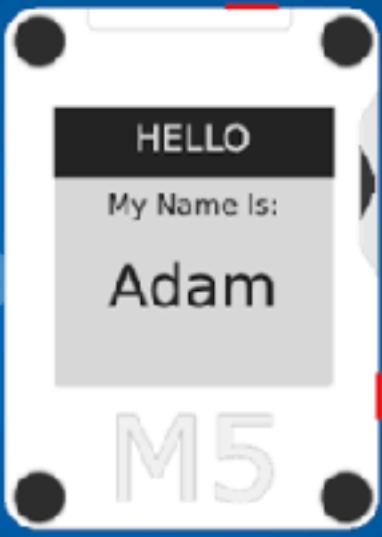
M5Stack Coreink



Adam Bryant

M5Stack CoreInk

V2

Written by Adam Bryant

Corrections and errors by Cyberminion and Pepsi [NB,NL]

Code tested by the M5Stack Community.

Arduino Code and samples provided by and also tested by the M5Stack Community.

Table of Contents

M5Stack E-Ink Devices

Written By
Adam Bryant

About this book

When I started writing this book I thought it would only reach around 10 pages long, with the support of the community it has grown way beyond what I had thought it would reach.

Another big change that has caused a delay in finishing this book is that instead of being just on the CoreInk (M5Stacks smallest E-Ink controller, it now covers the M5Papers which is M5Stack's E-reader development unit.

I would just like to thank M5Stack for creating these great devices as well as others in their range and the community for their ongoing help and support.

Thank you everyone, I hope I can continue to make helpful documents to support M5Stack products.

What is E-Ink technology?

E-ink display technology or E-paper technology is a radically different display technology to common LED and LCD displays. Unlike LED displays where separate LEDs are illuminated to produce colours, or LCD screens which use a backlight which is controlled by a liquid crystal "masking" layer E-Ink displays use capsules of Electro florescent ink that when given a charge draws them towards the front of the screen or away from the front of the screen. Unlike LED and LCD screens E-ink displays retain the display image even when powered of (if they have not been issued a clear screen command) and are only powered to change what is displayed on the screen.

Only using power for changing the information displayed on the screen makes them Ideal for low power operation like E-Book readers where they are often found and in more recent use as electronic price tags.

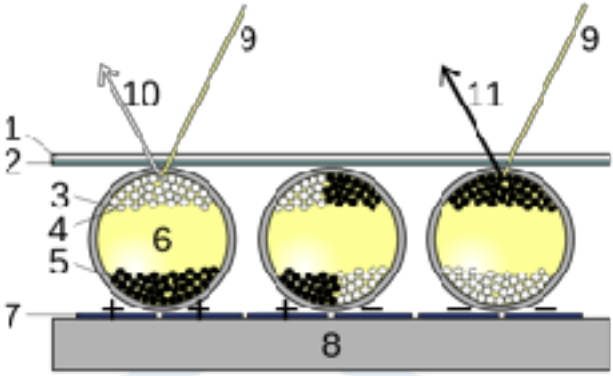


Diagram of how a 1 bit E-Ink screen works.

1, Front Glass Screen. 2, Front Electrode. 3, E-Ink Capsule.4, White E-Ink Pigment. 5, Black E-Ink Pigment. 6, Suspension Fluid. 7, Rear Pixel Electrode. 8, Back Panel. 9, External Light Source. 10, Reflected White Light. 11, Reflected Gray Light.

M5Stack use two different E ink displays for their E-Ink devices. The CoreInk uses a 1 bit display controlled by a GDEW0154M09 controller meaning that the screen points can only be black or white whereas the M5Paper employs a ED047TC1 EPD screen with an IT8951 screen controller allowing up to sixteen levels of gray scale.

While M5Stack does not produce a three colour or multi colour screen (as of the time of writing) some members have had success connecting and controlling the tricolour screen from Seeedstudio https://www.seeedstudio.com/Grove-Triple-Color-E-Ink-Display-1-54-p-2890.html?fbclid=IwAR2020qy9Zk4uhx4Zy7Z5UJ0kh8U8I0FglJ7DgAv-UcCBOxvQ-G8agZk02Y

Which connects to Port C and uses UART to communicate however, this is outside the scope of this document for the present time.

The M5Paper also has a capacitive five point multi touch screen powered by a GT911 controller.

CoreInk vs M5Paper

As of writing, M5Stack make two development devices that are equip with E-Ink screens. The biggest is M5Paper which is a small e-reader style device which has a 4.7" screen and the smallest is the CoreInk with a 1.54" screen. The CoreInk is powered by the ESP32 Pico D4 found in several of their smaller programable controllers like the CoreInk and the Atom While the M5Papers is built around the ESP32 DOWDQ6-V3 Found in the Core2. The CoreInks small low power design makes it useful for devices that don't need to change the information displayed on the screen very often. This makes the CoreInk perfect for use as a reprogrammable name tag, WIFI updatable price tag on shop shelves. or a thermometer that only needs to take readings after a set time.

CoreInk Specifications

Below is a list of the main important functions of the CoreInk.

ESP32 PICO D4	240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth
FLASH	4MB
INPUT VOLTAGE	5V@500mA
PORTS	USBC, HY2.0 4P (GROVE PORT A), 16P M-BUS, 8 PIN H.A.T
SCREEN	200X200 PX SPI E-INK **GPIO4, GPIO0, GPIO15, GPIO9, GPIO18, GPIO23**
PROGRAMMABLE BUTTONS	1X 3 Way Multifunction **GPIO37, GPIO38, GPIO39**. 1X NO **GPIO5**
LED	Green LED **GPIO10**
RTC	BM8563 **GPIO21, GPIO22, GPIO19**
BUZZER	Passive **GPIO2**
WIFI	**2.4GHz**
Bluetooth	Bluetooth V4.2 BR/EDR and Bluetooth LE
Battery	**390mAh @ 3.7V**
Size	**56x40x16mm**

M5Papers Specifications

Below is a list of the main important functions of theM5Papers.

ESP32 D0WDQ6-V3	240MHz dual core, 600 DMIPS, 520KB SRAM, Wi-Fi, dual mode Bluetooth
FLASH	16MB
PSRAM	8MB
INPUT VOLTAGE	5V@500mA
PORTS	USBC, HY2.0 4P X3 (PORT A, PORT B, PORT C)
SCREEN	EPD-ED047TC1 4.7" 540X960 E-INK Controlled by **IT8951 GPI14, GPIO13, GPIO15, GPIO13**
PROGRAMMABLE BUTTONS	1X 3 Way Multifunction **GPIO37, GPIO38, GPIO39**. 1X NO **GPIO5**
RTC	BM8563 **GPIO21, GPIO22, GPIO19**
SDCARD	**GPIO4, GPIO13, GPIO12, GPIO14**
WIFI	**2.4GHz**
Bluetooth	Bluetooth V4.2 BR/EDR and Bluetooth LE
Battery	**1150mAh @ 3.7V**
Size	**118x66x10mm**

Hardware Functions

The E-Ink Screen

The CoreInk uses a Good Display GDEW0154M09 200x200 pixel 1 bit black and white display connected directly to the ESP32 Pico D4 using SPI on the following pins

E-Ink PINS	ESP32 PICO D4 Pins
9 (Busy)	GPIO4
10 (Reset)	GPIO0
11 (Data/Command)	GPIO15
12 (Chip Select)	GPIO9
13 (Serial Clock)	GPIO18
14 (Serial Data)	GPIO23

Where as the M5Papers has an ED047TC1 EPD controlled by an IT8951 controller connected to the ESP32 D0WDQ6 using SPI on the following pins. It is worth noting that some of these pins are shared with the SDCard reader.

IT8951 PINS	ESP32 D0WDQ6
124 (MISO)	GPIO13
123 (MOSI)	GPIO12
122 (Serial Clock)	GPIO14
119 (Chip Select)	GPIO15

I/O Ports

On the top, bottom and rear of the CoreInk can be found three I/O ports. Shared between the three ports, we have ten I/O pins that we can use for connecting to external hardware.

The port on the top is the H.A.T connector and is compatible with the M5Stick H.A.T connector.

Pin	ESP32 PICO D4 Pins.
1	GND
2	5V OUT
3	GPIO26
4	GPIO36
5	GPIO25
6	BAT
7	3V3
8	5V IN

On the bottom of the CoreInk is found the Grove Port

PORT A	
Pin	**ESP32 PICO D4 Pins.**
1	GND
2	5V OUT *
3	GPIO32 (Pin 12)
4	GPIO33 (Pin 13)

* If more power is needed for units connected to this port. Connecting +5V via the Unit T will not cause damage.

In UIFlow, by default this is set to Port A as there is only one port however, if we select the custom option for a port, we can connect a unit to any two available pins on the CoreInk by telling UIFLow which pins to use.In order to connect a unit to different pins we will need to use a Dupont to Grove adapter.

On the rear of the CoreInk we find the sixteen pin M-Bus connector.

GPIO25	GPIO23 (**MOSI**)
GPIO26	GPIO34 (**MISO**)
GPIO36 (**ADC**)	GPIO18 (**SCK**)
GPIO22 (**SCL**)	GPIO21 (**SDA**)
GPIO14 (**TXD2**)	GPIO13 (**RXD2**)
3V3	RST
5V OUT	5V IN
BAT	GND

While this is physically the same as the CoreInk and CoreInk Plus H.A.T connector, placement of part on the various H.A.Ts prevent some of them from fitting to the CoreInk without an adapter.

The M5Papers doesn't have the MBus or the HAT connector but instead has three I/O ports, I2C/Digital I/O, Analogue and, UART. These three ports are connected to the following pins of the ESP32.

PORT A - I2C/Digital	
Pin	**ESP32 PICO D4 Pins.**
1	GND
2	5V OUT *
3	GPIO25
4	GPIO32

PORT B - Analoge	
Pin	**ESP32 PICO D4 Pins.**
1	GND
2	5V OUT *
3	GPIO26
4	GPIO33

PORT C - UART	
Pin	**ESP32 PICO D4 Pins.**
1	GND
2	5V OUT *
3	GPIO18
4	GPIO19

SDCard Reader

The M5Papers has an SDHC Micro slot for using an SDMicro cards as storage instead of needing to save a file to the memory of the ESP32. As of writing, it is not know what the technical limits of the card reader are.

The SD Card connector is connected to the following pins of the ESP32.

ESP32 Pins	SD Card Conector
GPIO4	Chip Select
GPIO13	MISO
GPIO12	MOSI
GPIO14	Serial Clock

Getting Started

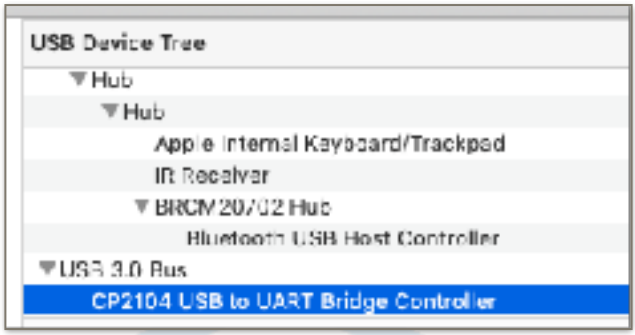
Before we can start programming the CoreInk and the M5Papers we will need to install some software on a host computer. Thankfully, due to the hard work of the programmers at M5Stack, we can use computers with OSX, Windows or linux based operating systems.

The Virtual Serial Driver.

The first piece of software that need to be installed is the virtual serial driver. This piece of software is need to allow the CoreInks CP2104 communication chip to talk to the host computer.

Goto https://docs.m5stack.com/#/en/arduino/arduino_development, scroll down and click on the button to download the driver for the OS your computer is using Follow the onscreen prompts to install the driver.

If installation was successful on an OSX based computer, you will see the following under the USB section of system report.



What to do if the driver installs but the computer does not see the CoreInk?

Sometimes Windows and OSX will cause problems with the driver resulting in the driver failing to work. This is a known issue with both operating sources and the walk around to fix it can be very time consuming. Below I will show you the steps I use to get the driver to work.

Cold booted computer.

To cold boot a computer, simply turn the computer completely off (standby and logging out DO NOT work.) and leave for ten minutes. This step is needed to allow the capacitors inside the computers ram to discharge, clearing any data out of the ram.

Step 1 Start the computer.

Start up the computer and log in but DO NOT run any programs. once the computer has loaded up and been left to settle down for a few minutes plug the core ink into a USB port, open the computers device manager and look for and devices that show problems. Right click on the device to open it and click on Remove Driver. Once the driver has been removed, shut down the computer and wait for ten minutes in order to cold boot the computer. This step is required because, while we have removed the faulty driver, some parts of it remain working in the memory that can only be removed from the computer being powered off.

Step 2 Install the new driver.

Start the computer from the cold booted stage and again DO NOT run any programs. Run the CP210x driver installer but do not connect any M5Stack controllers to the USB. Follow the onscreen prompts to install the drivers and when finished, shut the computer down again for another cold boot.

Step 3 Connect the M5Stack Controller.

Start the computer again, wait for it to finish loading up and connect an M5Stack controller. This should have worked this time and you should be able to see a usb to UART device in the various programming environments.

M5Burner

To program the CoreInk in UIFlow or Micropython we need to install some firmware. This firmware is found in the M5Burner program. Please note that some versions are called M5Electron Burner.

Step 1 - Download M5Burner

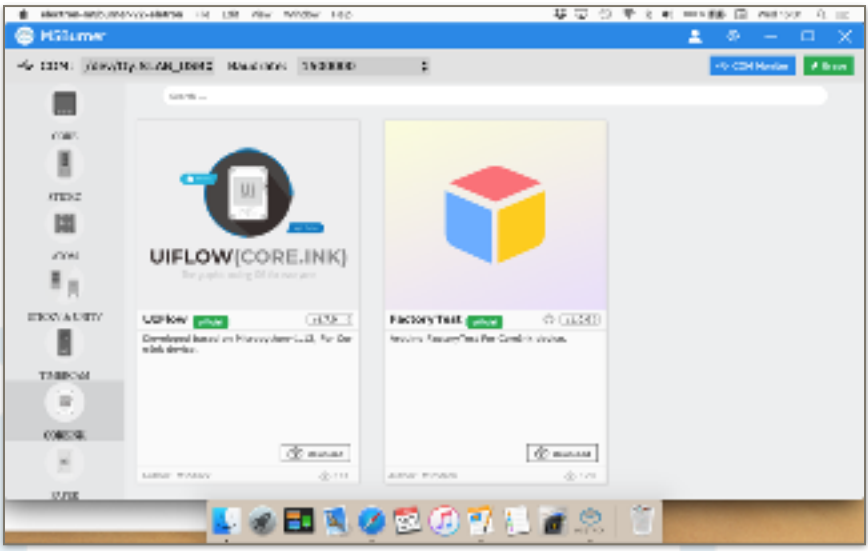
Goto https://m5stack.com/pages/download , move the mouse cursor over to the download button and click on the operating system you are using to download the appropriate M5Burner version.

Step 2 - Install M5Burner

Copy M5Burner to a suitable location (in OSX, copy it to the Launchpad/Applications folder.

Step 3 - Launch M5Burner

Open M5Burner



If /dev/tty.SLAB_USBtoUART is not shown next to Com then click on the two little black arrows to open the dropdown box and select the /dev/tty.SLAB_USBtoUART entry. It is worth noting that in Windows this will just appear as Com with a number after it.

Step 4 - Select Device firmware group

On the right hand side you will see a list of M5Stack programmable controllers. At the bottom is the CoreInk, click on the image of the CoreInk and the main window will change to show the available firmwares for the CoreInk. As of the time of writing this, there is only the UIFlow firmware and Factory test demo available.

Step 5 - Erase Existing Firmware

Before we can write any firmware to the CoreInk, we first have to erase the current loaded/installed firmware already on it. In the top right hand screen (you may need to resize to window to see it) is a big green button with "Erase" written on it. If you still have the CoreInk plugged in and visible in the com menu, press the Erase button and the following screen will appear.

Once you see the "Erase Successful" message you can click on the close button to return to the main screen.

Interesting point of note.

During the erase process, M5Stack devices do not always show a blank screen and sometimes they will show the last screen that was created in memory.

