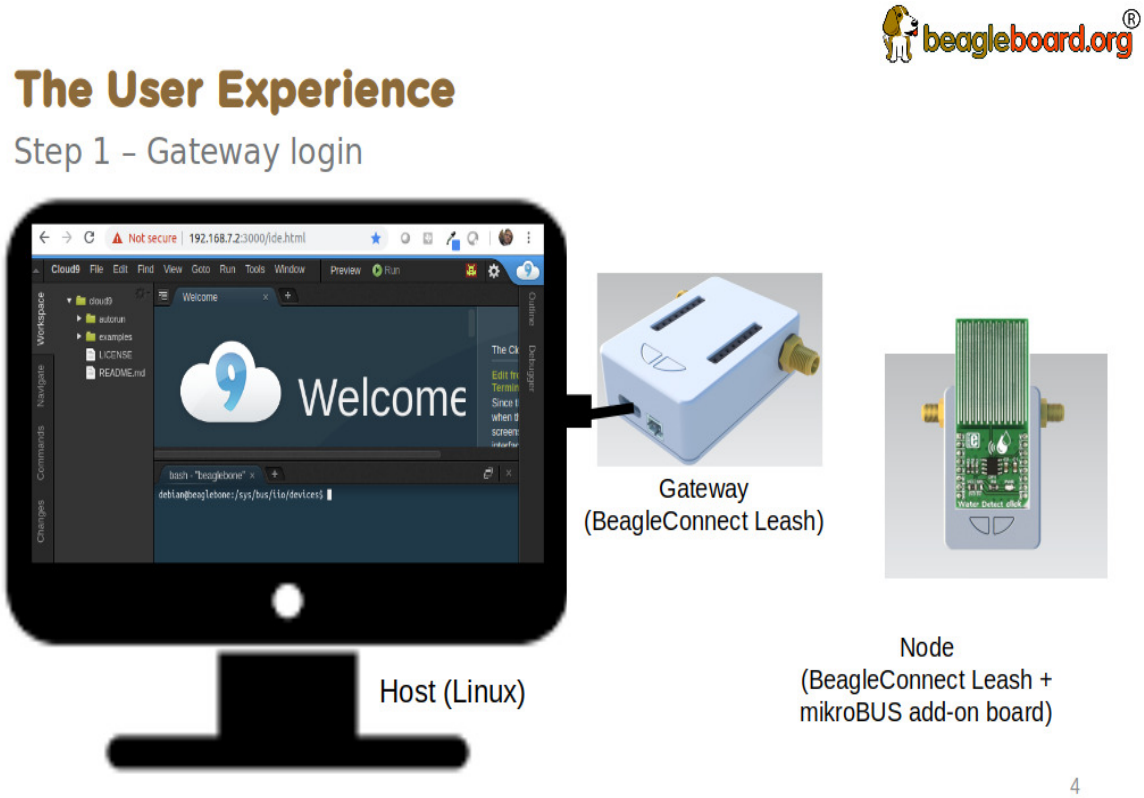
Todo: Add documentation to debug BCF zephyr application over serial terminal.

5.3 Using BeagleConnect Greybus

Note: This is still in development.

