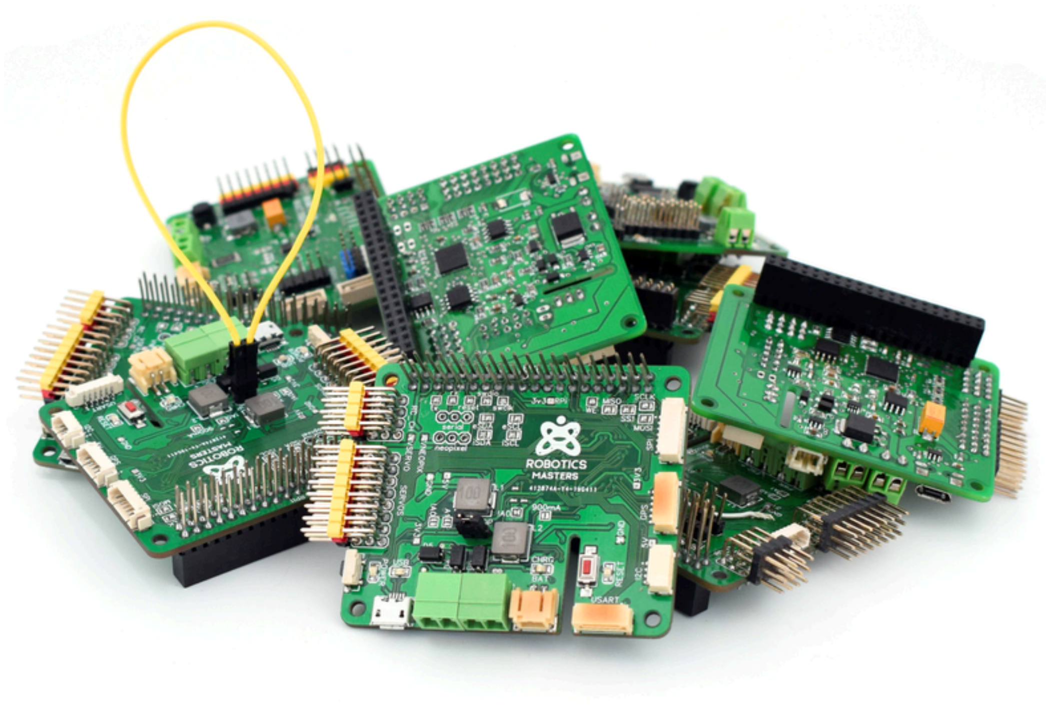
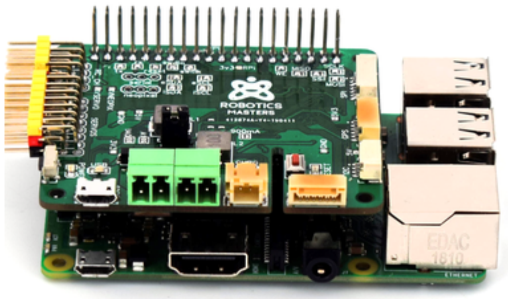
Arduino IDE: Creating Custom Boards

By wallarug in CircuitsArduino

Introduction: Arduino IDE: Creating Custom Boards





Over the past six months I have spent a lot of time porting different libraries to the Robo HAT MM1 board developed by Robotics Masters. This has lead to discovering a lot about these libraries, how they work behind the scenes and most importantly - what to do to add new boards in the future.

This is the first in a series of write ups I will be doing to help others who wish to port libraries for their boards. Many of the sources of information can be vague or difficult for outsiders to understand. I hope to ‘demystify’ and explain how to achieve a successful port for everyone.

Today, we will be looking at the Arduino Platform. It has over 700,000 different board variants around the world and is one of the most popular electronics platforms for education, industry and makers.

I could only find very limited sources of information on how to do this after many Google Searches. So I thought I would write about how I did it in detail.

Here we go!

Step 1: Before You Begin

Before you begin with porting a software library or firmware to your board, you must know a few key points about the technology you are using and be able to answer the questions below.

1. What processor are you using?
2. What architecture does it use?
3. Do I have access to the datasheet for this microprocessor?
4. Is there a similar board on the market that uses the same microprocessor?

These are very important. It will impact on many aspects of how you approach the development process.

Arduino boards commonly use a limited number of processor types and architectures. The most common being the ATMEGA range using the AVR architecture (Arduino Uno). There are newer generations of Arduinos becoming more common using the SAMD processors (ARM) and other more powerful processors. So it is important to check which one you are using.

The datasheet for a microprocessor is absolutely vital to ensure that the board responds as expected when you compile the firmware. Without it, you will not be able to set the correct pin output functions or configure serial ports.

Once you have all the information you need about the processor you are using, you can start to look at the software and modify it to work for your custom board.

Step 2: Overview

The hardest part of any project is finding a good starting point. This is no different. I struggled to find good tutorials with enough detail on how to create custom boards for Arduino. Most tutorials show you how to ‘add a custom board’ but not how to ‘create a custom board’. Here is a short summary of what is involved.

