VM5 GEN2 PROJECT TUTORIAL.

In this document we will show you how to setup Arduino IDE For programming the VM5 Device.

This will include how to setup flash options which are critical for operation as the correct partition manager has to be correctly setup, Setting up authentication for Private Product key for security will also prevent the Apps being loaded on Non authenticated devices.

The VM5 Structure is a custom SD Card launcher system there is the main menu which serves as a boot loader swap feature which can load exported bin files Aka the Mini apps which have the function such as word mode Etc. What the launcher essentially does is passes control of the system to these bin files and the bin files have a rollback operation which calls the system to load the menu.bin, The flashed firmware, Similarly we can also update the menu with a separate feature called Update, This works similarly to how the Apps work, the Menu can be updated with any changes in code, Exported to binary Rename binary file to menu.bin, create another duplicate and call this update.bin, Pass the menu.bin to the root of the Sd card, then add the Update.bin to the update folder, Goto settings/update and it will force the device to load the new menu system you just created, To keep that menu system the new menu.bin is loaded whenever you quit an App, which does the exact same thing as the Update feature, In short you could simply overwrite the menu.bin, Load an App then quit the App it will still update to the new menu launcher, Its simply added to simplify updating the menu system.

Updating Apps however is a simple process, Simply export any changes to binary, Got sketch folder and rename the binary file to the App name, Ensure its the same text, then overwrite the old app on the Sd card, The binary Apps are simply temporary operating programs, The App will stay in memory until you physically quit the App, So even turning the device off and on wont quit it.

The menu launcher is quite special, it uses preferences which are stored values on the memory of the VM5, The preferences are used with Apps so your theme settings and other settings are always available across the whole platform, The Apps are built in a similar way to the Menu launcher and share a lot of the menu handling, this gives the whole experience a more integrated feel, All the Apps are created from the same template, just the core function is different, this should make it easy to clone one of the Apps and build an entirely new function as most of the logic is part of the template, When programming an App for the VM5 Make sure to have the rollback to menu options installed or you will be stuck in that App when testing, Its also important not to flash the code directly to the device as you need to test that it functions correctly within the normal menu system.

The menu launcher doesn't have Anti piracy features installed so can be flashed directly to a brand new core2 device, Just make sure to have the Sd card installed and your menu.bin and Icons etc installed to the card, The Operating system wont work without it, What you will want to do first is setup your private key which is found on both the Authenticator app and All Apps provided, The section for this can be found in the Void Setup code block, Create your own private key here, And ensure this exact key is used on All your Apps.

Export the Authenticator and run it on your device After the menu loader has been installed, This authenticates the device with your private key and will remain on the device unless you re flash it with a new menu or burn some other code to it,

Now the Apps you create will only run on devices which have had your private key installed too.

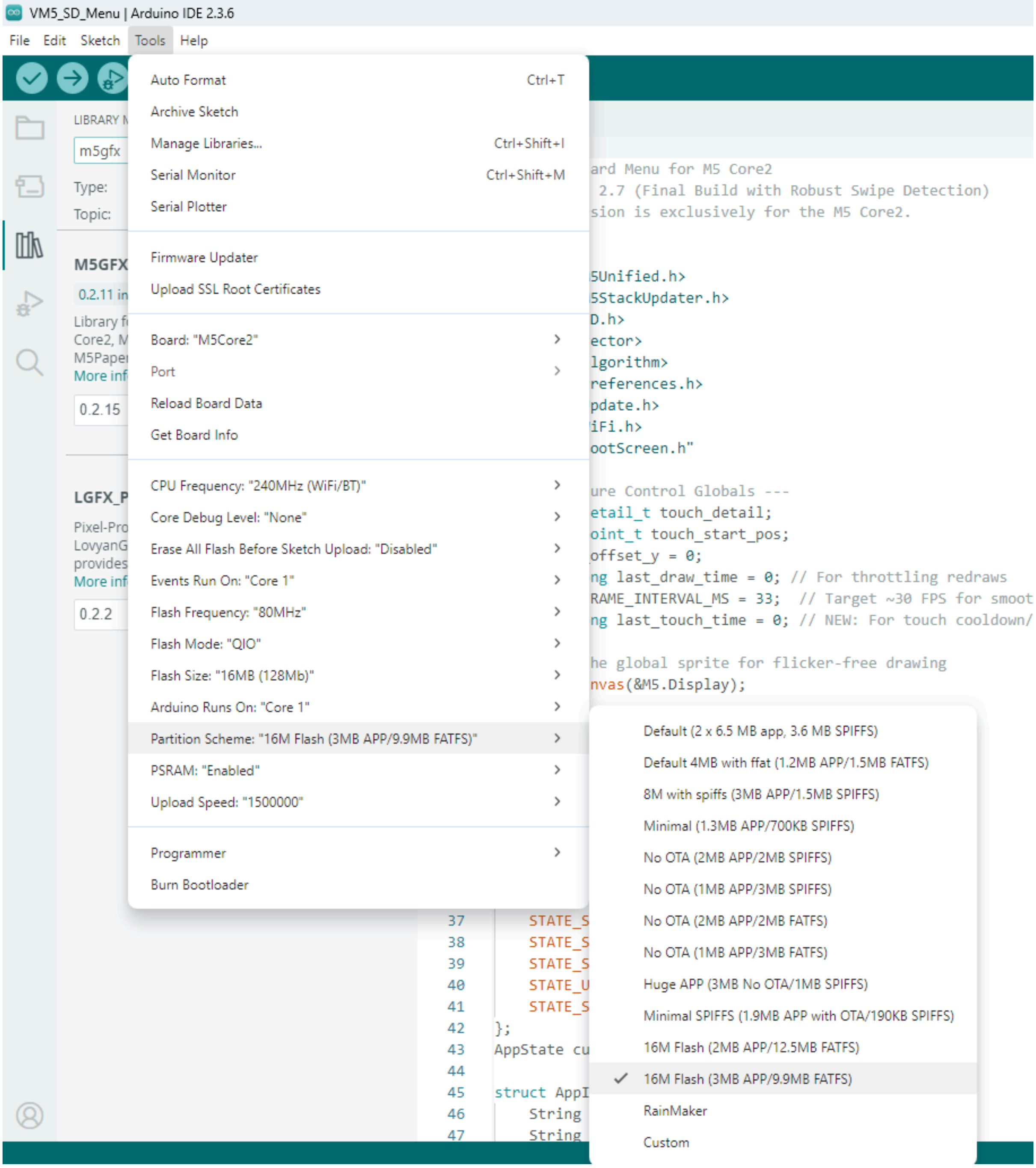
Ok lets setup the Arduino environment to use on your M5 Core2.

