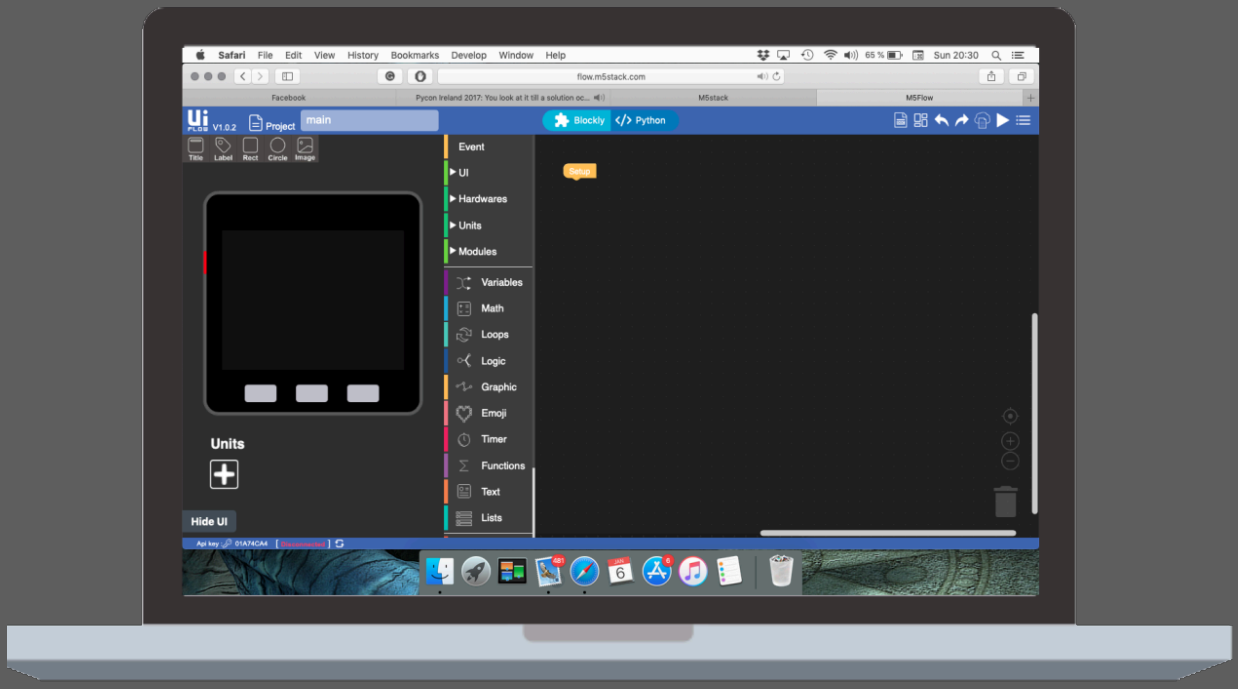
UIFlow Reference
V1.4.4

Written By
Adam Bryant



Introduction

UIFlow is the Integrated Development Environment (IDE) created by M5Stack and is the third incarnation of the programming environment.
In various documents references are made to M5Cloud and Moments, these are the two previous versions and are no longer available.

UIFlow contains two environments. Blocky, the default environment is a basic programming aimed at the young and beginners to programming. Micropython, the second environment is a text based environment that is at the core of all the firmware and functions of blocky. Once a programmer has grown beyond blocky, they can use the Micropython environment to dig deeper in to the functions and develop new blocks for Blocky or program the M5Stacks and M5Sticks at a lower level.