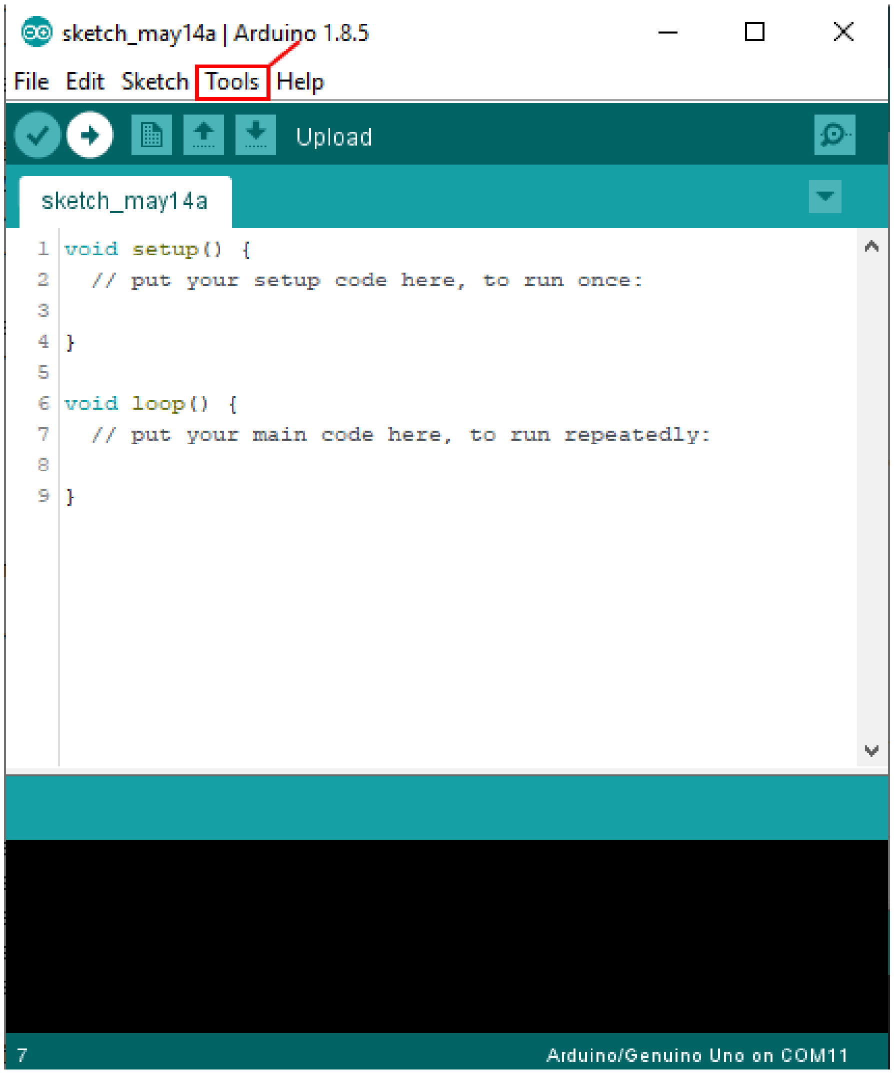
1. Download Existing Board Definitions and Copy
2. Updating Definition Files (variant.h, varient.cpp)
3. Create Board Entry (board.txt)
4. Update Board Version (platform.txt)
5. Preparing for Install (json)
6. Installing Board in Arduino IDE

Each step will be explained in detail below. There will also be extensive discussion on how each file interacts with each other to help clarify how everything works behind the Arduino IDE.

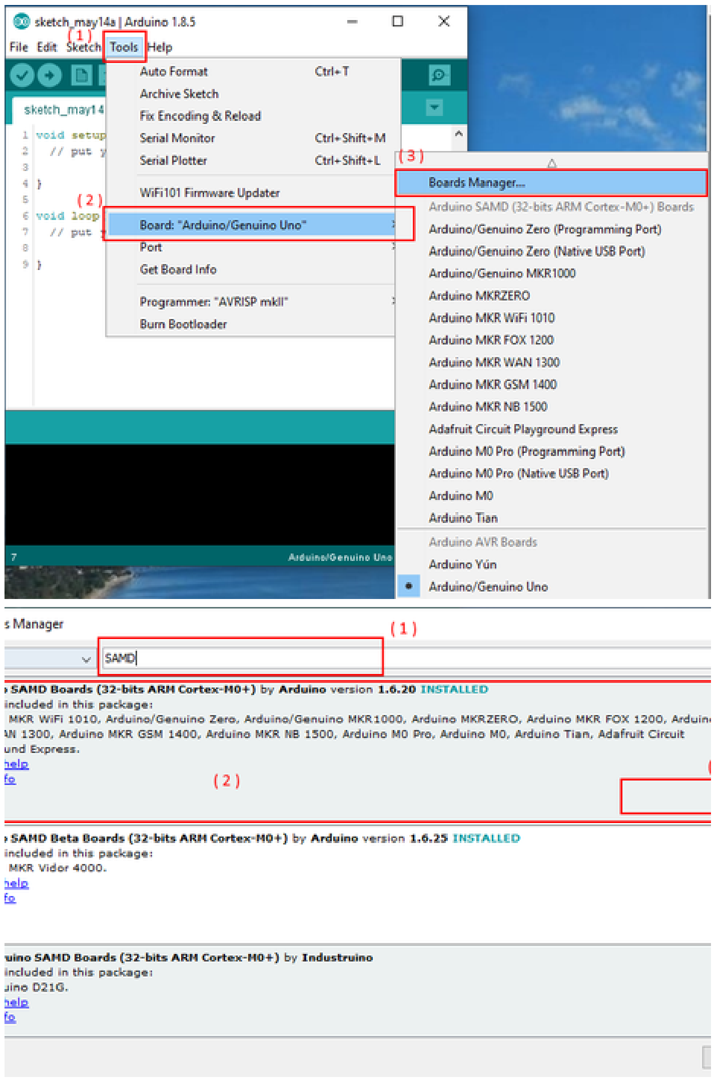
For this tutorial, I will be showing you how to create a custom board for SAMD processors. More specifically, the SAMD21G18A - which is the microprocessor used on the Robo HAT MM1 board which I was porting.

I also assume you already have Arduino IDE 1.8 or later downloaded. I used Arduino 1.8.9 at the time of writing.

Step 3: Downloading Existing Boards



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```


sketch_may14a | Arduino 1.8.5
File Edit Sketch Tools Help
(1)
(2)
(3)
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino SAMD (32-bits ARM Cortex-M0+) Boards
Arduino/Genuino Zero (Programming Port)
Arduino/Genuino Zero (Native USB Port)
Arduino/Genuino MKR1000
Arduino MKRZERO
Arduino MKR WiFi 1010
Arduino MKR FOX 1200
Arduino MKR WAN 1300
Arduino MKR GSM 1400
Arduino MKR NB 1500
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
s Manager
(1)
SAMD
(2)

› SAMD Boards (32-bits ARM Cortex-M0+) by **Arduino** version **1.6.20** INSTALLED
included in this package:
MKR WiFi 1010, Arduino/Genuino Zero, Arduino/Genuino MKR1000, Arduino MKRZERO, Arduino MKR FOX 1200, Arduin
AN 1300, Arduino MKR GSM 1400, Arduino MKR NB 1500, Arduino M0 Pro, Arduino M0, Arduino Tian, Adafruit Circuit
und Express.
help
fo

› SAMD Beta Boards (32-bits ARM Cortex-M0+) by **Arduino** version **1.6.25** INSTALLED
included in this package:
MKR Vidor 4000.
help
fo

uino SAMD Boards (32-bits ARM Cortex-M0+) by **Industruino**
included in this package:
uino D21G.
help
fo

The first step is to download the closest variant Arduino board which matches your board. For SAMD boards, this is the Arduino Zero.

Since Arduino IDE 1.6, the method for downloading new boards into the environment has been by adding special JSON files that software developers provide and then installing the custom boards using the “Boards Manager”. Previous versions of Arduino IDE used a different method that we won’t be discussing today. We will be creating our own JSON file later in this tutorial, however, we need to add the Arduino Zero board using this method first.

Lucky for us, the board we want to download does not need a JSON file because the JSON file is pre-bundled with Arduino IDE - so we just need to install the board from “Boards Manager”.