5.3.1 BeagleConnect wireless user experience

Enable a Linux host with BeagleConnect



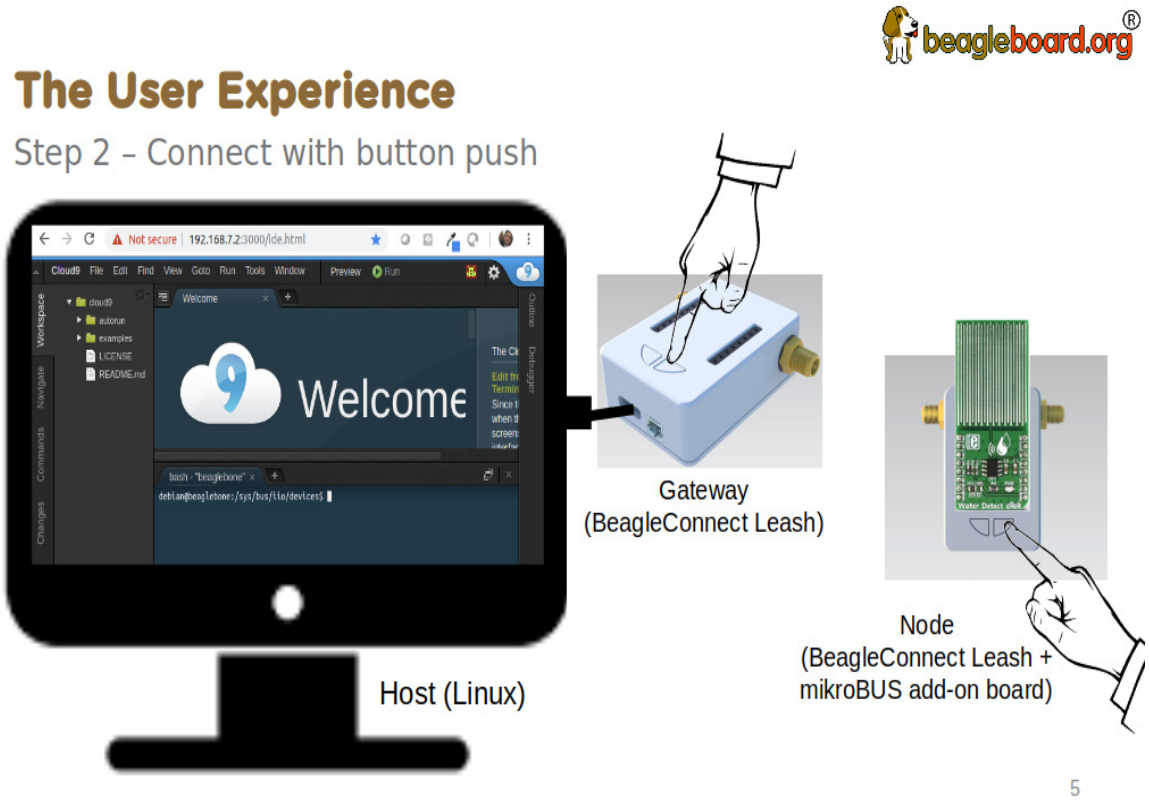
Log into a host system running Linux that is BeagleConnect™ enabled. Enable a Linux host with BeagleConnect™ by plugging a **BeagleConnect™ gateway device** into its USB port. You'll also want to have a **BeagleConnect™ node device** with a sensor, actuator or indicator device connected.

Note: BeagleConnect™ Freedom can act as either a BeagleConnect™ gateway device or a BeagleConnect™ node device.

Important: The Linux host will need to run the BeagleConnect™ management software, most of which is incorporated into the Linux kernel. Support will be provided for BeagleBoard and BeagleBone boards, x86 hosts, and Raspberry Pi.

