Keiser M7 bridge for Bike profile – Instructions.

Material needed:

2x ESP32 dev boards (such as M5 Stack lite for example).

USB-C cable.

2 jumper wires (or a paper clip or a piece of 1.5mm electrical wire).

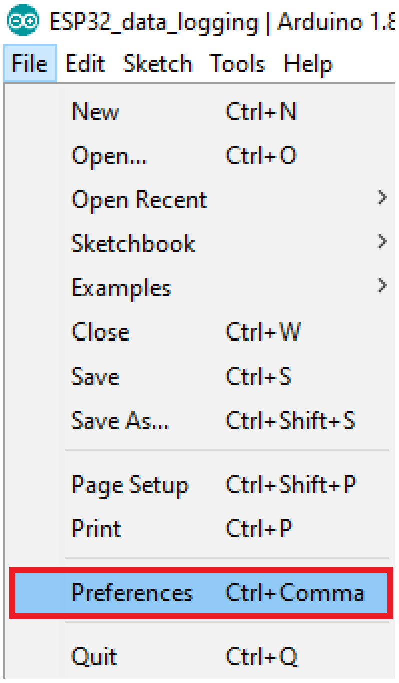
Arduino IDE software with ESP32 library installed.

https://www.arduino.cc/en/Guide/macOS

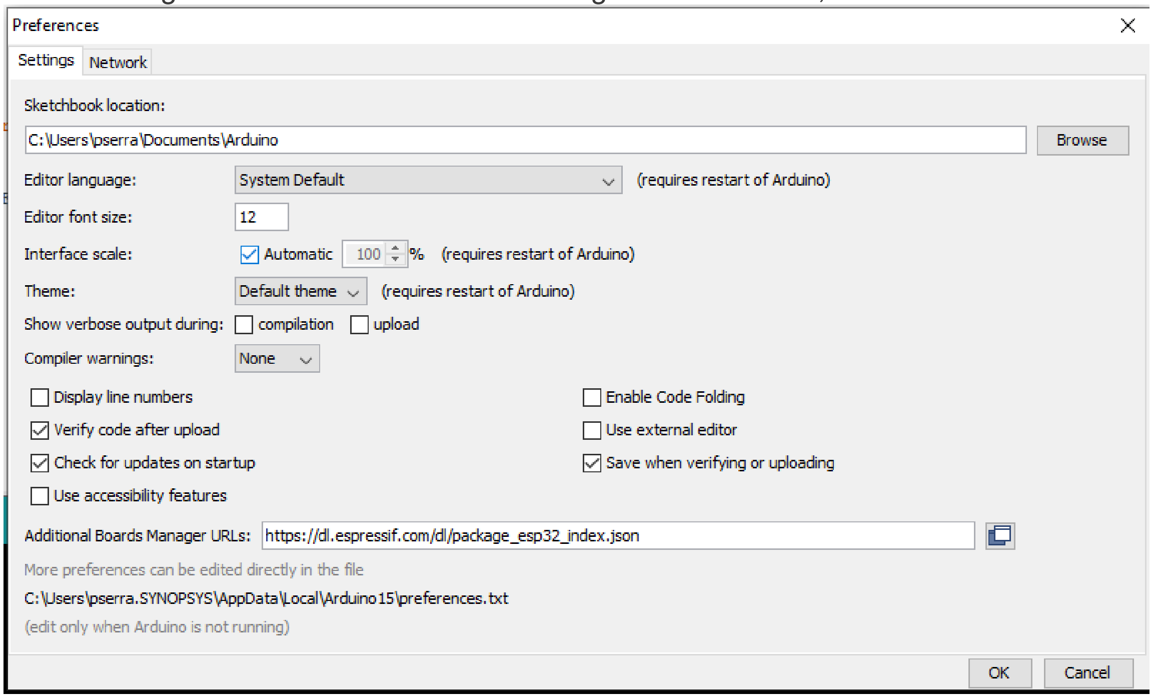
Installing ESP32 Add-on in Arduino IDE

To install the ESP32 board in your Arduino IDE, follow these next instructions:

1. In your Arduino IDE, go to **File**> **Preferences**



2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the “Additional Board Manager URLs” field as shown in the figure below. Then, click the “OK” button:



Preferences

Settings | Network

Sketchbook location: C:\Users\pserra\Documents\Arduino [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

□ Display line numbers

☑ Verify code after upload

☑ Check for updates on startup

□ Use accessibility features

□ Enable Code Folding

□ Use external editor

☑ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

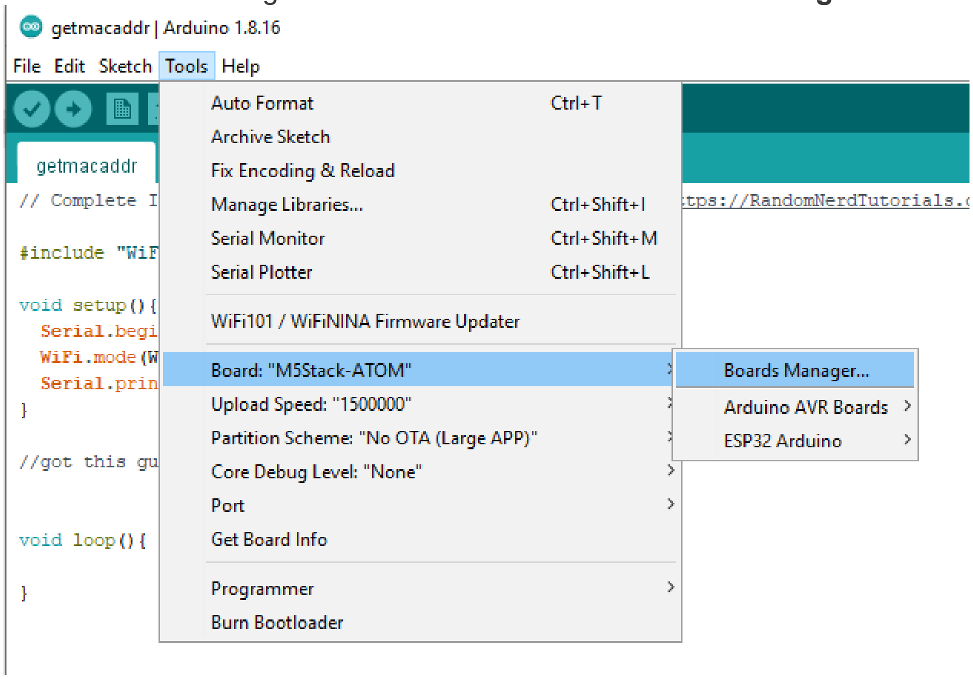
More preferences can be edited directly in the file

C:\Users\pserra.SYNOPSYS\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

[OK] [Cancel]