To do this, go to “Tools” then expand the ”Board” menu. At the top of the “Board” menu will be the “Boards Manager”. Click on this menu option to bring up the Boards Manager.

(See Images)

When the Boards Manager is opened, it will look at all the JSON files that it has stored in the Arduino IDE and then download the settings from the file. You should see a long list of available Arduino boards that you can install.

(See Images)

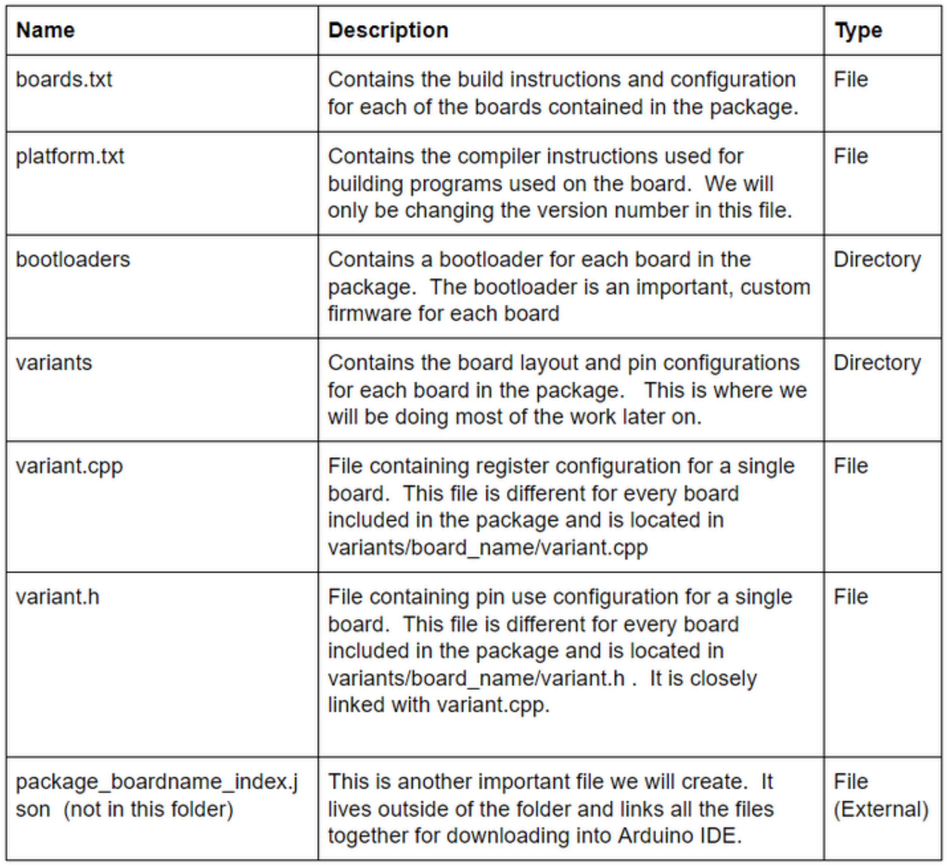
We are only interested in the “Arduino SAMD Boards (32-bits ARM Cortex-M0+)” board for this tutorial, but you could deviate and install the board you need at this point.
Please search for and install the “Arduino SAMD Boards (32-bits ARM Cortex-M0+)” board. This can be done by

clicking on the text from the list followed by the “Install” button on the right hand side which will appear after you click on the text. It will take a couple of minutes to install.

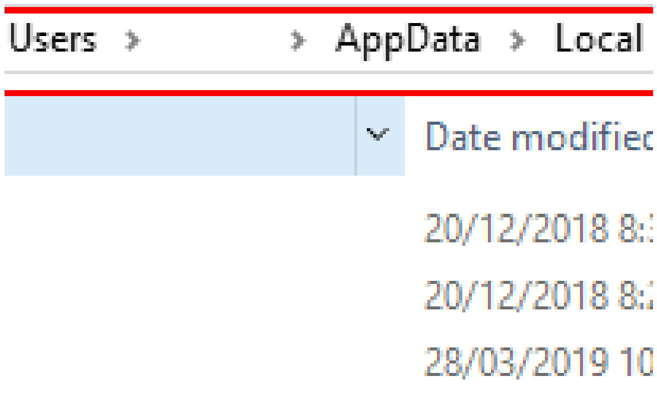
For more details installing new boards: Adafruit have a great tutorial here which explains how to install their Feature M0 boards.

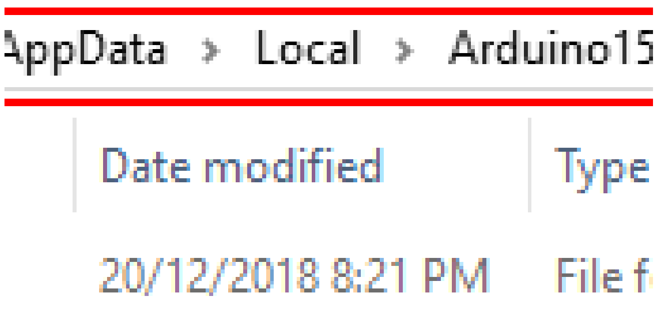
Now that the board files have been installed, we are able to copy them so that they can be modified for your custom board.

Step 4: Finding and Copying Board Files



Name	Description	Type
boards.txt	Contains the build instructions and configuration for each of the boards contained in the package.	File
platform.txt	Contains the compiler instructions used for building programs used on the board. We will only be changing the version number in this file.	File
bootloaders	Contains a bootloader for each board in the package. The bootloader is an important, custom firmware for each board	Directory
variants	Contains the board layout and pin configurations for each board in the package. This is where we will be doing most of the work later on.	Directory
variant.cpp	File containing register configuration for a single board. This file is different for every board included in the package and is located in variants/board_name/variant.cpp	File
variant.h	File containing pin use configuration for a single board. This file is different for every board included in the package and is located in variants/board_name/variant.h . It is closely linked with variant.cpp.	File
package_boardname_index.json (not in this folder)	This is another important file we will create. It lives outside of the folder and links all the files together for downloading into Arduino IDE.	File (External)





For Windows the Board Files are located in (remember to change username to your username):

