# HOW TO SET JaBIT ON LINUX ( UBUNTU 22.04 )

Firmware for the JaBIT Solo Miner ( Bitaxe v2.2 )

- Supports Stratum v1
- 300-350 GH/s
- WiFi connectivity (Only supports 2.4ghz connection)
- Mining statistics visible on OLED display
- Power management

## OPTION 1:  Installation Step by Step using binary image and config file.

- Step 1. Download the default esp-miner.bin image file and the sample config.cvs
- Step 2. Edit the config.cvs.sample  file  with your own parameters and rename it to config.cvs
- Step 3. Place your configuration file config.cvs & esp-miner.bin image in the same folder
- Step 4. Install bitaxetool using pip.
- Step 5. Flash the firmware onto JaBIT Solo Miner

- In this whole setup I am logged to my Ubuntu 22.04 devbox as `espuser` and my home directory is `/home/espuser`

## Step 1. Download the esp-miner.bin and config.cvs.sample

Download the default `esp-miner.bin` image file and the sample `config.cvs.sample` from

https://github.com/skot/ESP-Miner/releases/tag/v1.0

OR

https://github.com/developeralgo8888/ESP-Miner/releases/tag/JaBIT

## Step 2. Edit config.cvs.sample & save it as config.cvs

```
The items in RED needs to be updated with your own configurations and please
modify:

1. myWiFiSSID/myWiFiPassword with your home Wifi credentials (Only supports
   2.4ghz connection) ,
```

2. stratumurl/stratumport/stratumuser/stratumpass with the stratum server and login credentials that you would like to use.

3. Set the frequency of the Mining chip BM1397 between **200 - 800** Mhz (recommended **425 - 545** Mhz)

**Sample config.cvs.sample**

```
key,type,encoding,value
main,namespace,,
wifissid,data,string,myWiFiSSID
wifipass,data,string,myWiFiPassword
stratumurl,data,string,solo.ckpool.org
stratumport,data,u16,3333
stratumuser,data,string,myBitcoinWalletAddress.jabit
stratumpass,data,string,x
bm1397frequency,data,u16,475
bm1397voltage,data,u16,1400
```

my sample configuration file config.cvs.sample saved as config.cvs

```
key,type,encoding,value
main,namespace,,
wifissid,data,string,JaBIT-TestNet
wifipass,data,string,JaB1T@123@bitaxe
stratumurl,data,string,solo.ckpool.org
stratumport,data,u16,3333
stratumuser,data,string, bc1q4eznd85ntsphuq7epwp8lcrxufv84w9j2cngvk.jabit
stratumpass,data,string,x
bm1397frequency,data,u16,475
bm1397voltage,data,u16,1400
```

## Step 3. Place the binary file esp-miner.bin & config.cvs in same folder

```
espuser@devbox:~$  ls
esp-miner.bin     config.cvs
```

## Step 4. Install the bitaxetool

```
pip is included with Python 3.4+ but if you need to install it
check https://pip.pypa.io/en/stable/installation/


       espuser@devbox:~$ pip install bitaxetool

Defaulting to user installation because normal site-packages is not writeable
Collecting bitaxetool Downloading bitaxetool-0.1-py3-none-any.whl (11 kB)
Collecting esptool Downloading esptool-4.6.2.tar.gz (262 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 262.1/262.1 KB 2.5 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
Requirement already satisfied: PyYAML>=5.1 in /usr/lib/python3/dist-packages
(from esptool->bitaxetool) (5.4.1)
Collecting bitstring>=3.1.6 Downloading bitstring-4.0.2-py3-none-any.whl (46 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 KB 957.2 kB/s eta 0:00:00
Requirement already satisfied: cryptography>=2.1.4 in /usr/lib/python3/dist-
packages (from esptool->bitaxetool) (3.4.8)
Collecting ecdsa>=0.16.0
  Downloading ecdsa-0.18.0-py2.py3-none-any.whl (142 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 142.9/142.9 KB 2.8 MB/s eta 0:00:00
Collecting pyserial>=3.0
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 90.6/90.6 KB 1.9 MB/s eta 0:00:00
Collecting reedsolo<1.8,>=1.5.3
  Downloading reedsolo-1.7.0-py3-none-any.whl (32 kB)
Requirement already satisfied: six>=1.9.0 in /usr/lib/python3/dist-packages (from
ecdsa>=0.16.0->esptool->bitaxetool) (1.16.0)
Building wheels for collected packages: esptool
  Building wheel for esptool (setup.py) ... done
  Created wheel for esptool: filename=esptool-4.6.2-py3-none-any.whl size=338906
sha256=a1e15013da23ee7cc66f92f8fa47668b1526da4e5008319865ce49b7e0653751
  Stored in directory:
/home/espuser/.cache/pip/wheels/48/d3/c4/ebb120cf9d927e219ed969a217a25bedb428665f
77c3414b8d
Successfully built esptool
Installing collected packages: reedsolo, pyserial, ecdsa, bitstring, esptool,
bitaxetool
Successfully installed bitaxetool-0.1 bitstring-4.0.2 ecdsa-0.18.0 esptool-4.6.2
pyserial-3.5 reedsolo-1.7.0
```

## Step 4. Flash the firmware onto JaBIT Solo Miner using bitaxetool

```
espuser@devbox:~$  bitaxetool --port /dev/ttyUSB1 --config config.cvs --firmware
esp-miner.bin
```