You will need to install the M5Core boards, Goto the left hand side of the screen and click on boards, type in the search, M5STACK. And install the boards to your IDE.

Next goto Libraries and search for M5Unified, and M5GFX, Install both libraries,
You may also need some other libraries for the various resources the menu system uses, See the #include header files at the top of each sketch, Those are the required libraries needed for the project. You can copy the name of the header file if the compiler complains when trying to compile the project, Paste the text into the library search and see if its available, If not you can simply paste the text into google search and you should be able to find the whole library on github, Download the library and manually install it to the IDE From the tools dropdown/install from.zip

Once you have all the necessary libraries, And setup your private keys, you can now attempt to export or flash to the device.

This partition scheme must be used on All the project files for the VM5.



VM5_SD_Menu | Arduino IDE 2.3.6
File
Edit
Sketch
Tools
Help
LIBRARY M
m5gfx
Type:
Topic:
M5GFX
0.2.11 in
Library f
Core2, M
M5Paper
More inf
0.2.15
LGFX_P
Pixel-Pro
LovyanG
provides
More inf
0.2.2
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Port
Reload Board Data
Get Board Info
Programmer
Burn Bootloader
Default (2 x 6.5 MB app, 3.6 MB SPIFFS)
Default 4MB with ffat (1.2MB APP/1.5MB FATFS)
8M with spiffs (3MB APP/1.5MB SPIFFS)
Minimal (1.3MB APP/700KB SPIFFS)
No OTA (2MB APP/2MB SPIFFS)
No OTA (1MB APP/3MB SPIFFS)
No OTA (2MB APP/2MB FATFS)
No OTA (1MB APP/3MB FATFS)
Huge APP (3MB No OTA/1MB SPIFFS)
Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)
16M Flash (2MB APP/12.5MB FATFS)
✓ 16M Flash (3MB APP/9.9MB FATFS)
RainMaker
Custom

Board: "M5Core2"

CPU Frequency: "240MHz (WiFi/BT)"

Core Debug Level: "None"

Erase All Flash Before Sketch Upload: "Disabled"

Events Run On: "Core 1"

Flash Frequency: "80MHz"

Flash Mode: "QIO"

Flash Size: "16MB (128Mb)"

Arduino Runs On: "Core 1"

Partition Scheme: "16M Flash (3MB APP/9.9MB FATFS)"

