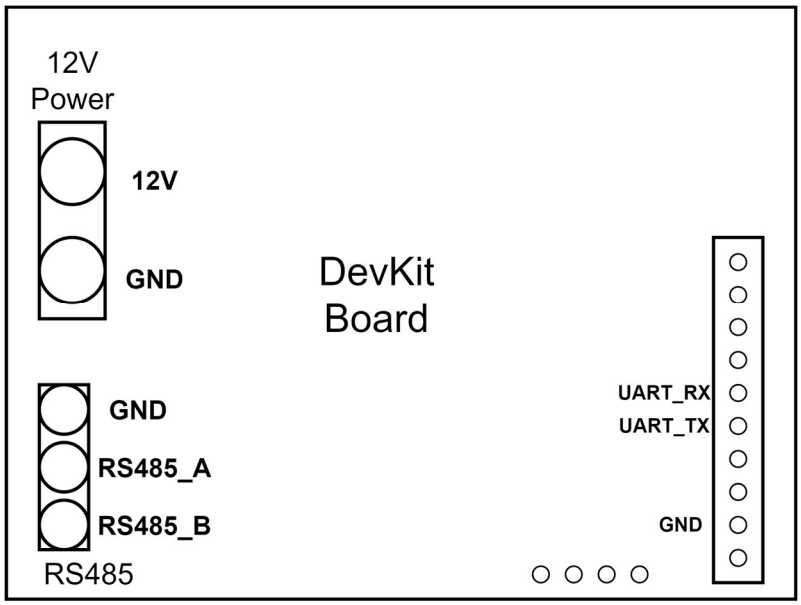
Bus Mini V3 DevKit Guide

Hardware Interface

- 12V Power Slot
- UART Pins
- RS485 Slot



Power On

The behaviour for powering on is as such:

1. LED begins in an OFF state.
2. The system will configure the sensors. This will take about 5 seconds.
3. Once configured, the LED should be ON.
4. Various other initialisations occur, and the background is stored.
5. The LED should be OFF now.
6. The process runs.

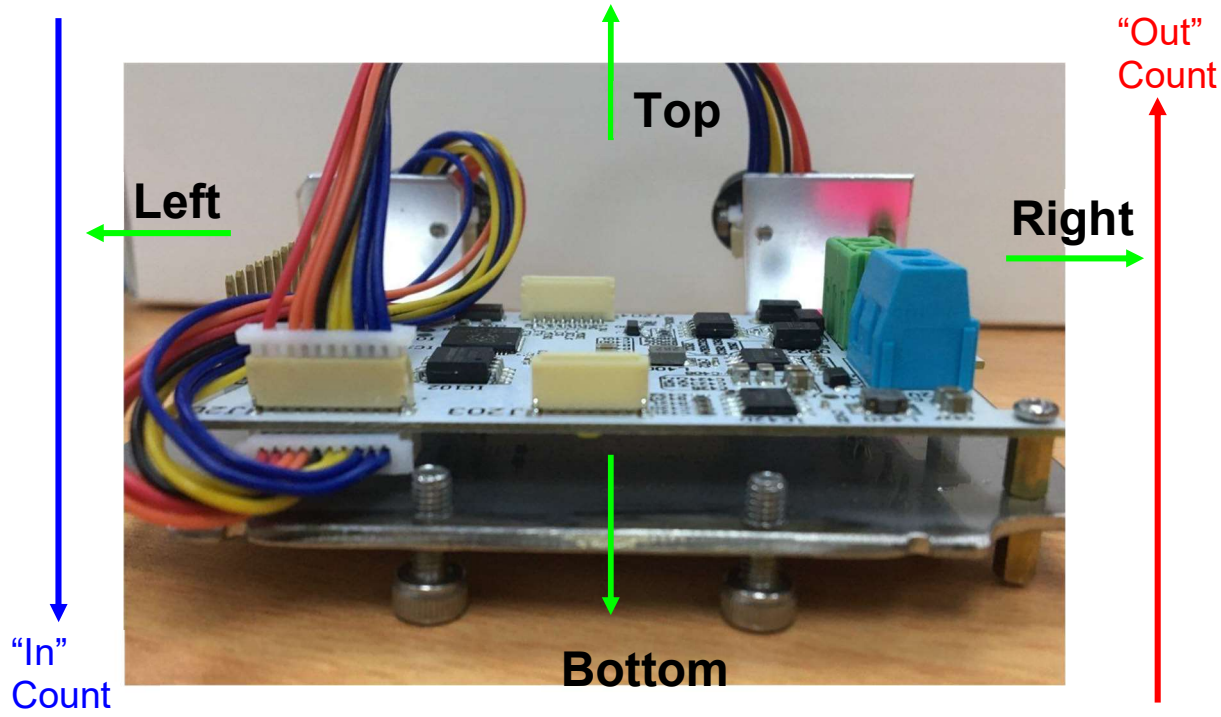
In short, you should expect the LED behaviour on start-up as such:
OFF->ON->OFF
The entire start-up process should take around 10-15 seconds.
After the start-up, the tracking and counting process will begin.
If this behaviour is not observed, you may have a faulty board.