C:\Users\username\AppData\Local\Arduino15\packages\

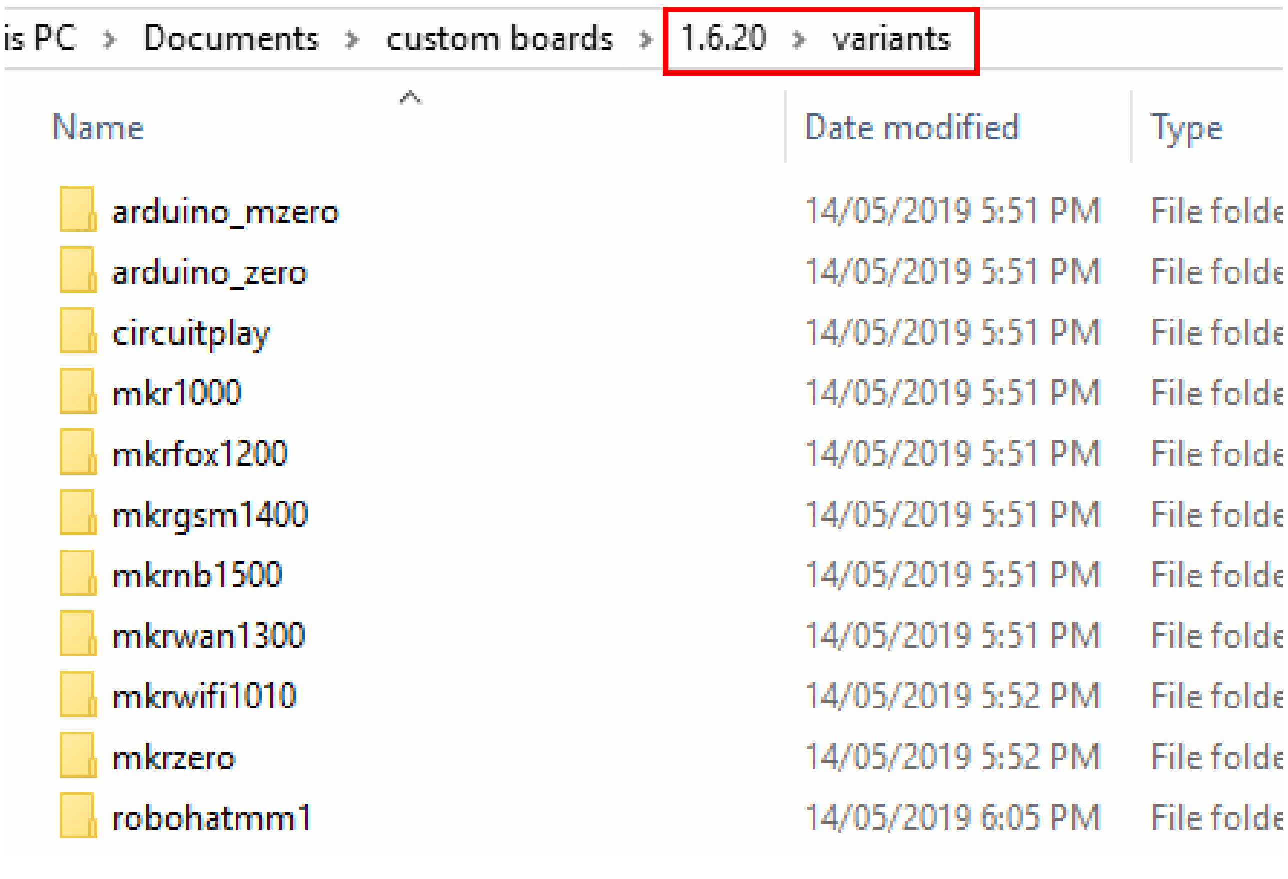
In this folder, you have to go a little deeper to get to the files that you need to copy to modify. For this tutorial we will go and get the Arduino Zero board files which will be installed to (remember to change username to your username):

C:\Users\username\AppData\Local\Arduino15\packages\arduino\hardware\samd

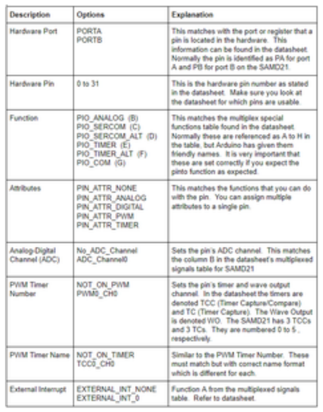
Copy the version-numbered folder located in this directory to a new folder in your Documents folder or folder of your choosing. For the purposes of this tutorial, I will put them in a new folder named ‘custom boards’ inside Documents.

The folder contains a number of folders and directories. The ones we will be using are noted in the table in the screenshot.

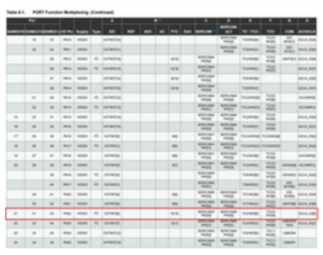
Step 5: Creating Variant


is PC › Documents › custom boards › 1.6.20 › variants

Name	Date modified	Type
arduino_mzero	14/05/2019 5:51 PM	File folde
arduino_zero	14/05/2019 5:51 PM	File folde
circuitplay	14/05/2019 5:51 PM	File folde
mkr1000	14/05/2019 5:51 PM	File folde
mkrfox1200	14/05/2019 5:51 PM	File folde
mkrgsm1400	14/05/2019 5:51 PM	File folde
mkrnb1500	14/05/2019 5:51 PM	File folde
mkrwan1300	14/05/2019 5:51 PM	File folde
mkrwifi1010	14/05/2019 5:52 PM	File folde
mkrzero	14/05/2019 5:52 PM	File folde
robohatmm1	14/05/2019 6:05 PM	File folde



Description	Options	Explanation
Hardware Port	PORTA PORTB	This matches with the port or register that a pin is located in the hardware. This information can be found in the datasheet. Normally the pin is identified as PA for port A and PB for port B on the SAMD21.
Hardware Pin	0 to 31	This is the hardware pin number as stated in the datasheet. Make sure you look at the datasheet for which pins are usable.
Function	PIO_ANALOG (B) PIO_SERCOM (C) PIO_SERCOM_ALT (D) PIO_TIMER (E) PIO_TIMER_ALT (F) PIO_COM (G)	This matches the multiplex special functions table found in the datasheet. Normally these are referenced as A to H in the table, but Arduino has given them friendly names. It is very important that these are set correctly if you expect the pinto function as expected.
Attributes	PIN_ATTR_NONE PIN_ATTR_ANALOG PIN_ATTR_DIGITAL PIN_ATTR_PWM PIN_ATTR_TIMER	This matches the functions that you can do with the pin. You can assign multiple attributes to a single pin.
Analog-Digital Channel (ADC)	No_ADC_Channel ADC_Channel0	Sets the pin's ADC channel. This matches the column B in the datasheet's multiplexed signals table for SAMD21
PWM Timer Number	NOT_ON_PWM PWM0_CH0	Sets the pin's timer and wave output channel. In the datasheet the timers are denoted TCC (Timer Capture/Compare) and TC (Timer Capture). The Wave Output is denoted WO. The SAMD21 has 3 TCCs and 3 TCs. They are numbered 0 to 5, respectively.
PWM Timer Name	NOT_ON_TIMER TCC0_CH0	Similar to the PWM Timer Number. These must match but with correct name format which is different for each.
External Interrupt	EXTERNAL_INT_NONE EXTERNAL_INT_0	Function A from the multiplexed signals table. Refer to datasheet.