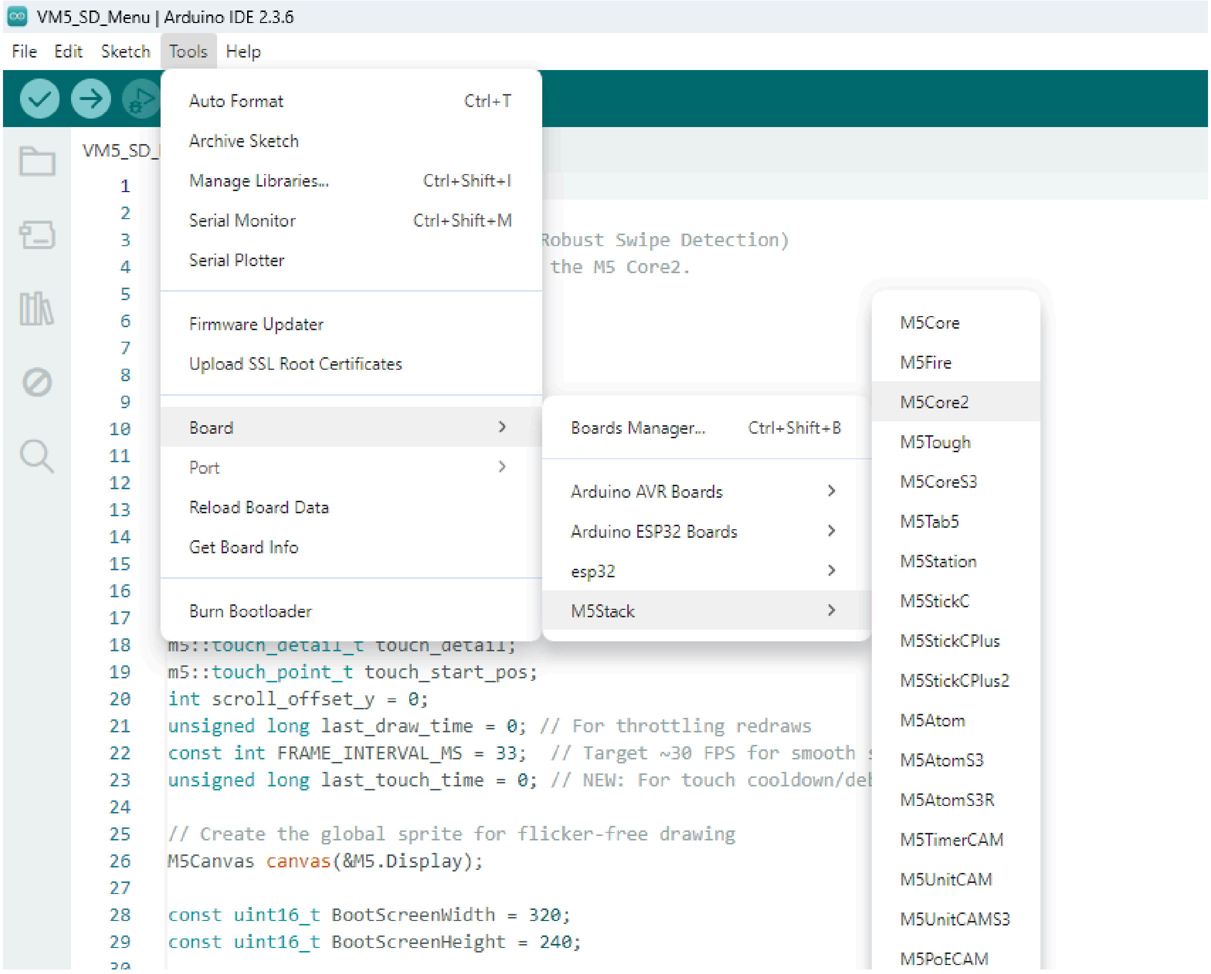
PSRAM: "Enabled"

Upload Speed: "1500000"

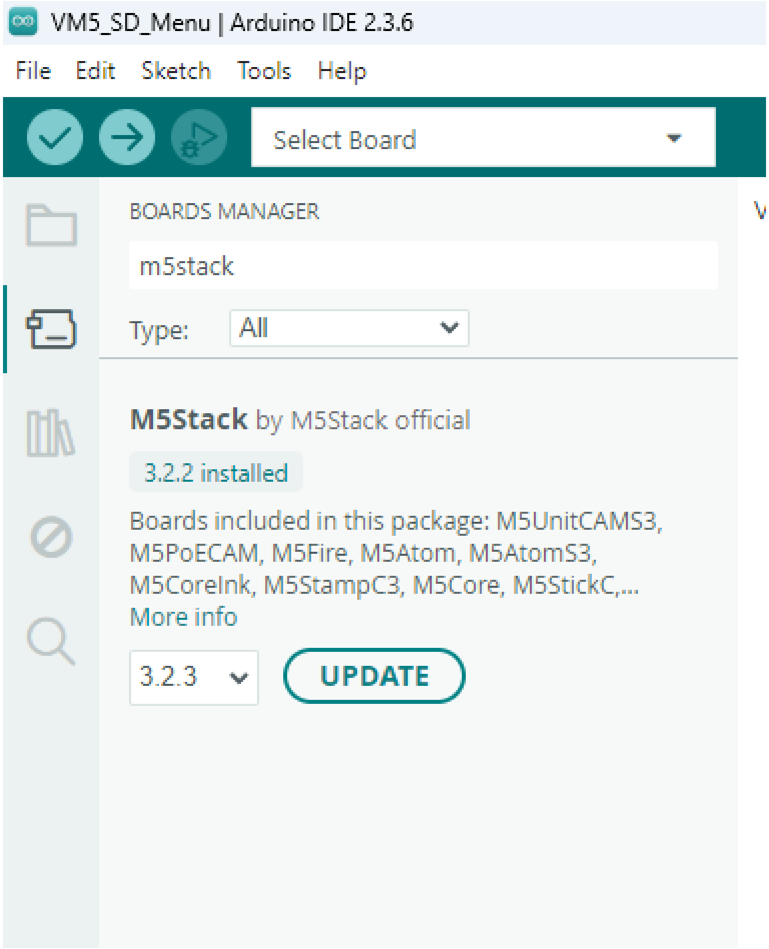
```
ard Menu for M5 Core2
 2.7 (Final Build with Robust Swipe Detection)
sion is exclusively for the M5 Core2.

5Unified.h>
5StackUpdater.h>
D.h>
ector>
lgorithm>
references.h>
pdate.h>
iFi.h>
ootScreen.h"

ure Control Globals ---
etail_t touch_detail;
oint_t touch_start_pos;
offset_y = 0;
ng last_draw_time = 0; // For throttling redraws
RAME_INTERVAL_MS = 33;  // Target ~30 FPS for smoot
ng last_touch_time = 0; // NEW: For touch cooldown/

he global sprite for flicker-free drawing
nvas(&M5.Display);

    STATE_S
    STATE_S
    STATE_S
    STATE_U
    STATE_S
};
AppState cu

struct AppI
    String
    String
```