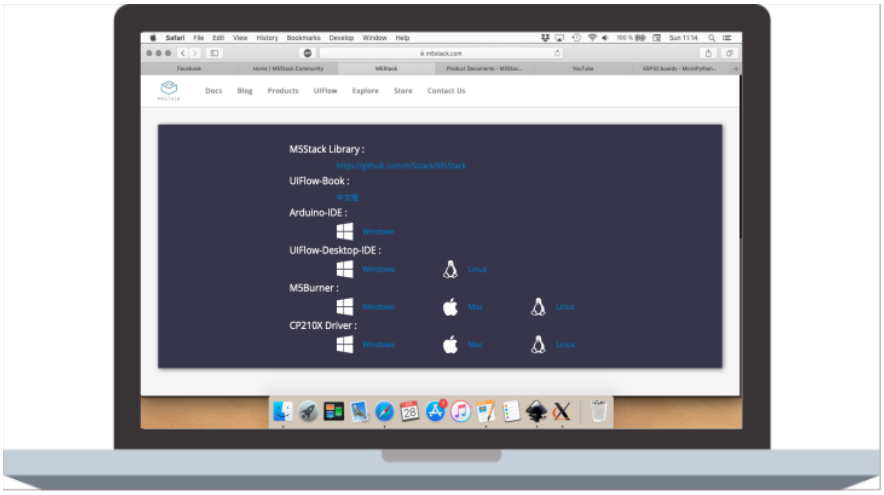
Blocks in UIFlow can be separated into three groups, actions, loops and values.
Actions are the blocks that contain command to preform tasks.
Loops are used for code that needs to be repeated.
Values provide numbers that can vary or be defined by the programmer.

I hope that this section will explain what the various blocks available in the UIFlow environment are. In another section I will show examples on how to use the various blocks to make things work.

UIFlow Setup

Installing the UIFlow Firmware.

To use UIFlow on the M5Stack devices and to also update UIFlow to the latest firmware versions you need to follow the following procedure that works on all M5Stacks and M5Sticks. First we need to go to the M5Stack homepage found at https://www.m5stack.com/
At the top of the screen you will see the title bar. Click on **Explore** and then **Downloads** from the menu that drops down and, you will be taken to this screen.