We are now ready to start creating a custom board entry. For this tutorial, I will be using the Robo HAT MM1 as the example. As previously mentioned, it is a SAMD based board which is closest matched to the Arduino Zero build.

We will start by going into the variants folder and copying the arduino_zero folder located in there. We will rename the new board variant 'robohatmm1'. You can call yours whatever you like.

(See Image)

Inside the robohatmm1 folder will be the two pins that we need to start editing: variant.cpp and variant.h. Open both of them up.

This took me a while to work out, so I will explain it here to save you some time. The variant.cpp file contains a very large array of pins which is referenced throughout variant.h. All the pin references in variant.h are a reference to the pin configuration at a particular position in the variant.cpp array.

(See Screenshot with Two Files)

So, most of your edits will be occurring in both of these files but you need to make sure that if you change the order of the pins in variants.cpp - you must change the references in the header file (variants.h).
For the Robo HAT MM1, I only needed to change some of the pins and functions. This was done in variants.h. I added some new PWM pins since the SAMD21 is able to handle 12 PWM channels. I also added some names for Servos, Signals (instead of ADC/Digital) and custom mapping to the right functions - such as SPI, UART and I2C.

The important thing to note is to double check that the array references you use for the functions outlined in variants.h match that of the pin in varaints.cpp - which we will now look at.

Variants.cpp is a very powerful and important file. It does all the hard work of setting the pins to match the hardware settings. The easiest way to explain this is with an example and explanation of each part.

(See Table In Screenshot)

Example (extract from variants.cpp)

```
{ PORTA, 22, PIO_SERCOM, PIN_ATTR_NONE, No_ADC_Channel, NOT_ON_PWM, NOT_ON_TIMER,
EXTERNAL_INT_NONE }, // SDA
```

This is the first pin in the variants.cpp array for the Robo HAT MM1. The table from the datasheet has been provided as an image (grey table).

(See Images)

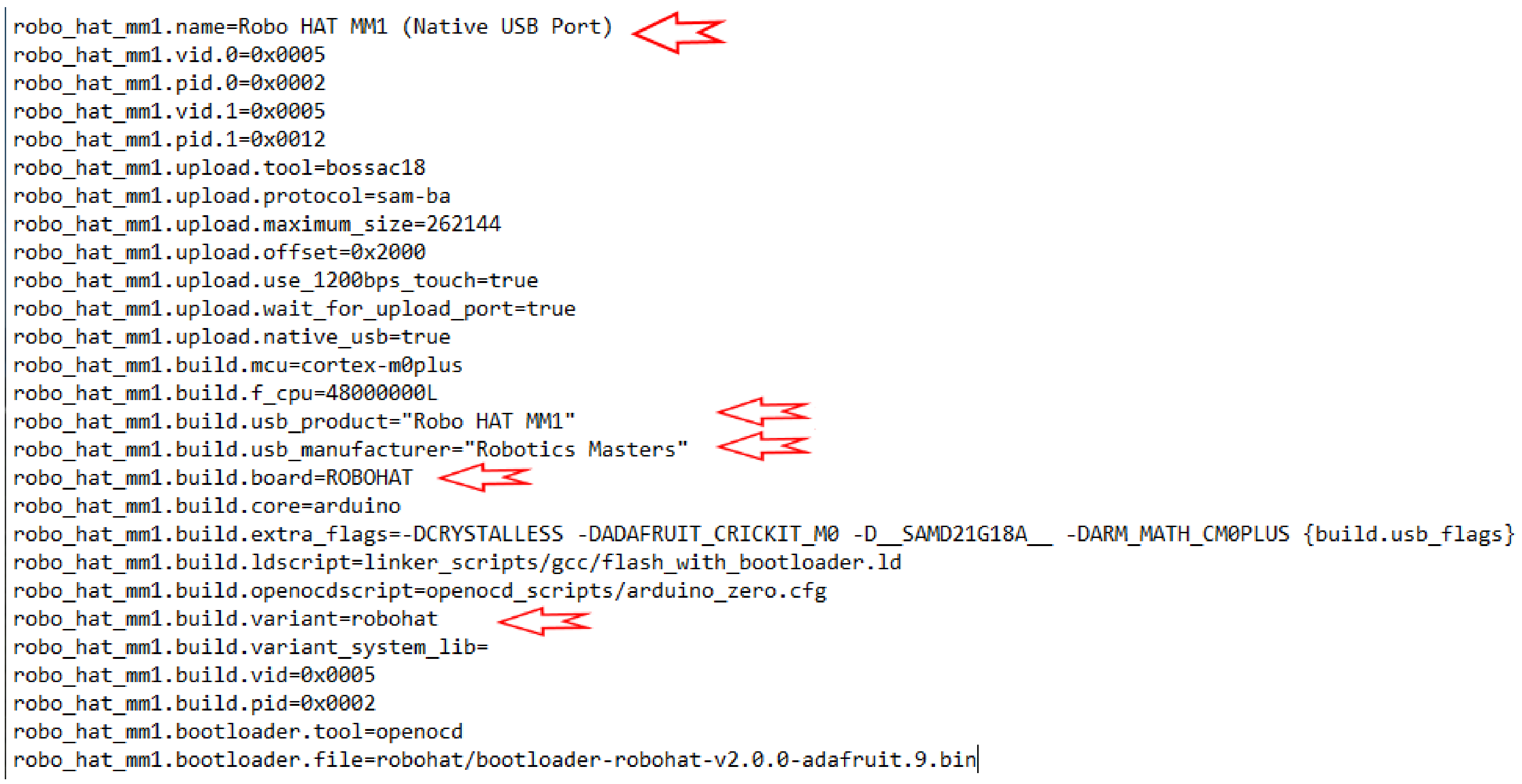
This particular pin is used for the I2C Data Pin Function. Taking a look at the table, we are able to see that this pin is able to be used as an I2C SDA pin (good start!).
The pin is named “PA22” which is short for PORTA on pin 22. Right away we can set the PORT and pin number for this pin.