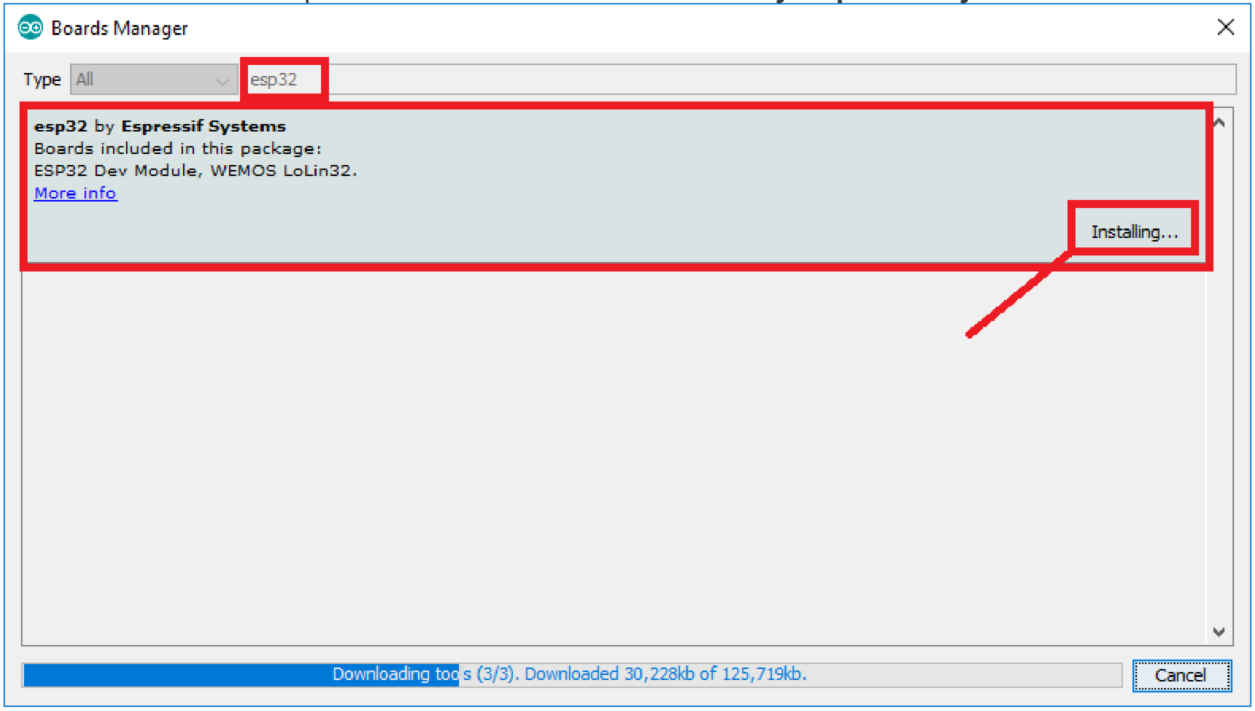
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



getmacaddr | Arduino 1.8.16

```
// Complete I                                          tps://RandomNerdTutorials.c

#include "WiF

void setup(){
  Serial.begi
  WiFi.mode(W
  Serial.prin
}

//got this gu


void loop(){

}
```

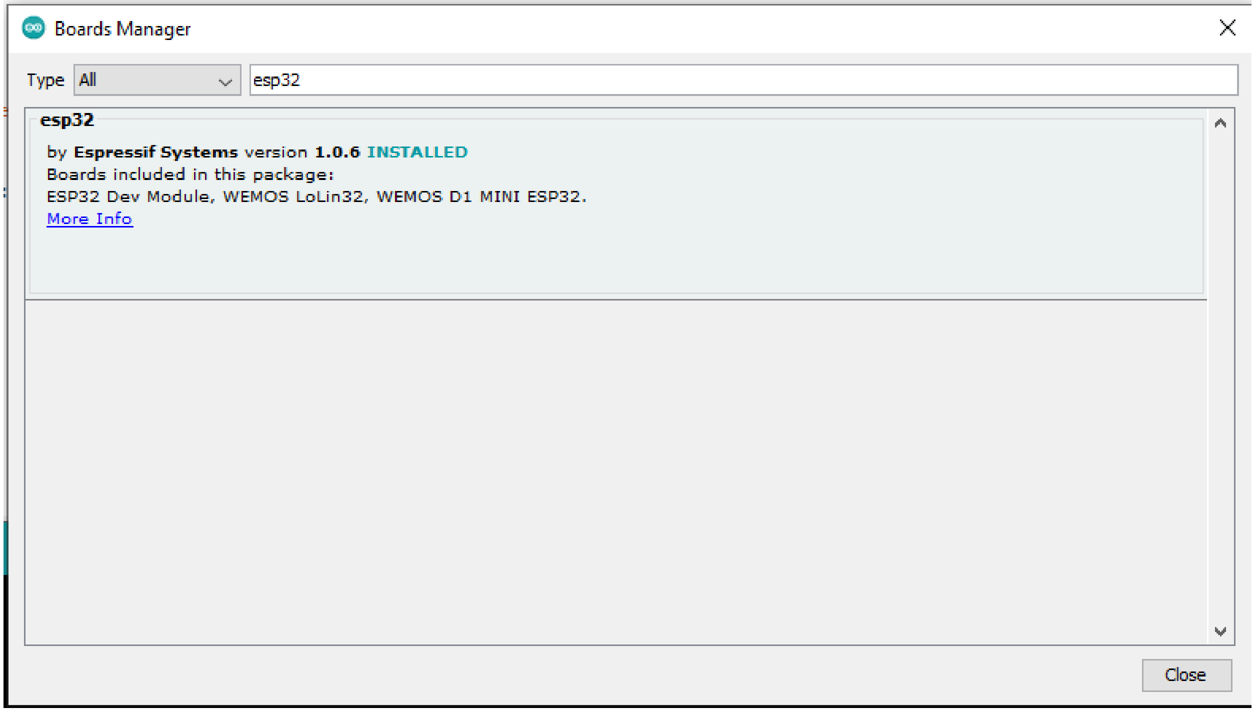
4. Search for **ESP32** and press install button for the "**ESP32 by Espressif Systems**":



esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

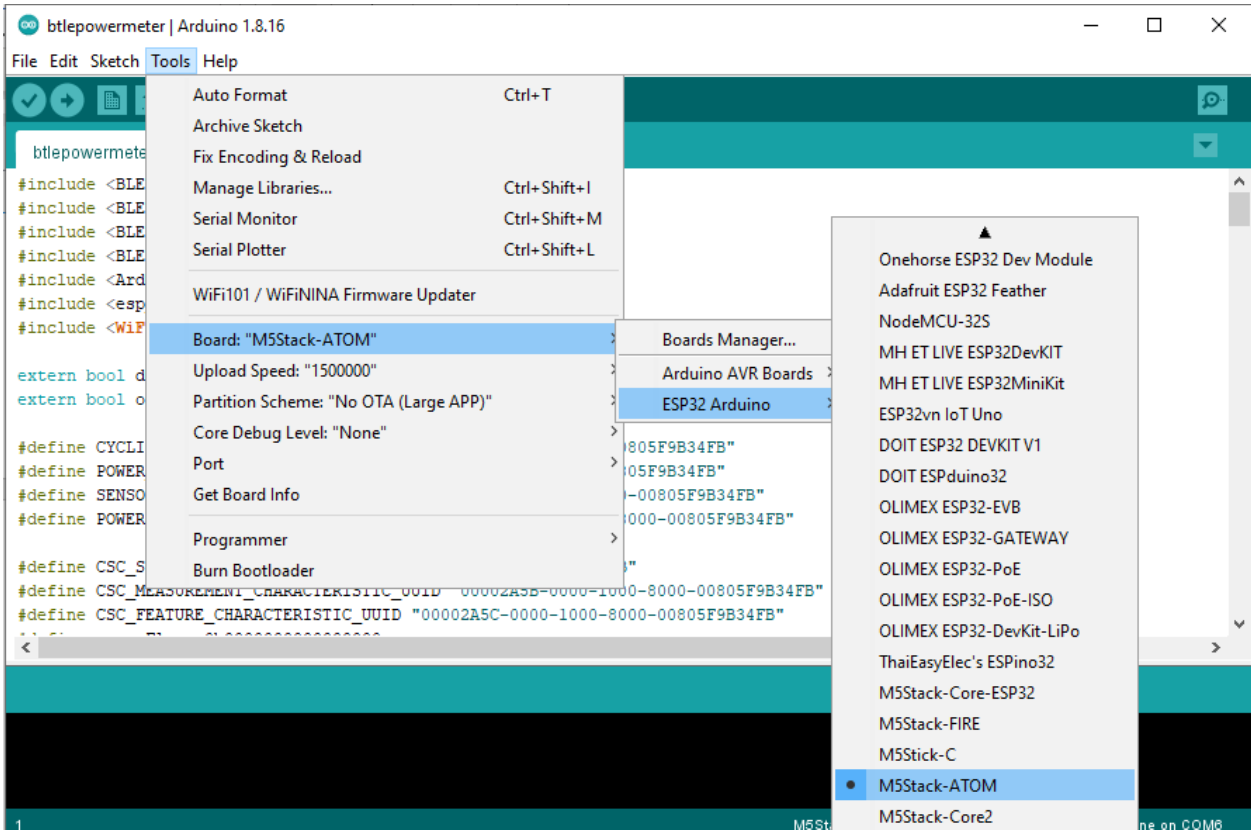
Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

5. That's it. It should be installed after a few seconds.



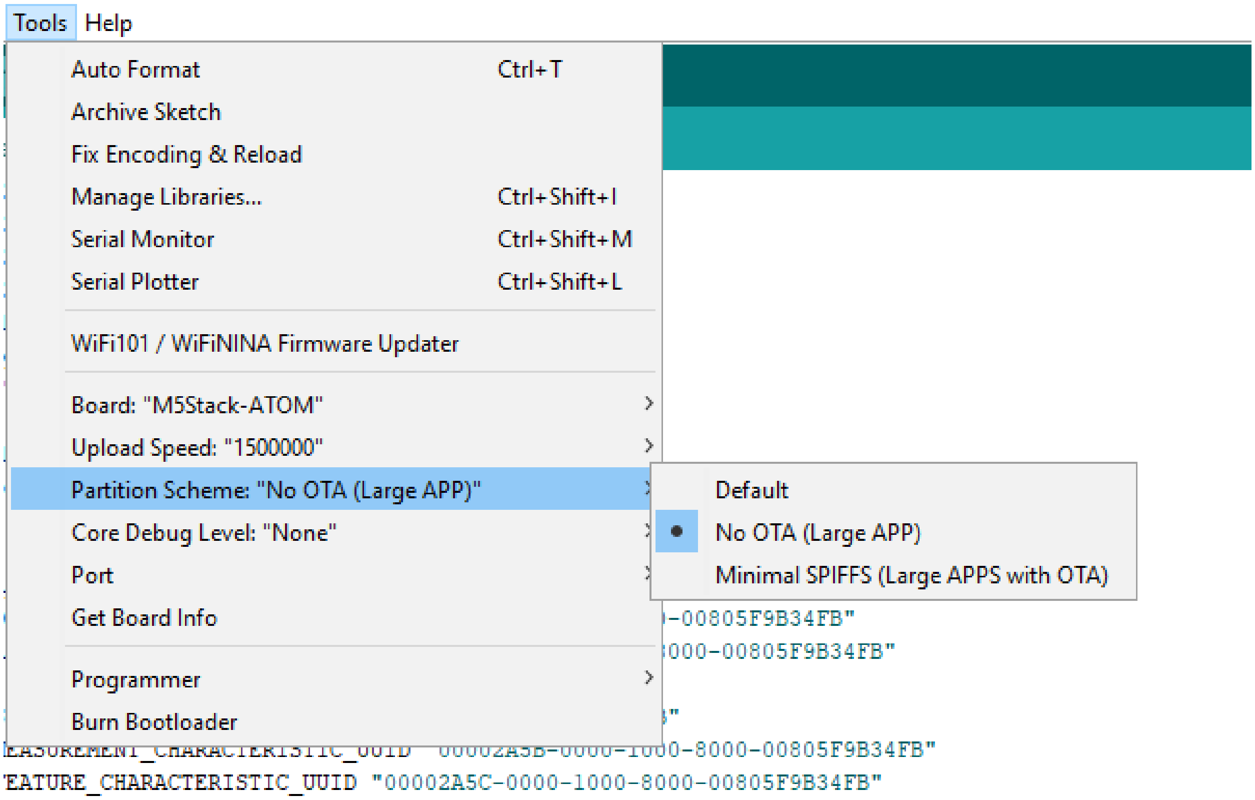
esp32
by **Espressif Systems** version **1.0.6** **INSTALLED**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

Select this for M5 stack:



```
#include <BLE
#include <BLE
#include <BLE
#include <BLE
#include <Ard
#include <esp
#include <WiF

extern bool d
extern bool o

#define CYCLI                                                   805F9B34FB"
#define POWER                                                  05F9B34FB"
#define SENSO                                                  -00805F9B34FB"
#define POWER                                                  000-00805F9B34FB"

#define CSC_S                                                  "
#define CSC_MEASUREMENT_CHARACTERISTIC_UUID "00002A5B-0000-1000-8000-00805F9B34FB"
#define CSC_FEATURE_CHARACTERISTIC_UUID "00002A5C-0000-1000-8000-00805F9B34FB"
```