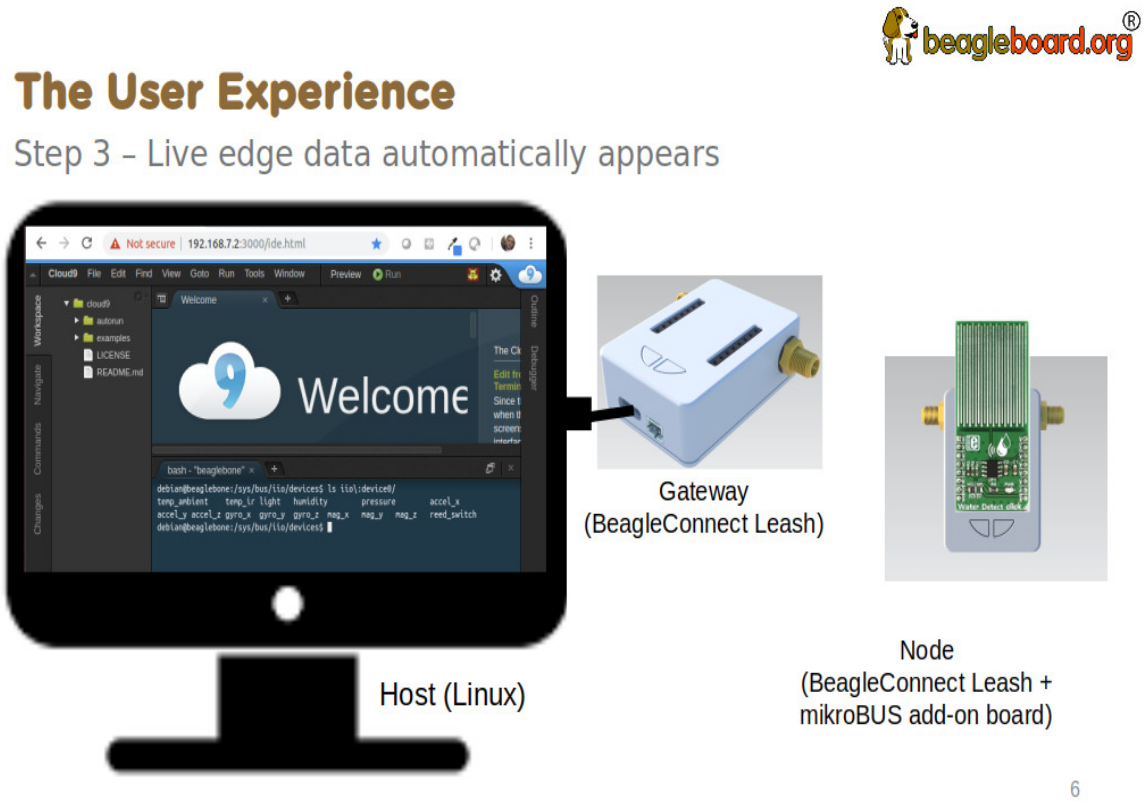
#TODO#: Clean up images

Connect host and device



Initiate a connection between the host and devices by pressing the discovery button(s).

Device data shows up as files



New streams of self-describing data show up on the host system using native device drivers.

High-level applications, like Node-RED, can directly read/write these high-level data streams (including data-type information) to Internet-based MQTT brokers, live dashboards, or other logical operations without requiring any sensor-specific coding. Business logic can be applied using simple if-this-then-that style operations or be made as complex as desired using virtually any programming language or environment.

5.3.2 Components

BeagleConnect™ enabled host Linux computer, possibly single-board computer (SBC), with BeagleConnect™ management software and BeagleConnect™ gateway function. BeagleConnect™ gateway function can be provided by a BeagleConnect™ compatible interface or by connecting a BeagleConnect™ **gateway** device over USB.

Note: If the Linux host has BLE, the BeagleConnect™ **gateway** is optional for short distances

---

BeagleConnect™ Freedom Board, case, and wireless MCU with Zephyr based firmware for acting as either a BeagleConnect™ gateway device or BeagleConnect™ node device.

- In BeagleConnect™ **gateway** device mode: Provides long-range, low-power wireless communications, Connects with the host via USB and an associated Linux kernel driver, and is powered by the USB connector.
- In BeagleConnect™ **node** device mode: Powered by a battery or USB connector Provides 2 mikroBUS connectors for connecting any of hundreds of Click Board mikroBUS add-on devices Provides new Linux host controllers for SPI, I2C, UART, PWM, ADC, and GPIO with interrupts via Greybus

BeagleConnect gateway device

Provides a BeagleConnect™ compatible interface to a host. This could be a built-in interface device or one connected over USB. BeagleConnect™ Freedom can provide this function.

BeagleConnect node device

Utilizes a BeagleConnect™ compatible interface and TODO

BeagleConnect compatible interface

Immediate plans are to support Bluetooth Low Energy (BLE), 2.4GHz IEEE 802.15.4, and Sub-GHz IEEE 802.15.4 wireless interfaces. A built-in BLE interface is suitable for this at short range, whereas IEEE 802.15.4 is typically significantly better at long ranges. Other wired interfaces, such as CAN and RS-485, are being considered for future BeagleConnect™ gateway device and BeagleConnect™ node device designs.