Click on tools, set board to M5Core2 then set the Partition scheme to whats shown on the image, Make sure everything matches to what the image shows, Com port is only required if you are directly flashing the device, For most of the code you will be exporting the code to binary, You do this by selecting Sketch/Export to binary. Once the binary compiles, Goto sketch/Open sketch folder, Once the folder opens, Click and open the Build folder to locate the Binary files, The one you want is always listed at the top, Ignore the other 3 files, Rename the top file to whatever the App is called ensure that the .bin Extension isn't renamed otherwise it will corrupt the bin file, If you are exporting the Menu, Rename the file to menu.bin, and duplicate it with update.bin like we discussed above, These files can be passed on to your customers to easily update their system, Likewise with App files, Caution! Do not rename App files to menu.bin or update.bin, If you load these as updates you loose the rollback safety if the device panic boots, it will never reach the original menu system, Only update Apps by overwriting the originals on the card.

Below are some images of the ide setup and configuration to get you started.



```
Robust Swipe Detection)
the M5 Core2.

m5::touch_detail_t touch_detail;
m5::touch_point_t touch_start_pos;
int scroll_offset_y = 0;
unsigned long last_draw_time = 0; // For throttling redraws
const int FRAME_INTERVAL_MS = 33;  // Target ~30 FPS for smooth 
unsigned long last_touch_time = 0; // NEW: For touch cooldown/deb

// Create the global sprite for flicker-free drawing
M5Canvas canvas(&M5.Display);

const uint16_t BootScreenWidth = 320;
const uint16_t BootScreenHeight = 240;
```



m5stack

Type: All

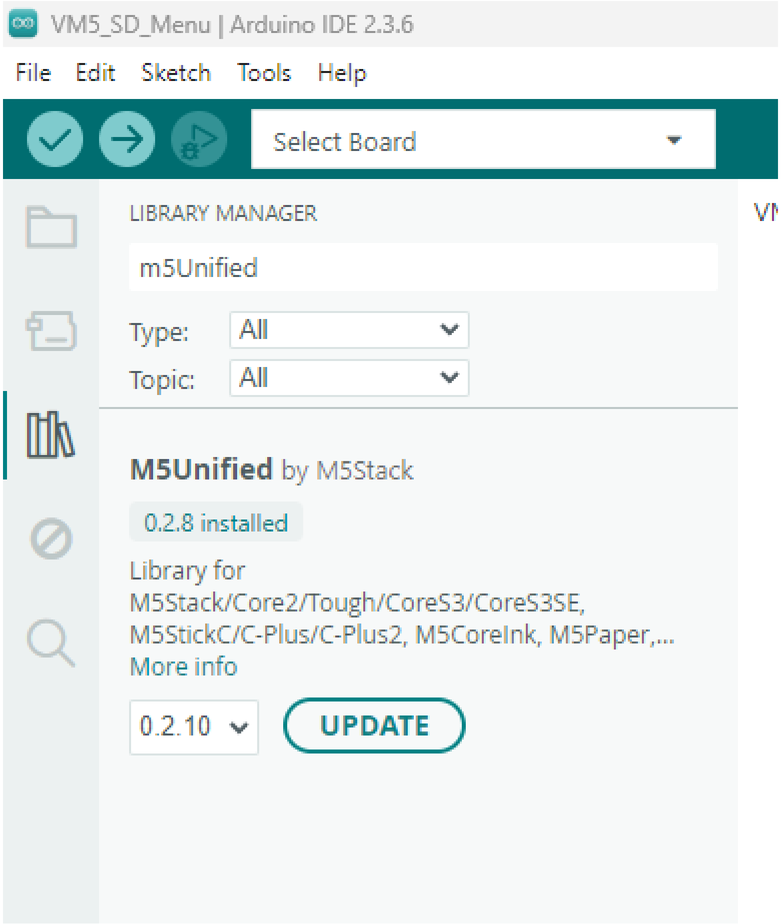
M5Stack by M5Stack official

3.2.2 installed

Boards included in this package: M5UnitCAMS3, M5PoECAM, M5Fire, M5Atom, M5AtomS3, M5CoreInk, M5StampC3, M5Core, M5StickC,...