The next thing we need to do is set the pin as a serial communication port. The pin has serial communication available through function C (PIO_SERCOM) using SERCOM 3 and D using SERCOM5 (PIO_SERCOM_ALT). For the purposes of the Robo HAT MM1, we are using SERCOM3 for I2C communication. This is on Function C; aka. PIO_SERCOM for variants.cpp.

Since we plan to only use this pin as an I2C SDA, there is no need to set any of the other functions. They can all be set as “No” or “Not” options from the previous table. However, if we did want to use the other functions - we could go across the datasheet and put them all into the correct spaces. It is all in the datasheet.

Modifying the variant files can take some time. Be careful and always triple check.

Step 6: Create a Board Definition



```
robo_hat_mm1.name=Robo HAT MM1 (Native USB Port)
robo_hat_mm1.vid.0=0x0005
robo_hat_mm1.pid.0=0x0002
robo_hat_mm1.vid.1=0x0005
robo_hat_mm1.pid.1=0x0012
robo_hat_mm1.upload.tool=bossac18
robo_hat_mm1.upload.protocol=sam-ba
robo_hat_mm1.upload.maximum_size=262144
robo_hat_mm1.upload.offset=0x2000
robo_hat_mm1.upload.use_1200bps_touch=true
robo_hat_mm1.upload.wait_for_upload_port=true
robo_hat_mm1.upload.native_usb=true
robo_hat_mm1.build.mcu=cortex-m0plus
robo_hat_mm1.build.f_cpu=48000000L
robo_hat_mm1.build.usb_product="Robo HAT MM1"
robo_hat_mm1.build.usb_manufacturer="Robotics Masters"
robo_hat_mm1.build.board=ROBOHAT
robo_hat_mm1.build.core=arduino
robo_hat_mm1.build.extra_flags=-DCRYSTALLESS -DADAFRUIT_CRICKIT_M0 -D__SAMD21G18A__ -DARM_MATH_CM0PLUS {build.usb_flags}
robo_hat_mm1.build.ldscript=linker_scripts/gcc/flash_with_bootloader.ld
robo_hat_mm1.build.openocdscript=openocd_scripts/arduino_zero.cfg
robo_hat_mm1.build.variant=robohat
robo_hat_mm1.build.variant_system_lib=
robo_hat_mm1.build.vid=0x0005
robo_hat_mm1.build.pid=0x0002
robo_hat_mm1.bootloader.tool=openocd
robo_hat_mm1.bootloader.file=robohat/bootloader-robohat-v2.0.0-adafruit.9.bin
```

Once you have your variant files ready, it should be straight forward from here. Most of the work will be copying and pasting or modifying and updating files.

Starting with boards.txt.

(See Image)

You will want to copy and paste a board definition that is already there. I would recommend the Arduino Zero again.

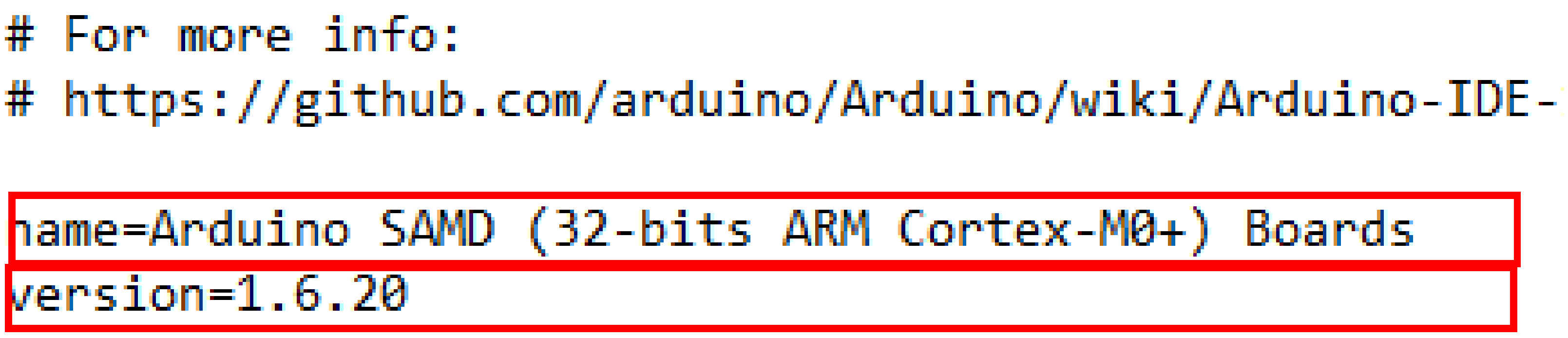
For simplicity, only change the board name (first line), usb_product, usb_manufacturer, and variant (robohat). You can customise the other arguments later on to suit your needs - such as a custom bootloader or different USB VID/PIDs for identifying your board.

The board variant must match the name given to the folder created at the start. For this tutorial I called it ‘robohatmm1’.

It would also be recommended changing the first part of each line to match your board name. In the screenshot it has been changed to ‘robo_hat_mm1’. You should choose a name for your board with the same format.

That is all for boards.txt unless you want to do further modifications mentioned above later.