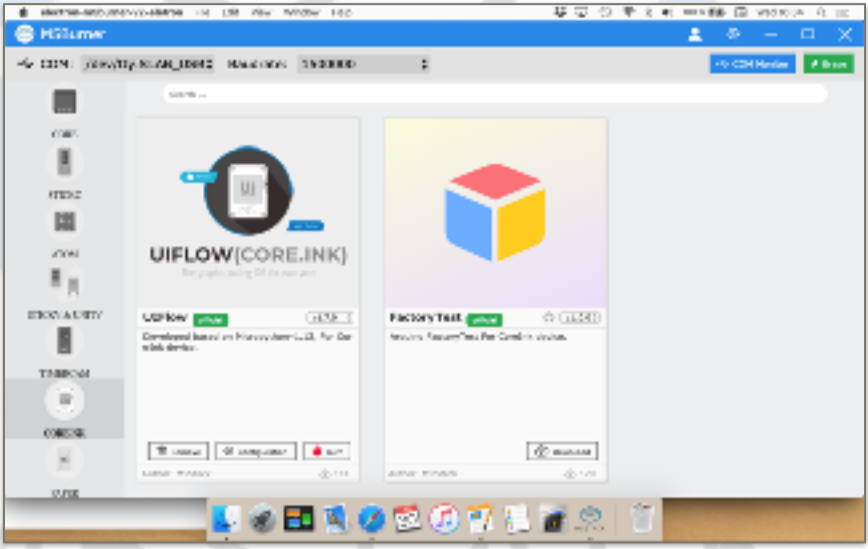
Step 6 - Download Firmware

Next we need to download the latest firmware for the CoreInk. On the righthand side of the UIFlow firmware is a

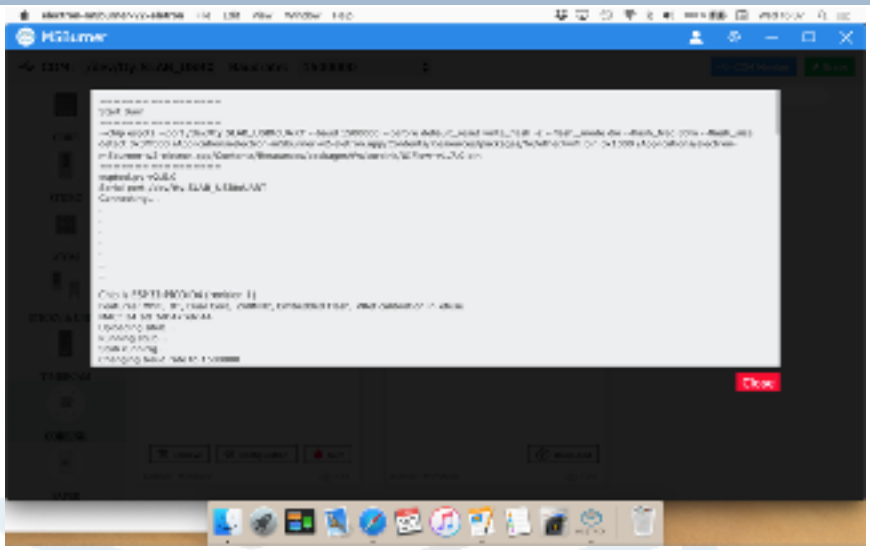
dropdown menu that by default should show the latest version available, Click on the box and select the latest version. Once the version has been selected click on the "Download" button at the bottom of the panel to download the firmware

Step 7 - Burn Firmware

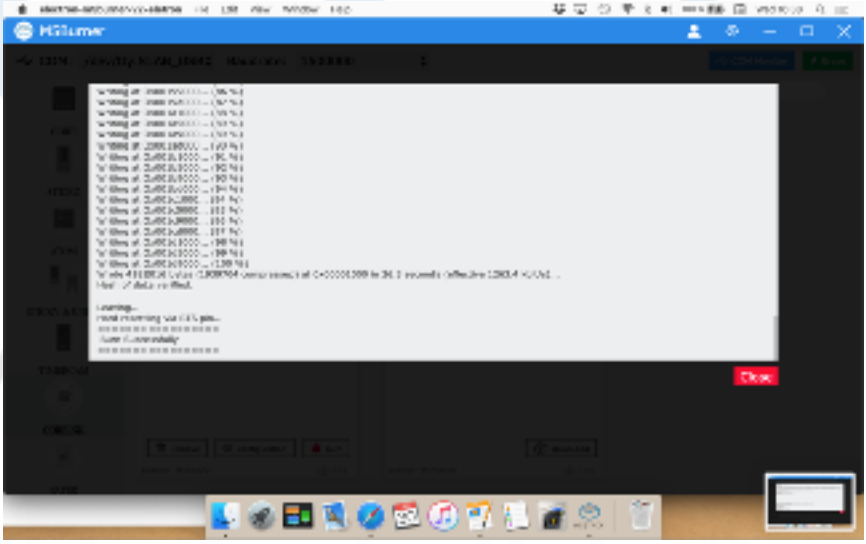
Once the firmware has finished downloading, the bottom of the panel will change to show some new buttons.



Click on the "Burn" button to start the firmware installation process and the window will change to show this:



The burn process will start and if successful will show the following screenshot.



If you don't see the "Burn Successful" message at the bottom of the screen try scrolling the box to make it visible. If the firmware install was successful, the CoreInk will beeb to show its reset and attempt to connect to the local WIFI.

Connecting the CoreInk to WIFI

If for some reason WIFI setup through M5Burner failed to work you can manually set the WIFI SSID and password to connect after the firm ware install.

To do this first start up the CoreInk and immediately pull the multifunction button down, this will enter the setup menu.

Press the multifunction switch down so that an arrow appears to the right of "WIFI Config" and press the multifunction button in to select the menu option and a screen like the following will be shown.

On the screen will we show the SSID of the temporary access point you will need to connect to and an IP address of 192.168.4.1. of the page on the CoreInk for changing the WIFI details.

Connect to the SSID using your computers WIFI manager.

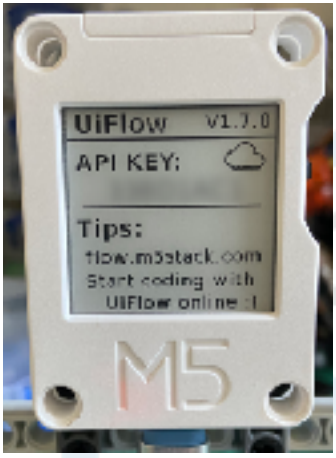
Then using an internet web page browser connect to the web address http://192.168.4.1 which is a page hosted on all M5Stack controllers.

Select the WIFI network that you wish to connect to, type in the password and click on configure to connect.If you typed the password correct you will get the success message and the screen

will change on the CoreInk to show an API number which should be unique to each device.

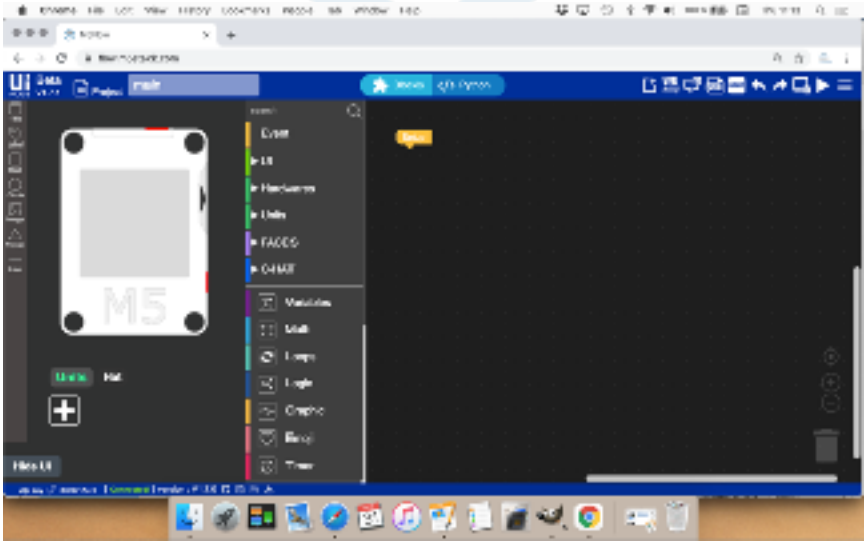
Connecting the CoreInk to UIFLow

This step is not needed when using Arduino and platform.io.



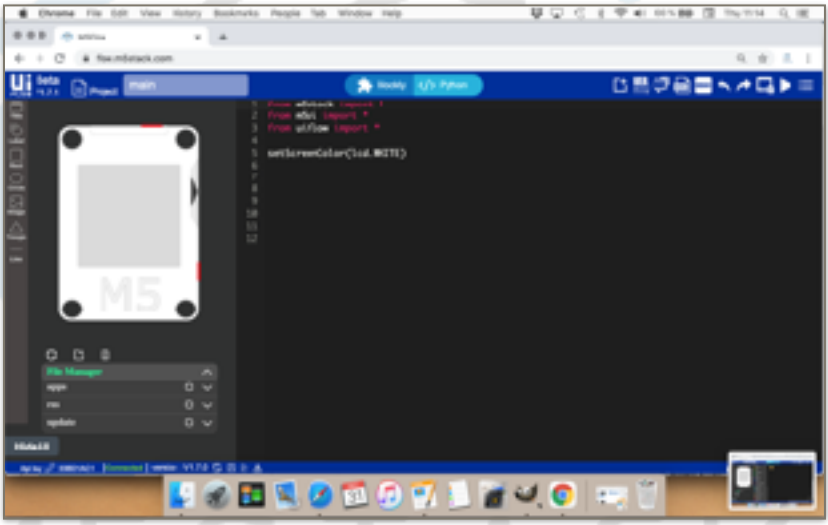
Now that we have our M5Stack CoreInk connected to the internet we need to connect the device to the online UIFlow Server. To link our core to the server, we need the API number that is now shown on the screen after each restart I have blurred my key here but normally the key is changed every time the firmware is installed or reinstalled..

Open a new page in the internet browser and goto https://flow.m5stack.com/



This is UIFlow's main screen. Across the top is the title bar which shows the current revision of the UIFlow web server. Next to that is a box that we can use to set our project name. When we want to save the program to a computer the file name will use the project name we specify here.

In the middle of the title bar there is the two IDE buttons. Blocky is the default IDE shown above, Python takes us to the Code view where we can directly edit and code in Micropython.



On the right of the title bar we have a set of Icons. These icons and their functions are as follows.

• **Registered User** - There are functions that require users to be registered in order for some functions to work. Clicking on this icon allows you to log in using the forum credentials used on the M5Forums found at https://community.m5stack.com We logged in, this icon will change to show the image a user has set up as an avatar on the forums.

• **New File** - This will clear the current open program ready to start a new program.

• **VER** - This will allow you to change between the latest stable version and the latest test version.

• **Forum** - Takes you to the M5Stack Forum.

• **DOC** - Takes you to the UIFlow online Documentation.

• **Demo** - Opens a dropdown menu containing available preinstalled demos.

• **Undo** - Undo a code change.

• **Redo** - redo a code change.

• **Manager** - Allows you to add or remove libraries and files to the onboard storage.

• **Run** - This downloads the program to the the ram to test the program when you restart the program is lost as it is not premaritally saved.

• **Main Menu** - Contains UIFlow settings and options to save the code to a computer or to a device perinatally.

The Settings is where we go to configure UIFlow to connect to and M5Stack, stick or core.

The API Key box is where you type in the number shown of the screen of the M5Paper or CoreInk.

Language is to set which language you need UIFlow to be in if English is not your language.

Next you click on the icon that represents the device you are connecting to UIFlow. The icons are as follows:

Basic Core,

Fire Core,

Stick C,

Stick C Plus,

Atom Lite,

Atom Matrix,

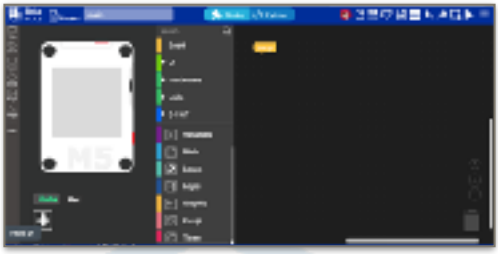
Core 2,

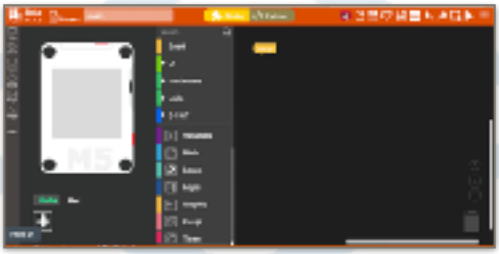
Core 2 AWS,

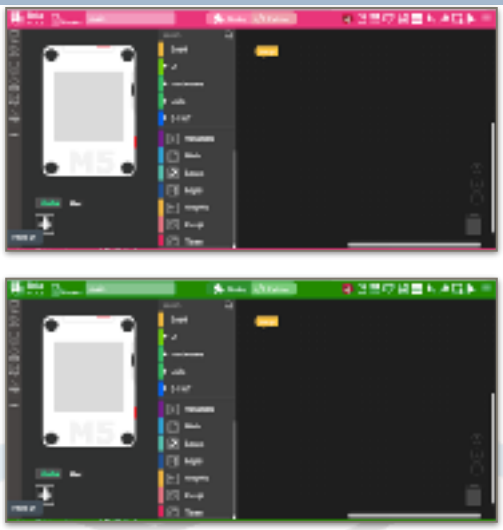
CoreInk,

M5Paper,

Below these icon are the IDE colour themes. this changes the blue bars in the IDE to the selected colour theme.







Under the colour themes is the M5Burner download links. Currently there are only links to the Windows and OSX versions.

Next we have the Official manual. This is downloadable in Chinese, Japanese, Russian and English.

And last is a series of videos for installing the firmware and using UIFLow.

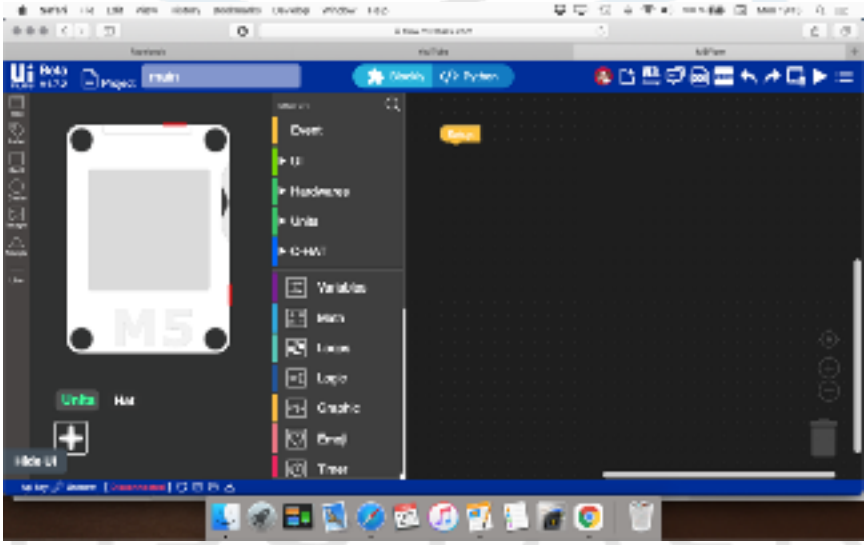
Once these are set you can then click OK to connect UIFlow to your device.

Sometimes UIFlow or the device will not connect to the UIFLow server, this is often down to a bad internet connection or the server being overloaded by users trying to connect online.

Programming in UIFlow and Micropython

When we return to the main screen after setting the API and selecting the CoreInk or M5Paper, the CoreInk or M5Paper will be shown on the left of the screen.

In the middle of the screen is found the list of available functions and on the right is the programming space with the default "Setup" block already placed.



If we switch to a Micropython IDE, separate to UIFlow we need to place the following line of code at the top of our file:

```
from m5stack import *
from m5ui import *
from uiflow import *
```

These three lines tell the Micropython interpreter inside the CoreInk what files need to be loaded in order to find the required functions however, in the UIFlow version of the Micropython IDE, the lines are already added for us.

UIFlow Blocks

Event Blocks

Setup

The Setup block is required by all programs as it works in the background to import the library required for our programs to run. This block is the equivalent of Arduino's Void Setup function and contains functions that you only want to run once at the beginning of the program.

In the Blocky windows we can't see much but if we switch to the </> Python window we see the following Micropython code.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x222222)
```

We can see in this code snippet, that this block imports the three core libraries required by all programs and sets the initial screen colour to an almost black colour. By changing the value highlighted in red, we can change the initial screen colour. For more information on setting colours, check out the colour blocks.Graphical and GUI Function.

Loop

After this we have the main program loop. Inside of this block we place the main program that we want to continuously repeat.

The Micropython code for this block is

```
while True:
```

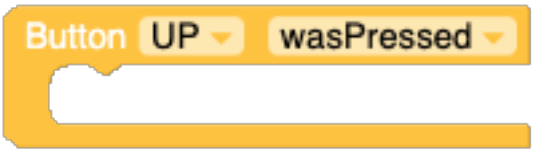
Example

In the following example we will use a loop to continuously step through the Set screen Background Colour block. If this is run without the wait blocks then the screen will look purple because the screen is changing faster then the human eye can see. To be able to see the colour change we have to add a wait block to make the program wait one second before moving between the blocks.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x222222)

while True:
  setScreenColour(0xff0000)
  wait(1)
  setScreenColour(0x3366ff)
  wait(1)
  wait_ms(2)
```

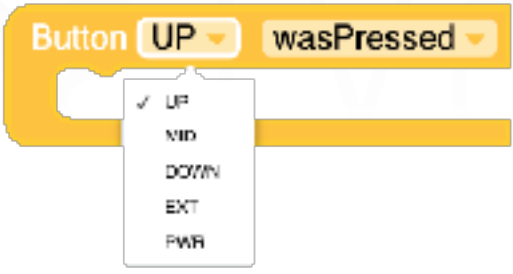
For more information on the wait block, check out the timer section of the book.

Button



As mentioned in the hardware section, the CoreInk has a three way button, a power button and a user programmable push button whereas the M5Paper only has the three way button and a power button. To control the buttons in UIFlow we can use the button loop block or the button value blocks.

To assign a function to a button on the CoreInk or M5Paper, click the arrow next to "UP" and the following buttons will be available



The Buttons are available to these blocks are

UP, MID and DOWN for the multi function.

EXT for the button on the top of the CoreInk.

PWR for the Power button on the side of the CoreInk and rear of the M5Paper.

Each of the buttons can have four events which can be accessed by clicking the arrow next to "wasPressed" and are as follows.

Was Pressed - This is the normal mode of operation and waits until a single short press of the button is detected before running the code inside it.

Was Released - This waits until the button is pressed and released before running the code placed inside it.

Long Press - Like the "Was Pressed" block, it waits until the button is pressed however, this block on activates if the button is pressed and held down for a few seconds.

Was Double Pressed - This function checks to see if the put-on was pressed and released twice within a few milliseconds before running the code inside it.

The Micropython code for the Button Loop is as follows.

buttonUP_wasPressed():
buttonMID_wasPressed():
buttonDOWN_wasPressed():
buttonEXT_wasPressed():
buttonPWR_wasPressed():

For the buttons and

buttonUP_wasPressed():
buttonUP_wasReleased():
buttonUP_pressFor():
buttonUP_wasDoublePress():

It is worth noting that the long press command in Micropython is pressFor.