More info

3.2.3



VM5_SD_Menu | Arduino IDE 2.3.6

m5Unified

Type: All

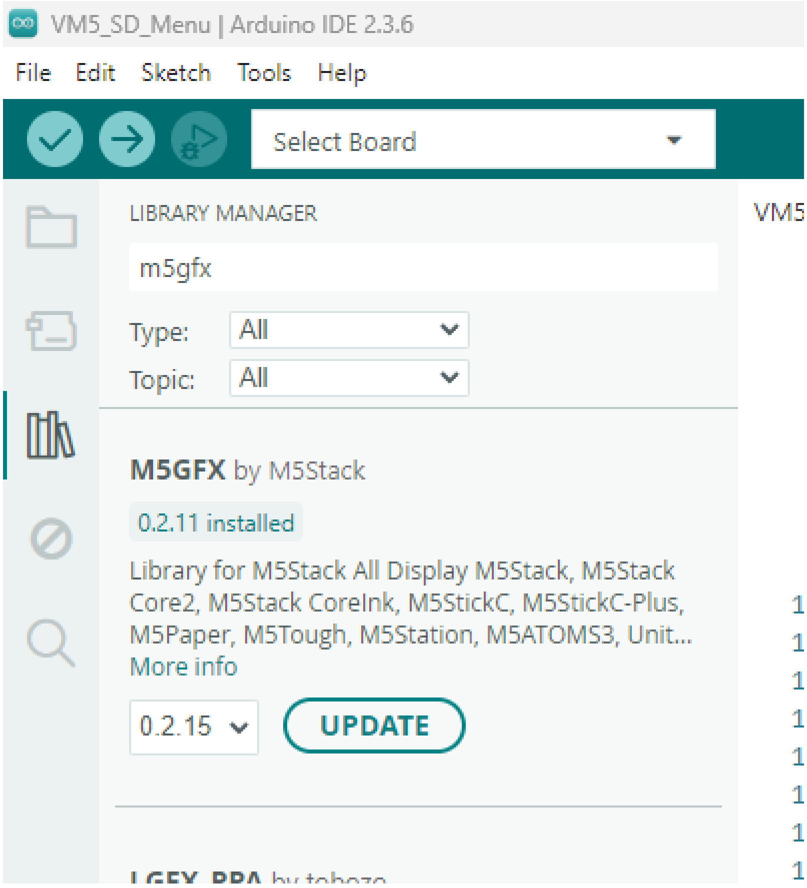
Topic: All

M5Unified by M5Stack

0.2.8 installed

Library for M5Stack/Core2/Tough/CoreS3/CoreS3SE, M5StickC/C-Plus/C-Plus2, M5CoreInk, M5Paper,...