Greybus

Todo: Find a place for the following notes:

- The device interfaces get exposed to the host via Greybus BRIDGED_PHY protocol
- The I2C bus is probed for a an identifier EEPROM and appropriate device drivers are loaded on the host
- Unsupported Click Boards connected are exposed via userspace drivers on the host for development

5.3.3 What's different?

So, in summary, what is so different with this approach?

- No microcontroller code development is required by users
- Userspace drivers make rapid prototyping really easy
- Kernel drivers makes the support code collaborative parts of the Linux kernel, rather than cut-and-paste

5.4 Using Arduino Zephyr Template

The Arduino Core API module for zephyr leverages the power of Zephyr under an Arduino-C++ style abstraction layer thus helping zephyr new-comers to start using it without worrying about learning new APIs and libraries.

Using this template you can run arduino code on your BeagleConnect Freedom.

5.4.1 Pin Numbering For BeagleConnect Freedom

You will see pins over `Mikrobus-1` and `Mikrobus-2` that can be used for Arduino code. You can set each pin to either **read signals (input)** for things like buttons & sensors or you can set them to **send signals (output)** to things like LEDs and motors. This lets you interact with and control the physical world using Arduino code on BeagleConnect Freedom board.

Commonly used GPIOs are specified:

- **D0-D18** Digital GPIO pins
- **A0-A5** ADC GPIO pins
- **D2** and **D6** PWM GPIO pins

Reference to all GPIO pins are shown in the below images.