**Sample output of the above command:**

```
Connecting to port: /dev/ttyUSB1
Flashing firmware: esp-miner.bin
Flashing config: config.cvs

Creating NVS binary with version: V2 - Multipage Blob Support Enabled

Created NVS binary: ===> /tmp/config.bin
esptool.py v4.6.2
Serial port /dev/ttyUSB1
Connecting....
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (revision v0.1)
Features: WiFi, BLE
Crystal is 40MHz
MAC: 34:85:18:9c:54:18
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00009000 to 0x0000efff...
Flash will be erased from 0x00010000 to 0x000c0fff...
Compressed 24576 bytes to 448...
Wrote 24576 bytes (448 compressed) at 0x00009000 in 0.3 seconds (effective 722.8
kbit/s)...
Hash of data verified.
Compressed 724896 bytes to 454125...
Wrote 724896 bytes (454125 compressed) at 0x00010000 in 40.3 seconds (effective
143.8 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

# OPTION 2:  Installation Step by Step from Scratch

This is a detailed roadmap to walk you through the installation process.

- In this whole setup I am logged to my Ubuntu 22.04 **devbox** as `espuser` and my home directory is `/home/espuser`

NOTE:  Where ESP-IDF v4.4.x and ESP-IDF tools will be installed:  Use `ESD-IDF v4.4.x`

```
ESP-IDF v4.4.x   ➔   /home/espuser/esp/esp-idf
ESP-IDF Tools    ➔   /home/espuser/.espressif
```

## Setting up Development Environment

These are the steps for setting up the ESP-IDF for your ESP32-S3.



# Step 1. Install Prerequisites

In order to use ESP-IDF with the ESP32-S3, you need to install some software packages based on your Operating System. This setup guide will help you on getting everything installed on Linux and macOS based systems.

### For Linux Users:

To compile using ESP-IDF you will need to get the following packages. The command to run depends on which distribution of Linux you are using:

- Ubuntu and Debian:

```
espuser@devbox:~$  pwd
/home/espuser
```

  - Using **sudo** command change to root to install the prerequisite packages.

```
espuser@devbox:~$  sudo apt-get install git wget flex bison gperf python3
python3-pip python3-venv cmake ninja-build ccache libffi-dev libssl-dev
dfu-util libusb-1.0-0
```

  - CMake version 3.16 or newer is required for use with ESP-IDF. Run “tools/idf_tools.py install cmake” to install a suitable version if your OS versions doesn’t have one.

## Step 2. Get ESP-IDF v 4.4.x

*NOTE: ESP_IDF v5.x currently does not work with JaBIT (Bitaxe V2.2) Firmware.*

To build applications for the ESP32-S3, you need the software libraries provided by Espressif in ESP-IDF repository.

To get ESP-IDF, navigate to your installation directory and clone the repository with `git clone`, following instructions below specific to your operating system.

Open Terminal, and run the following commands:

```
espuser@devbox:~$  mkdir -p ~/esp
espuser@devbox:~/esp$  cd ~/esp
espuser@devbox:~/esp$  git clone --recursive https://github.com/espressif/esp-
idf.git
```

ESP-IDF will be downloaded into `~/esp/esp-idf`.

## Step 3. Set up the tools

Aside from the ESP-IDF, you also need to install the tools used by ESP-IDF, such as the compiler, debugger, Python packages, etc, for projects supporting ESP32-S3.

In order to install tools for all supported targets please run the following command:

```
espuser@devbox:~$  cd ~/esp/esp-idf
espuser@devbox:~/esp/esp-idf$  ./install.sh all
```

### Customizing the tools installation path

The scripts introduced in this step install compilation tools required by ESP-IDF inside the user home directory: `$HOME/.espressif` on Linux. If you wish to install the tools into a different directory, set the environment variable `IDF_TOOLS_PATH` before running the installation scripts.

## Step 4. Set up the LLVM 15.x or 16.x and Clang compiler

Install Clang and LLVM by installing LLVM 15.x or 16.x (llvm-esp-16.0.0-20230516-linux-amd64.tar.xz ) from

```
https://github.com/espressif/llvm-project/releases/tag/esp-16.0.0-20230516
```

by downloading and extracting it in `/.espressif/tools` folder. You will now have a new `esp-clang` folder in `/.espressif/tools` folder

```
espuser@devbox:~$  cd ~/.espressif/tools
espuser@devbox:~/.espressif/tools$  wget https://github.com/espressif/llvm-project/releases/download/esp-16.0.0-20230516/llvm-esp-16.0.0-20230516-linux-amd64.tar.xz
espuser@devbox:~/.espressif/tools$  tar -xvf llvm-esp-16.0.0-20230516-linux-amd64.tar.xz
espuser@devbox:~/.espressif/tools$  ls -l esp-clang
```

## Step 5. Set up the environment variables

The installed tools are not yet added to the PATH environment variable. To make the tools usable from the command line, some environment variables must be set. ESP-IDF provides another script which does that.

In the terminal where you are going to use ESP-IDF, run:

Note the space between the leading dot and the path!

```
espuser@devbox:~/esp/esp-idf$   . $HOME/esp/esp-idf/export.sh
```

If you plan to use esp-idf frequently, you can create an alias for executing `export.sh`:

1. Copy and paste the following command to your shell's profile (`.profile`, `.bashrc`, `.zprofile`, etc.)

```
2.  alias get_idf='. $HOME/esp/esp-idf/export.sh'
```

3. Refresh the configuration by restarting the terminal session or by running `source [path to profile]`, for example, `source ~/.bashrc`.

Now you can run `get_idf` to set up or refresh the esp-idf environment in any terminal session.

Technically, you can add `export.sh` to your shell's profile directly; however, it is not recommended. Doing so activates IDF virtual environment in every terminal session (including those where IDF is not needed), defeating the purpose of the virtual environment and likely affecting other software.

If you have to use your shell's profile (`.profile`, `.bashrc`, `.zprofile`, etc.) to set up the PATH for ESP-IDF v4.4.x and tools, python etc then here is my sample of `.bashrc`. Open and edit your shell's profile and add similar  lines.