0.2.10



VM5_SD_Menu | Arduino IDE 2.3.6

m5gfx

Type: All

Topic: All

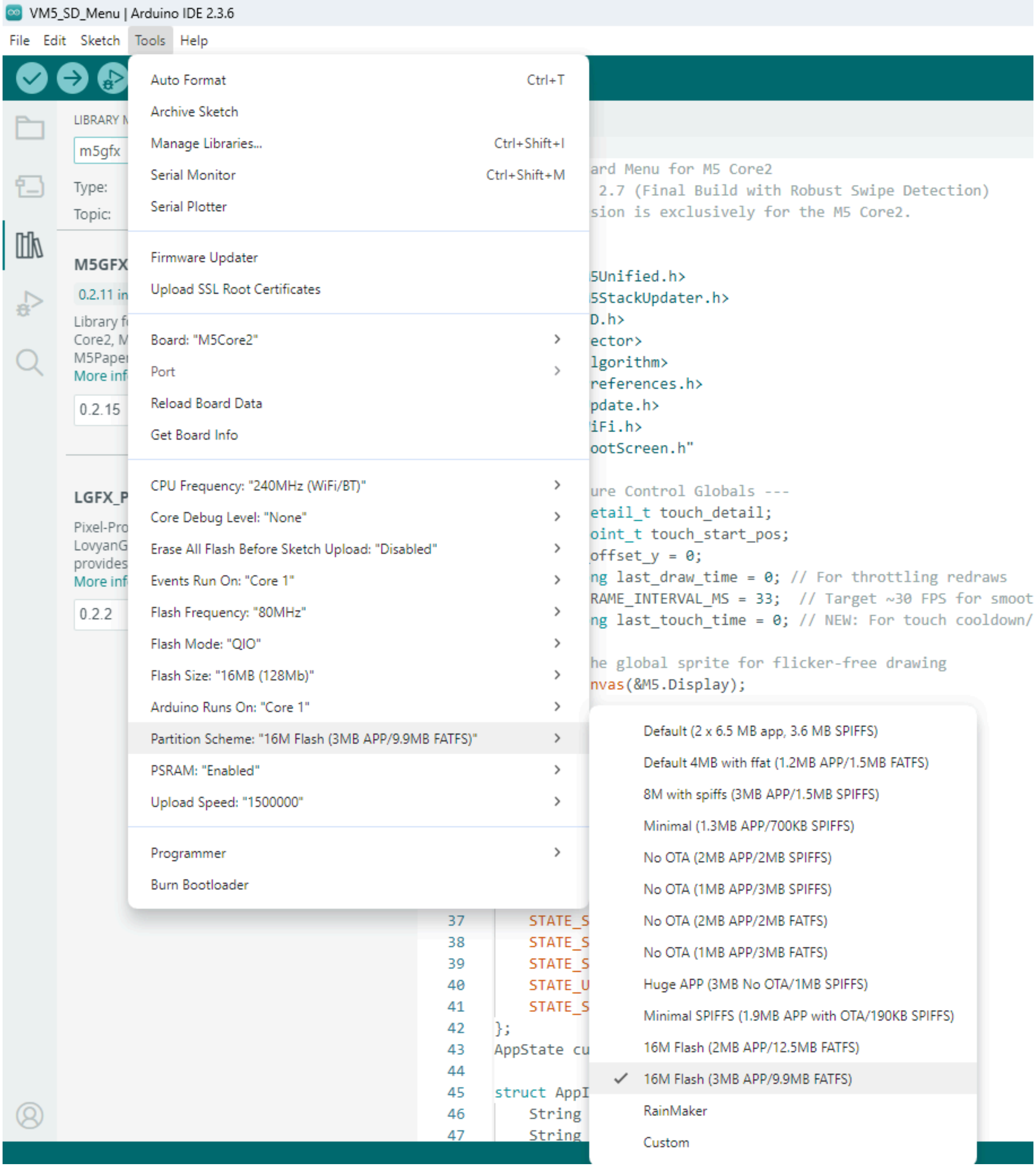
M5GFX by M5Stack

0.2.11 installed

Library for M5Stack All Display M5Stack, M5Stack Core2, M5Stack CoreInk, M5StickC, M5StickC-Plus, M5Paper, M5Tough, M5Station, M5ATOMS3, Unit...

More info

0.2.15



Board: "M5Core2"

CPU Frequency: "240MHz (WiFi/BT)"

Core Debug Level: "None"

Erase All Flash Before Sketch Upload: "Disabled"

Events Run On: "Core 1"

Flash Frequency: "80MHz"

Flash Mode: "QIO"

Flash Size: "16MB (128Mb)"

Arduino Runs On: "Core 1"

Partition Scheme: "16M Flash (3MB APP/9.9MB FATFS)"

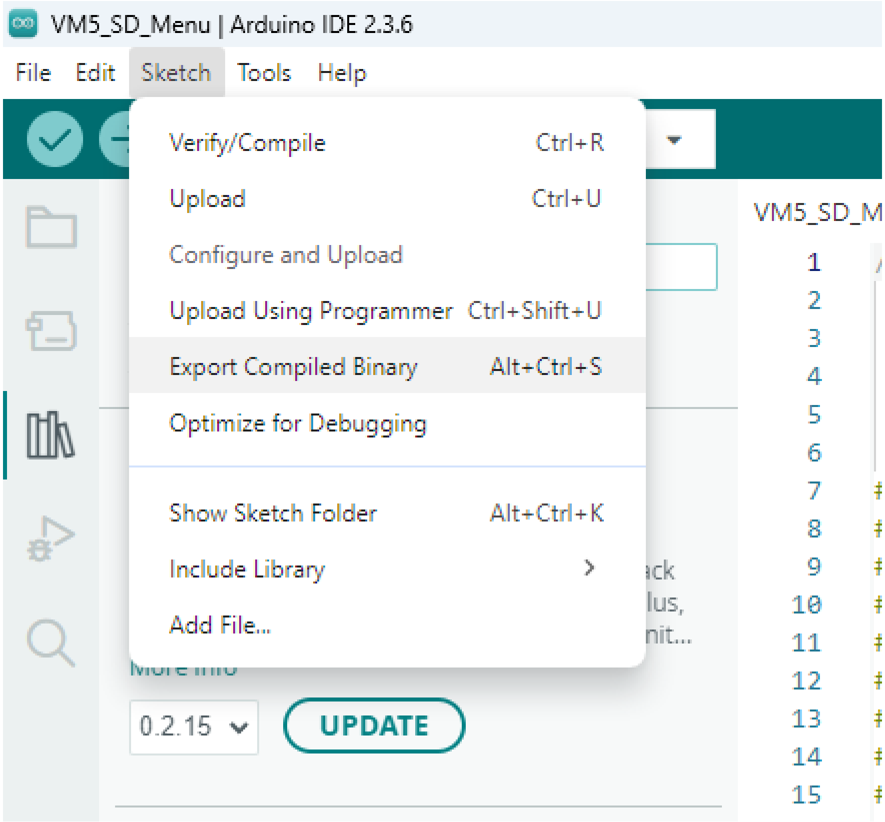
PSRAM: "Enabled"

Upload Speed: "1500000"

- Default (2 x 6.5 MB app, 3.6 MB SPIFFS)
- Default 4MB with ffat (1.2MB APP/1.5MB FATFS)
- 8M with spiffs (3MB APP/1.5MB SPIFFS)
- Minimal (1.3MB APP/700KB SPIFFS)
- No OTA (2MB APP/2MB SPIFFS)
- No OTA (1MB APP/3MB SPIFFS)
- No OTA (2MB APP/2MB FATFS)
- No OTA (1MB APP/3MB FATFS)
- Huge APP (3MB No OTA/1MB SPIFFS)
- Minimal SPIFFS (1.9MB APP with OTA/190KB SPIFFS)
- 16M Flash (2MB APP/12.5MB FATFS)
- ✓ 16M Flash (3MB APP/9.9MB FATFS)
- RainMaker
- Custom