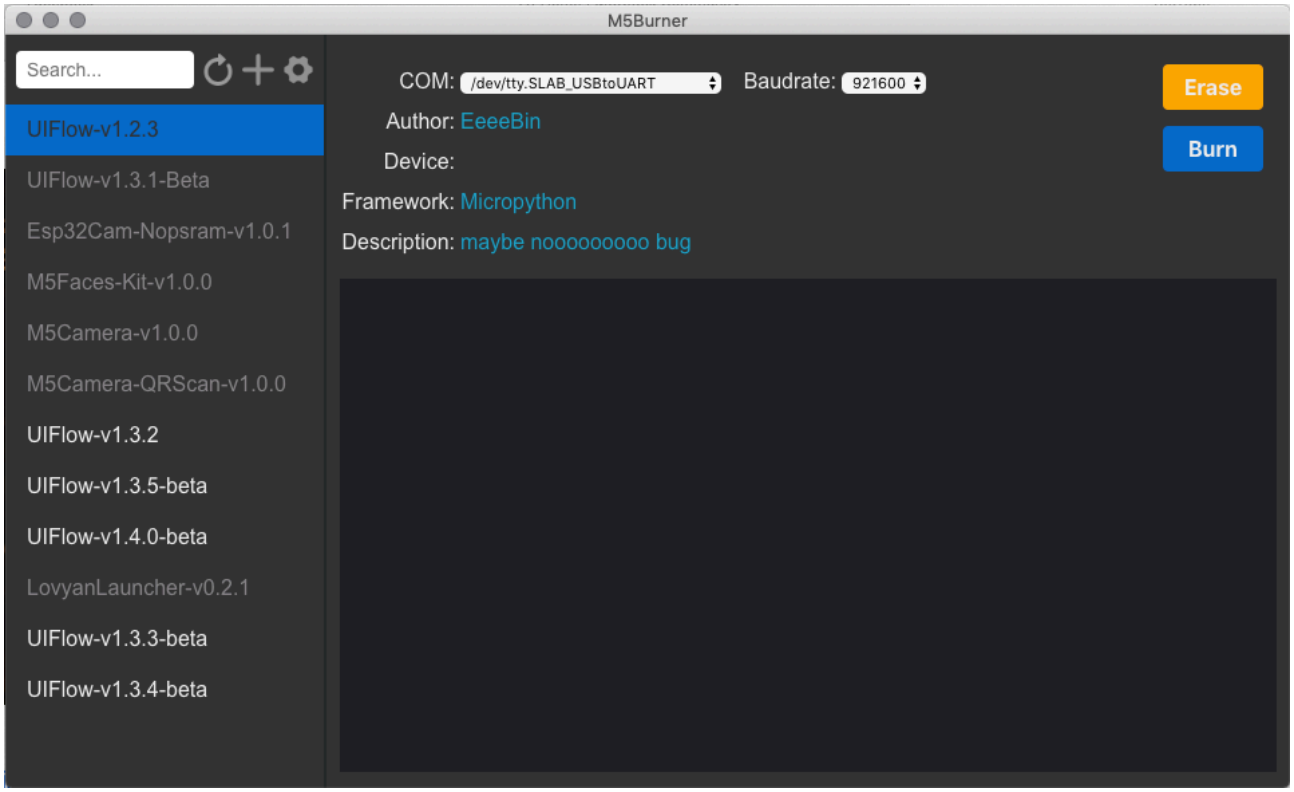


If you don't have the **CP210X** driver installed that click on that and install it before proceeding. The **CP210X** driver is needed for the computer to communicate with the M5Stack range of products. On OSX based computers there is an issue where it does not always install properly due to security issues.

In OSX there is a security issue that causes M5 burner to pop up a message saying it is corrupt. To get around this you need to open the command prompt and type
sudo spctl --master-disable
Download and run M5Burner (don't do anything else), Once finished you need to type
sudo spctl --master-enable
in the command line to reset security settings. I'm not sure what is causing the corrupt message but, disabling and re-enabling **spctl** just for this app has not caused me any problems.

Once the driver is installed you can download the M5Burner app for your computer which is needed to install and update firmware.
When M5Burner starts up you will be presented with an interface matching the screenshot on the following page. The image on the next page of the new M5Firmware burner shows list on the right that contains the current available firmwares while the buttons on the left allow you to wipe and reinstall the firmware. When changing firmwares or switching to the Arduino programming environment, it is wise to erase the existing firmware first. Make sure that the M5Stack or M5Stick is plugged in, select a firmware version (you can upgrade and downgrade firmware from M5Burner) in the column on the left and wait for it to download.

Select the port it has appeared as and and from the second drop down box select wether the connected device is an English M5Stack, Chinese M5Stack or and M5StickC. Once you have selected the correct device you can press "Burn". Burning a stack firmware on a stick or vice versa will not damage a device, it will just result in hardware not working as expected. The panel in the bottom right is a text box which show the progress of firmware erasing and installation. when the firmware has finished installing, the panel will show "Instillation complete, staying in



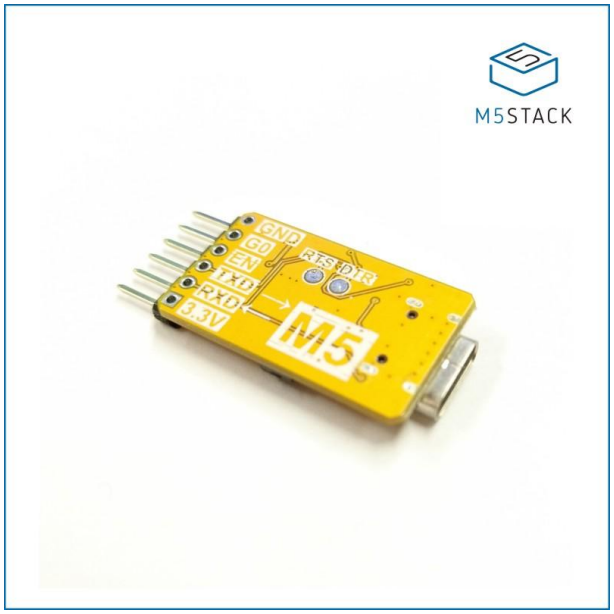
boot mode." wait about a minute, close down the burner program and then restart the M5Stack or M5Stick to start using the new firmware.

Sometimes the M5Stack or M5Stick will not appear on the computer as a port, Sometimes this is due to the driver not loading and sometimes it is down to a faulty lead. The first thing to check in this instance is the lead. The leads supplied come folded up which sometimes results in the conductors braking. when buying a new lead, always go for a high quality charge and sync cable and cheep cables often do not work.