```
IDF_PATH="/home/espuser/esp/esp-idf"
IDF_TOOLS_PATH="/home/espuser/.espressif"
IDF_PYTHON_ENV_PATH="/home/espuser/.espressif/python_env/idf4.4_py3.10_env"
OPENOCD_SCRIPTS="/home/espuser/.espressif/tools/openocd-esp32/v0.11.0-esp32-
20221026/openocd-esp32/share/openocd/scripts"
LIBCLANG_PATH="/home/espuser/.espressif/tools/esp-clang/lib"
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/usr
/games:/usr/local/games:/opt/OpenJDK17U-jdk_x64_linux_hotspot_>
```

## Step 6. Set up JaBIT Production firmware project folder

Download the latest production firmware from the bitaxe community  github repository and extract to folder.

- https://github.com/skot/ESP-Miner/tree/nonce_checking

OR

- https://github.com/developeralgo8888/ESP-Miner/tree/nonce_checking

```
espuser@devbox:~$  git clone --branch nonce_checking https://github.com/skot/ESP-Miner.git

OR

espuser@devbox:~$  git clone --branch nonce_checking https://github.com/developeralgo8888/ESP-Miner.git

espuser@devbox:~$  cd ESP-Miner
espuser@devbox:~/ESP-Miner$
```

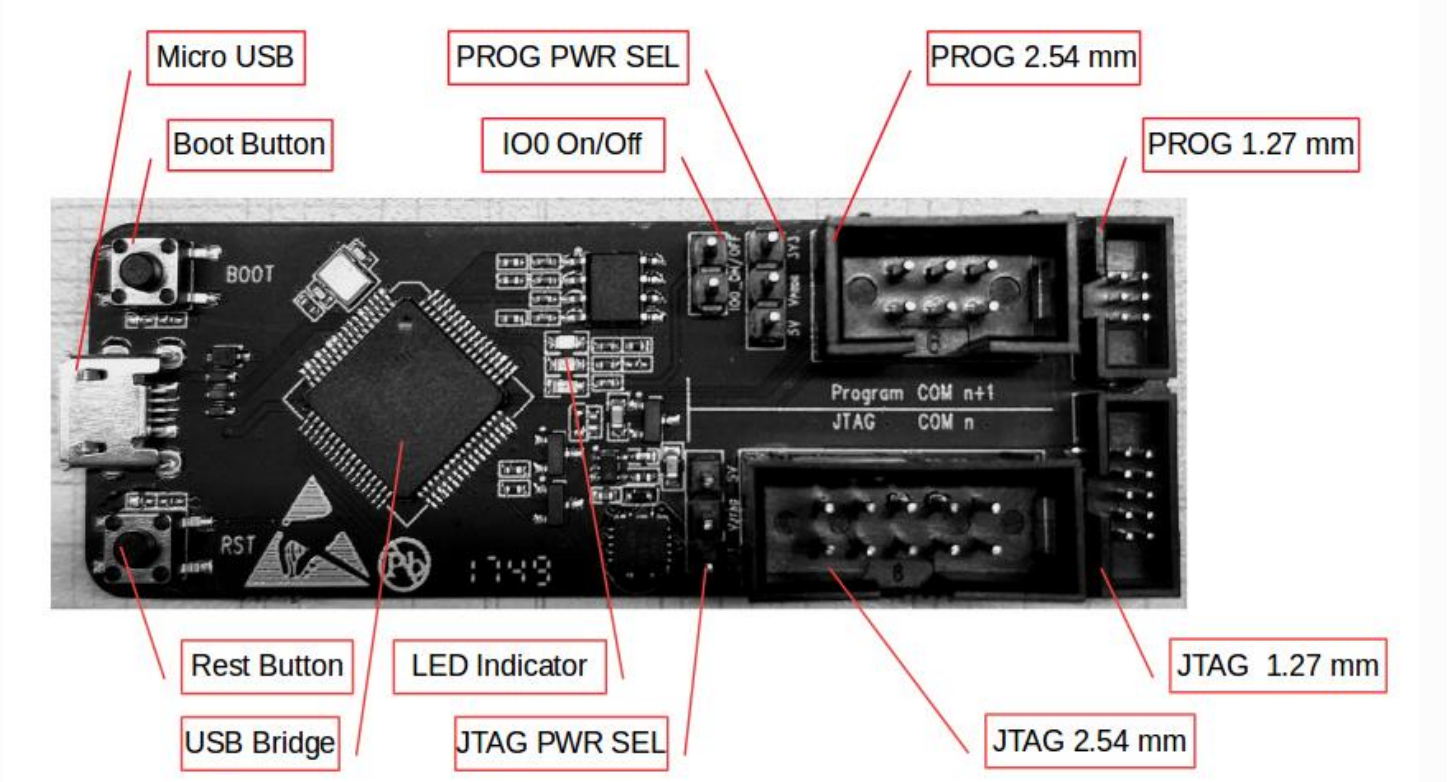
## Step 7. Connect  to your JaBIT Solo Miner .

Now connect your JaBIT Solo Miner which contains ESP32-S3 chip as controller to the computer and check under which serial port the board is visible.

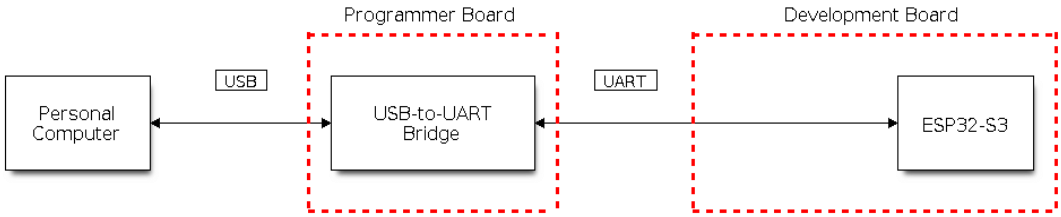
Currently JaBIT solo miner supports two ways of connecting to JaBIT solo Miner and flashing the firmware.

1. Using ESP-PROG (USB-to_UART Bridge)  and TAG-Connect TC2030-IDC-NL cable
2. Using ESP-PROG (USB-to_UART Bridge)  and Mini JTAG (J10) on JaBIT Solo Minier


The figure below describes the areas of each function on the ESP-Prog board. For JaBIT Solo Miner make sure that selector clip for PROG PWR SEL is set to 3.3V



External USB-to-UART Bridge Connection diagram




Serial ports have the following naming patterns:

- **Linux**: starting with `/dev/tty`