Timers (Part 1)

Timer Callback Loop

The timer callback look creates a timer that can contain actions that are set to run under certain time critical events. Click on the light box with **"timer1"** in it to change the name of the timer. The timer callback loop must always be used as it creates the timer functions used by the following timer blocks.

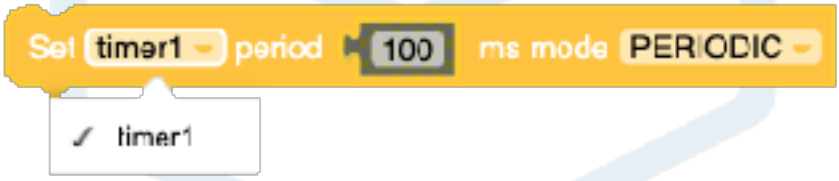
The Micropython code for the loop is as follows:@timerSch.event('timer1')

```
@timerSch.event('timer1')
def ttimer1():
  # global params
  pass
```

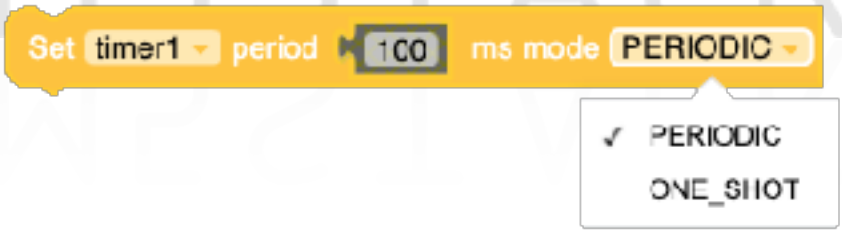
Set timer Period

The set timer period timer block is used to chose which of the defined timers are to be used in periodic or One Shot mode along with their time length in milliseconds.

If you click on the down arrow next to timer1 a dropdown list will appear showing a list of defined timers.



If you click on the arrow next to Periodic then another dropdown list appears with the options "Periodic" and "One_Shot".



The Micropython code for set timer Period block in periodic mode is

```
timerSch.setTimer('timer1', 100, 0x00)
```

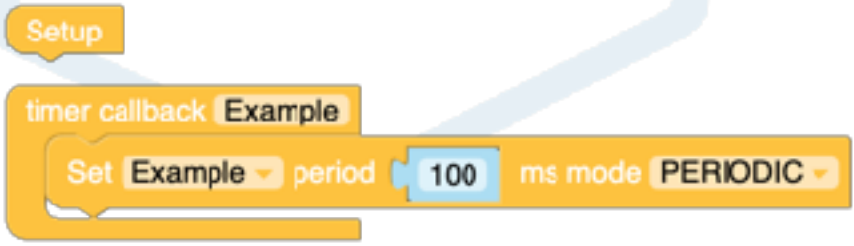
and for "One Shot" mode,

```
timerSch.setTimer('timer1', 100, 0x01)
```

It is worth noting that the only difference in the code is the 0x00 used to set "Periodic" mode and 0x01 to set "One Shot" mode in Micropython.

Examples

The following examples show how to set the different modes.



The Micropython code for this example is as follows.

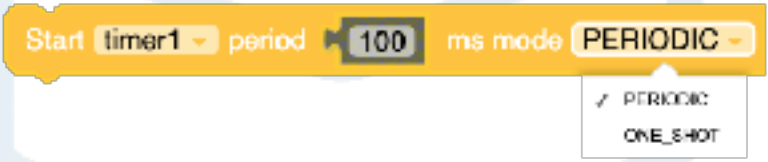
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

@timerSch.event('Example')
def tExample():
  # global params
  timerSch.setTimer('Example', 100, 0x00)
  pass
```

The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

@timerSch.event('Example')
def tExample():
  # global params
  timerSch.setTimer('Example', 100, 0x01)
  pass
```

Start Timer

Starts a timer chosen from a dropdown list of timers found by clicking the dropdown arrow. The time value can be set with a maths block or a variable block. The operating mode can be changed by clicking the down arrow on the right of the block.



The Micropython code for the stop block is:

```
timerSch.run('timer1, 100, 0x00)
```

For "Periodic" mode and

```
timerSch.run('timer1', 100, 0x01)
```

For "One Shot" Mode.

It is worth noting that the only difference in the code is the 0x00 used to set "Periodic" mode and 0x01 to set "One Shot" mode in Micropython.

Example.

Stop Timer

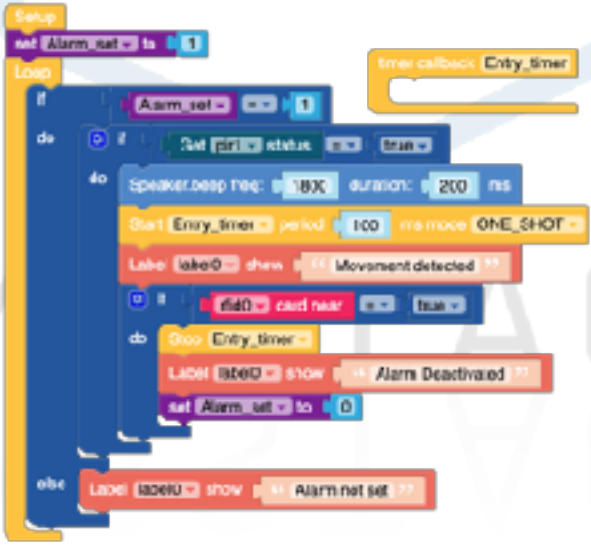
Stops the timer selected in the dropdown box.

The Micropython code for the stop block is:

```
timerSch.stop('timer1')
```

Example.

The following example (as mentioned earlier) has been created to show how the various time blocks react with each other but also rely on each other to work.



The Micropython code for this is quite long and is more advanced then previous demos

```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(0x222222)
pir1 = unit.get(unit.PIR, unit.PORTB)
rfid0 = unit.get(unit.RFID, unit.PORTA)

Alarm_set = None

label0 = M5TextBox(22, 101, "Text",
lcd.FONT_Default, 0xFFFFFF, rotate=0)

@timerSch.event('Entry_timer')
def tEntry_timer():
  global Alarm_set
  pass

Alarm_set = 1
while True:
  if Alarm_set == 1:
    if (pir1.state) == True:
      speaker.tone(1800, 200)
```

```
            timerSch.run('Entry_timer', 100,
0x01)
            label0.setText('Movement detected')
            if (rfid0.isCardOn()) == True:
              timerSch.stop('Entry_timer')
              label0.setText('Alarm
Deactivated')
              Alarm_set = 0
        else:
          label0.setText('Alarm not set')
        wait_ms(2)
```

Timers Part 2
Wait Timer

The timer blocks shown so far are not the only timer blocks available in UIFlow. Below are timer blocks that are used to cause delays in sections of programs.

Wait Seconds

This wait block allows us to make the program wait or pause for a time defined in seconds using a variable or a maths block.

The Micropython code for this block is

```
wait(1)
```

Wait Milliseconds

The Wait Milliseconds block on the other hand allows us to define a wait or pause in milliseconds.

The Micropython code for this block is

```
wait_ms(1)
```

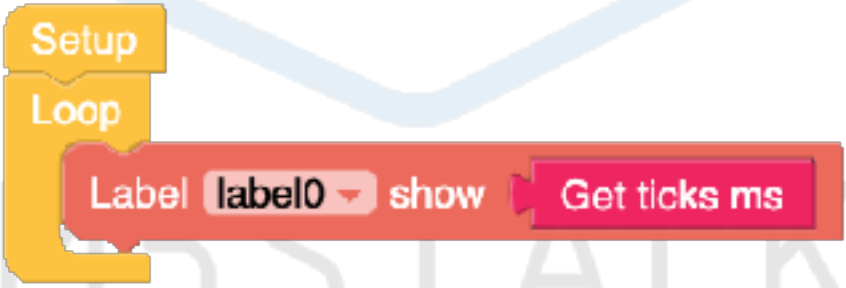
Get Ticks ms

The last timer block we have is Get Ticks ms, and no, this block has nothing to do with blood sucking mites! Get ticks ms returns the amount of millisecond size the CoreInk or M5Papers has been active.

The Micropython code for this is

```
time.ticks_ms()
```

Unlike the wait blocks, the Get Ticks ms block is a value block and must be used with other blocks like a text block.



this example use a label placed in a loop to just show how many seconds a device has been active.

The Micropython code for this example is as follows.

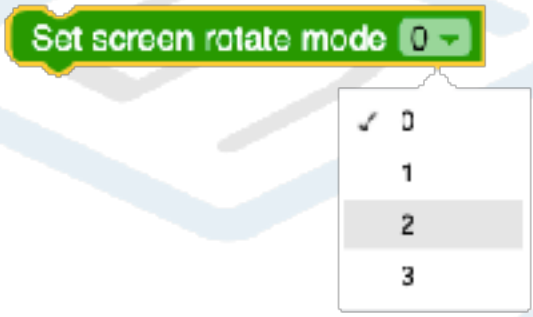
```
from m5stack import *
from m5ui import *
from uiflow import *
import time

setScreenColour(lcd.WHITE)

label0 = M5TextBox(49, 100, "Text",
lcd.FONT_Default, lcd.BLACK, rotate=0)


while True:
  label0.setText(str(time.ticks_ms()))
  wait_ms(2)
```

Graphical and GUI Function.

Screen

Set Screen Rotate Mode.

Set screen rotate mode is used to control the direction that the content of the screen will be shown.



Clicking the down arrow next to the number zero will show the dropdown menu that allows you to set the direction. Which way the screen content is shown when a number is selected is as follows:

- 0 - Normal view,
- 1 - Rotated 90 degrees clockwise,
- 2 - Rotated 180 degrees clockwise,
- 3 - Rotated 90 degrees anticlockwise (or 270 clockwise depending on your point of view.)

The Micropython code for this is

```
lcd.setRotation(0)
```

where the number in the brackets is 0, 1, 2, or 3.

Set Screen Background Colour

Set screen background colour is used to set the colour of the screen background. The CoreInk and M5Paper are mono e-ink displays and so can only be set to black and white.

Clicking the down arrow next to Black will show the dropdown menu showing the available background colours of Black and White.

The Micropython functions for this are

```
setScreenColour(lcd.BLACK)
```

or

```
setScreenColour(lcd.White)
```

Set Screen Show

The Set Screen Show is used to make information in our code visible on screen. Unlike set screen partial show, the whole screen is redrawn which is why there is a refresh rate of no less than fifteen seconds.

The Micropython code for this block is

```
CoreInkShow()
```

Warning: do not use a refresh rate below 15 seconds for the CoreInk.

Set Screen Partial Show

Set Screen Partial Show allows us to redraw only part of the screen.

The Micropython code for this block is:

```
CoreInkParitalShow(0, 0, 0, 0)
```

The arguments for this block are as follows:

- Horizontal top left start position of area to be partially refreshed,
- Vertical top left start position of area to be partially refreshed,
- Width of area to be partially refreshed,
- Height of area to be partially refreshed,

When a program is run on the CoreInk or M5Paper that uses the partial refresh block, only a section of the screen will flash black and white several times instead of the whole screen flashing black and white.

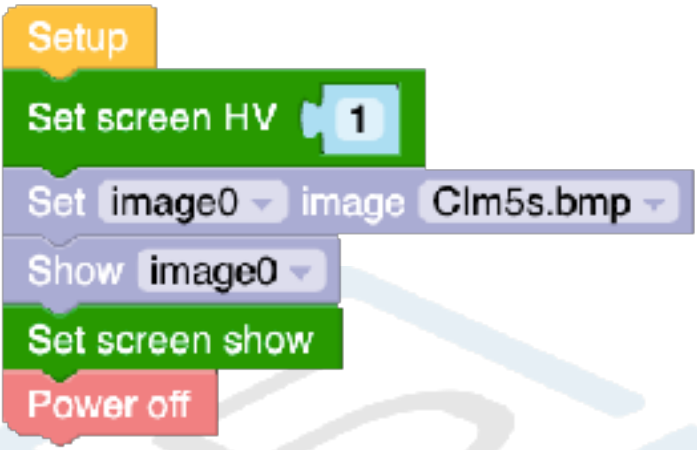
Set Screen HV

The Set Screen HV block is used to control the change pump that generates the high voltage needed for the e-ink screen draw. When set to 0 the charge pump is turned off the data last drawn to the screen will start to fade once the CoreInk is turned off using the Power Off function. With Set Screen HV set to 1, the data is permanently drawn on the screen until the screen is set a clear command.. The default setting for this is 1 which turns the charge pump on.

The Micropython command is :`CoreInkSetHV(0)`

If the CoreInk is powered off using the power button then the SetScreenHV function is never activated and the data will still fade.

The following example show the Set Screen HV and Power off blocks in use,



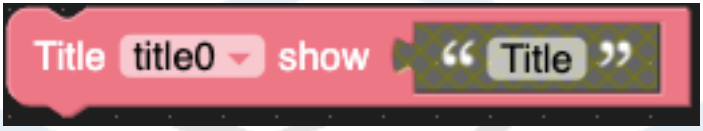
The Micropython code for this example is as follows:

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(lcd.WHITE)

image0 = M5Img(0, 0, "res/CIm5s.jpg",
visible=True, invert=False, threshold=128)

coreInkSetHV(1)
image0.changeImg("res/CIm5s.bmp")
image0.show()
coreInkShow()
power.off()
```

Title

Title Show

The blocks in this section are just for controlling a title bar that can be used at the top of the drawing area.

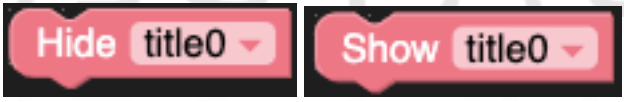


Title Show is used to set the text to be displayed in the title bar.

The Micropython code for this block is:

```
title0.setTitle('Title')
```

The text can be a string of text or a variable that has been converted into a string i.e:



The Hide Title block is used to hide the title bar when its not needed and Show Title block is used to show the title bar after it has been hidden.

The Micropython code for these blocks are:

```
title0.hide()
title0.show()
```

Set Title Colour block is used to set the text colour of the title bar. Setting the colour to black makes the text visible if the background is set to white, and setting to white makes the text visible when the background colour is set to black.

The Micropython code for this block is:

```
title0.setFgColour(lcd.BLACK)
```

Set Title BackgroundColor is used to set the colour of the title bar around the text.

The Micropython code for this block is:

```
title0.setBgColor(lcd.BLACK)
```

Label

Just like the Title Show, the Label Show is used to display a string of text or a value converted to a string.

The Micropython code for this block is:

```
label0.setText()
```

The Set Label X Y block is used to set the position that the label can be shown on the screen using a mathematical value block. If the values are replaced with variable blocks we can then move the label around the screen.

The Micropython code for this block is:

```
label0.setPosition(0, 0)
```

Set Label Font block is used to chose which font the string will be displayed with.

The available fonts options are as follows:

- Default,
- Default Small,
- Ubuntu,
- Comic,
- Dejavu 18,
- Dejavu 24,
- Dejavu 40,
- Dejavu 56,
- Dejavu 72,

The Micropython code for this block is:

```
label0.setFont(lcd.FONT_DefaultSmall)
```

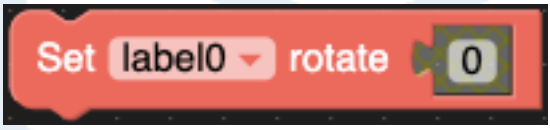
The Set Label Show/Hide is used to show or hide the label or sting of text.

The Micropython codes for this block is:

```
label0.hide()
```

and

```
label0.show()
```



The Label Rotate block is used to set the angle that the text is shown on the screen and uses a mathematical value block for the value. the values are from 0 - 360

The Micropython code for this block is:

```
label0.setRotate(0)
```

Set Label Colour block is used to set the text colour of the label. Setting the colour to black makes the text visible if the background is set to white and setting to white makes the text visible, when the background colour is set to black.

The Micropython code for this block is:

```
label0.setColour(lcd.BLACK)
```

or

```
label0.setColour(lcd.WHITE)
```

Rectangle, Circle and Triangle

To use a rectangle, circle or triangle you just drag a rectangle to the CoreInks screen and the following code gets added to our program. Here I am only showing the blocks for the rectangle as the circle and triangle are mostly the same. Once I have finished explaining the rectangle blocks, I will only cover the extra blocks not covered by the rectangle blocks.