If you have checked the lead is working with another non M5Stack device but it is still not working, then then next stage is to try to test on another computer. If it works on the second computer then the driver needs to be uninstalled and reinstalled on the first computer. **DO NOT uninstall and reinstall the driver without rebooting the computer between uninstall and reinstall and the faulty driver will remain in the computers memory. Turning the computer off, drains the memory and removes the faulty driver from RAM ready for a clean install.**

UIFlow Setup

Sometimes even after trying the above we still can't programme the M5Stack. When we get to this then we have one more thing to try. The M5Stack has a UART port broken out as a blue connector on some bases or as pins on the I/O base or the internal bus connector, (M5Sticks to do have access to the UART connections.) Using these pins we can bypass the internal USB to UART port and program using an Arduino in ISP mode or by using another USB to UART adapter. M5 Stack provide two such adapters that we can use.



To use these programmers we need to connect the TXD from the programmer to the RXD2 (GPIO16) of the M5Stack and RXD from the Programmer to TXD2 (GPIO17) of the M5Stack. GND of the Programmer connects to GND off the M5Stack and 3.3V or 3V3 of the programmer connects to 3V3 off the M5Stack. Once the connections have bee checked and rechecked to make sure they are correct, we can program the M5Stack or update the firmware as normal.

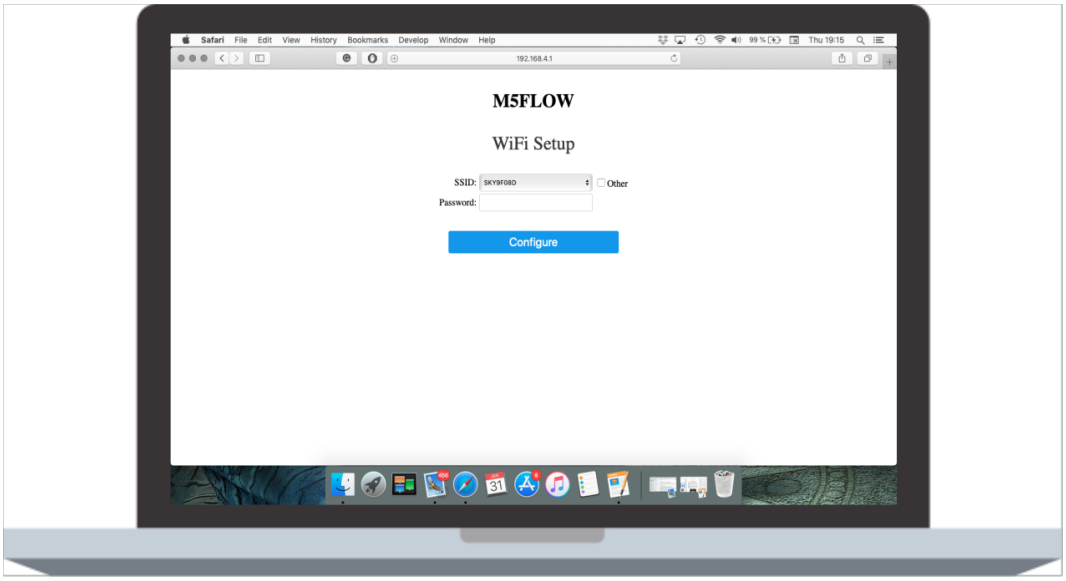
If none of the above work, then its time to contact M5Stack via the email:
support@m5stack.com

Connecting the M5Stack to UIFlow.

Press the red button on the side of the M5Stack to switch it on (assuming its charged but switched off.) the start screen will load up asking you to connect to the web address shown on the screen (the SSID changes every time the firmware is updated).