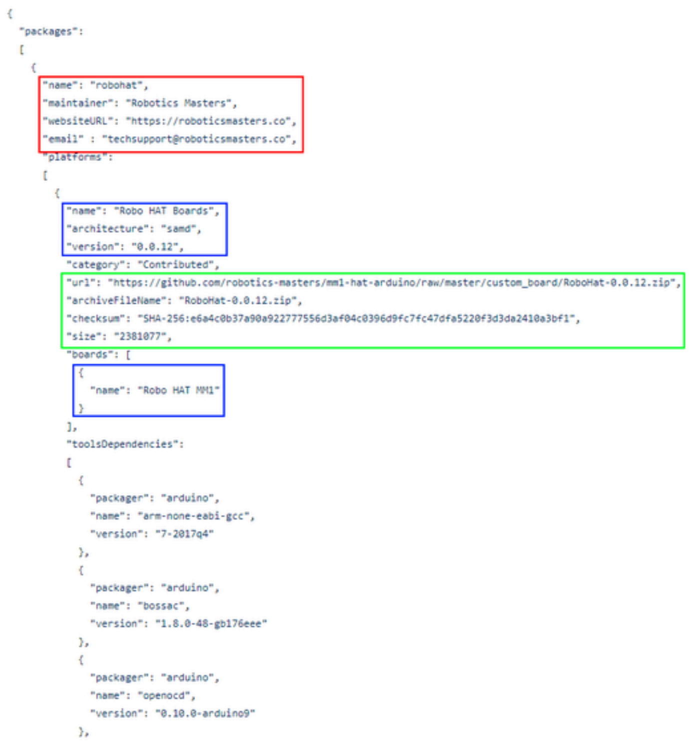
Step 7: Update Board Version



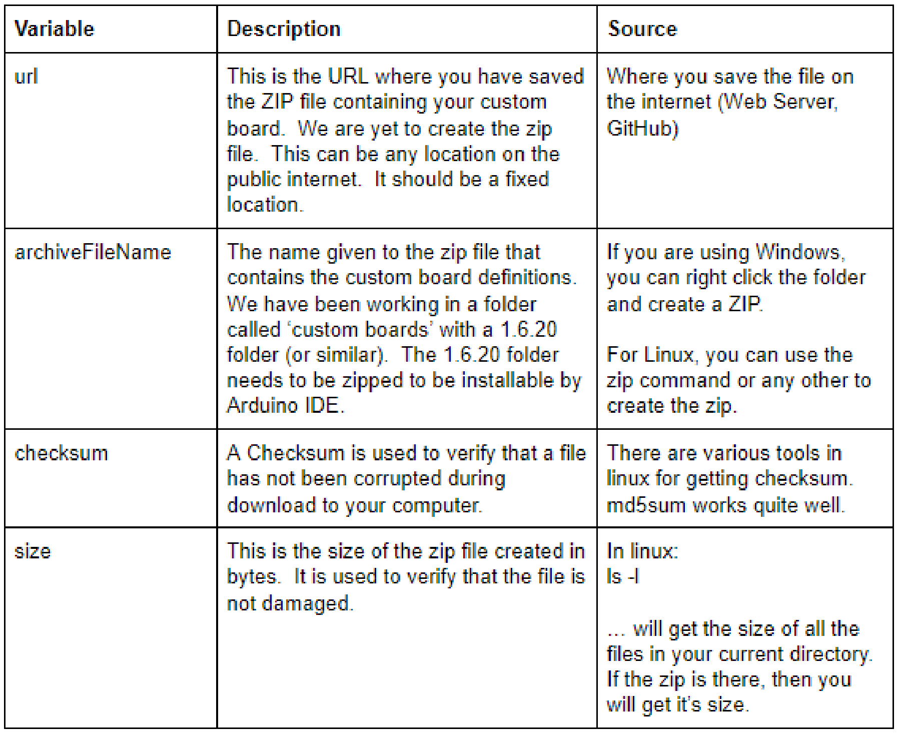
```
# For more info:
# https://github.com/arduino/Arduino/wiki/Arduino-IDE-

name=Arduino SAMD (32-bits ARM Cortex-M0+) Boards
version=1.6.20
```

In platforms.txt change the name to the name of your custom board. Also change the version number. Remember what you set this to, we will need it later.

Step 8: Create JSON Package File



```
{
  "packages":
  [
    {
      "name": "robohat",
      "maintainer": "Robotics Masters",
      "websiteURL": "https://roboticsmasters.co",
      "email" : "techsupport@roboticsmasters.co",
      "platforms":
      [
        {
          "name": "Robo HAT Boards",
          "architecture": "samd",
          "version": "0.0.12",
          "category": "Contributed",
          "url": "https://github.com/robotics-masters/mm1-hat-arduino/raw/master/custom_board/RoboHat-0.0.12.zip",
          "archiveFileName": "RoboHat-0.0.12.zip",
          "checksum": "SHA-256:e6a4c0b37a90a922777556d3af04c0396d9fc7fc47dfa5220f3d3da2410a3bf1",
          "size": "2381077",
          "boards": [
            {
              "name": "Robo HAT MM1"
            }
          ],
          "toolsDependencies":
          [
            {
              "packager": "arduino",
              "name": "arm-none-eabi-gcc",
              "version": "7-2017q4"
            },
            {
              "packager": "arduino",
              "name": "bossac",
              "version": "1.8.0-48-gb176eee"
            },
            {
              "packager": "arduino",
              "name": "openocd",
              "version": "0.10.0-arduino9"
            },
```



Variable	Description	Source
url	This is the URL where you have saved the ZIP file containing your custom board. We are yet to create the zip file. This can be any location on the public internet. It should be a fixed location.	Where you save the file on the internet (Web Server, GitHub)
archiveFileName	The name given to the zip file that contains the custom board definitions. We have been working in a folder called 'custom boards' with a 1.6.20 folder (or similar). The 1.6.20 folder needs to be zipped to be installable by Arduino IDE.	If you are using Windows, you can right click the folder and create a ZIP. For Linux, you can use the zip command or any other to create the zip.
checksum	A Checksum is used to verify that a file has not been corrupted during download to your computer.	There are various tools in linux for getting checksum. md5sum works quite well.
size	This is the size of the zip file created in bytes. It is used to verify that the file is not damaged.	In linux: ls -l … will get the size of all the files in your current directory. If the zip is there, then you will get it's size.

In order to install your board in Arduino IDE, you will need to create a JSON file that you will import. The JSON file tells Arduino IDE where to get the files to install the board, what extra packages are needed and a few other bits of metadata.

It is very important that you save this file outside of the folder we have just been working in.

Most of the file can be copied and pasted into your one. You will only need to change the "boards" section and the other metadata at the top of the file. See screenshot for what should be updated.

(See Image)

- **Red Section:** This is metadata that users can use for finding help. It has very little technical relevance.
- **Blue Section:** All of these are important. They show in the Arduino IDE. Name, Architecture and Version will all be shown to anyone who is trying to install the package. This is where you need to put the version number from platforms.txt. The second blue section is the list of boards that are included in the package. You could have multiple boards.
- **Green Section:** This section needs further explanation.

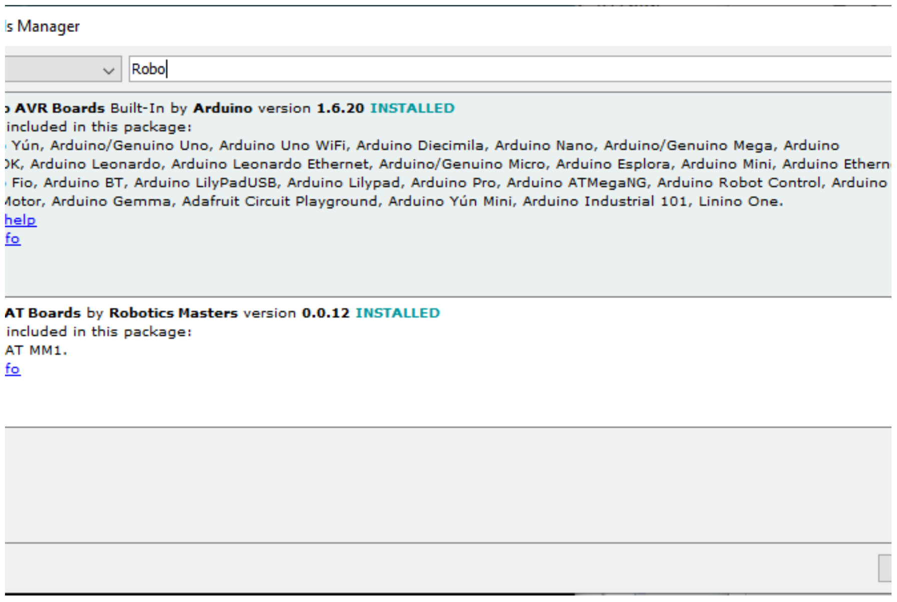
(See Table in Images)

Once you have created the zip file, got the zip file checksum and file size, you are now able to upload the zip file to a location. You will need to put that URL into the 'url' field. If the name or any of the above details are incorrect, your custom board will fail to install.