```
    rectangle0 = M5Rect(77, 92, 26, 26,
lcd.BLACK, lcd.BLACK)
    circle0 = M5Circle(93, 129, 13,
lcd.BLACK, lcd.BLACK)
    triangle0 = M5Triangle(98, 47, 72, 73,
124, 73, lcd.BLACK, lcd.BLACK)
    line0 = M5Line(M5Line.PLINE, 55, 133,
115, 133, lcd.BLACK)
```

Set rectangle Width and Height blocks are used to control the size of the rectangle in the CoreInk's drawing area.

The Micropython codes for these blocks are :

```
rectangle0.setSize(30, 30)
rectangle0.setSize(height=30)
rectangle0.setSize(width=30)
```

The Set Rectangle X and Y blocks are used to control the position of the rectangle pithing the CoreInks drawing area.

The Micropython codes for these blocks are :

```
rectangle0.setPosition(30, 30)
rectangle0.setPosition(x=30)
rectangle0.setPosition(y=30)
```

The Micropython code for this block is:

```
rectangle0.hide()
```

or

```
rectangle0.show()
```

Set Rectangle Show/Hide block is used to display the rectangle on the screen or hide the rectangle.

Rectangles consist of the rectangle and a border, this block controls the border that surrounds the rectangle and can be set to black or white.

The Micropython code for this block is:

```
rectangle0.setBorderColour(lcd.BLACK)
```

or

```
rectangle0.setBorderColour(lcd.WHITE)
```

While the previous block controls the border colour, this block controls the colour or fill of the rectangle.

The Micropython code for this block is:

```
rectangle0.setBgColour(lcd.BLACK)
```

or

```
rectangle0.setBgColour(lcd.BLACK)
```

The Set Circle Radius block works like the width and height blocks, but controls the radius of the circle.

The Micropython code for this block is:

```
circle0.setSize(20)
```

and is controlled with a mathematical value block.

The Set Triangle block is another block that is different in that you can control both the X and the Y coordinates of the three points of the triangle block using a mathematics block or variables.

The Micropython code for this block is:

```
triangle0.setSize(0, 0, 0, 0, 0, 0)
```

The last GUI block that does share functions with the rectangle block is the Set Line Block.

The Set Line block has more in common with the triangle in that you can control the X and Y position of each end of the line.

The Micropython code for this block is:

```
line0.setSize(0, 0, 0, 0)
```

Images on the CoreInk and the M5Paper.

M5Stack E-Ink devices support images but there are some specific requirement that the images must meet.

For the CoreInk images must be:

- 1 Bit colour depth,
- Bmp or Jpg format,
- 200 X 200px in height and width,
- 50KB or under when saved,

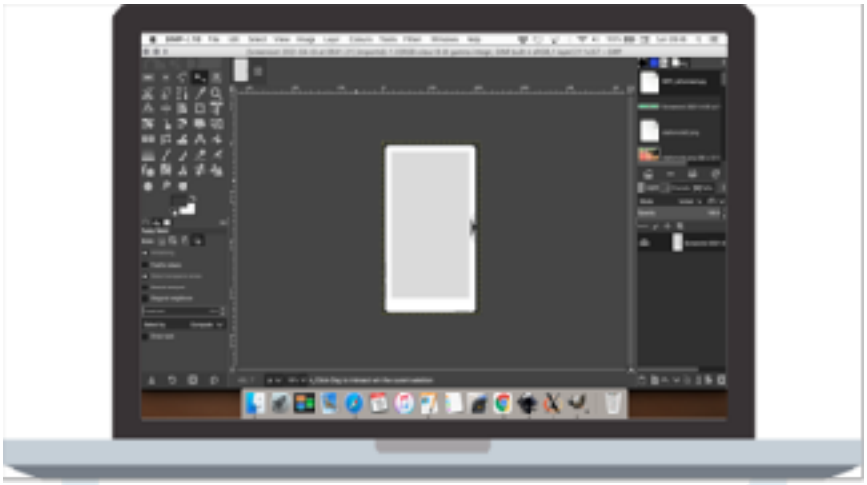
While it is possible to make images bigger and the memory can handle the extra size, the CoreInk has a minimum refresh rate of 15 seconds which makes it unusable for animations.

For the M5Paper, the image requirements are as follows:

- 16 Bit grayscale,
- Bmp, Jpg or PNG format,
- 960 X 540px in Height ad Width,
- 50KB or under when saved,

These requirements may feel quite restraining however, they are more than enough for making a simple image such as icons, that don't require animation.

In order to create an image or edit an existing image to fit the E-Ink requirements, we can use a graphics program called G.I.M.P.

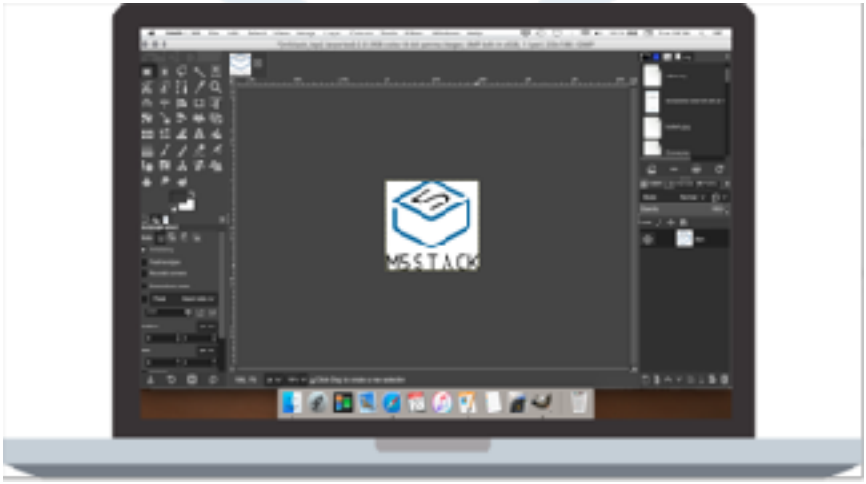


To make a start we first need an example image. For the example image I will use the M5Stack logo seen on the background of all pages.

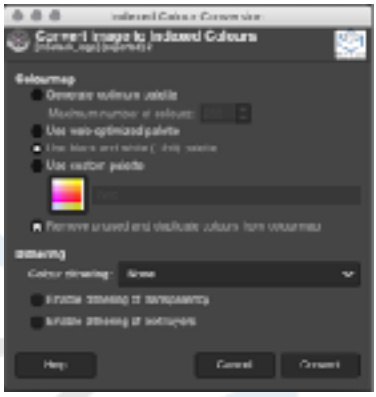
In order for this to be useable we first need to scale the image. We could keep the image to its current size and change it with code but we can save space in the ESP32's memory but scaling the image to the screen size before uploading it to the CoreInks storage.

In G.I.M.P. go to `Image > Scale Image` and this dialog will appear. Change the Width to 200px and the height will change to 186px.

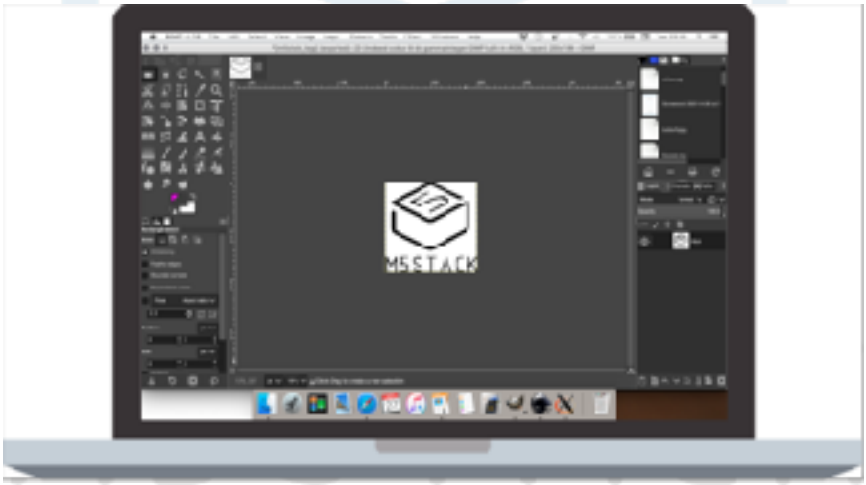
Click on the `Scale` button and the image will change.



Next we need to reduce the colour depth. To reduce the colour depth we need to click on `Image > Mode > Indexed` to bring up this dialog. Click on the round box next to `Use Black and White (1-bit) palette`.



Leave all other options as they are and click on `Convert` to return to the image.

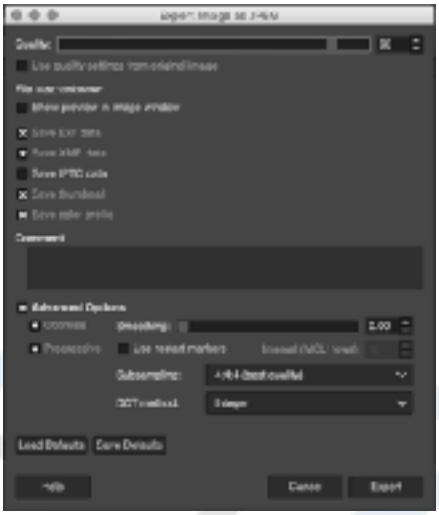


Now that we have reduce the image in size and colour it is time to save the image. The CoreInk supports supports both Jpgs and Bmp image formats.

When you click `File > Save` in G.I.M.P, images are saved to its .xcf file format. In order to save in .jpg or .bmp you need to go to `File > Export As`.

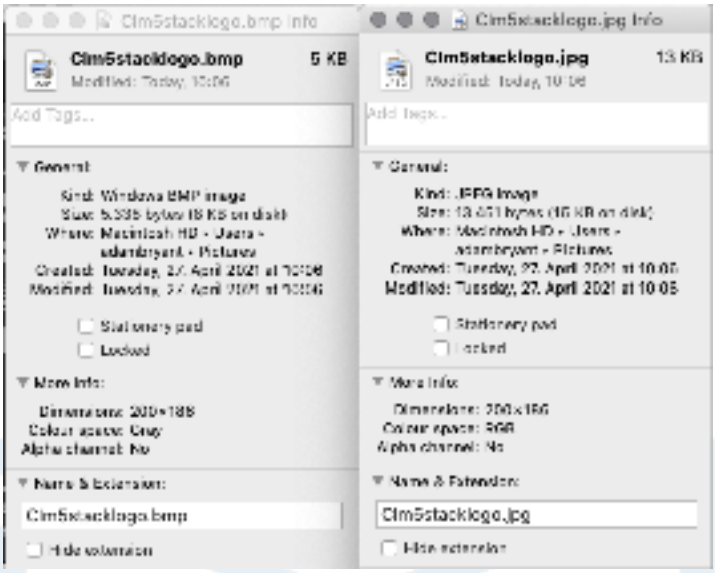
At the top of the dialog is where we type in the file name and at the bottom is the Select File Type list.

We don't really need the file type list and we know that the images must be saved with a .Bmp or .Jpg extension. If we type the extension after the file name (like in the screenshot) G.I.M.P. will automatically select the file type for us and bring up the appropriate export dialog for us once we press the `Export` button.



Click on the box next to `Advanced Options` to show the extra setting seen in the above dialog if they are not already visible.

All the options selected are not support in micropython and will increase the file size of the saved image. Deselect all the options in the dialog and set `Subsampling` to `4:2:0(chroma quartered)` and then click `Export.`



In this screenshot you can see the difference in file size when exported to .Bmp and .Jpg file formats even though both image visibly look the same.

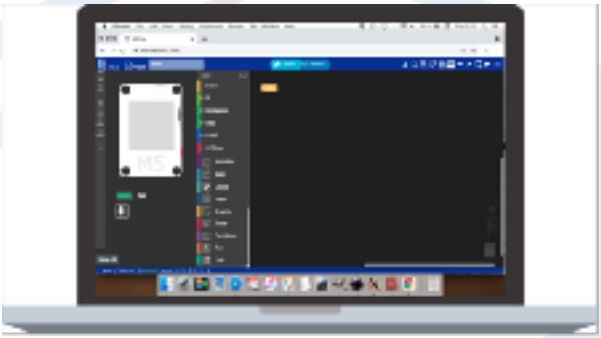
M5stack file names also have a requirement and that is that they must be in the 7.3 format i.e: 1234567.jpg or 1234567.bmp.

If the names are too long like the ones in the dialogs above you will see the Unsupported Image Type message on the screen.

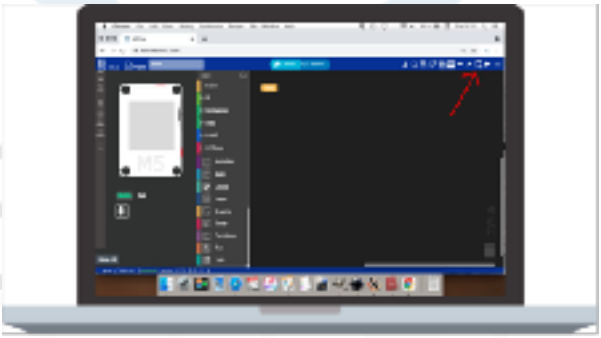
Uploading images to the CoreInk and M5Paper.

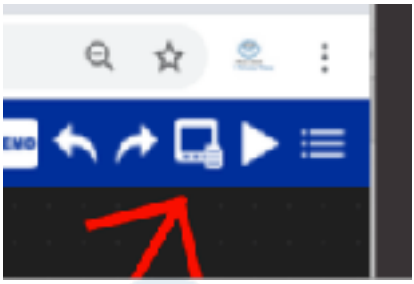
Now that we have the prepared images ready to be used we now need to upload them to the CoreInk and M5Paper. The easiest way to do this is from within UIFlow.

Open UIFlow and connect to the CoreInk or M5Paper.

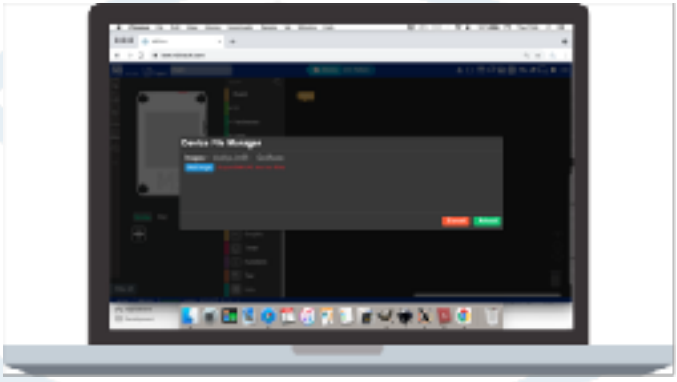


Next click on the third blue icon in from the right,

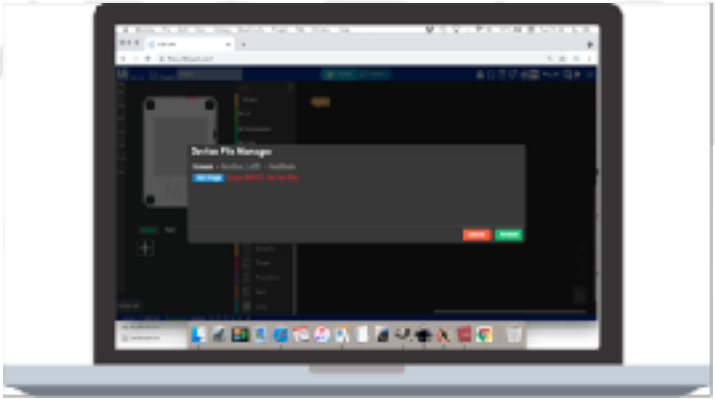




This will bring up the file upload dialog.

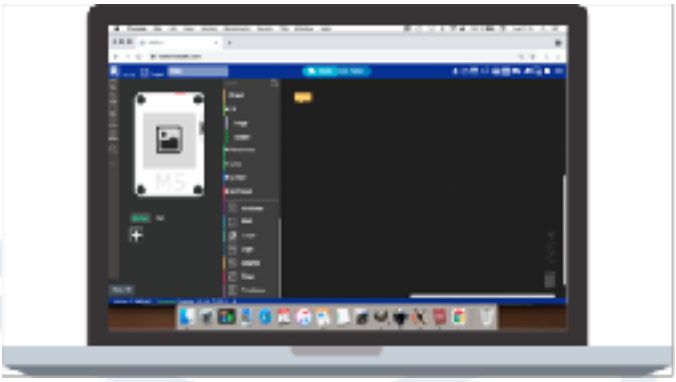


Click on the `Open Image` button and the File open dialog will pop up, click on the images, press OK and they will be uploaded to the CoreInks memory.

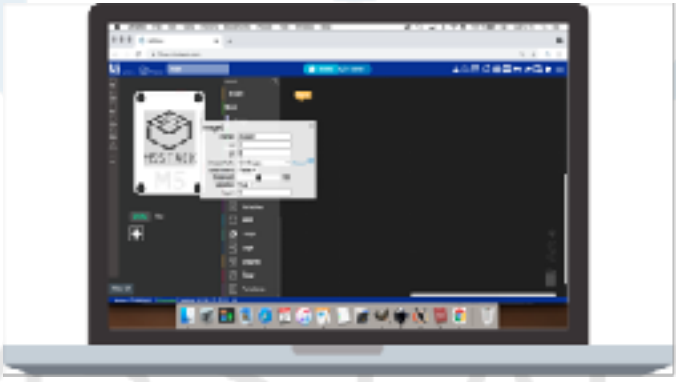


Next click on the `Cancel` button to close the dialog.

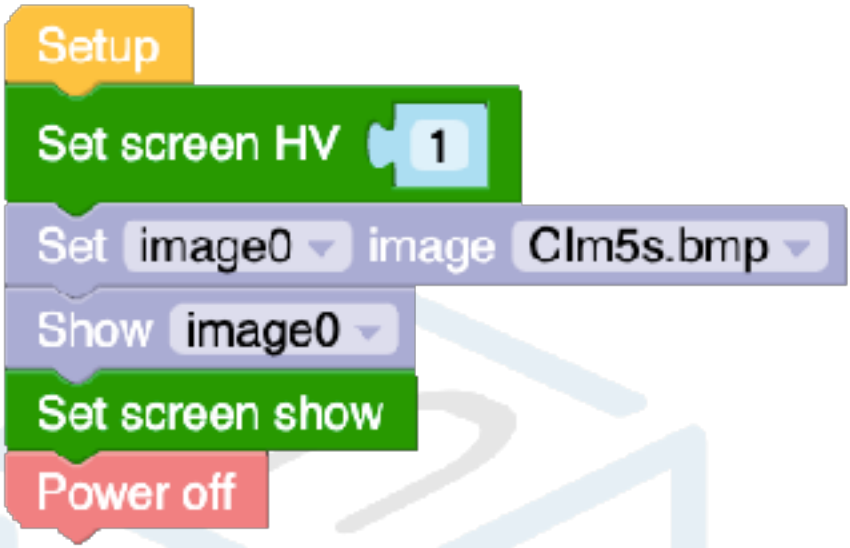
To get the Images to show on the CoreInk or M5Paper's screen we now need to add an image UI element to the screen.



Click on the icon and the image setting dialog will appear,



Click on the dropdown box next to `Image path` and chose the image to be displayed. The icon will change to the image we uploaded but to make it appear on the screen, we need a small piece of code.



When run the screen will show the uploaded and selected image.

Internal Hardware

Speaker

(**M5Paper Only**)

Speaker.Beep Frequency

The Speaker Beep Frequency block allows us to make sounds by setting the frequency of the sound and the duration of the sound. We can use the value blocks to manually set the frequency and duration or we can replace these value blocks with a variable block or a value returned from a hardware block.

The Micropython code for this is:

```
speaker.tone()
```

Example

In the following example, when run, will play two tones from the M5Stacks internal speaker.

The Micropython code for this example is as follows from m5stack import *

```
from m5ui import *
from uiflow import *

setScreenColour(0x222222)

while True:
  speaker.tone(1800, 200)
  wait_ms(200)
  speaker.tone(1500, 200)
  wait_ms(200)
  wait_ms(2)
```

Speaker Volume

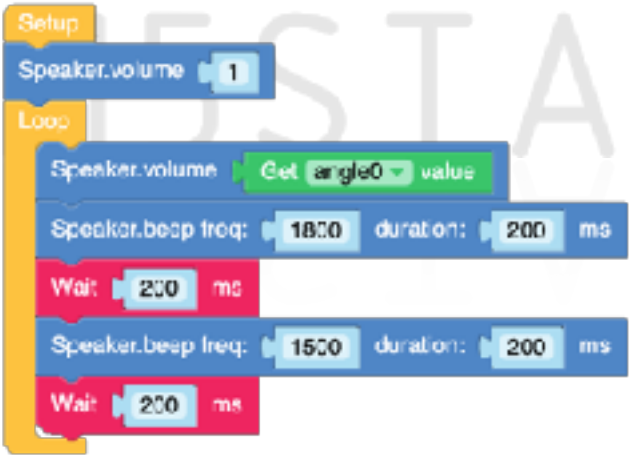
Speaker volume allows us to set the volume of the speaker sounds. The volume has to be initially set before the main loop and can be changed later in the loop. We can use the value block to manually set the volume or or we can replace these value blocks with a variable block or a value returned from a hardware block.

The Micropython code for this is:

```
speaker.setVolume()
```

Example

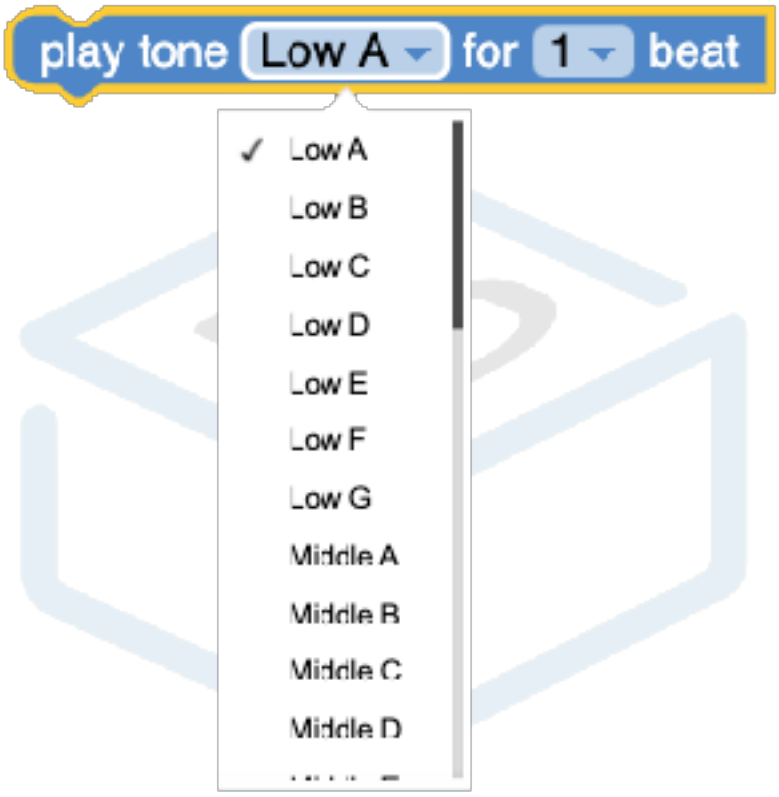
In the following example, I have just added the volume block that gets its value from the angle sensor connected to port B.



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColour(0x222222)
angle0 = unit.get(unit.ANGLE, unit.PORTB)

speaker.setVolume(1)
while True:
  speaker.setVolume((angle0.read()))
  speaker.tone(1800, 200)
  wait_ms(200)
  speaker.tone(1500, 200)
  wait_ms(200)
  wait_ms(2)
```

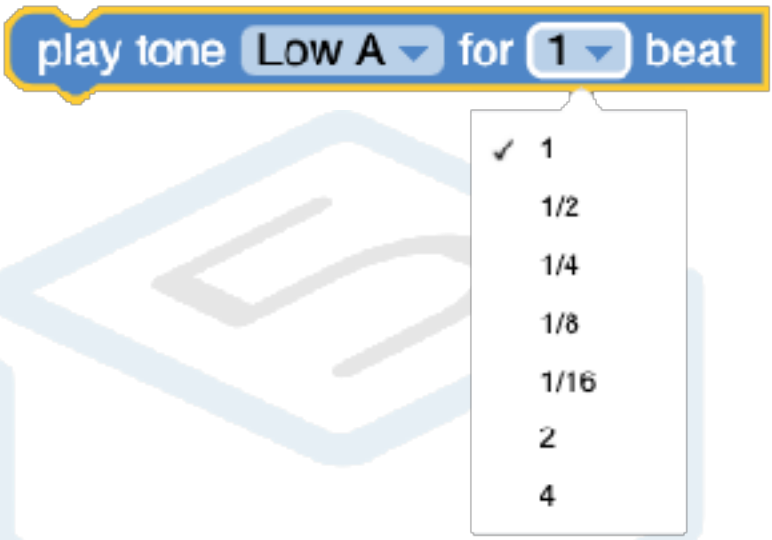
Play Tone



The Play Tone block allows users to play musical notes and set the duration in beats or fractions of Beats. Clicking on the arrow to the right of the note pops up a scrolling list of available programmed notes, likewise clicking on the arrow next to beat display the beat size list.

The Micropython code for this is:

```
speaker.sing(220, 1)
```



Example

The following example plays three note in the loop continuously.

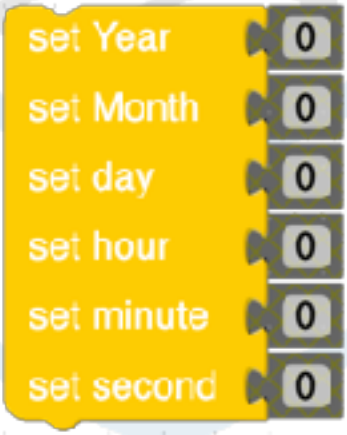
And the Micropython code for this is

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x222222)

while True:
  speaker.sing(131, 1)
  speaker.sing(165, 1)
  speaker.sing(196, 1)
  wait_ms(2)
```

Real Time Clock

Both the CoreInk and the M5Paper use the BM8563 RTC clock for keeping time and time sensitive programs. The blocks in this section are used for setting and controlling time.

Time and Date configuration block

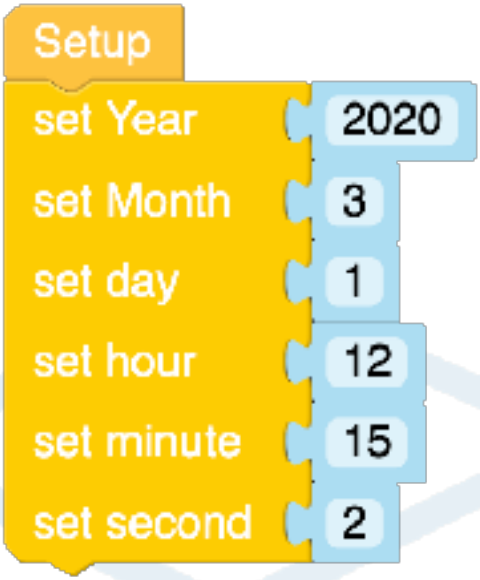


Used to configure the initial time and date settings for the internal clock. We can manually set the values by typing them in or we can use other blocks like variables.

The Micropython code for this block is

```
rtc.setTime(0, 0, 0, 0, 0, 0)
```

Example



To set the time and date options, you would configure this block and place it in the setup stage before the main loop.

The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

rtc.setTime(2020, 3, 1, 12, 15, 2)
```

The number in the brackets are from left to right and represent the values of the blocks from top to bottom.

This block doesn't not show anything on the screen but sets these values in the code background.

To view these values on screen, we need to use the following blocks.

So far there is no ability to receive the time and date from the global radio clock signal to automatically set time and date on the CoreInk or M5Paper.

Get Year

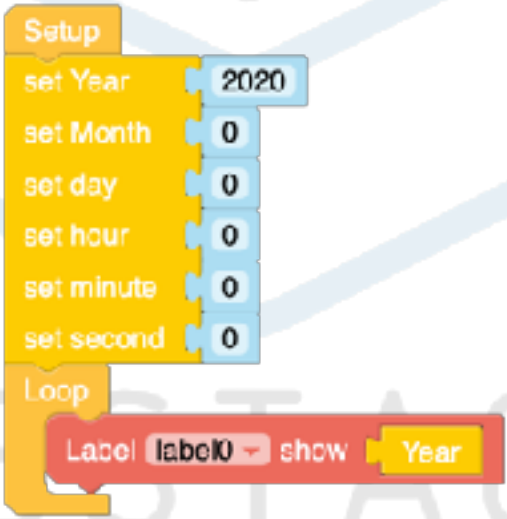
Gets the current year held in the real time clocks memory.

The micropython code for this block is

```
rtc.now()[0]
```

Example

In this example the code uses a label to display the date configured using the setup block.



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(18, 53, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(2020, 3, 25, 14, 21, 0)
while True:
  label0.setText(str(rtc.now(0)))
  wait_ms(2)
```

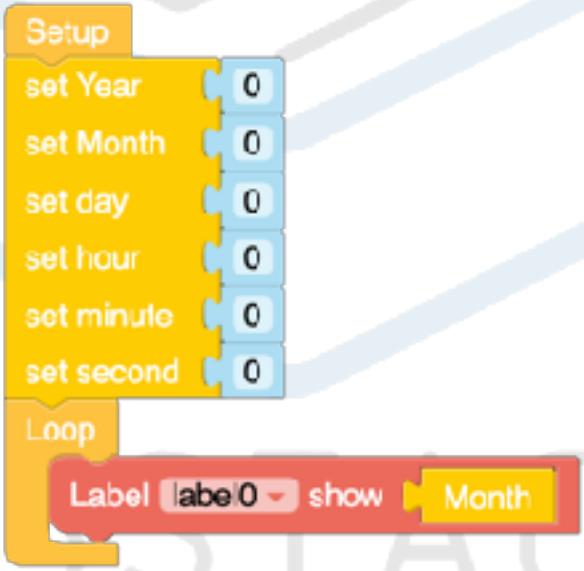
Get Month

Gets the month from the RTC.

The micropython code for this block is

```
rtc.now()[1]
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()[1]))
  wait_ms(2)
```

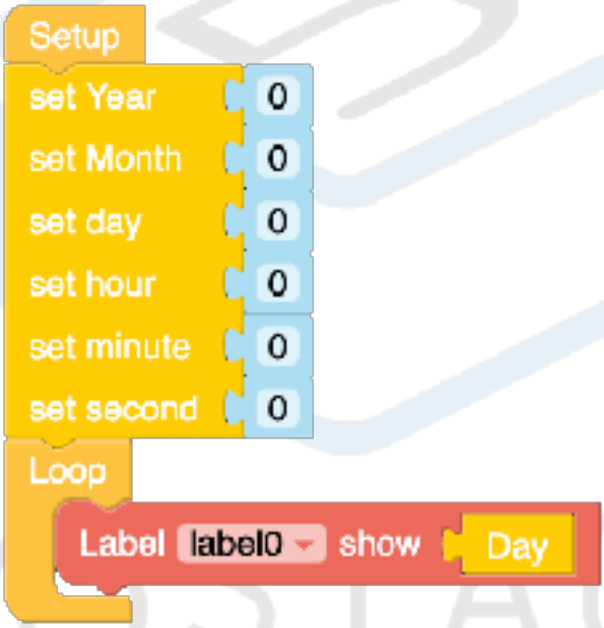
Get Day

Gets the day from the RTC.

The micropython code for this block is

```
rtc.now()[2]
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()[2]))
  wait_ms(2)
```

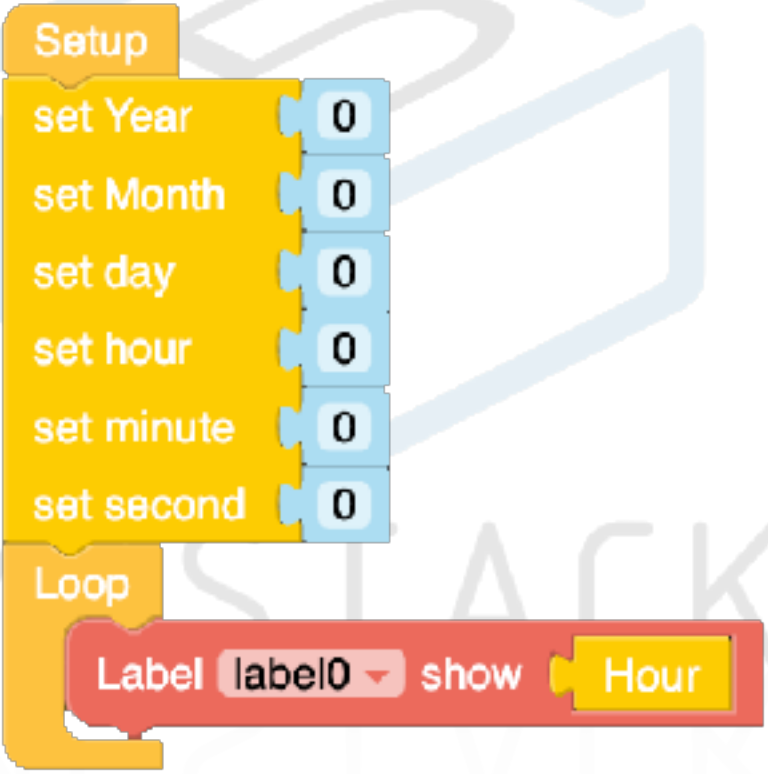
Get Hour

Gets the hour from the RTC.

The micropython code for this block is

```
rtc.now()[3]
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()[3]))
  wait_ms(2)
```

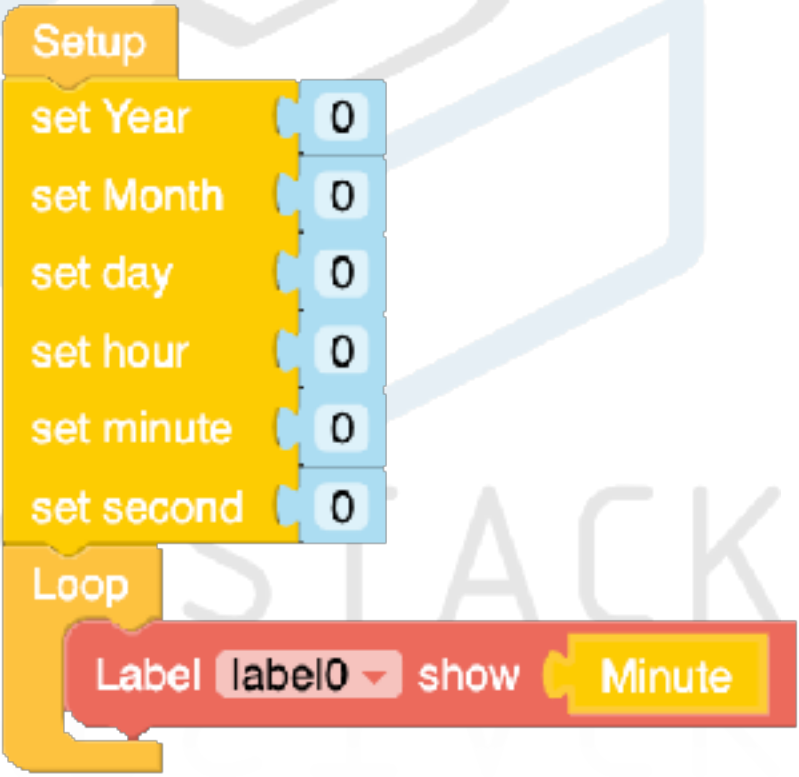
Get Minute

Gets the minute from the RTC.

The micropython code for this block is

```
rtc.now()[4]
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()[4]))
  wait_ms(2)
```

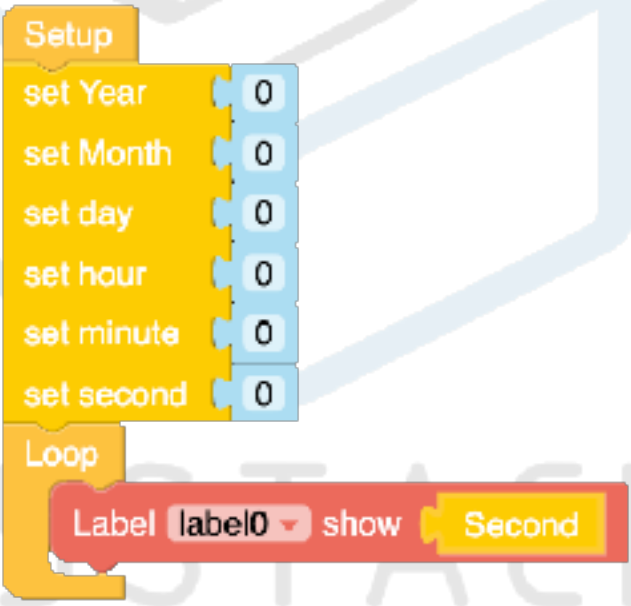
Get Second

Gets the seconds from the RTC.

The micropython code for this block is