```
espuser@devbox:~$  cd ESP-Miner
espuser@devbox:~/ESP-Miner$  ls /dev/ttyUSB*
/dev/ttyUSB0    /dev/ttyUSB1
```

Keep the port name handy as you will need it in the next steps. In this case we use `/dev/ttyUSB1`

# Step 8. Configure Your Project

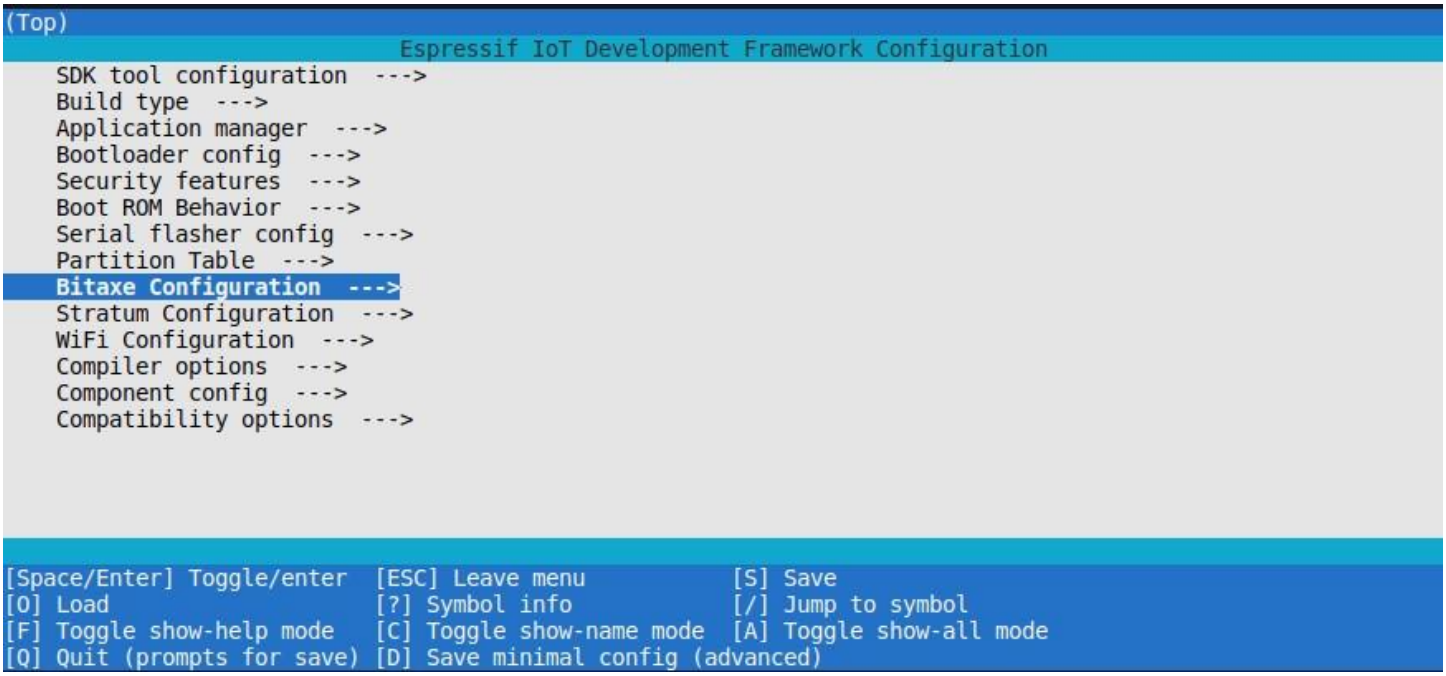
Navigate to your `ESP-Miner` project directory which contains `nonce_checking branch` source code , set ESP32-S3 as the target, and run the project configuration utility `menuconfig`.

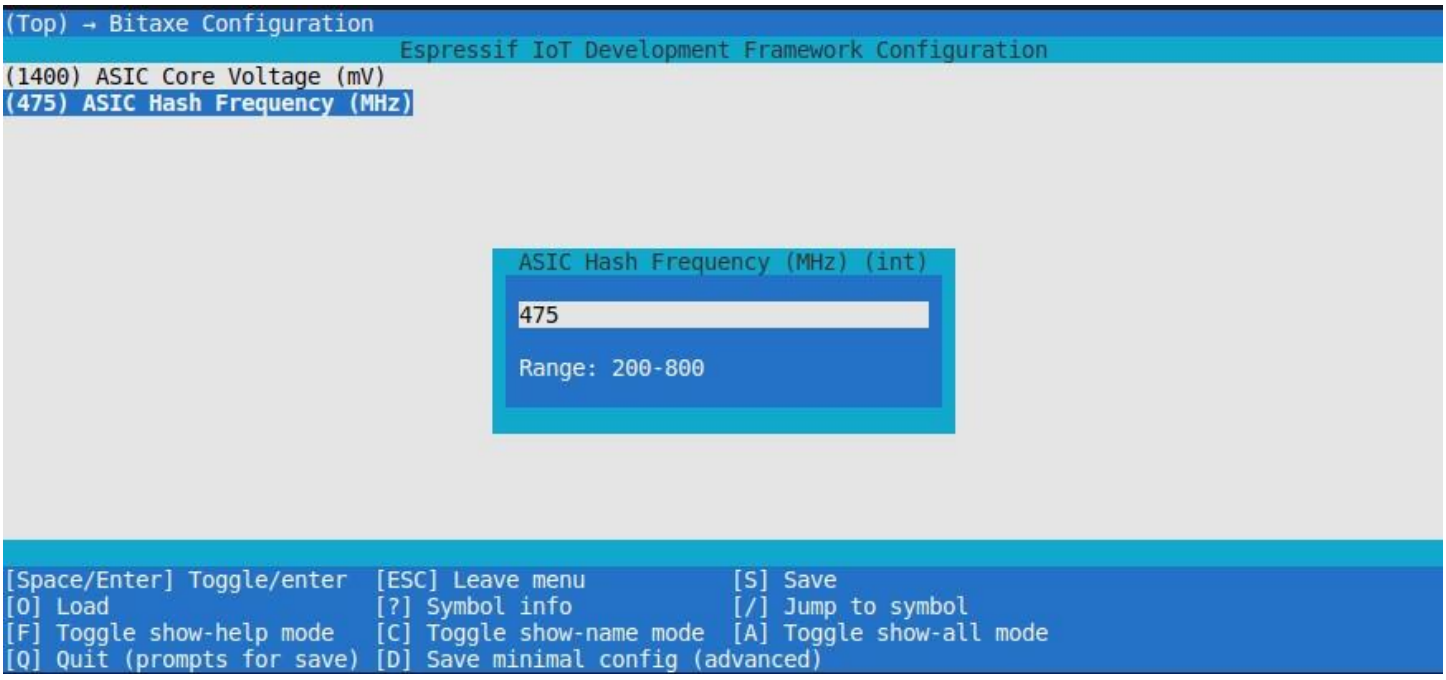
```
espuser@devbox:~$  cd ESP-Miner