Upload Speed to 1500000, and set the partition scheme to “No OTA (Large-App)”



Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Core Debug Level: "None"
Port
Get Board Info
Programmer
Burn Bootloader
Default
● No OTA (Large APP)
Minimal SPIFFS (Large APPS with OTA)

Board: "M5Stack-ATOM"

Upload Speed: "1500000"

Partition Scheme: "No OTA (Large APP)"

```
-00805F9B34FB"
000-00805F9B34FB"

"
EASUREMENT_CHARACTERISTIC_UUID "00002A5B-0000-1000-8000-00805F9B34FB"
EATURE_CHARACTERISTIC_UUID "00002A5C-0000-1000-8000-00805F9B34FB"
```

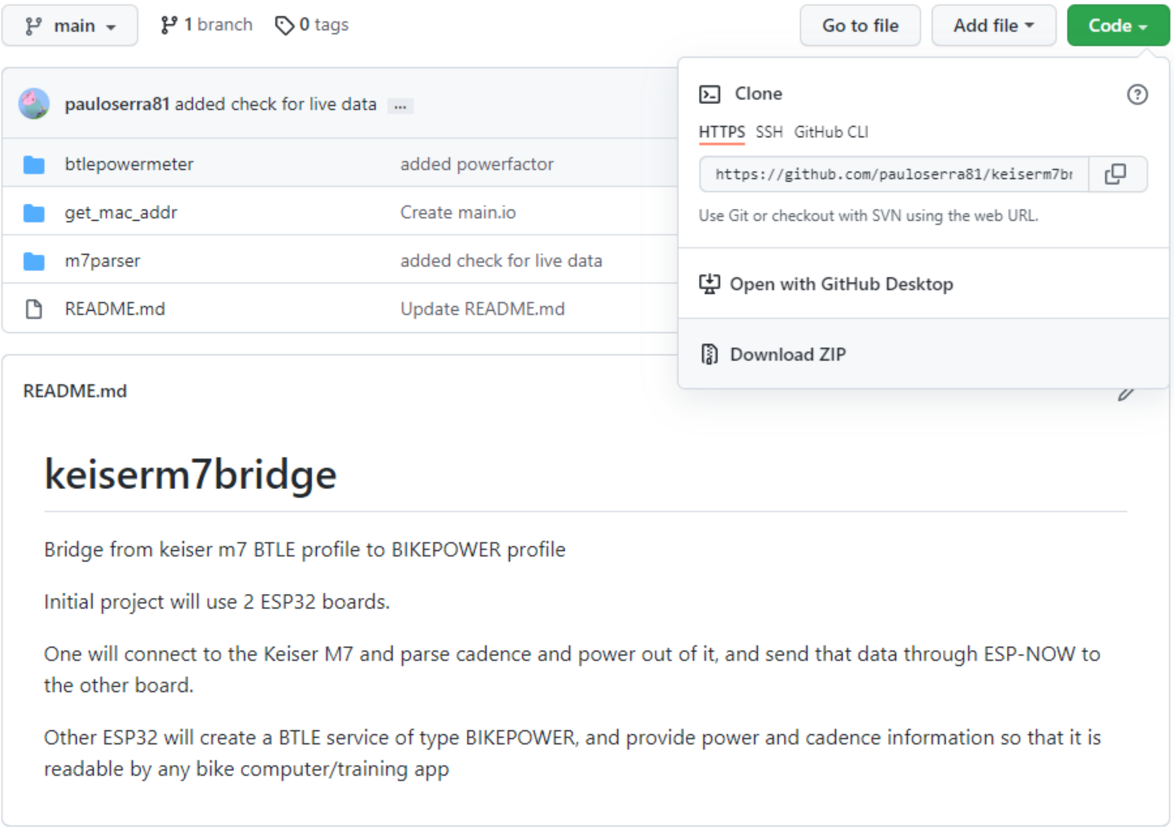
Now we are all set to go.

Download all the code from:

https://github.com/pauloserra81/keiserm7bridge

Click on “code” and then download zip.

Extract it to a location of your choice.



pauloserra81 added check for live data ...

btlepowermeter	added powerfactor
get_mac_addr	Create main.io
m7parser	added check for live data
README.md	Update README.md

```
https://github.com/pauloserra81/keiserm7br
```

Use Git or checkout with SVN using the web URL.

README.md

keiserm7bridge

Bridge from keiser m7 BTLE profile to BIKEPOWER profile

Initial project will use 2 ESP32 boards.

One will connect to the Keiser M7 and parse cadence and power out of it, and send that data through ESP-NOW to the other board.

Other ESP32 will create a BTLE service of type BIKEPOWER, and provide power and cadence information so that it is readable by any bike computer/training app

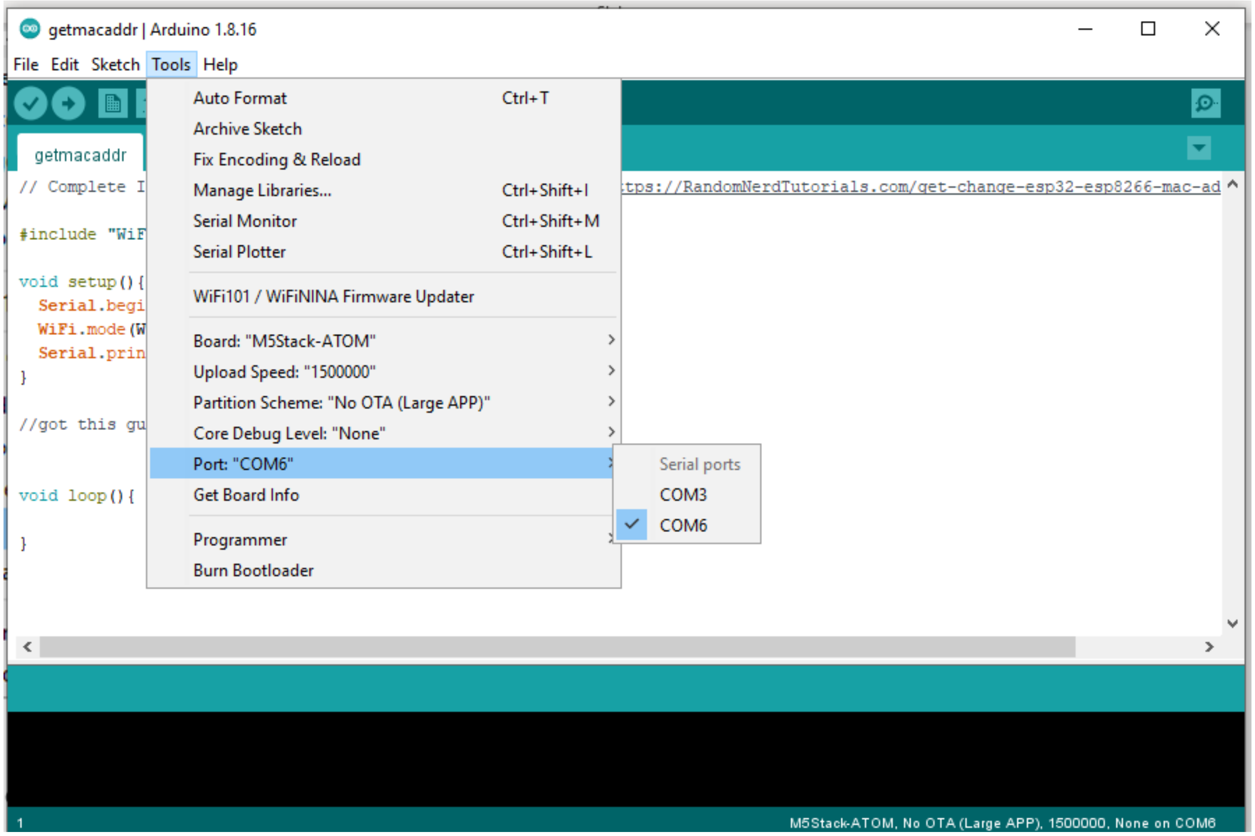
Find out your board MAC address.