```
rtc.now()[5]
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()[5]))
  wait_ms(2)
```

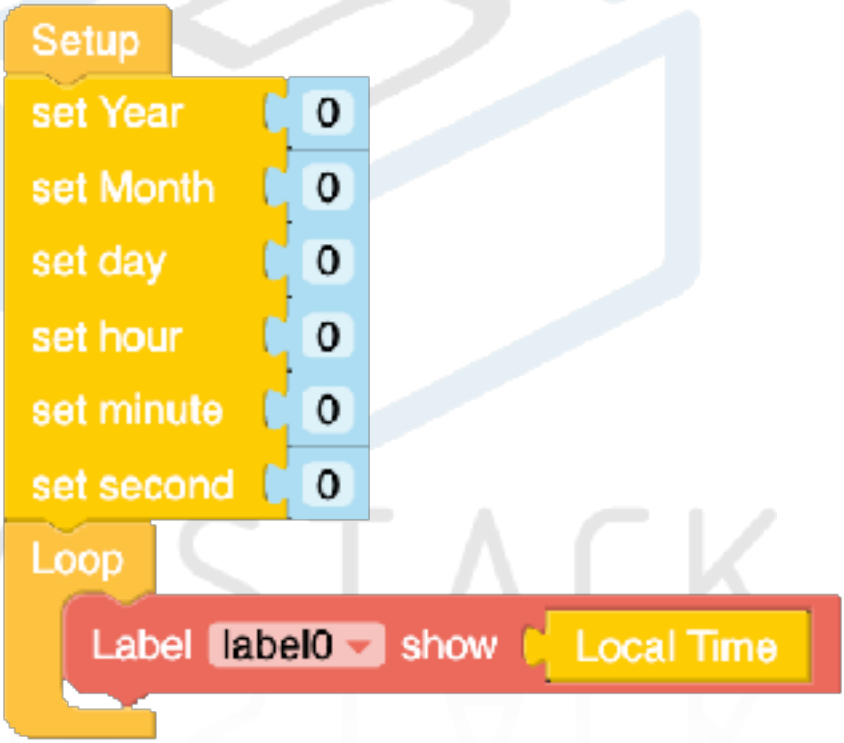
Get Local Time

Gets the local from the RTC.

The micropython code for this block is

```
rtc.now()
```

Example



The Micropython code for this example is as follows.

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColour(0x111111)

label0 = M5TextBox(22, 57, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

rtc.setTime(0, 0, 0, 0, 0, 0)
while True:
  label0.setText(str(rtc.now()))
  wait_ms(2)
```

Network Time Protocol

So far I have shown you how to manually set the time a data on the real time clock. We can get these setting automatically changed every time the CoreInk and M5Paper is restarted by using Network Time Protocol (hereafter NTP).

The NTP blocks are as follows:

Init NTP time with Host

The Init NTP time with host block is for choosing which NTP server will be used. by clicking the down arrow a dropdown list of preprogrammed servers is selectable.

If a suitable NTP server is not available, we can use the following block to define a different server to use.

The number on the end is used to se how many hours a chosen timezone is ahead of GMT.

The Micropython code for these two blocks is the same

```
ntp = ntptime.client(host='cn.pool.ntp.org',
timezone=8)
```

The only difference is in the way that UIFLow adds the code to the program for us.

Get Time Stamp

The Get Timestamp block is used to retrieve the current date and time however it returns it as numbers

```
>>> ntp = ntptime.client(host='cn.pool.ntp.org', timezone=0)
>>> print(str(ntp.getTimestamp()))
665660512
```

The above screenshot taken from Mu editor shows how the timestamp is returned when the time zone is set to "0" for GMT.

Mu Editor can be downloaded from https://codewith.mu

Touch Coordinates

M5Paper Only

Get Touch Coordinates

The Get touch Coordinate returns the X and Y positions touched on the screen of the M5Paper and returns as a comma separated string.

The Micropython code for this block is:

```
touch.read()
```

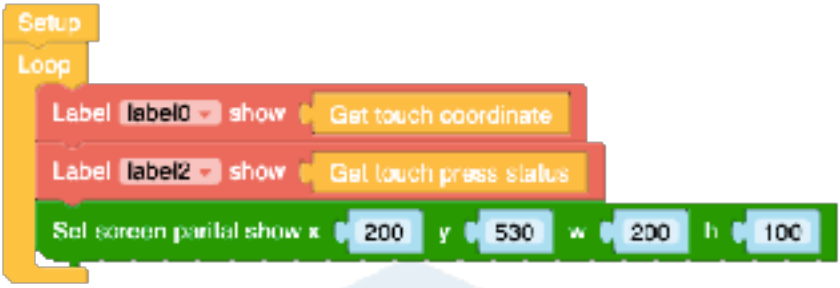
Get touch Status

The touch press status is used to read if the screen has been touched or not and returns true if it detects contact or false if there is no contact.

The Micropython code for this is:

```
touch.status()
```

The following example refreshes a partial section of the screen only around the text labels just to show the status and position of a finger touch.



The Micropython code for this example is:

```
from m5stack import *
from m5ui import *
from uiflow import *
from m5stack import touch

setScreenColour(15)

label0 = M5TextBox(210, 541, "Text",
lcd.FONT_DejaVu24, 0, rotate=0)
label1 = M5TextBox(210, 541, "Text",
lcd.FONT_DejaVu24, 0, rotate=0)
label2 = M5TextBox(213, 589, "Text",
lcd.FONT_DejaVu24, 0, rotate=0)

while True:
  label0.setText(str(touch.read()))
  label2.setText(str(touch.status()))
  lcd.partial_show(200, 530, 200, 100)
  wait_ms(2)
```

Battery

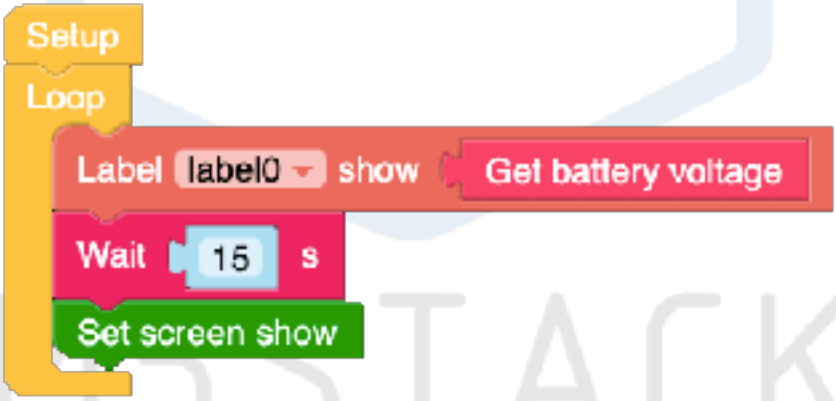
Get battery Voltage

The Get Battery Voltage returns the current battery voltage.

The Micropython code for this block is

```
bat.voltage()
```

The following example shows a simple battery monitor on the CoreInk and M5Paper.



LED

CoreInk Only

On the top left hand corner of the M5Stick C is a red led. To control this led you use the following blocks.

LED On

Turns the red led on the top of the M5Stick C on.

The Micropython code for this is,

```
M5Led.on()
```

LED Off

Turns the red led on the top of the M5Stick C off.

The Micropython code for this is,

```
M5Led.off()
```

LED Value

The LED Value block is used to turn the LED on or off using a number which is 0 for off and 1 for on.

The Micropython code for this is,

```
M5Led.value(0)
```

The Micropython code for this example is:

```
from m5stack import *
from m5ui import *
from uiflow import *
import time

setScreenColour(15)

label0 = M5TextBox(228, 497, "Text",
lcd.FONT_DejaVu24, 0, rotate=0)

while True:
  label0.setText(str(bat.voltage()))
  wait(15)
  lcd.show()
  wait_ms(2)
```

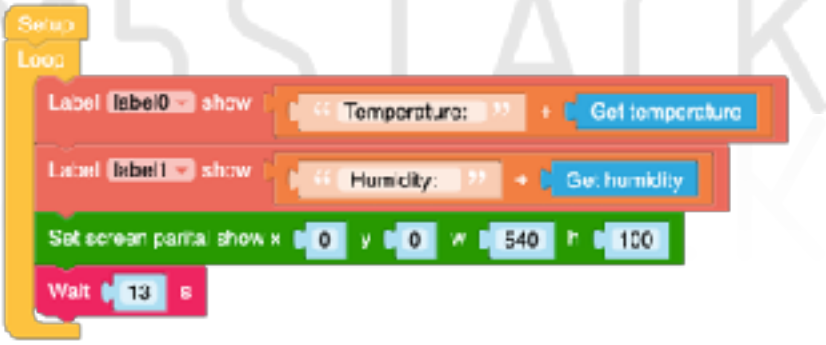
SHT30

To access the internal SHT30 sensor we have only two blocks humidity and temperature.

Both of these blocks return the values of the humidity and temperature being read by the internal SHT30 sensor. The Micropython code for these blocks are as follows.

```
sht30.humidity
sht30.temperature
```

In order to use these block we need to use additional blocks. In the following example we use a label to show the readings and set a partial screen refresh to avoid the need to refresh the whole screen when I am only using a small part of the screen.



And the Micropython code for this example is as follows

```
from m5stack import *
from m5ui import *
from uiflow import *
import time

setScreenColour(15)

label0 = M5TextBox(5, 5, "Text",
lcd.FONT_DejaVu40, 0, rotate=0)
label1 = M5TextBox(0, 70, "Text",
lcd.FONT_DejaVu40, 0, rotate=0)

while True:
  label0.setText(str((str('Temperature:  ')
+ str((sht30.temperature)))))
  label1.setText(str((str(' Humidity:   ') +
str((sht30.humidity)))))
  lcd.partial_show(0, 0, 540, 100)
  wait(13)
  wait_ms(2)
```

SD Card slot(TBC)

Unfortunately, as of writing (March14th 2021) SD card support has not been enabled in UIFlow.

Power

Power On

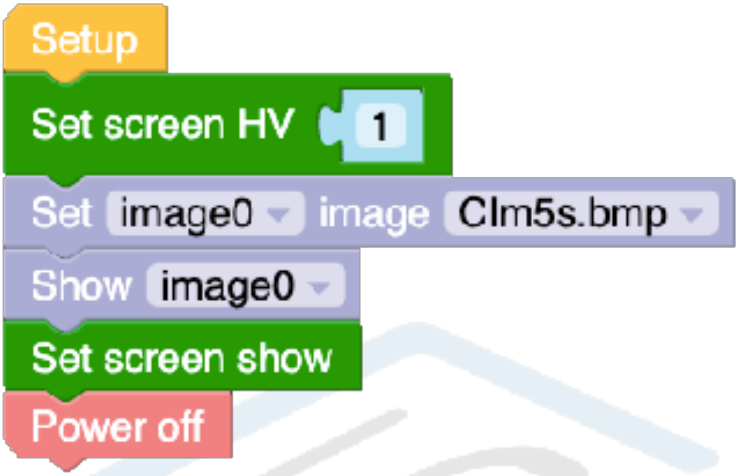
Power Off

The Power Off function is used to fully deactivate the CoreInk from within the code. To call the function in Micropython we use the following command:

```
power.off()
```

Or by using the following UIFlow block:

In the following example the Power Off block is used in combination with the Set Screen HV block in order to retain the data on the CoreInks screen after power off.



The Micropython code for this example is as follows:

```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(lcd.WHITE)

image0 = M5Img(0, 0, "res/CIm5s.jpg",
visible=True, invert=False, threshold=128)

coreInkSetHV(1)
image0.changeImg("res/CIm5s.bmp")
image0.show()
coreInkShow()
power.off()
```

Restart after Seconds ( )

To use the Restart after seconds function, you use the following Micropython command:

```
power.restart_after_seconds(0)
```

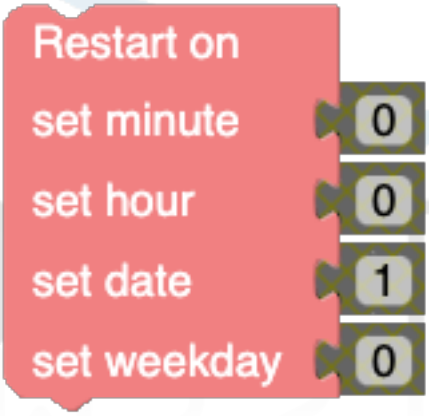
or use this UIFLow block:

Restart on Time

To use the Restart on Time function, you need to use:

```
power.restart_on(minutes=0, hours=0, date=1, weekday=0)
```

or use the UIFLow block:



Timers Part 3
Watch Dog Timer

The Watch Dog timer (here after referred to as WDT) is used to monitor the program to check if it has crashed or not. In the event of a crash the WDT will wait for the specified time to see it is reset by the wdt.feed function and if not, will hard reset the ESP32.

There are two parts to the Watch Dog timer, the Init WDT

```
wdt = WDT(timeout=2000)
```

and the Feed WDT

```
wdt.feed()
```

And these functions can be accessed by calling:

```
from machine import WDT
```

In UIflow the WDT is called by adding the Init WDT Timer block which automatically adds the import call to the Setup block;

and to reset the WDT timer we use the WDT Feed block:

The WDT Feed block need to be placed somewhere suitable in the main loop to reset the WDT time but you have to make sure that the timer is set long enough for the loop to reach the WDT Feed block otherwise you would get false resets.

Programming with Arduino

To program the CoreInk in the Arduino we need to download the latest version of the Arduino IDE and configure the environment to communicate with the CoreInk and other M5Stack controllers.

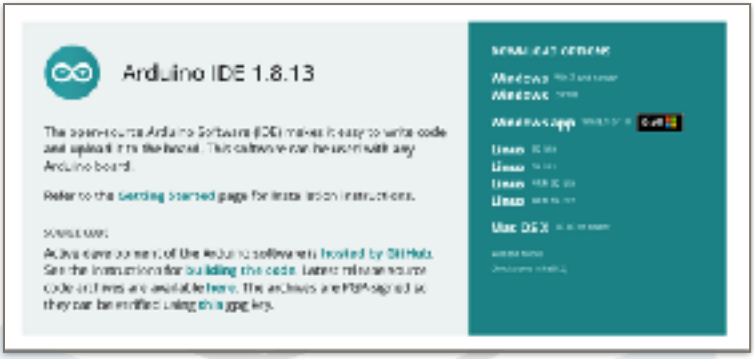
Step 1

Install the CP210X driver as instructed in a previous chapter.

Step 2

Goto https://www.arduino.cc move the curse over to software and click on downloads.

Scroll down to downloads and download the latest version of the Arduino IDE for your operating system.



At the next page you can choose to download the Arduino IDE without a donation but, I recommend donating and downloading.

Step 3

Next create a folder on your computer, copy the downloaded Arduino program into that folder and open the Arduino IDE.

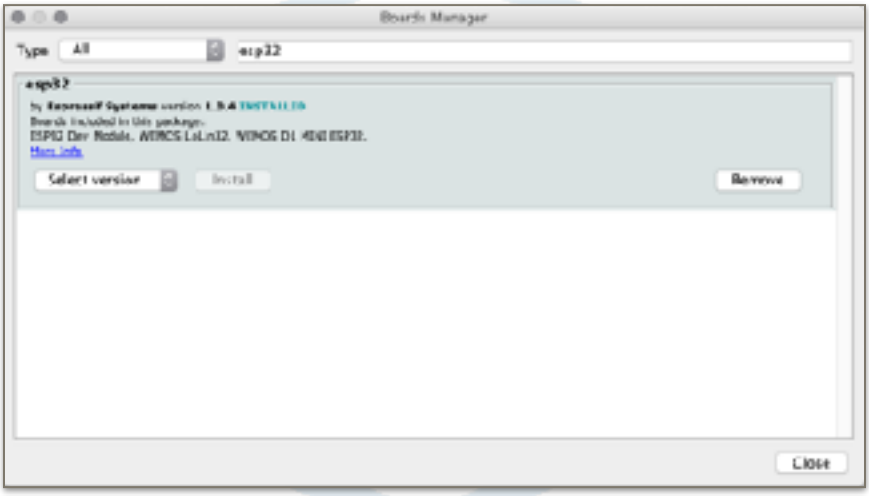
Step4

Next goto File>Preferences>Settings, (or Arduino>Preferences>setting in OSX)

And type in https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json into the box next to Additional Board Manager URL's

Click on the OK button to close this window and then goto Tools>Board>Board Manager.