espuser@devbox:~/ESP-Miner$ idf.py set-target esp32s3
espuser@devbox:~/ESP-Miner$ idf.py menuconfig
```

After opening a new project, you should first set the target with `idf.py set-target esp32s3`. Note that existing builds and configurations in the project, if any, will be cleared and initialized in this process.

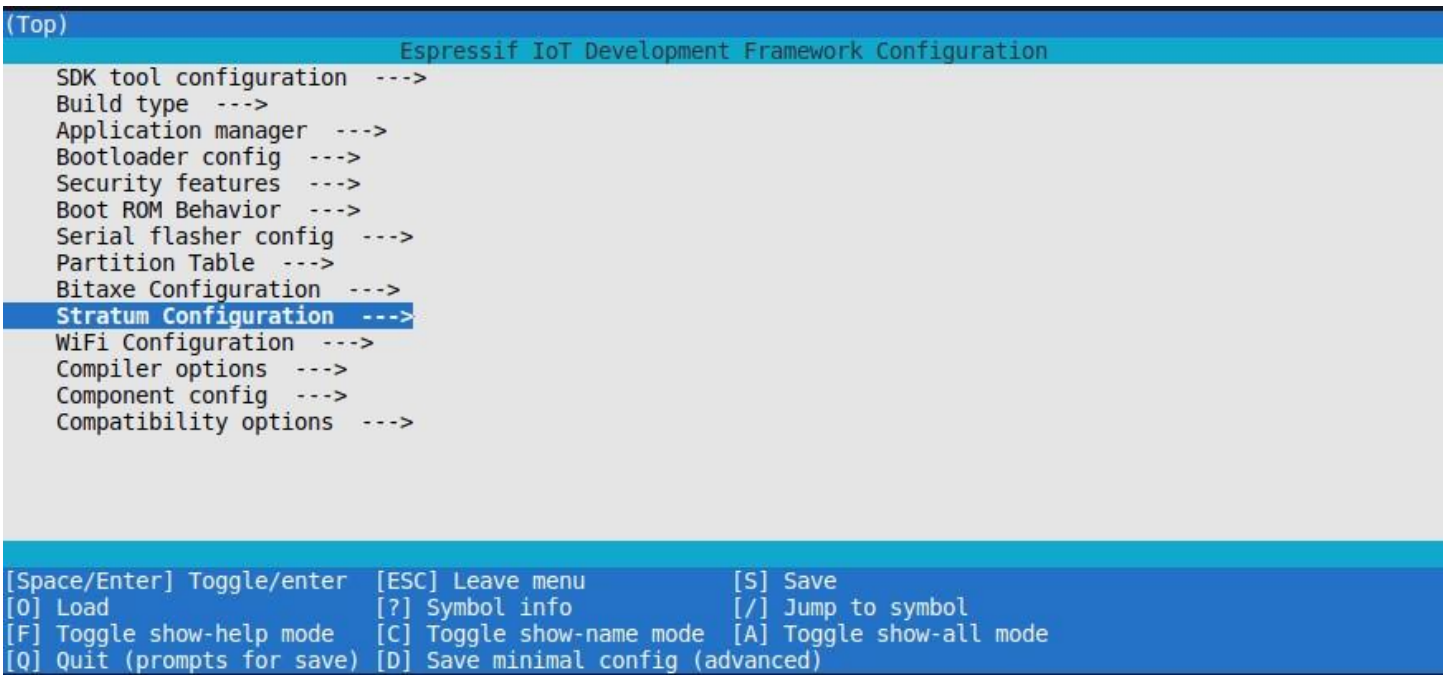
NOTE: Make sure to save all your configuration changes while using `menuconfig` menu by pressing `S` everytime after changes  or when you press `Q` it will ask for you save your changes , please select `Yes` on your Keyboard.