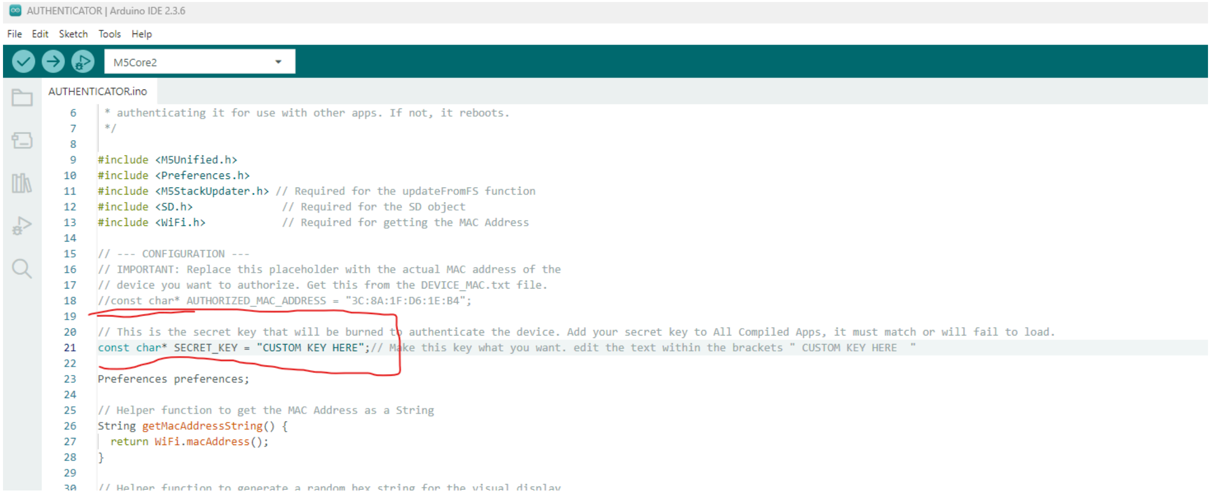
```
ard Menu for M5 Core2
 2.7 (Final Build with Robust Swipe Detection)
sion is exclusively for the M5 Core2.

5Unified.h>
5StackUpdater.h>
D.h>
ector>
lgorithm>
references.h>
pdate.h>
iFi.h>
ootScreen.h"

ure Control Globals ---
etail_t touch_detail;
oint_t touch_start_pos;
offset_y = 0;
ng last_draw_time = 0; // For throttling redraws
RAME_INTERVAL_MS = 33;  // Target ~30 FPS for smoot
ng last_touch_time = 0; // NEW: For touch cooldown/

he global sprite for flicker-free drawing
nvas(&M5.Display);
```

```
    STATE_S
    STATE_S
    STATE_S
    STATE_U
    STATE_S
};
AppState cu

struct AppI
    String
    String
```



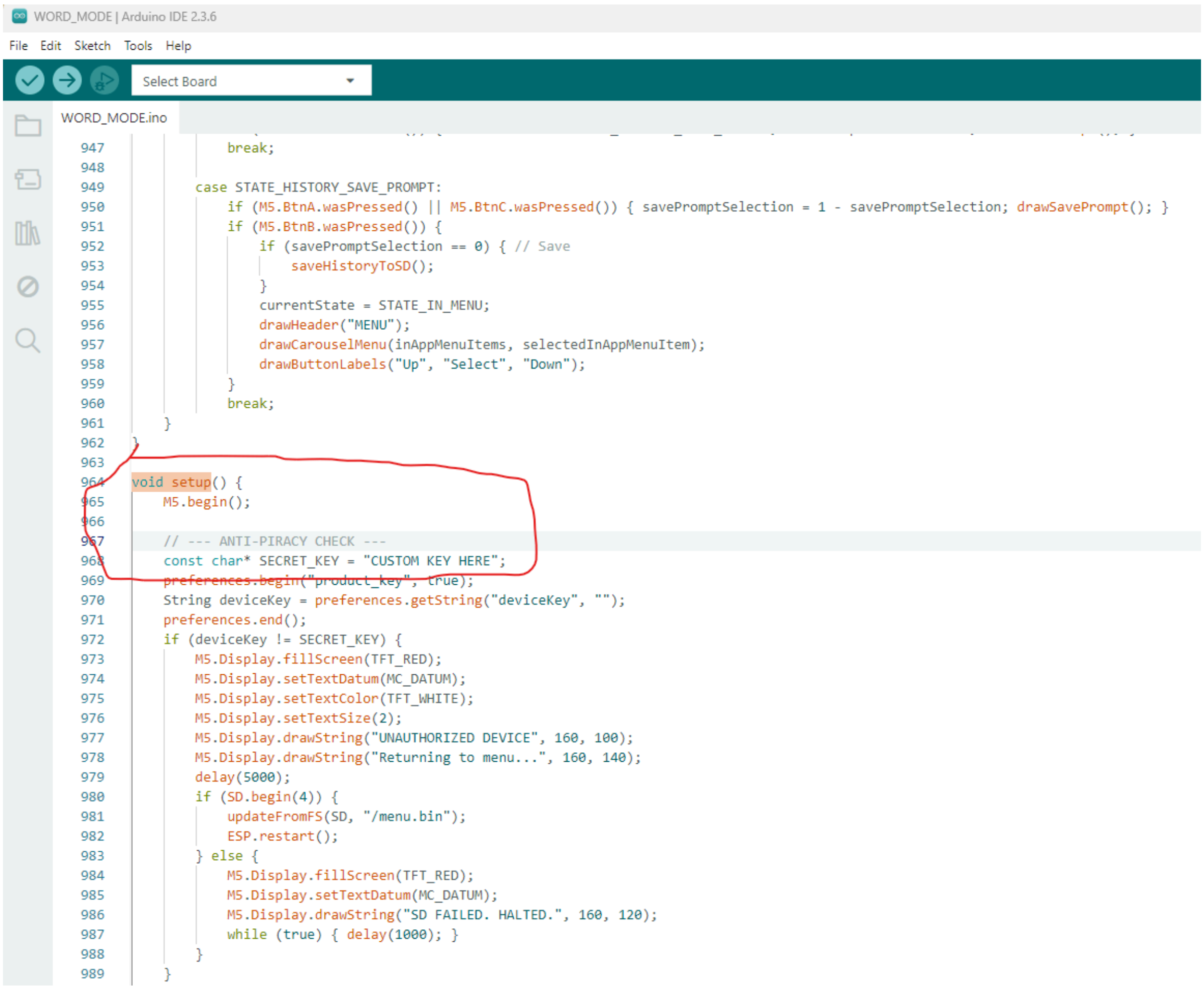
VM5_SD_Menu | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Configure and Upload
Upload Using Programmer Ctrl+Shift+U
Export Compiled Binary Alt+Ctrl+S
Optimize for Debugging
Show Sketch Folder Alt+Ctrl+K
Include Library
Add File...
More info
0.2.15
UPDATE
VM5_SD_M