As we are using 2 boards, we need to get the MAC address for one of them. This is going to be our "receiver" board.

To do that, open the file named "getmacaddr.ino" in Arduino IDE.

Connect the USB-C cable to one of the M5 Stack devices, install any drivers asked by the operating system.

Select the virtual COM that will appear here:



getmacaddr | Arduino 1.8.16

```
// Complete I                                         tps://RandomNerdTutorials.com/get-change-esp32-esp8266-mac-ad

#include "WiF

void setup(){
  Serial.begi
  WiFi.mode(W
  Serial.prin
}

//got this gu


void loop(){

}
```

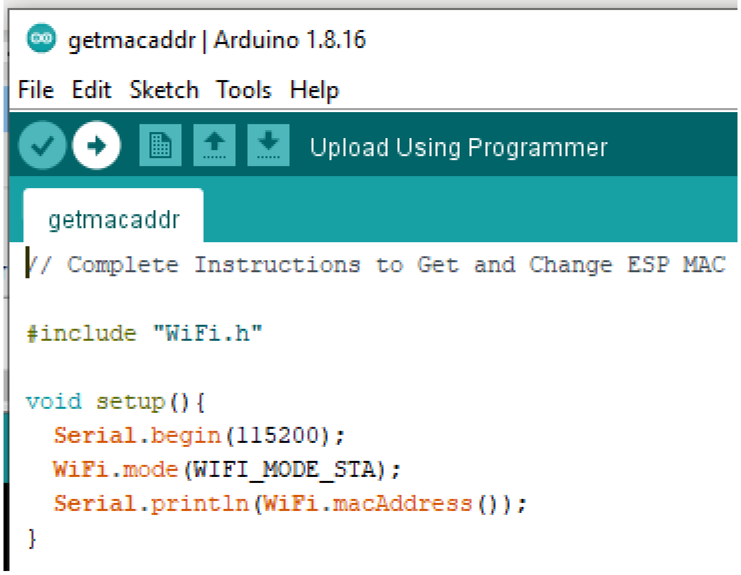
Board: "M5Stack-ATOM"
Upload Speed: "1500000"
Partition Scheme: "No OTA (Large APP)"
Core Debug Level: "None"
Port: "COM6"

1 M5Stack-ATOM, No OTA (Large APP), 1500000, None on COM6

(If you have more than one, it should be the one with highest number.

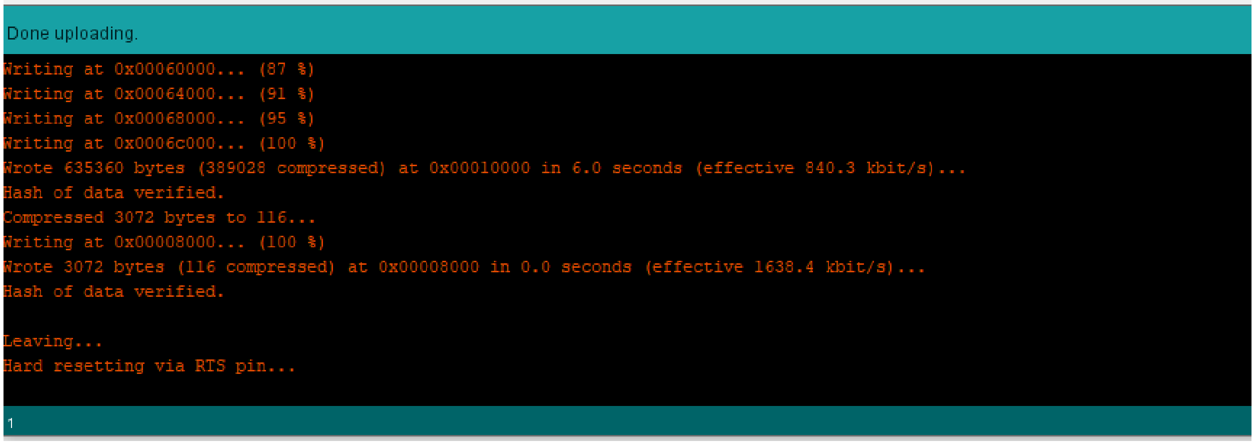
If it fails, try the other ones.

Now press the “upload” button to send the code to the device. (The -> arrow)



```
// Complete Instructions to Get and Change ESP MAC

#include "WiFi.h"

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}
```

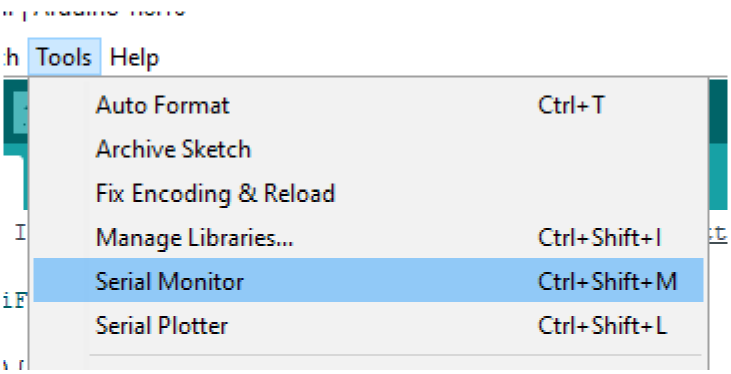
Wait for it to compile, build the sketch and upload to device. Once that is done it should read like this:

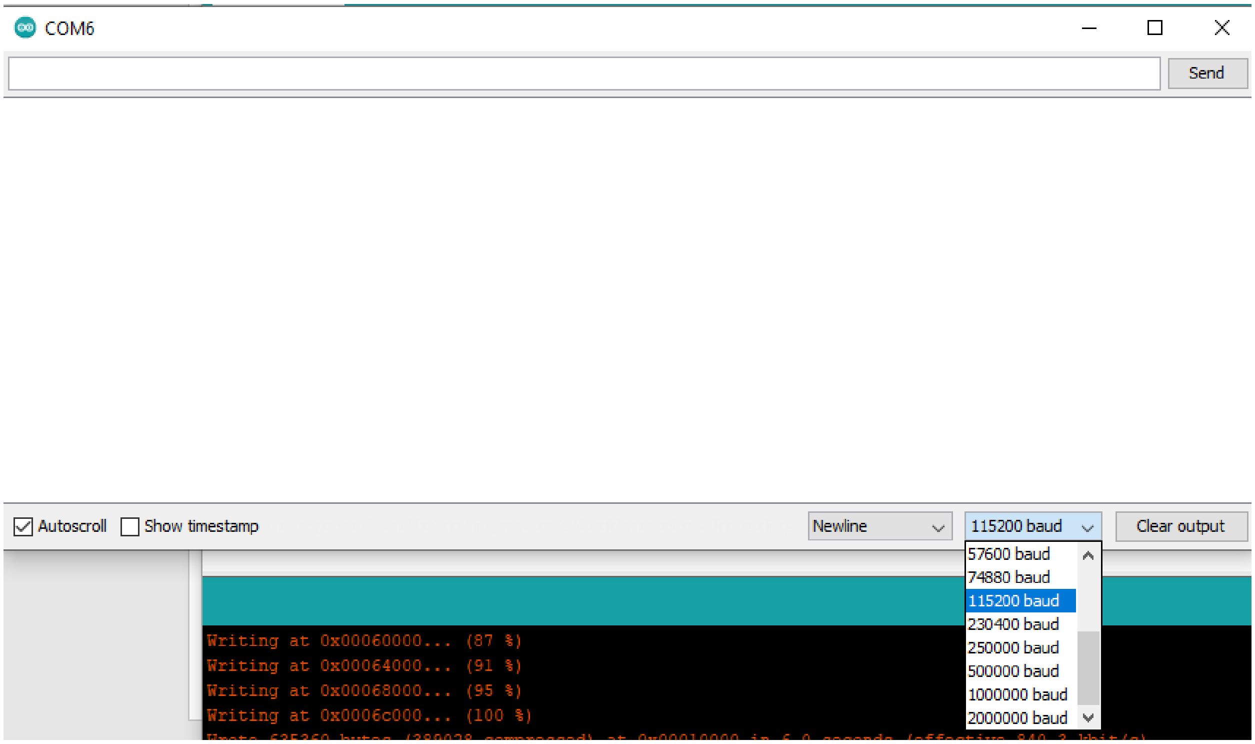


Done uploading.

```
Writing at 0x00060000... (87 %)
Writing at 0x00064000... (91 %)
Writing at 0x00068000... (95 %)
Writing at 0x0006c000... (100 %)
Wrote 635360 bytes (389028 compressed) at 0x00010000 in 6.0 seconds (effective 840.3 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 116...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (116 compressed) at 0x00008000 in 0.0 seconds (effective 1638.4 kbit/s)...
Hash of data verified.

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

Now, open the Serial Monitor at a baud rate of 115200


h Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L



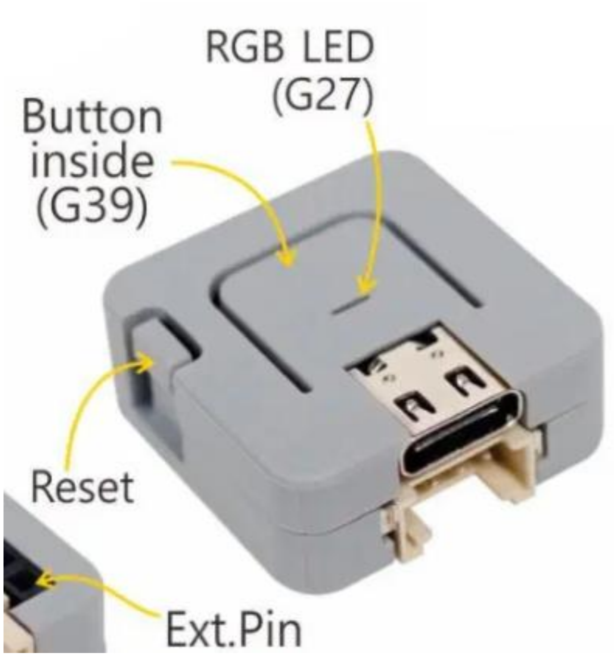
☑ Autoscroll □ Show timestamp

Newline

115200 baud

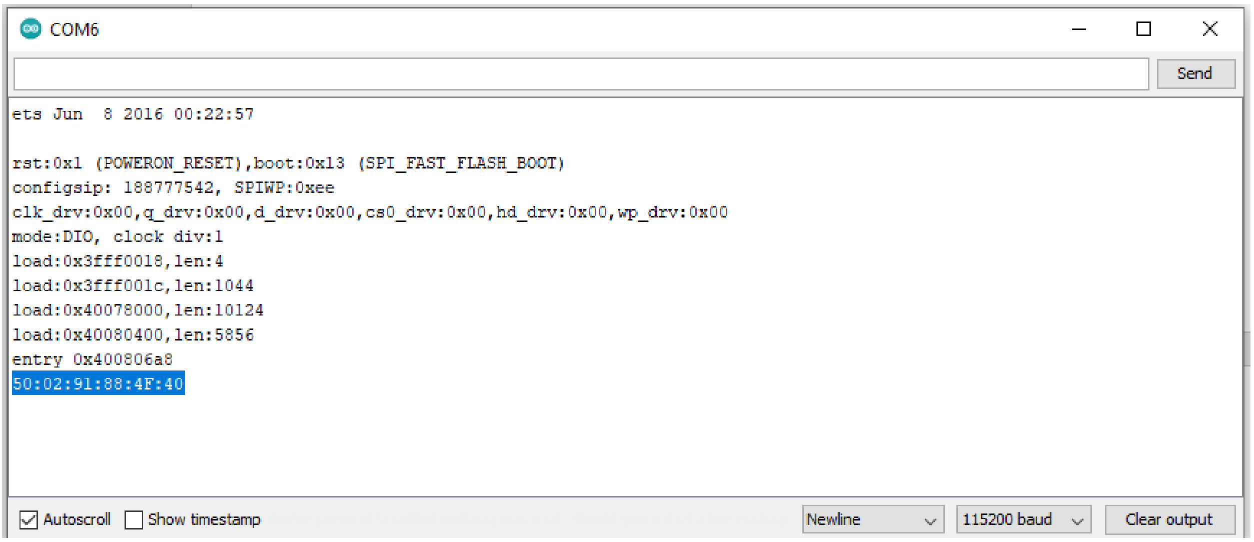
```
Writing at 0x00060000... (87 %)
Writing at 0x00064000... (91 %)
Writing at 0x00068000... (95 %)
Writing at 0x0006c000... (100 %)
Wrote 635360 bytes (389028 compressed) at 0x00010000 in 6.0 seconds (effective 840.3 kbit/s)
```

One that is open, press the reset button on the M5Stack (with it still connected to the computer).



The side thingy.

Now, if you look at the serial port, you should read something like this.