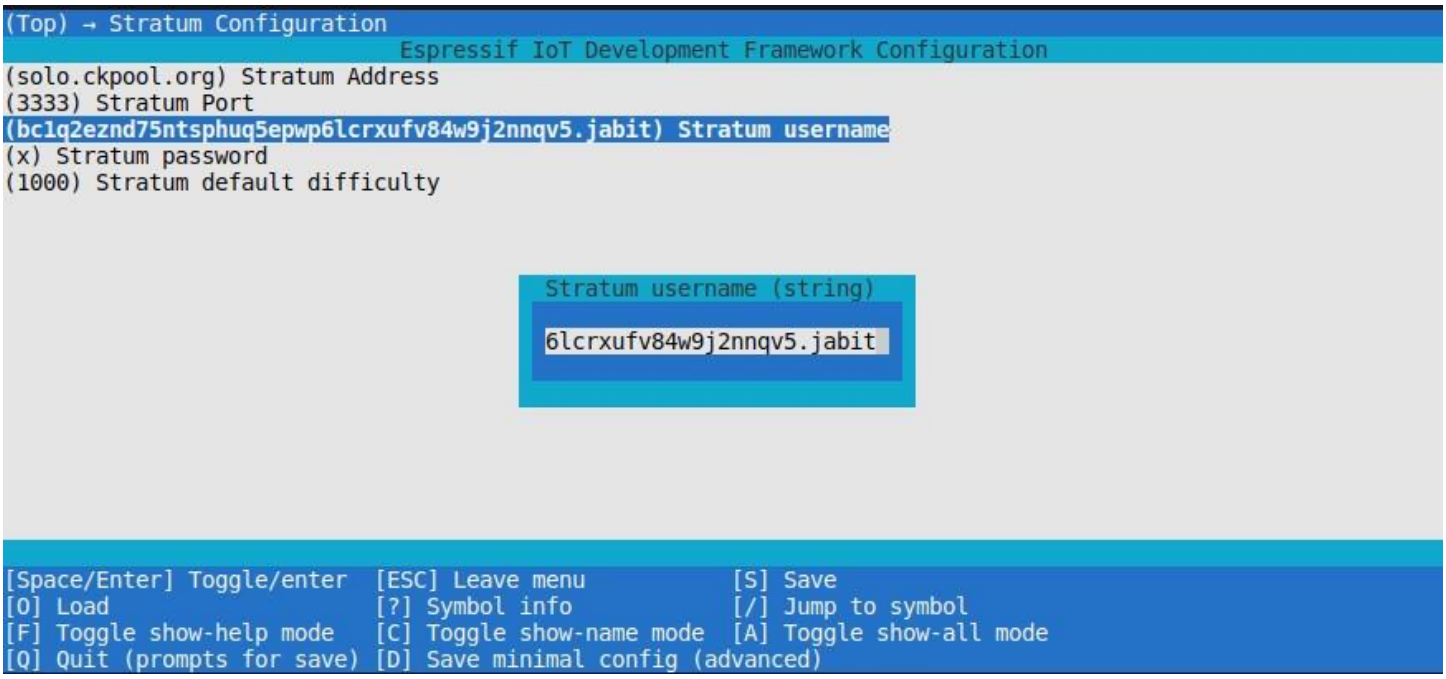
Under Bitaxe configuration , Configure your JaBIT Solo Miner Frequency  within 200 – 800 MHz (Recommended **425 MHz**)



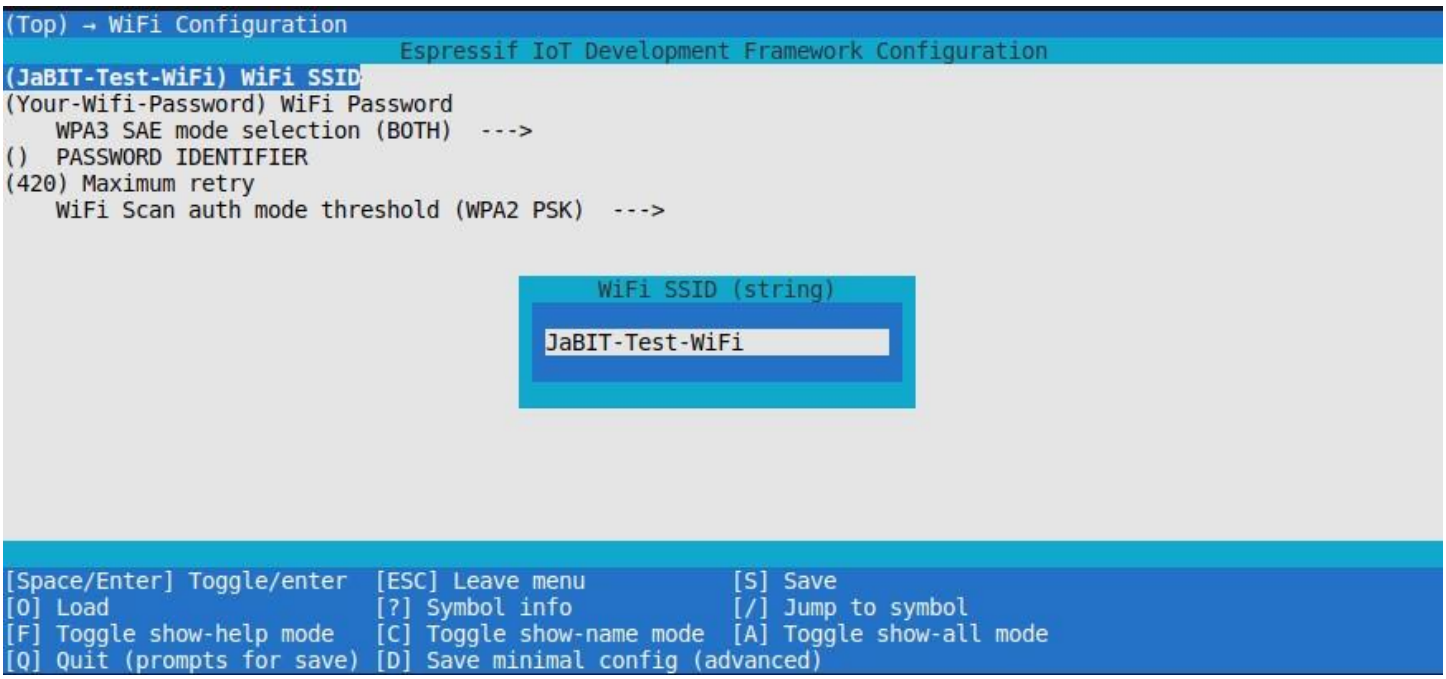



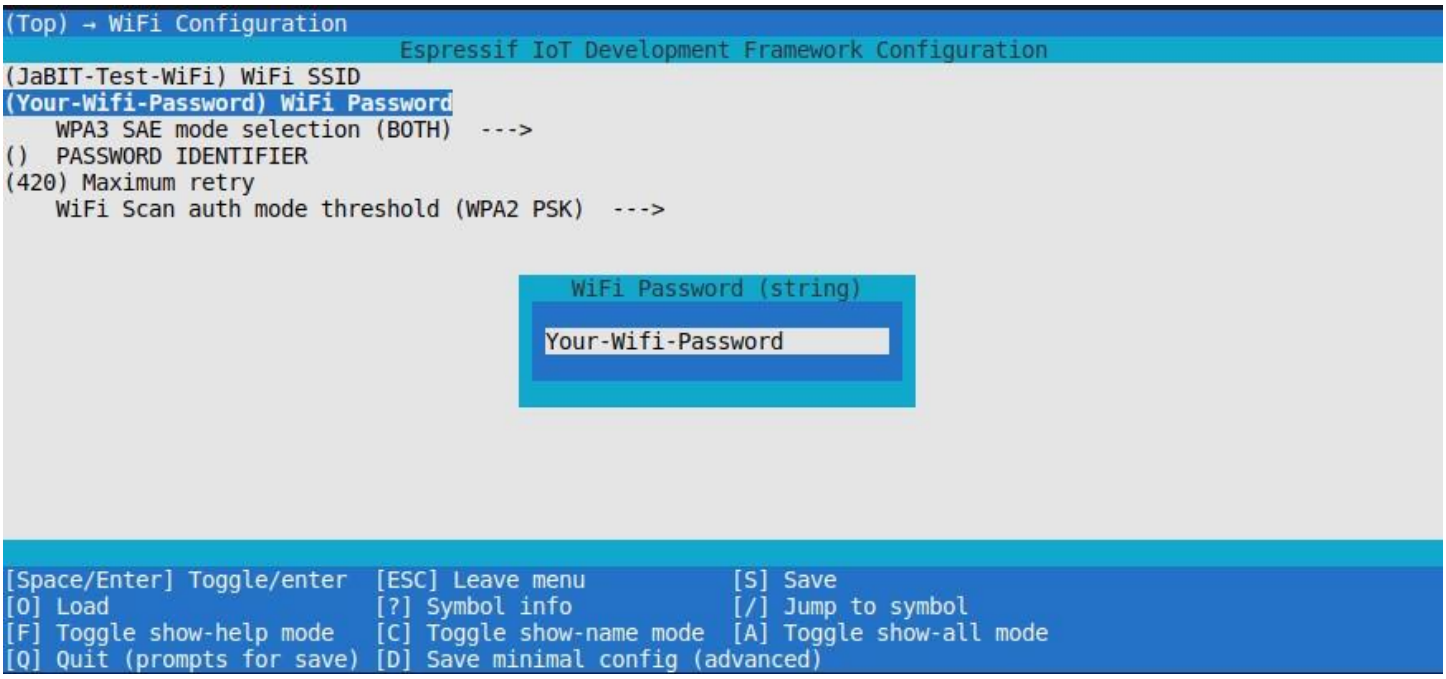
Under your Stratum configuration,  Configure your Bitcoin address and other pool parameters



Under WiFi Configuration  screen configure your own **WiFi SSID** and **Password** for your local WiFi Network




## Step 9.  Build and Flash Firmware onto the JaBIT Solo Miner