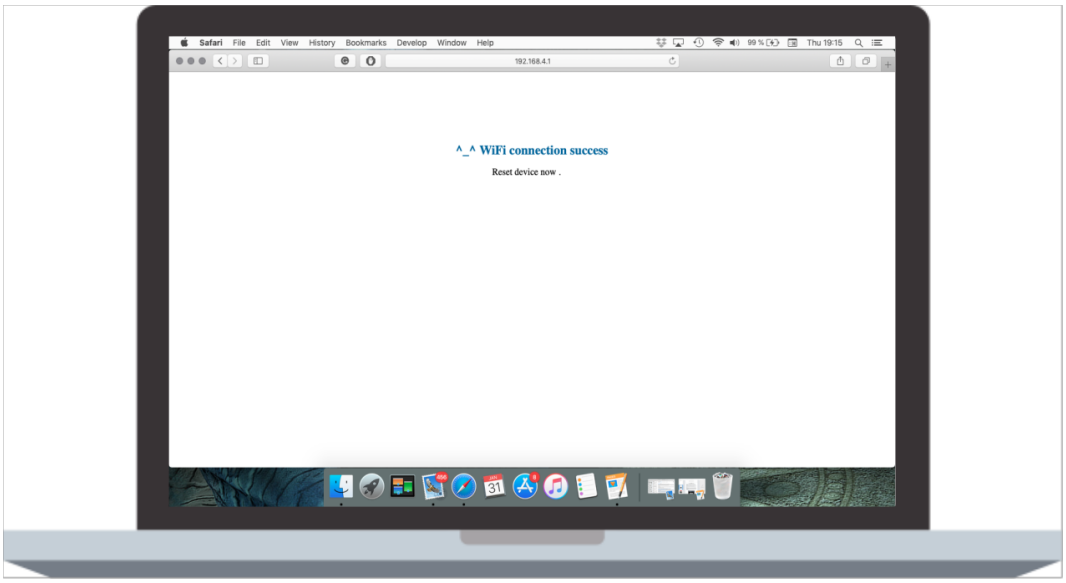
On your computer you need to go to the wifi connection, click on it to bring up the list of available wifi devices and click on the name that matches the screen of the M5Stack.

Once connected, you need to go to 192.168.4.1 which will bring you to the M5Stack wifi setup page.



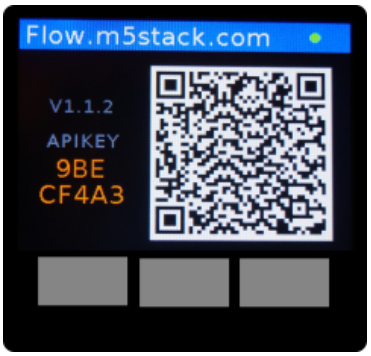
UIFlow Setup

Select your wifi network that your computer normally connects to, type in the password and wait for it to connect.



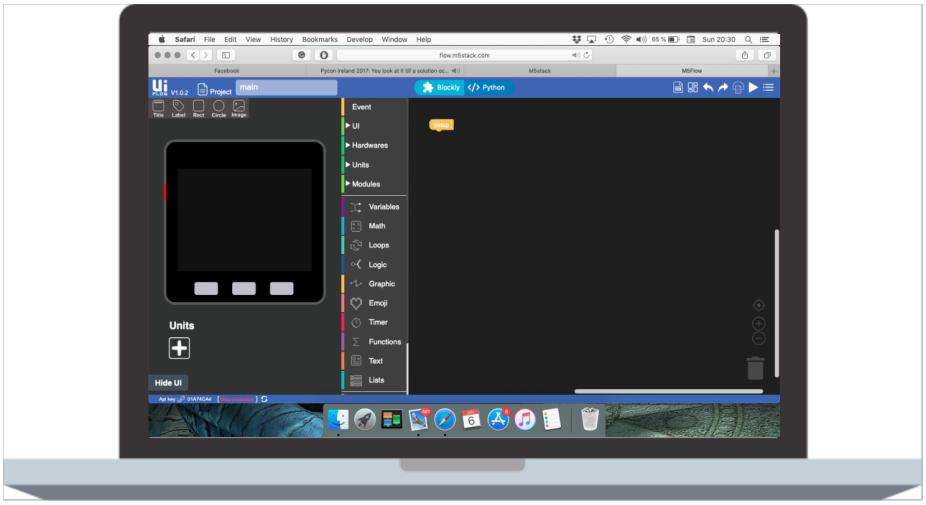
If it works, the webpage will say connection successful

The computer will disconnect from the M5Stack and reconnected to your normal wifi network. The M5Stack will now restart and bring up the UIFlow page for a few seconds and then attempt to connect to the wifi connection and UIFlow. the screen will change to show a code and a QR