```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 188777542, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5856
entry 0x400806a8
50:02:91:88:4F:40
```

☑ Autoscroll □ Show timestamp

Newline 115200 baud

Numbers like the highlighted 50:02:91:88:4F:40 is going to be your boards MAC address.

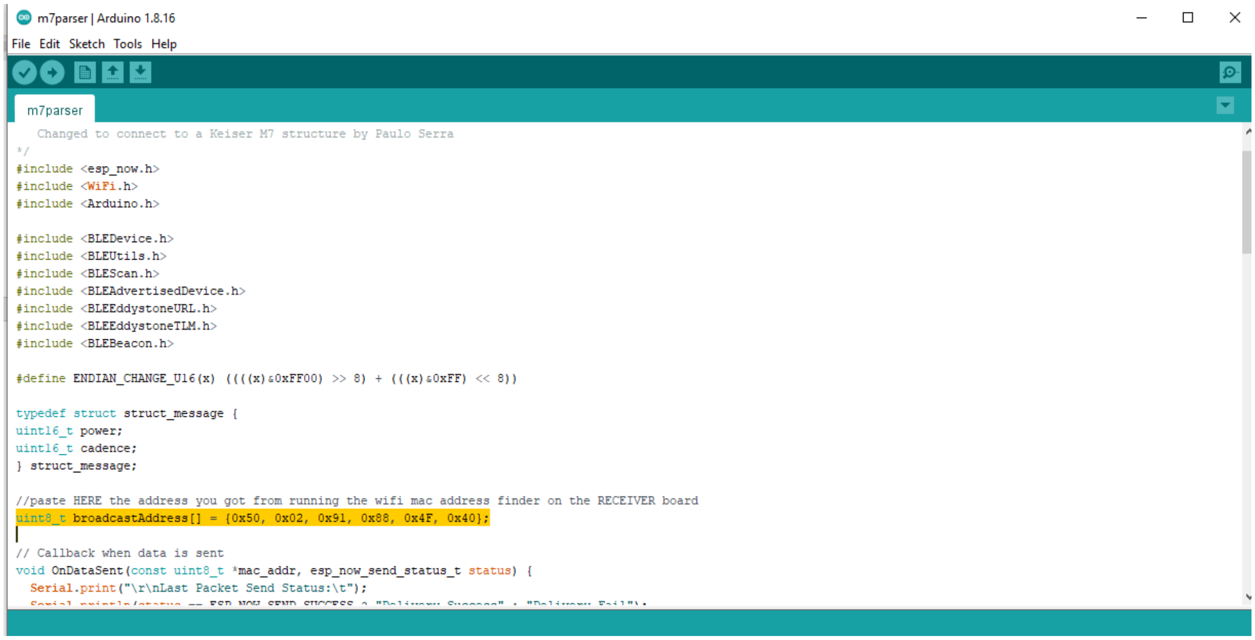
Write that down somewhere and place a sticker on post-it on that board so you know which one has that MAC ADDR.

Almost done …

Now disconnect that board from the computer and connect the other one. (The one without the post-it).

Open the file named: m7parser.ino

This the code that reads data from the bike and sends it to the post-it board.



```
  Changed to connect to a Keiser M7 structure by Paulo Serra
*/
#include <esp_now.h>
#include <WiFi.h>
#include <Arduino.h>

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>
#include <BLEEddystoneURL.h>
#include <BLEEddystoneTLM.h>
#include <BLEBeacon.h>

#define ENDIAN_CHANGE_U16(x) ((((x)&0xFF00) >> 8) + (((x)&0xFF) << 8))

typedef struct struct_message {
uint16_t power;
uint16_t cadence;
} struct_message;

//paste HERE the address you got from running the wifi mac address finder on the RECEIVER board
uint8_t broadcastAddress[] = {0x50, 0x02, 0x91, 0x88, 0x4F, 0x40};

// Callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
```

In the first lines of the file, there is this variable called “broadcastAddress”.

On the field values, place what you got from the previous code, with the format as shown.

Save the file, and press “upload”.

It will build the sketch and upload it to the board.

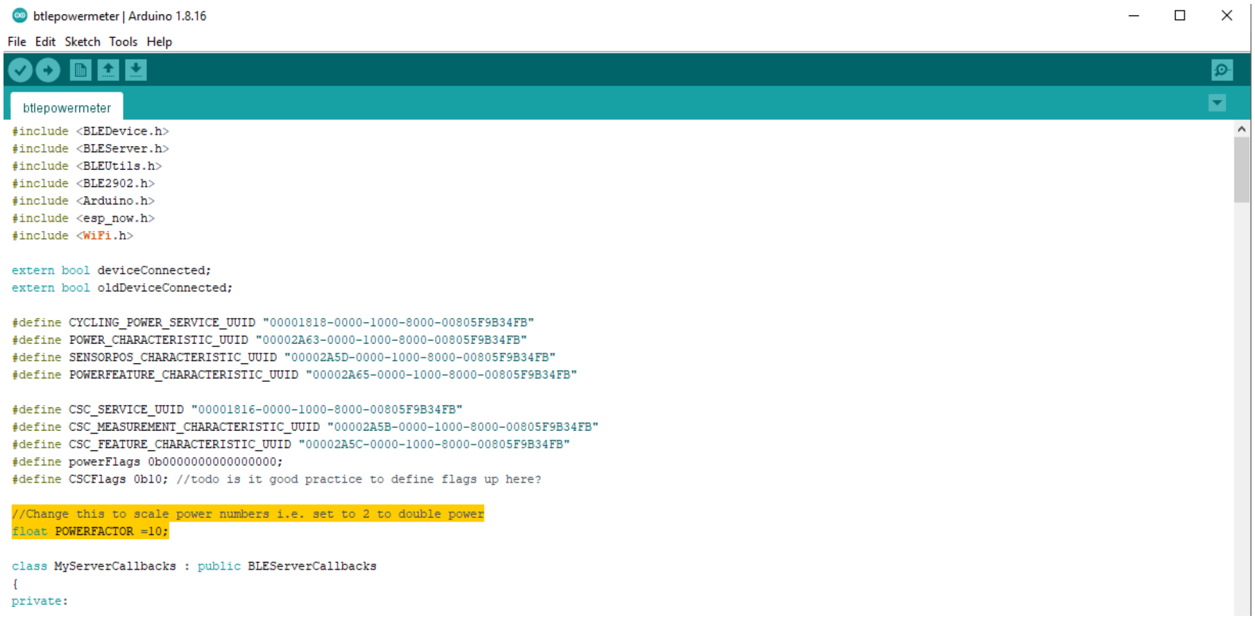
That guy is done!!!

Now open the file: btlepowermeter.ino

Connect the board with the post-it to the computer.

If you want to change the power multiplication factor, there is a variable called “POWERFACTOR” at the beginning of the file. The power output will then be that number * what comes out of the M7.

Set to 10x by default.