```
espuser@devbox:~$  cd ESP-Miner
espuser@devbox:~/ESP-Miner$ idf.py -p  /dev/ttyUSB1 flash monitor

(To exit the serial monitor, type Ctrl-].)
```

**If the flashing of the production firmware is successful, you should see something similar to the output below and your small OLED screen should start displaying mining information.**

```
espuser@devbox:~/ESP-Miner$ idf.py -p /dev/ttyUSB1 flash monitor

Executing action: flash
Running ninja in directory /home/espuser/ESP-Miner/build
Executing "ninja flash"...
[1/5] cd /home/espuser/ESP-Miner/bui.../home/espuser/ESP-Miner/build/esp-
miner.bin
esp-miner.bin binary size 0xb0fb0 bytes. Smallest app partition is 0x100000
bytes. 0x4f050 bytes (31%) free.
[2/5] Performing build step for 'bootloader'
[1/1] cd /home/espuser/ESP-Miner/build/bootloader/esp-idf/esptool_py &&
/home/espuser/.espressif/python_env/idf4.4_py3.10_env/bin/python
/home/espuser/esp/esp-idf/components/partition_table/check_sizes.py --offset
0x8000 bootloader 0x0 /home/espuser/ESP-Miner/build/bootloader/bootloader.bin
Bootloader binary size 0x5240 bytes. 0x2dc0 bytes (36%) free.
[2/3] cd /home/espuser/esp/esp-idf/components/esptool_py && /usr/bin/cmake -D
IDF_P...ner/build" -P /home/espuser/esp/esp-
idf/components/esptool_py/run_serial_tool.cmake
esptool.py esp32s3 -p /dev/ttyUSB2 -b 460800 --before=default_reset --
after=hard_reset write_flash --flash_mode dio --flash_freq 80m --flash_size 2MB
0x0 bootloader/bootloader.bin 0x10000 esp-miner.bin 0x8000
partition_table/partition-table.bin
esptool.py v3.3.2
Serial port /dev/ttyUSB2
Connecting....
Chip is ESP32-S3
Features: WiFi, BLE
Crystal is 40MHz
MAC: 34:85:18:a6:f0:44
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00005fff...
Flash will be erased from 0x00010000 to 0x000c0fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Compressed 21056 bytes to 13153...
Writing at 0x00000000... (100 %)
Wrote 21056 bytes (13153 compressed) at 0x00000000 in 0.5 seconds (effective
319.0 kbit/s)...
Hash of data verified.
Compressed 724912 bytes to 454173...
Writing at 0x00010000... (3 %)
Writing at 0x0001bfbb... (7 %)
Writing at 0x0002717c... (10 %)
Writing at 0x0002fecf... (14 %)
```