Make sure to also upload your **package_boardname_index.json** file to a public location on the internet. GitHub is a good option.

The Robo HAT MM1 custom board file can be found here.

Step 9: The Final Step - Install Your Custom Board!

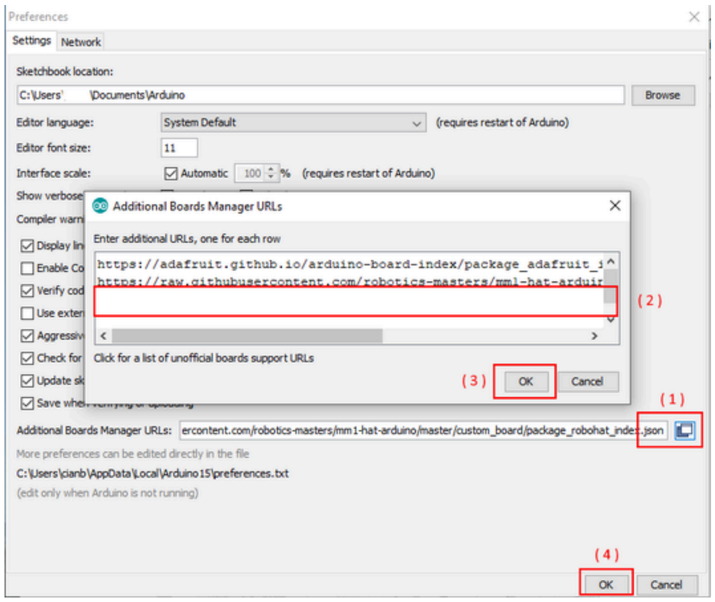


ls Manager

Robo

› AVR Boards Built-In by Arduino version 1.6.20 INSTALLED
included in this package:
· Yún, Arduino/Genuino Uno, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino/Genuino Mega, Arduino
OK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino/Genuino Micro, Arduino Esplora, Arduino Mini, Arduino Ethern
· Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino
Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
help
fo

AT Boards by Robotics Masters version 0.0.12 INSTALLED
included in this package:
AT MM1.
fo



Preferences

Settings | Network

Sketchbook location: C:\Users\ \Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 11

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

Additional Boards Manager URLs

Enter additional URLs, one for each row

```
https://adafruit.github.io/arduino-board-index/package_adafruit_i
https://raw.githubusercontent.com/robotics-masters/mm1-hat-arduir
```

Click for a list of unofficial boards support URLs

Additional Boards Manager URLs: ercontent.com/robotics-masters/mm1-hat-arduino/master/custom_board/package_robohat_index.json

More preferences can be edited directly in the file
C:\Users\cianb\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

All going well, you should now be able to include your JSON file in Arduino IDE and install your custom board.

Including the JSON file is easy! In Arduino IDE - just go to “File” > “Preferences” and copy and paste the location (URL) of your package_boardname_index.json to the “Additional Boards Manager URLs” section on the bottom of the Preferences menu.

(See Image)

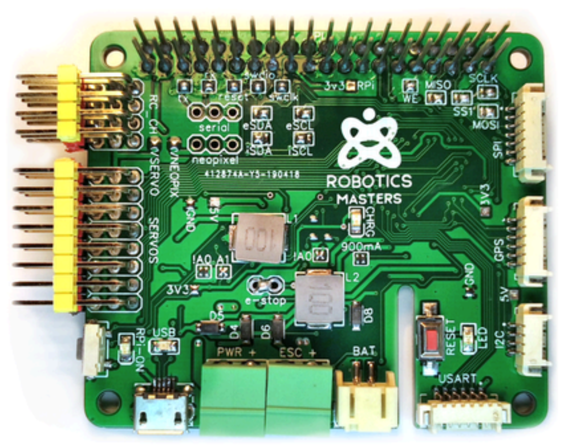
Then hit OK.

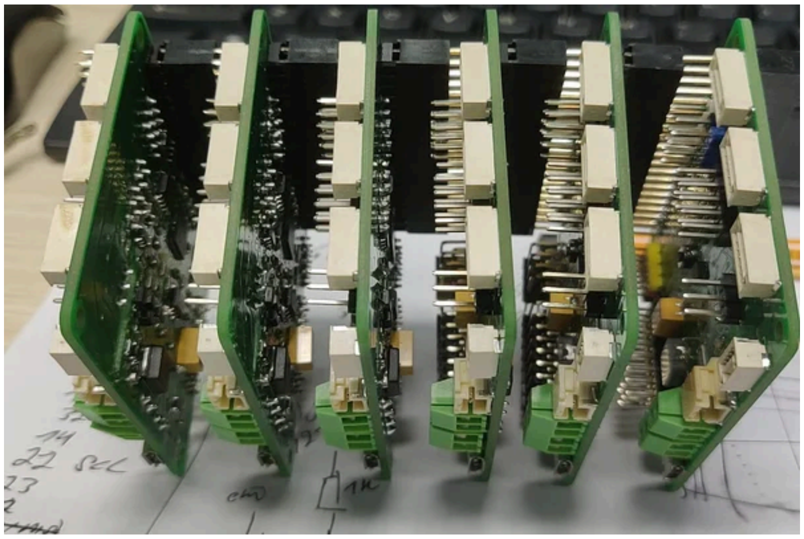
Run through the instructions from above for installing new boards if you have forgotten. Remember to look for your custom board this time!

(See Image)

For more details installing new boards: Adafruit have a great tutorial here which explains how to install their Feature M0 boards.

Step 10: Conclusion





This has been fun creating custom Arduino boards. There is still a lot for me to experiment with in the future (adding additional serial ports) but it has been a great learning experience!

Be sure to check out the Crowd Supply campaign as well. It ends on June 11 2019.

https://www.crowdsupply.com/robotics-masters/robo-...

I hope this helped you or you had fun reading along!

Thanks!