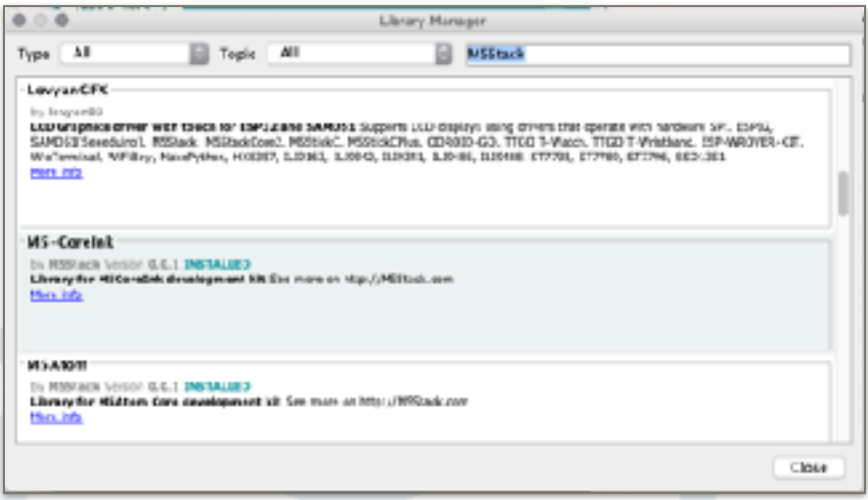
Click in the search bar, type ESP32



Click on the dropdown box to select the latest version (1.0.3 as of writing) and then click on install. When Installation has finished we can close this window and return to the main Arduino window.

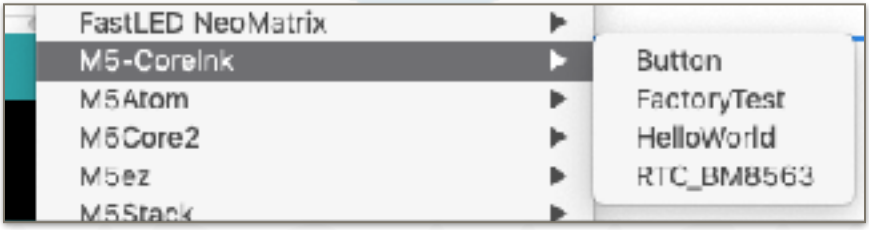
Step 5

Now goto Sketch>Include Library>Manage Libraries.



Type in M5Stack, click on M5-CoreInk, click on install and then click close to return to the main window again.

If we goto File>Examples and scroll down, there will be a CoreInk menu with some prewritten examples available to look at.



Arduino CoreInk Examples

As of writing, there are only four examples for the CoreInk found in the Arduino library.

These four demos are as follows:

- Button,
- Factory Test,
- Hello World,
- RTC_BM8563

Due to the size of the source code's and the fact that they are several files long, I wont be posting the Factory test code here.

The Button Example.

In order to program the CoreInk in the Arduino IDE we first have to call the library that hold the import API commands. This is done by adding:

```
#include "CoreInk.h"
```

This file contains some definitions used only by the CoreInk and Calls other libraries that contain functions we need. Next we need to create a page that we want to display information on. This page is held in memory until all the pages items are created and drawn to the E-Ink display from memory.

To create the page in memory we use the following:

```
Ink_Sprite InkPageSprite(&M5.M5Ink);
```

If we look at the API we find that this line creates a function called InkPageSprite as a child of the Ink_Sprite and assigns it the value of M5Ink as a child of M5.

Next the M5 functions are called in the void setup() and linked to the M5Ink object.

```
M5.begin();
```

The following code checked to see if the screen functions have been called and checks to see if the screen has been initialised. If it hasn't been initialised an error message is sent out over the USB to be read by a computer that may be connected

```
 if( !M5.M5Ink.isInit())
    {
        Serial.printf("Ink Init faild");
        while (1) delay(100);
    }.
```

Next we need to clear the screen by calling:

```
   M5.M5Ink.clear();
```

We then pause the program with

```
delay(1000);
```

before creating the Page sprite that will hold the graphical information we wish to display.

```
if( InkPageSprite.creatSprite(0,0,200,200
,true) != 0 )
        {
```

```
            Serial.printf("Ink Sprite creat
faild");
        }
```

This function is placed in an if function so that if the values are incorrect, we will get a message over the USB telling us that the programmed failed to create the page.

Next we create the function that creates the object to be shown on the screen and displays the object.

This example was written in the void Setup() function as it only need to be ran once. If we wanted to have several messages to be displayed then it needs to be written in the void loop() function.

An important note about E-Ink displays and looped functions. especially the GDEW0154M09 used in the CoreInk.

The refresh rate of the screen is slow and needs a delay of fifteen seconds at the end of the loop. If the delay is any lower it could cause damage to the screen.

The Hello World Example


HelloWorld | Arduino 1.8.13
HelloWorld
#include "M5CoreInk.h"
Ink_Sprite InkPageSprite(&M5.M5Ink);
void setup() {
M5.begin();
if(!M5.M5Ink.isInit())
{
Serial.printf("Ink Init faild");
while (1) delay(100);
}
M5.M5Ink.clear();
delay(1000);
//creat new refresh Sprite
if(InkPageSprite.creatSprite(0,0,200,200,true) != 0)
{
Serial.printf("Ink Sprite creat faild");
}
InkPageSprite.drawString(25,50,"Hello World");
InkPageSprite.pushSprite();
}
void loop() {
}

The RTC_BM8563 Example

Arduino CoreInk API

E-Ink

Before using the E-Ink screen operation function, you need to create an instance and pass in the screen driver address. Call M5.M5Ink.isInit() and initialize with M5.M5Ink.clear();.

```
#include "CoreInk.h"

Ink_Sprite InkPageSprite(&M5.M5Ink);

void setup()
{
    M5.begin();
    digitalWrite(LED_EXT_PIN,LOW);
    if( !M5.M5Ink.isInit())
    {
        Serial.printf("Ink Init faild");
        while (1) delay(100);
    }
    M5.M5Ink.clear();
}
```

createSprite();

Syntax:

int creatSprite(uint16_t posX, uint16_t posY, uint16_t width = 200, uint16_t height = 200, bool copyFromMem = true);

Creates an image area, configure whether to get image data from the buffer or from screen driver . By default this is set to true which means it will retrieve data from the screen buffer.

The x coordinate of the >create image area must be an integer multiple of 8. (eg: 0 , 8 , 16...) Otherwise, the display will not work properly.

Example:

```
if( InkPageSprite.creatSprite(0,0,200,200,true) != 0 )
{
   Serial.printf"Ink Sprite creat faild");
}

Serial.printf("Ink Sprite creat successful");
```

clear();

Syntax:

void clear(int cleanFlag = CLEAR_DRAWBUFF);

Description: Clear the content of the image area, CLEAR_DRAWBUFF (clear the image data that has not yet been pushed) CLEAR_LASTBUFF (clear the cache area of the image data refreshed last time, and refresh the screen completely when it is pushed again)

Example:

InkPageSprite.clear();

deleteSprite();

Syntax:

int deleteSprite();

Description: Release the created image area

Example:

InkPageSprite.deleteSprite();

pushSprite();

Syntax:

int pushSprite();

Description: Push the edited image data to the image area (after using the drawing API to operate, you need to push to refresh the screen)

Example:

InkPageSprite.drawString(10,50,"Hello World!",&AsciiFont8x16);

InkPageSprite.pushSprite();

drawPix();

Syntax:

void drawPix(uint16_t posX,uint16_t posY,uint8_t pixBit);

Description: draw a single pixel(the parameter pixBit is black when 0 is passed in, and white when 1 is passed in)

Example:

InkPageSprite.drawPix(100,100,0);

InkPageSprite.pushSprite();

FillRect();

Syntax:

void FillRect(uint16_t posX, uint16_t posY, uint16_t width, uint16_t height, uint8_t pixBit);

Description: draw a rectangle (the parameter pixBit is black when 0 is passed in, and white when 1 is passed in)

Example:

InkPageSprite.FillRect(0,0,100,100,0);

InkPageSprite.pushSprite();

drawFullBuff();

Syntax:

void drawFullBuff(uint8_t buff, bool bitMode = true);*

void drawBuff(uint16_t posX, uint16_t posY, uint16_t width, uint16_t height, uint8_t imageDataptr);*

Description: To draw the entire page, you need to pass in the complete page Buff

drawChar();

Syntax:

void drawChar(uint16_t posX,uint16_t posY,char charData,Ink_eSPI_font_t fontPtr);*

Description: draw characters

Example:

InkPageSprite.drawChar(35,50,'M');

InkPageSprite.pushSprite();

drawString();

Syntax:

void drawString(uint16_t posX,uint16_t posY,const char charData,Ink_eSPI_font_t* fontPtr = &AsciiFont8x16);*

Description: draw string

Example:

InkPageSprite.drawString(35,50,"Hello World!");

InkPageSprite.pushSprite();

getSpritePtr();

Syntax:

Description: Get image buff data

uint8_t getSpritePtr(){ return _spriteBuff;}*

Description: Get image width coordinate data

uint16_t width(){ return _width;}

Description: Get image height coordinate data

uint16_t height(){ return _height;}

Description: Get image X coordinate data

uint16_t posX(){ return _posX;}

Description: Get Y coordinate data of image area

uint16_t posY(){ return _posY;}

System & Button & RTC

begin();

Syntax:

int begin(bool InkEnable = true, bool wireEnable = false);

Description: Initialize E-Ink, RTC, I2C, Speaker

Example:

```
#include "CoreInk.h"


void setup() {
  M5.begin(true,false);
}
```

update();

Syntax:

void update();

Description: refresh device button/buzzer status

Example:

```
if( M5.BtnPWR.wasPressed())
{
  Serial.printf("Btn wasPressed!");
}
M5.update();
```

Button

Description: detects the device key state

M5.BtnUP.wasPressed()

M5.BtnDOWN.wasPressed()

M5.BtnMID.wasPressed()

M5.BtnEXT.wasPressed()

M5.BtnPWR.wasPressed()

Speaker

Description: Buzzer Drive

void end();

void mute();

void tone(uint16_t frequency);

void tone(uint16_t frequency, uint32_t duration);

void beep();

void setBeep(uint16_t frequency, uint16_t duration);

void update();

void write(uint8_t value);

void setVolume(uint8_t volume);

*void playMusic(const uint8_t *music_data, uint16_t sample_rate);*

M5.Speaker.tone(2700);

RTC

Please use *M5.begin()*; to initialize RTC related functions before using

typedef struct RTC_Time

```
    {
        int8_t Hours;
        int8_t Minutes;
        int8_t Seconds;
        RTC_Time() : Hours(),Minutes(),Seconds(){}
        RTC_Time(int8_t h,int8_t m,int8_t s) :
Hours(h),Minutes(m),Seconds(s){}
    } RTC_TimeTypeDef;

    typedef  struct RTC_Date
    {
        int8_t WeekDay;
        int8_t Month;
        int8_t Date;
        int16_t Year;
        RTC_Date() : WeekDay(),Month(),Date(),Year(){}
        RTC_Date(int8_t w,int8_t m,int8_t d, int16_t y) :
WeekDay(w),Month(m),Date(d),Year(y){}
    } RTC_DateTypeDef;
```

SetTime();

Syntax:

void SetTime(RTC_TimeTypeDef RTC_TimeStruct)*

Description:

Set RTC Time Struct

GetTime();

Syntax:

void GetTime(RTC_TimeTypeDef RTC_TimeStruct)*

Description:

Get RTC Time Struct

SetDate();

Syntax:

void SetDate(RTC_TimeTypeDef RTC_DateStruct)*

Description:

Set RTC Date Struct

GetDate();

Syntax:

void GetDate(RTC_TimeTypeDef RTC_DateStruct)*

Description:

Get RTC Date Struct

Example:

```
#include "CoreInk.h"

Ink_Sprite InkPageSprite(&M5.M5Ink);

RTC_TimeTypeDef RTCtime;
```

```
    RTC_DateTypeDef  RTCDate;

    char timeStrbuff[64];

    void flushTime(){
        M5.rtc.GetTime(&RTCtime);
        M5.rtc.GetDate(&RTCDate);

        sprintf(timeStrbuff,"%d/%02d/%02d %02d:%02d:%02d",
                        RTCDate.Year,RTCDate.Month,RTCDate.Date,
RTCtime.Hours,RTCtime.Minutes,RTCtime.Seconds);

        InkPageSprite.drawString(10,100,timeStrbuff);
        InkPageSprite.pushSprite();
    }

    void setupTime(){

      RTCtime.Hours = 23;
      RTCtime.Minutes = 33;
      RTCtime.Seconds = 33;
      M5.rtc.SetTime(&RTCtime);

      RTCDate.Year = 2020;
      RTCDate.Month = 11;
```

```
  RTCDate.Date = 6;
  M5.rtc.SetDate(&RTCDate);
}

void setup() {

    M5.begin();
    if( !M5.M5Ink.isInit())
    {
        Serial.printf("Ink Init faild");
        while (1) delay(100);
    }
    M5.M5Ink.clear();
    delay(1000);
    //creat ink refresh Sprite
    if( InkPageSprite.creatSprite(0,0,200,200,true) != 0 )
    {
        Serial.printf("Ink Sprite creat faild");
    }
    setupTime();
}

void loop() {
  flushTime();
  delay(15000);
}
```

shutdown()

function overload-1:

Power off and wake up again via PWR button

void shutdown();

function overload-2:

Turn off the power and wake up the device via RTC at the end of the delay based on the number of seconds of incoming delay.

int shutdown(int seconds);

function overload-3:

Turn off the power, pass in the RTC time structure that specifies a certain point in time, and wake up the device via RTC when the **hours, minutes**, and **seconds** of that time are met.

int shutdown(const RTC_TimeTypeDef &RTC_TimeStruct);

function overload-4:

Turn off the power, pass in the RTC time structure specified for a certain point in time, and wake up the device by RTC when the **week number, day number,** and **time** of that point in time match at the same time.

int shutdown(const RTC_DateTypeDef &RTC_DateStruct, const RTC_TimeTypeDef &RTC_TimeStruct);

Example:

```
#include "CoreInk.h"
void setup()
{
    M5.begin();
    digitalWrite(LED_EXT_PIN,LOW);
    M5.update();

    M5.M5Ink.clear();
    delay(500);
}

void loop()
{

    if( M5.BtnPWR.wasPressed())
    {
        Serial.printf("Btn %d was pressed
\r\n",BUTTON_EXT_PIN);
        digitalWrite(LED_EXT_PIN,LOW);
        M5.shutdown(5);
        //M5.shutdown(RTC_TimeTypeDef(10,2,0));
```

```
    }

    //M5.rtc.SetAlarmIRQ(RTC_TimeTypeDef(10,2,0));

    M5.update();
}
```

Examples and Demos

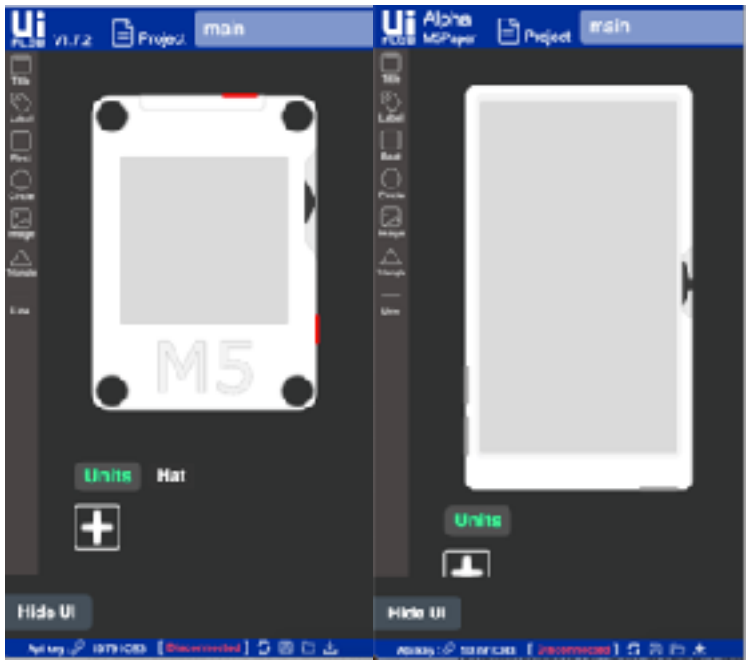
Example 1 - Hello world

There are two main basic programs new users are taught when learning to use micro-controllers. In this example I will show you how to print "Hello World" on the CoreInk and M5Papers screen. using three different programming environments.

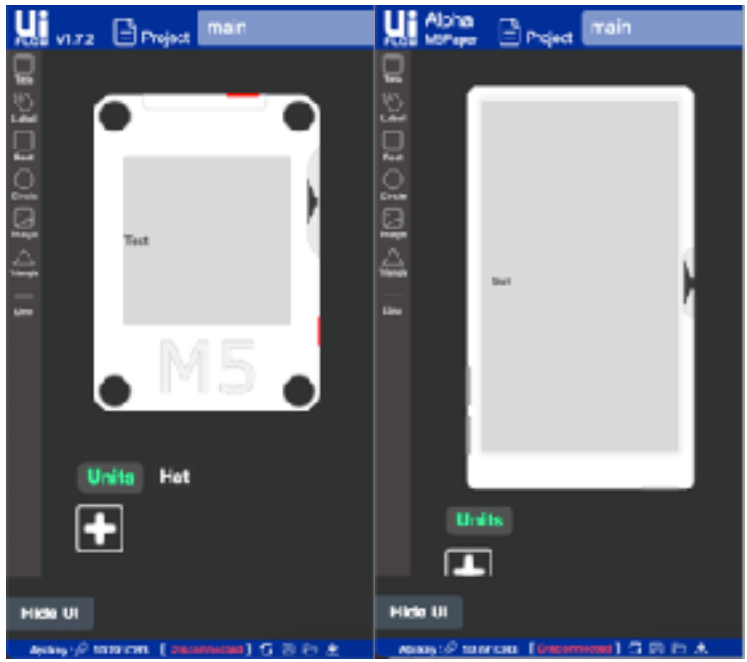
Example 1.1 Hello World UIFlow

Creating the Hello World program in UIFLow is the easiest of the three methods as it involves dragging and dropping code blocks and filling in the values. All we need to do is:

Step 1 - Start UIFlow, set the CoreInk or M5Paper in the settings along with each devices api code.



Step 2 - From the menu on the left, click on the label and drag it to the middle left of the screen.