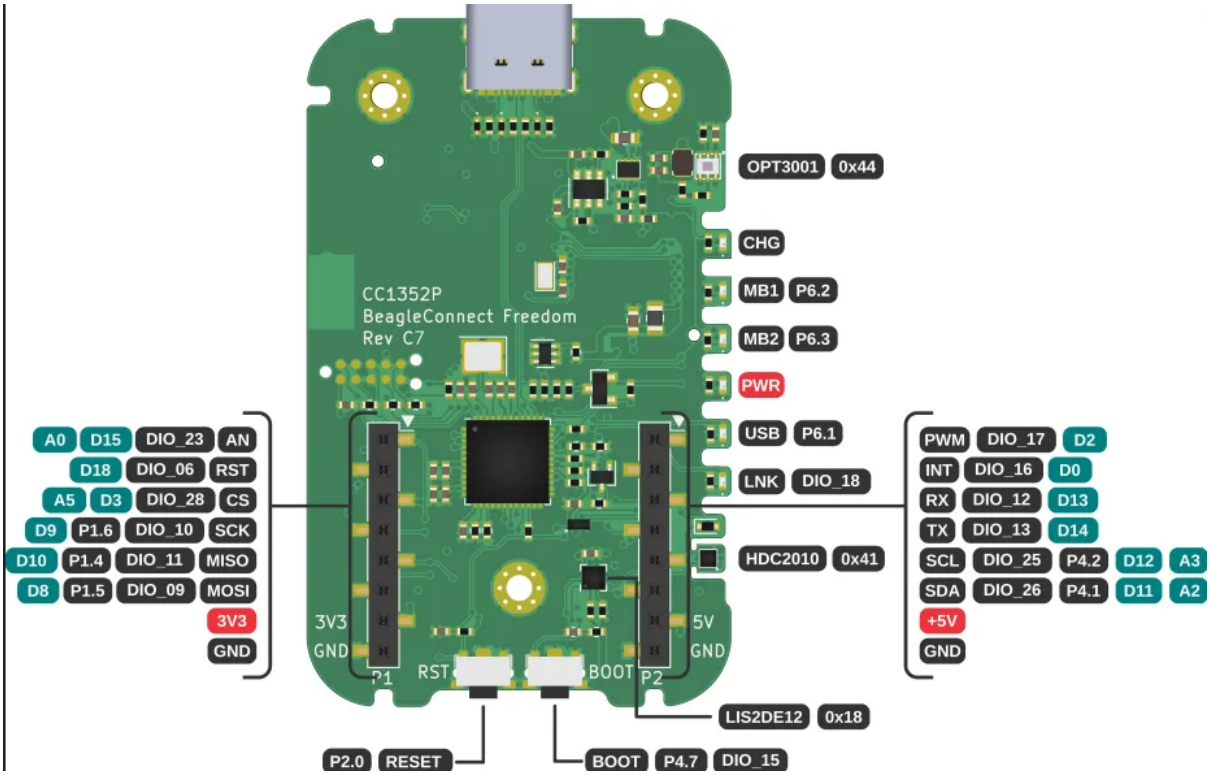


Fig. 5.1: Mikrobus-1 front annotated arduino pinout

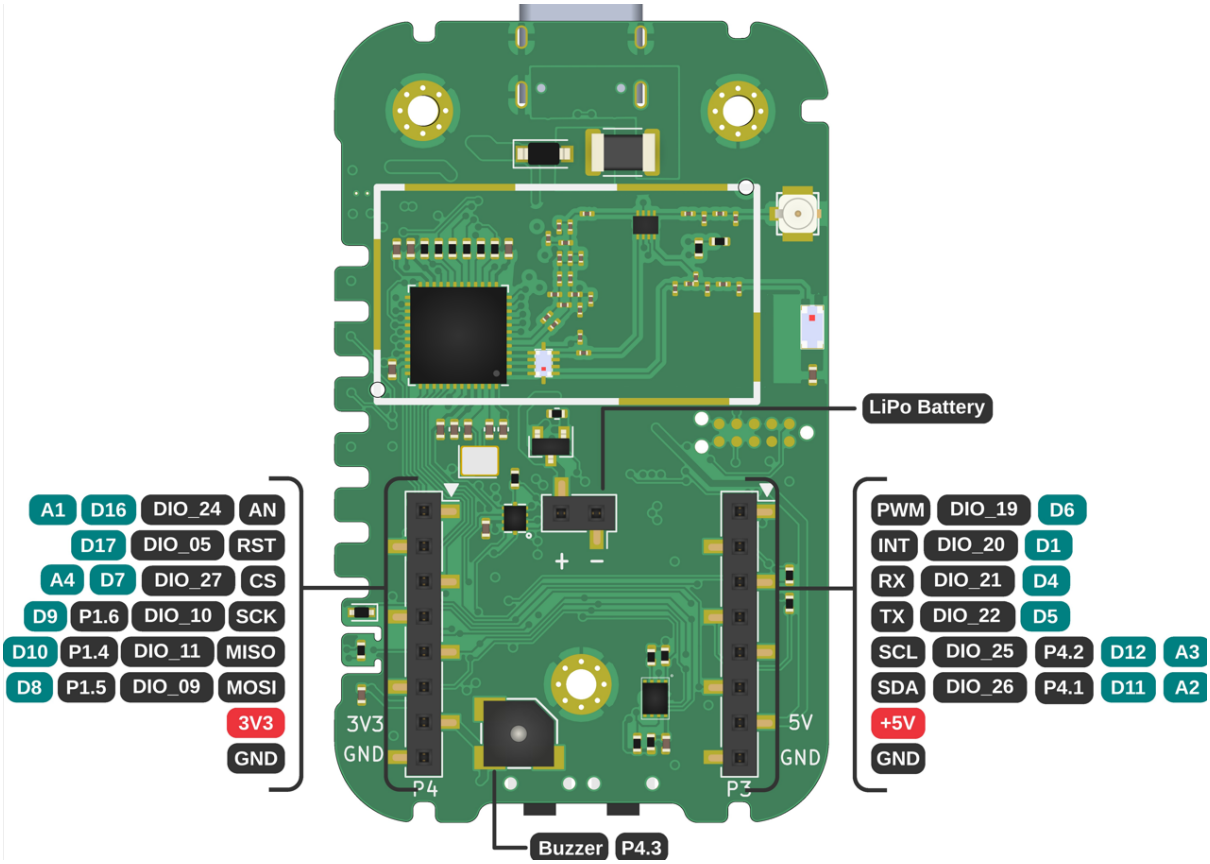


Fig. 5.2: Mikrobus-2 back annotated arduino pinout

5.4.2 Setup Arduino workspace

If this is your first time using zephyr, Install Zephyr SDK and install `cc1352-flasher` using command `pip install cc1352-flasher` then reboot your system before following the steps below.