Data Output

The RS485 interface is configured as such: 115200 8N1
For any counting event triggered, the data is output via the RS485. The format is as such:

0x41 0x41	"In" Count	"Out" Count
2 bytes header	1 byte	1 byte

"In" Count refers to when there is a crossing from the top to the bottom.
"Out" Count refers to when there is a crossing from the bottom to the top.
See the image below for orientation reference.



Once the data is received, a response of **0x00** must be sent back to acknowledge reception. If this response is not sent back, subsequent counts will accumulate until the current data is acknowledged.

NOTE: No timestamp is provided and it is the developer's responsibility to keep a tally of the counts.

Examples

- If an 'in' count is triggered, the payload to receive is: **0x41 0x41 0x01 0x00**
- If an 'out' count is triggered, the payload to receive is: **0x41 0x41 0x00 0x01**
- Sometimes you may get simultaneous 'in' and 'out' triggers, say 2 'in' and 1 'out'. In that case, the payload to receive is: **0x41 0x41 0x02 0x01**

In all cases, you must send back a **0x00** to acknowledge them.

Configuration

Connect the UART Pins to your workstation.
The UART is configured as such: 115200 8N1
The device is little-endian.
You may read and write configurations to the device via the UART.

The format of the UART packet is as such:

Address	Read/Write	Value
1 byte	1 byte	2 bytes

The 2 bytes value at the end only matters if you are writing to the address.

The format of the response is as such:

0x46 0x46	Value
2 bytes header	2 bytes

Possible configurations

Address	Type	R/W?	Min	Max
0x0	Ceiling Height Configuration (mm)	R,W	500	5000
0x1	Height Threshold Value (mm)	R,W	500	5000
0xA	Signal Threshold (kpcs/spad)	R,W	0	3000

Example

Assume that the ceiling height value is 2000 mm.
To read the ceiling height value, send: **0x00 0x00 0x00 0x00**
it should return: **0x46 0x46 0xD0 0x07**
Decoding 0xD0 0x07 (note that it is little-endian) will give 2000.

To write a ceiling height value of 2200 mm,
2200 encodes to 0x98 0x08.
Then, send: **0x00 0x01 0x98 0x08**.
it should return: **0x46 0x46 0x98 0x08**, to indicate successful write.

Commands

Commands work similarly to configurations, but the return value will always be zero.

Address	Type	R/W?	Min	Max
0x4	**Reset Trackers**	W	1	1
0x5	**Update Background**	W	0	100
0x9	**Get Sensor data + Tracker Data**	W	0	0

Reset Trackers

To reset the trackers, send **0x04 0x01 0x01 0x00**.
it should return: **0x46 0x46 0x00 0x00**
Nothing will be observed.

Update Background

To update the background, send **0x05 0x01 0x64 0x00**.
it should return: **0x46 0x46 0x00 0x00**
The LED will light up, indicating that the background is being updated.
Then, the LED will turn off

Get Sensor Data + Tracker Data

To update the background, send **0x09 0x01 0x00 0x00**.
it should return: **0x46 0x46 0x00 0x00**.

Then, you must read 256 bytes for the 16x8 sensor data.
Each value is an Int16 value and the data is transmitted row-by-row (i.e. row major order).

Then, read 6 x 8 bytes for the tracker data. Each tracker data is as follows:

x position	y position	new status	active status	excluded status	-
2 bytes, int16	2 bytes, int16	1 byte, bool	1 byte, bool	1 byte, bool	1 byte padding

The x position is in the range of [0,76], while the y position is in the range of [0,36]. Hence, to properly plot the tracker data, you will need to upscale the sensor data to 76x36.

The *new* status refers to whether the tracker is newly initialised.
The *active* status refers to whether the tracker is currently active.
The *excluded* status refers to whether the tracker has been excluded from counting.

Example Scripts

Three Python scripts are provided as examples. All scripts require pyserial and numpy. Minimum Python version tested is 3.8.10. You may check the requirements.txt for the list of requirements for all the scripts if you wish to run them.

All the scripts are tested on a Raspberry Pi 4B+ using the built-in TTY Serial interface for the UART interface and a RS485-to-USB Serial Converter for the RS485 interface.

uart_config.py
Usage: python uart_config.py <UART TTY device> <address> <r/w> <data>
Demonstrate sending configurations and commands via UART

rs485_read.py
Usage: python rs485_read.py <RS485 TTY device>
Read event data via RS485 and print to stdout

flask_dual_test_aio.py
Additional requirements: OpenCV and Flask
Usage: python flask_dual_test_aio.py <UART TTY device> <RS485 TTY device>
Run a HTTP Server to visualise the sensor data and trackers, with configurations, accessible via port 5000