Authenticator App custom key location. Edit this text within the " " Quote marks, Donot remove the Quotes, Ensure not to leave spaces example "MYPRODUCTKEY" "123456" Ensure the key matches within your App found in the Void setup block.



AUTHENTICATOR | Arduino IDE 2.3.6
File Edit Sketch Tools Help
M5Core2
AUTHENTICATOR.ino

```
 * authenticating it for use with other apps. If not, it reboots.
 */

#include <M5Unified.h>
#include <Preferences.h>
#include <M5StackUpdater.h> // Required for the updateFromFS function
#include <SD.h>             // Required for the SD object
#include <WiFi.h>           // Required for getting the MAC Address

// --- CONFIGURATION ---
// IMPORTANT: Replace this placeholder with the actual MAC address of the
// device you want to authorize. Get this from the DEVICE_MAC.txt file.
//const char* AUTHORIZED_MAC_ADDRESS = "3C:8A:1F:D6:1E:B4";

// This is the secret key that will be burned to authenticate the device. Add your secret key to All Compiled Apps, it must match or will fail to load.
const char* SECRET_KEY = "CUSTOM KEY HERE";// Make this key what you want. edit the text within the brackets " CUSTOM KEY HERE  "

Preferences preferences;

// Helper function to get the MAC Address as a String
String getMacAddressString() {
  return WiFi.macAddress();
}

// Helper function to generate a random hex string for the visual display
```



```
            break;

        case STATE_HISTORY_SAVE_PROMPT:
            if (M5.BtnA.wasPressed() || M5.BtnC.wasPressed()) { savePromptSelection = 1 - savePromptSelection; drawSavePrompt(); }
            if (M5.BtnB.wasPressed()) {
                if (savePromptSelection == 0) { // Save
                    saveHistoryToSD();
                }
                currentState = STATE_IN_MENU;
                drawHeader("MENU");
                drawCarouselMenu(inAppMenuItems, selectedInAppMenuItem);
                drawButtonLabels("Up", "Select", "Down");
            }
            break;
    }
}

void setup() {
    M5.begin();

    // --- ANTI-PIRACY CHECK ---
    const char* SECRET_KEY = "CUSTOM KEY HERE";
    preferences.begin("product_key", true);
    String deviceKey = preferences.getString("deviceKey", "");
    preferences.end();
    if (deviceKey != SECRET_KEY) {
        M5.Display.fillScreen(TFT_RED);
        M5.Display.setTextDatum(MC_DATUM);
        M5.Display.setTextColor(TFT_WHITE);
        M5.Display.setTextSize(2);
        M5.Display.drawString("UNAUTHORIZED DEVICE", 160, 100);
        M5.Display.drawString("Returning to menu...", 160, 140);
        delay(5000);
        if (SD.begin(4)) {
            updateFromFS(SD, "/menu.bin");
            ESP.restart();
        } else {
            M5.Display.fillScreen(TFT_RED);
            M5.Display.setTextDatum(MC_DATUM);
            M5.Display.drawString("SD FAILED. HALTED.", 160, 120);
            while (true) { delay(1000); }
        }
    }
```

All apps have this block inside Void Setup, Ensure this key matches your Authenticator App.

Note that some Samples form the CORE2 Libraries do not play well with the Sd card launcher system, Particularly if the code sample requires SD Card, I would highly recommend to keep a complete backup of your Working Edits made, And work on the projects granularly, which does involve alot of testing, But avoids making big changes and the App fail loosing alot of work, Even the simple app modes are quite complex, I've included statements within the code blocks to help you understand the core functions and logic, Experimenting is the best way to learn.