1. Create a workspace folder:

```
mkdir arduino-workspace
cd arduino-workspace
```

2. Setup virtual environment

```
python -m venv .venv
source .venv/bin/activate
pip install west
```

3. Setup Zephyr app:

```
west init -m https://openbeagle.org/beagleconnect/arduino-zephyr-template .
west update
```

4. Setup Arduino module

```
ln -srf modules/lib/ArduinoCore-API/api modules/lib/Arduino-Zephyr-API/cores/
↪arduino/.
```

5. Install python deps

```
pip install -r zephyr/scripts/requirements-base.txt
```

Arduino Code

You can find `main.cpp` file in the directory `arduino-workspace/arduino-zephyr-template/src/` which contains your arduino code. The default code prints `Hello World` on the serial monitor. Since you are already in the `arduino-workspace` directory, then proceed with writing the following code.

```
nano arduino-zephyr-template/src/main.cpp
```

Listing 5.1: main.cpp

```
#include <Arduino.h>

void setup() {
    Serial.begin(115200);
}

void loop() {
    Serial.println("Hello World");
    delay(5000);
}
```

Press `CTRL+O` and `ENTER` to save, `CTRL+X` to exit.

Important: You must start your `main.cpp` code with `#include <Arduino.h>`.

Build the Arduino directory

Before flashing, run the command below to build the `arduino-zephyr-template` for the board `beagleconnect_freedom`.

```
west build -b beagleconnect_freedom arduino-zephyr-template -p
```

Note: Only if you are following the steps from the beginning then the above command will work. Otherwise, make sure that you are in the `arduino-workspace` directory and setup virtual environment using command `source .venv/bin/activate`.

Flash BeagleConnect Freedom

Make sure that your BeagleConnect Freedom is connected with your linux system via USB.

```
west flash
```

Serial Output

Considering your BeagleConnect Freedom is connected to `/dev/ttyACM0` you can see the serial output coming from your BeagleConnect Freedom.

```
tio /dev/ttyACM0
```

5.4.3 Arduino blink code running on BeagleConnect Freedom

For BeagleConnect Freedom LNK LED will work as `LED_BUILTIN` in Arduino code.

First you have to modify `main.cpp` located in the directory `arduino-workspace/arduino-zephyr-template/src/` created at the time of setup.

Listing 5.2: main.cpp

```
#include <Arduino.h>

void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage␣
↪level)
delay(1000);                      // wait for a second
digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage␣
↪LOW
delay(1000);                      // wait for a second
}
```

Note: For specifying high and low output states, use `HIGH` and `LOW`. Avoid using boolean `1` and `0` as they may not be compatible.

Before doing `Build` and `Flash`, you must activate the virtual environment in the `arduino-workspace` directory which has been created earlier.

```
source .venv/bin/activate
```

Now, execute the build command.

```
west build -b beagleconnect_freedom arduino-zephyr-template -p
```

Make sure your BeagleConnect Freedom is connected to your linux system via USB.

Finally, flash using the command below. The LNK LED of BeagleConnect will start blinking after flashing is complete.

```
west flash
```

Tip: You can try more Arduino examples on BeagleConnect Freedom.

Chapter 6

Support

All support for BeagleConnect Freedom design is through BeagleBoard.org community at BeagleBoard.org forum.

6.1 Production board boot media

- BeagleConnect Freedom Rev C7

6.2 Certifications and export control

6.2.1 Export designations

- HS: 8471504090
- US HS: 8473301180
- EU HS: 8471707000

6.2.2 Size and weight

- Bare product dimensions (without antenna): 63 x 56 x 16.6 mm
- Bare product weight (with antenna): 53.2 g
- Full package dimensions: 188 x 85 x 35 mm
- Full package weight: 95.2 g

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 BeagleConnect Freedom repository.

6.3.2 Software docs

For BeagleConnect Freedom specific software projects you can checkout all the BeagleConnect 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/bcf

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://git.beagleboard.org/beagleconnect/freedom. Versions released into production are noted below.

Table 6.1: BeagleConnect Freedom board change history

Rev	Changes	Date	By
C7	Initial production version	2023-03-08	JK