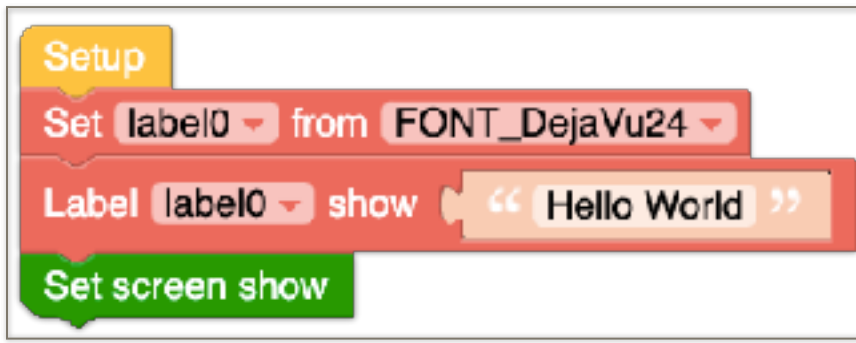


Step 3 - Next click on the "Label" menu in the middle of the screen to bring up the label blocks.

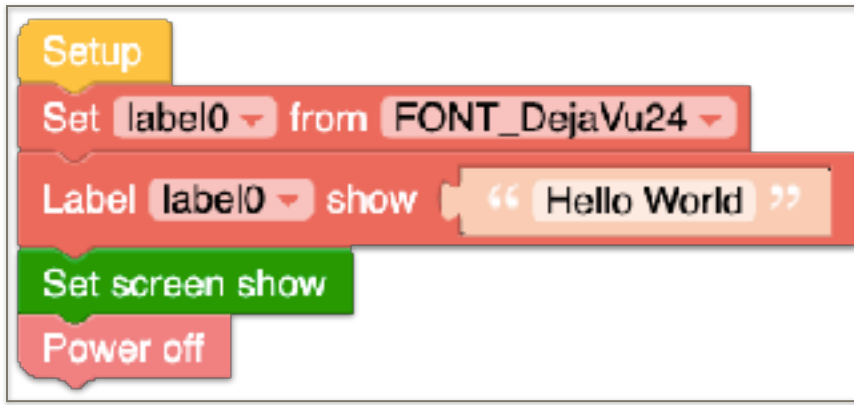
Step 4 - Click on the Set Font block to place it on in the programming area and drag it to the setup block and change the setting to FONT_Dejavu24.

Step 5 - Click on the Label menu again and now click on the Label Show " " block to add to the screen under the set font block and change "Hello M5" text to Hello World.

Step 6 - In order for the text to be shown on the screen we need to use the Screen Show block. When we send interface items to an E-Ink device, the data is stored "offscreen" in memory. Click on the Screen block to open the screen menu and click on the Set Screen Show block to add it to the program under Label show block.



Step 7 - Now that we have hello world displayed on the screen, we no longer need to have the CoreInk or M5Paper powered on. E-ink screens have the benefit of retaining items on screen even when powered off. to switch the CoreInk or M5Paper off, click on the Hardware block to open the hardware menu and then click on the Power block to open the power menu. Click on the Power off block and place it under the Set Screen Show block.



Step 8 - Upload to CoreInk or M5Paper and your done.

If you press the play button the text will be shown on the screen but devices will not power off. Click on the the setting menu in UIFLow and click on Download to perminatly save it to the CoreInk or M5Paper.

The CoreInk or M5Paper will restart and run the now stored program will be run from the internal memory and on the CoreInk the green LED on the top will go out to show the CoreInk has powered off.

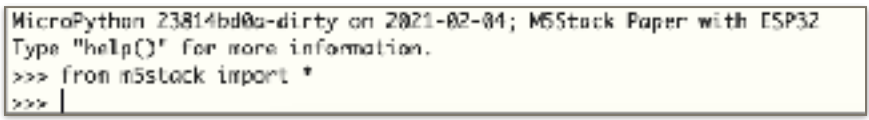
Example 1.2 Hello World Micropython

While UIFlow programs are built on micropython and we can cheat by building the program and switch into the Micropython IDE for the code, in this version I will show you how to create the program in an IDE called Mu https://codewith.mu

Once Mu has been opened, the CoreInk will be able to be connected directly to REPL however on the M5Paper, you need to switch to APP MODE in order to connect to REPL to start coding.

Step 1 - First wee need to import the libraries that contain the command we need to make a program. In UIFlow these libraries are automatically added to the programs as they are needed.

Type in ***from m5stack import ** and hit return and REPL will look like nothing has happened.



This is normal but it has accepted the command. Next we need to import the other libraries, again pressing return after each line.

from m5ui import *

from uiflow import *

import time

Step 2 - Next we have to set the screen colour to white The default colour to in order for the text to show in the default colour of black.

This is done with the following line

setScreenColour(15)

This clears the screen ready for the next step

Step 3 - Add a label with the following command

label0 = M5TextBox(58, 449, "Hello World", lcd.FONT_DejaVu24, 0, rotate=0)

Nothing will show on the screen until the next step.

Step 4 - To make Hello world appear of screen use

lcd.show()

This make the CoreInk or M5Paper transfer the message from memory to the screen.

Step 5 - Unlike the CoreInk, the screen on the M5Paper draws a bit too quick. In order for Hello World to remain on screen after power down we need to add a delay. Delays are added with the following code.

wait(5)

This pauses the processor from moving onto the next command.

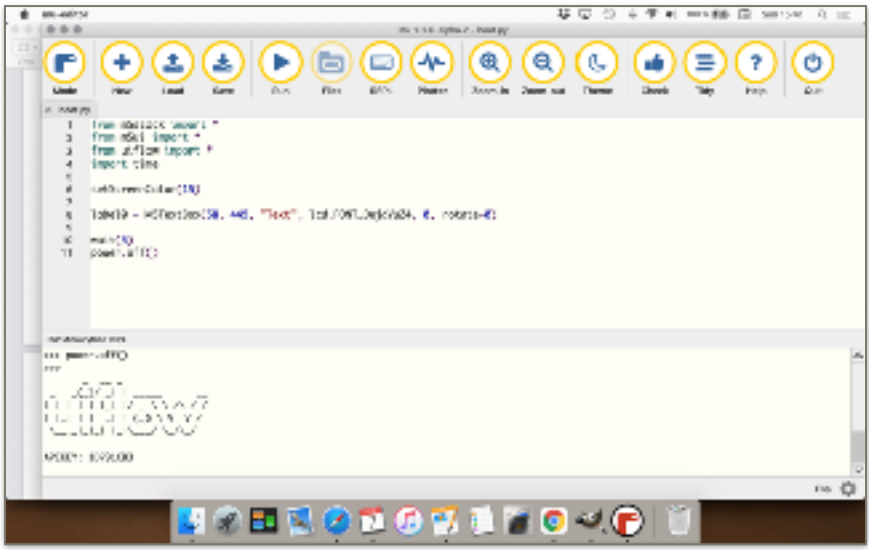
Step 6 - To power down the CoreInk or M5Paper we use

power.off()

and the CoreInk or M5Paper will switch off.

If we restart the M5Paper or CoreInk, the program will be erased from memory and forgotten. In order to make the program

that stays stored on the devices we need to turn the command into a program. This is done by copying the commands into the top panel of MU.



Next click on the Save button to save the program as helloworld.py and then change the CoreInk or M5Papers operating mode to USB mode.

Click o the REPL icon to close it and then the files icon to open the files saved on the devices.

Example 1.3 Hello World Arduino

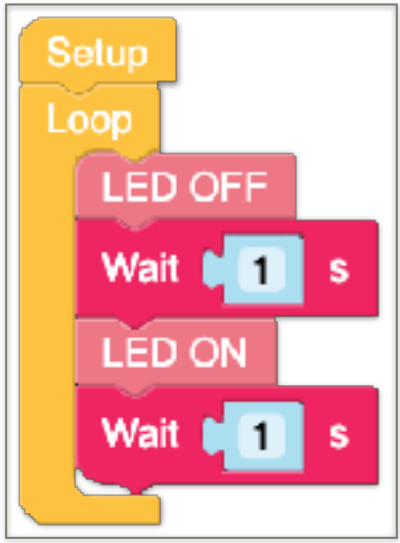
Example 2 - Blink

The second basic example taught to beginners in the "Blink" example. This involves flashing an LED on and off. Only the CoreInk has an unbuild LED so I will show you how to flash that

LED then how to connect and external LED to both the CoreInk and M5Paper and make it flash using the three different programming environments.

Example 2.1 Blink UIFlow

The CoreInk unlike the M5Paper has a green LED on the top edge near the HAT connector. This can be controlled using this simple program



This is all well and good for a simple LED but the M5Stack has some more interesting LED items in the form of plug in units.

For the blink example, we will be using the HEX RGB LED unit.

The HEX RGB Led unit contains 37 SK6812 programmable LEDs in a small hexagonal tile.

Step 1 - To use the HEX RGB we use a small cable and plug it into Port A on the CoreInk or M5Paper.

Step 2 - Next, open UIFlow and under the CoreInk or M5Paper image, click on the big "+".

Example 2.2 Blink Micropython

Example 2.3 Blink Arduino

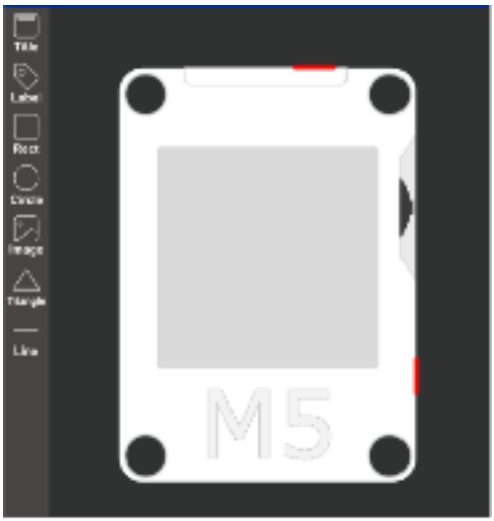
Example 3.1 - Electronic Name Tag

Example 3.2 - Electronic Price Tag

In this example I will show you how to make a simple electronic price tag. This is very similar to the Name Tag but has the ability to be upgraded to updated over WIFI and retain its display even when powered off.

The Graphical interface.

To build the Graphical interface for the electronic price tag I am using the GUI designer found in UIFLow. When UIFlow is first started and set to CoreInk, we are presented with an empty screen.



Step 1

First we drag a rectangle onto the screen then click on it to open the preferences.

Change the settings to the following:

- Name: Background
- X Position: 0
- Y Position 0
- Width: 200
- Height:50
- Border Colour: Black
- Background Colour: Black
- Layer:5

I have used the Rectangle element here because the Title element is restricted in size.

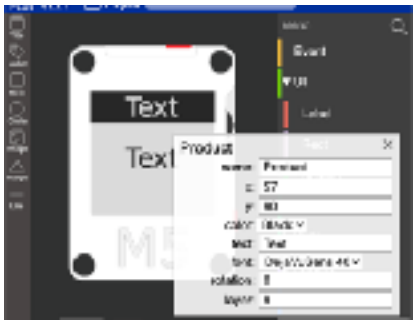
Step 2

Next add a label with the following settings.

- Name: Manufacturer
- X Position: 57
- Y Position: 8
- Colour: White
- Text: Text
- Font: DejaVuSans 40
- Rotation: 0
- Layer: 8

Step 3

Next we add another Label which will be used to hold the name of the product.

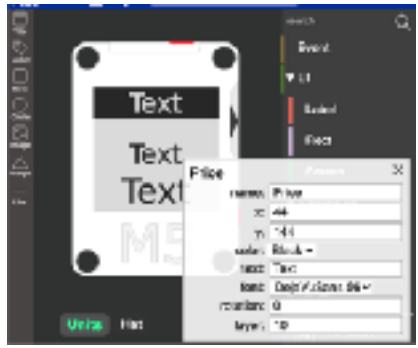


With the following settings:

- Name: Product
- X Position: 57
- Y Position: 90
- Colour: Black
- Text: Text
- Font: DejaVuSans 40
- Rotation: 0
- Layer: 9

Step 4

Add a last Label



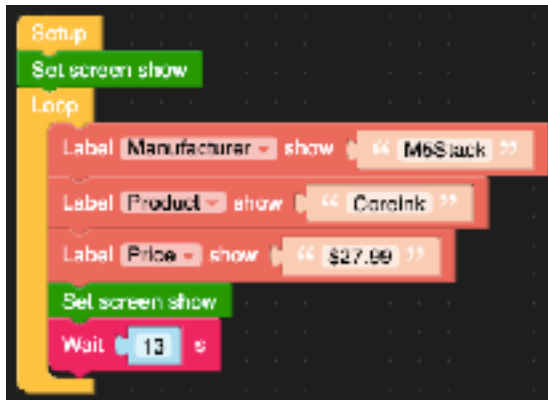
With the following Setting.:

- Name: Price
- X Position: 57
- Y Position: 90
- Colour: Black
- Text: Text
- Font: DejaVuSans 56
- Rotation: 0
- Layer: 10

The "Text" that appears in the labels is place holder text. when we come to create the code, this text is replaced and only the new text set in code will be shown on the CoreInk's screen. The layers are also user definable but, I have left them at their default values as set in UIflow.

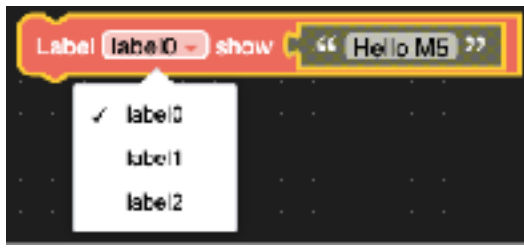
The Code.

To make the CoreInk show the information we wish to display, we need to use the following UIFlow code.



The Set Screen Show placed between the Setup and Loop blocks is used to wipe the screen of any existing information.

Next we have the three Label blocks. If we hadn't renamed the Labels earlier the Label block would show:



Changing the the names of the UIFlow Labels to something more descriptive allows is to see which label show which text value.

After the Labels we have another Set Screen Show block that is used to write the text in our blocks to the screen. Because the screen has a slow refresh rate of fifteen seconds we need to add a delay of thirteen second which with the two second loop delay gives us the refresh rate of fifteen seconds needed to avoid damaging the display.

This is a very simple electronic price tag but because it lays down the basic foundations, we can extend it to add additional functions.

For example, We could set up an MQTT server which handles request from CoreInks for price updates and then the server would query a database for the latest price, send it to the CoreInk which would then update with the new price.

Hardware Accessories

As of writing there is only one official hardware add-on released for the CoreInk and none for the M5Paper. It this section I will show you this add-on and one possible example on how to use it.

The CoreInk ProtoBase.

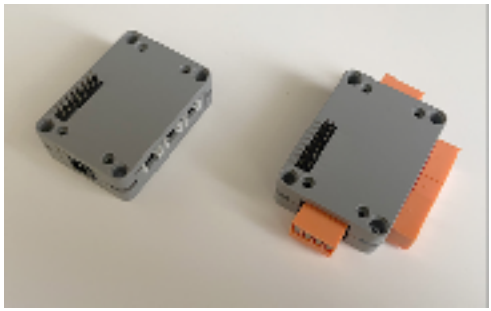
The CoreInk Proto Base is a case that has been designed to fit under the CoreInk with the same physical footprint.

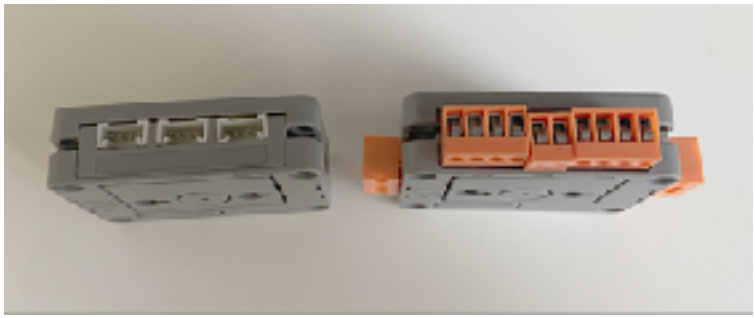
The base back contains the following components:

- Top and Bottom Case parts,
- PCB with MBus connections pre soldered.
- A power jack,
- 3x Grove ports (1 black, 1 red and, 1 blue)
- 3 Removable surround sections of the case,
- 4x HT3.96R 4Pin connectors,
- 1x HT3.96R 2Pin connector.
- 4x M2*6mm case crews
- Allen key/wrench for the screws.

The PCB

As mentioned previously, the pcb comes with the 16 pin (duel 8P pin) Mbus connector pre soldered in place but, it also has spaces to use the power jack and Grove connectors or the HT3.96 screw connectors. placement of the soldered point means that you can not use all the connectors at the same time and in order to make the case close.





In the above photo's you can see that I am using white Grove connectors instead of the coloured connectors as a project I am working on requires analog inputs only.

Bonus Section

Using the Seeedstudio Grove triple colour displays with M5Stack.

While the E-Ink screens in this section are not manufactured or endorsed by M5Stack, It is worth adding them as they make use of the four pin "Grove" connector found on M5Stack controllers.

Seeedstudio produce two grove comparable screens, the 1.54" and the 2.13" screens. The also produce a 2.7" screen but they are not compatible with M5Stack.

Datasheets.

Appendix 1
Data-sheet Catalogue.

This page is a catalogue of the known data sheets pertaining to modules used in the CoreInk and M5Paper products. Please note: while every attempt is made to keep this list up to date, due to the ever changing electronics environment, the list may become inaccurate as data-sheets are posted online or are removed.

Good Display GDEW0154M09

EPD_ED047TC1

GT911 Multitouch Controller

ESP32 D0WDQ6

ESP32 PICO D4