```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <Arduino.h>
#include <esp_now.h>
#include <WiFi.h>

extern bool deviceConnected;
extern bool oldDeviceConnected;

#define CYCLING_POWER_SERVICE_UUID "00001818-0000-1000-8000-00805F9B34FB"
#define POWER_CHARACTERISTIC_UUID "00002A63-0000-1000-8000-00805F9B34FB"
#define SENSORPOS_CHARACTERISTIC_UUID "00002A5D-0000-1000-8000-00805F9B34FB"
#define POWERFEATURE_CHARACTERISTIC_UUID "00002A65-0000-1000-8000-00805F9B34FB"

#define CSC_SERVICE_UUID "00001816-0000-1000-8000-00805F9B34FB"
#define CSC_MEASUREMENT_CHARACTERISTIC_UUID "00002A5B-0000-1000-8000-00805F9B34FB"
#define CSC_FEATURE_CHARACTERISTIC_UUID "00002A5C-0000-1000-8000-00805F9B34FB"
#define powerFlags 0b0000000000000000;
#define CSCFlags 0b10; //todo is it good practice to define flags up here?

//Change this to scale power numbers i.e. set to 2 to double power
float POWERFACTOR =10;

class MyServerCallbacks : public BLEServerCallbacks
{
private:
```

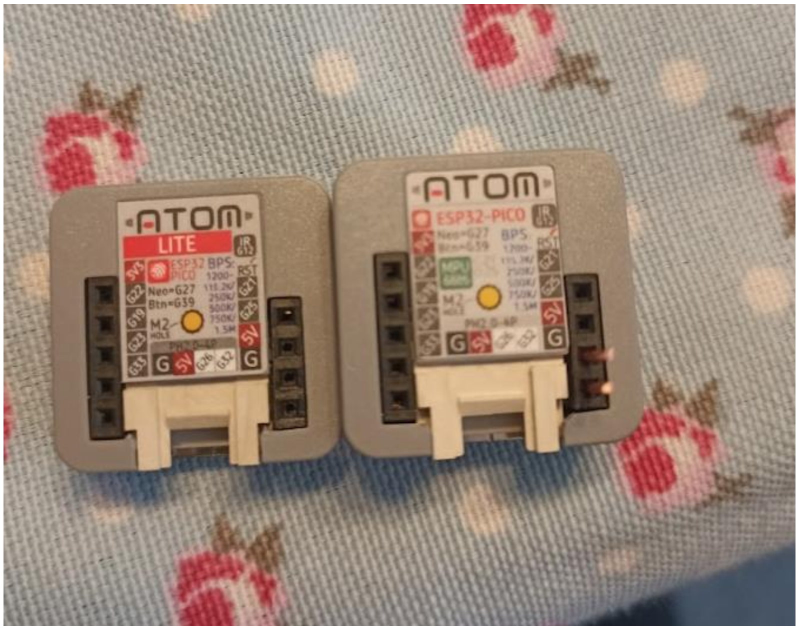
Again, press “upload”, wait for it to finish and it’s done.

Now we just need to find a power source for the devices. Easiest thing is to use a phone charger or a powermeter and connect the USB-C cable to it.

We can wire the power pins on both M5Stacks together, so that we only need 1 cable.

(I didn’t have two ATOM lite available at the time, but same applies)

Get some wires that fit nicely in the 5V and G pin.

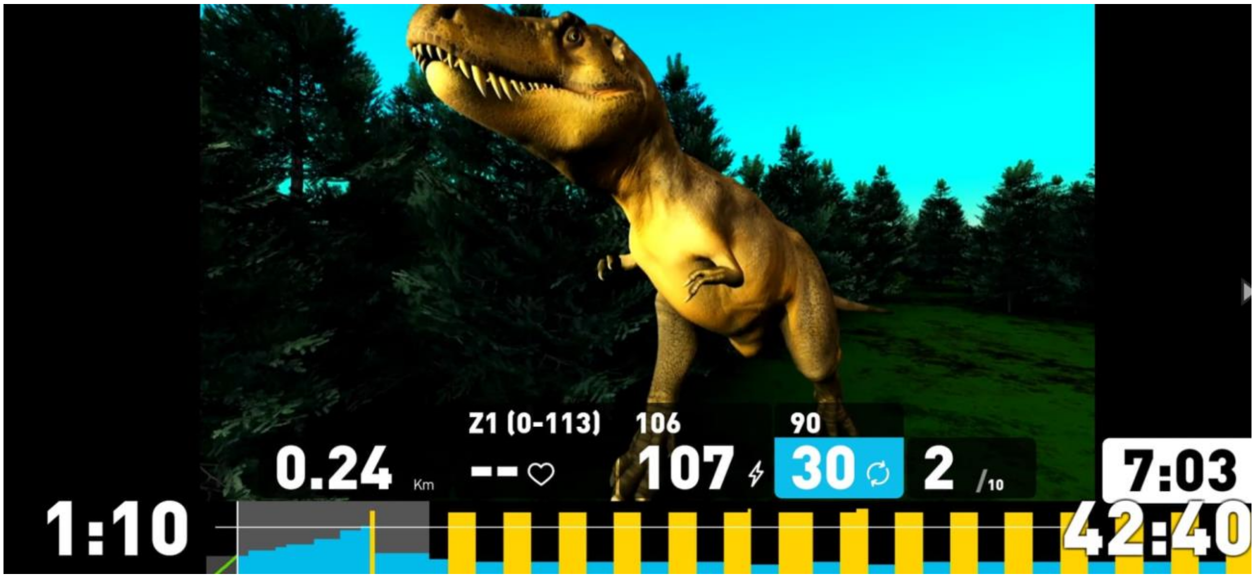


Turn it around, and connect 5V to 5V, G to G.

Now plug the USB-C cable to one of them (does not matter which) and they are ready to transmit.

A powermeter named “KeiserSoze” will be available to connect to on SYSTM / Zwift / RGT / Elemnt Bolt / ROAM / whatever.

When it picks up live data from the M7 bike, it will show live power / cadence data.



Enjoy !