```
Writing at 0x00035f22... (17 %)
Writing at 0x0003b740... (21 %)
Writing at 0x00041f46... (25 %)
Writing at 0x00047fca... (28 %)
Writing at 0x0004da25... (32 %)
Writing at 0x000530d5... (35 %)
Writing at 0x00058944... (39 %)
Writing at 0x0005e2b6... (42 %)
Writing at 0x00063b4f... (46 %)
Writing at 0x00068b88... (50 %)
Writing at 0x0006dceb... (53 %)
Writing at 0x00072ba4... (57 %)
Writing at 0x00078014... (60 %)
Writing at 0x0007d4b5... (64 %)
Writing at 0x000830e7... (67 %)
Writing at 0x0008890a... (71 %)
Writing at 0x0008e570... (75 %)
Writing at 0x000947de... (78 %)
Writing at 0x0009a08f... (82 %)
Writing at 0x000a2156... (85 %)
Writing at 0x000a94ab... (89 %)
Writing at 0x000b12f2... (92 %)
Writing at 0x000b6d72... (96 %)
Writing at 0x000bcde8... (100 %)
Wrote 724912 bytes (454173 compressed) at 0x00010000 in 10.4 seconds (effective
557.6 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 512.8
kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Executing action: monitor
Running idf_monitor in directory /home/espuser/ESP-Miner
Executing "/home/espuser/.espressif/python_env/idf4.4_py3.10_env/bin/python
/home/espuser/esp/esp-idf/tools/idf_monitor.py -p /dev/ttyUSB2 -b 115200 --
toolchain-prefix xtensa-esp32s3-elf- --target esp32s3 /home/espuser/ESP-
Miner/build/esp-miner.elf -m
'/home/espuser/.espressif/python_env/idf4.4_py3.10_env/bin/python'
'/home/espuser/esp/esp-idf/tools/idf.py' '-p' '/dev/ttyUSB2'"...
--- idf_monitor on /dev/ttyUSB2 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
x<x( boo<ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x2a (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3808,len:0x1668
```

```
load:0x403c9700,len:0xbcc
load:0x403cc700,len:0x2fac
entry 0x403c9954
I (25) boot: ESP-IDF v4.4.4-dirty 2nd stage bootloader
I (25) boot: compile time 05:08:53
I (25) boot: chip revision: v0.1
I (27) boot_comm: chip revision: 1, min. bootloader chip revision: 0
I (34) boot.esp32s3: Boot SPI Speed : 80MHz
I (39) boot.esp32s3: SPI Mode       : DIO
I (44) boot.esp32s3: SPI Flash Size : 2MB
I (48) boot: Enabling RNG early entropy source...
I (54) boot: Partition Table:
I (57) boot: ## Label            Usage          Type ST Offset   Length
I (65) boot:  0 nvs              WiFi data        01 02 00009000 00006000
I (72) boot:  1 phy_init         RF data          01 01 0000f000 00001000
I (80) boot:  2 factory          factory app      00 00 00010000 00100000
I (87) boot: End of partition table
I (91) boot_comm: chip revision: 1, min. application chip revision: 0
I (98) esp_image: segment 0: paddr=00010020 vaddr=3c090020 size=1746ch ( 95340)
map
I (124) esp_image: segment 1: paddr=00027494 vaddr=3fc97970 size=05340h ( 21312)
load
I (129) esp_image: segment 2: paddr=0002c7dc vaddr=40374000 size=0383ch ( 14396)
load
I (133) esp_image: segment 3: paddr=00030020 vaddr=42000020 size=80e3ch (527932)
map
I (233) esp_image: segment 4: paddr=000b0e64 vaddr=4037783c size=10128h ( 65832)
load
I (256) boot: Loaded app from partition at offset 0x10000
I (256) boot: Disabling RNG early entropy source...
I (267) cpu_start: Pro cpu up.
I (267) cpu_start: Starting app cpu, entry point is 0x40375480
0x40375480: call_start_cpu1 at /home/espuser/esp/esp-
idf/components/esp_system/port/cpu_start.c:148

I (0) cpu_start: App cpu up.
I (281) cpu_start: Pro cpu start user code
I (281) cpu_start: cpu freq: 160000000
I (281) cpu_start: Application information:
I (284) cpu_start: Project name:     esp-miner
I (289) cpu_start: App version:      1
I (294) cpu_start: Compile time:     Jun 22 2023 05:08:47
I (300) cpu_start: ELF file SHA256:  36f5140952dc22c5...
I (306) cpu_start: ESP-IDF:          v4.4.4-dirty
I (311) heap_init: Initializing. RAM available for dynamic allocation:
I (318) heap_init: At 3FCA0DF8 len 00048918 (290 KiB): D/IRAM
I (325) heap_init: At 3FCE9710 len 00005724 (21 KiB): STACK/DRAM
I (331) heap_init: At 3FCF0000 len 00008000 (32 KiB): DRAM
I (338) heap_init: At 600FE000 len 00002000 (8 KiB): RTCRAM
I (344) spi_flash: detected chip: gd
I (348) spi_flash: flash io: dio
```

```
W (352) spi_flash: Detected size(16384k) larger than the size in the binary image
header(2048k). Using the size in the binary image header.
I (366) sleep: Configure to isolate all GPIO pins in sleep state
I (372) sleep: Enable automatic switching of GPIO sleep configuration
I (379) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
I (400) miner: Welcome to the bitaxe!
```

If there are no issues by the end of the flash process, the board will reboot and start Mining.

NOTE: `Currently every time you want to make changes to the running firmware on JaBIT you will need to recompile the firmware with the new changes and flash it to the JaBIT Solo Miner`