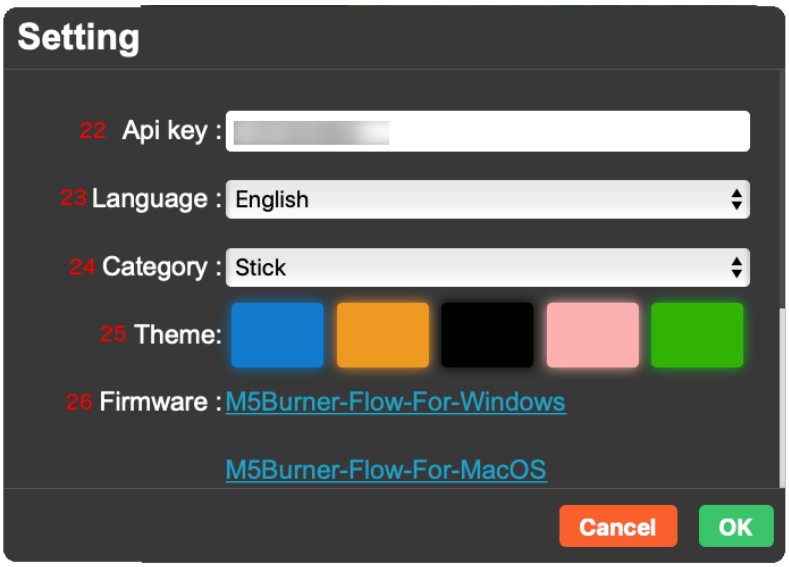


code. This is your API key and will be needed in the next step.

Go to flow.m5stack.com



Go back into the setting window and click on the API key box. (22),



Type in the code shown on the M5Stacks screen and then hit OK.
When you return to UIFlows main menu, look to the bottom left corner, it will show your API key and should have Connect in green.

My screen says DISCONNECTED and not Connected!
Don't Panic, look at your M5Stack and there should be a little green circle in the top right corner of the screen to show if it is connected or not, if its red it means it didn't connect to the internet, Restart the M5Stack and hit the right hand button to go into the WIFI menu, if it doesn't connect here, check your wifi settings and try again. Sometime another device on the network will stop it connecting, If the settings are correct, retry and it should connect.
Important note: Once in a while my own M5Stack refuses to connect if there are too many devices active on the network..

Event Blocks

The event folder contains the main basic functions that will be used by nearly all of our code in UIFlow. This folder contains the basic loop, buttons and timer functions.

Events

The set-up 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 blockly 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 *

setScreenColor(0x222222)
```

We can see in this code snippet that this block imports the three core libraries required by all programs ad set 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.

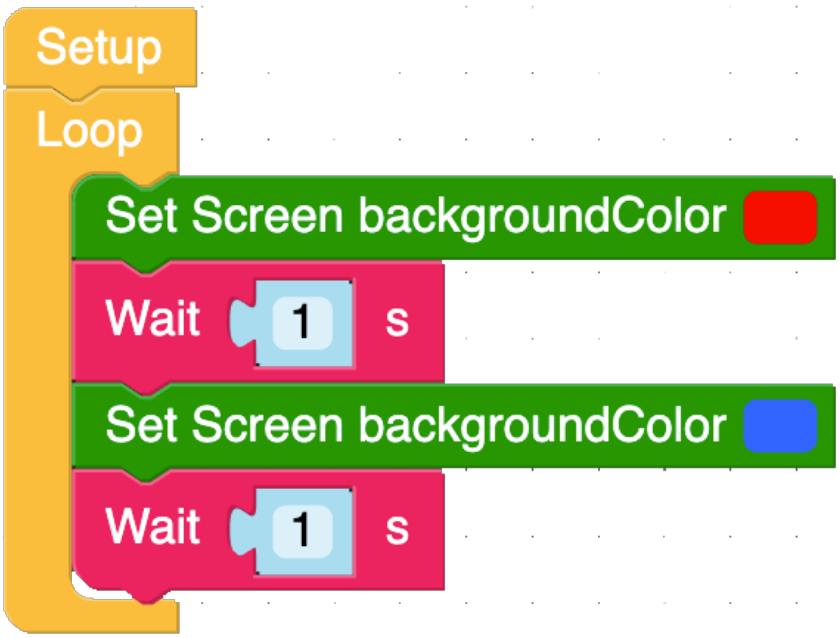
Event Blocks

Loop

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

Example

In the following example I am using a loop to continuously step through the SetscreenColour blocks. 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 I have added a wait block to make the program wait one second before moving between the blocks.

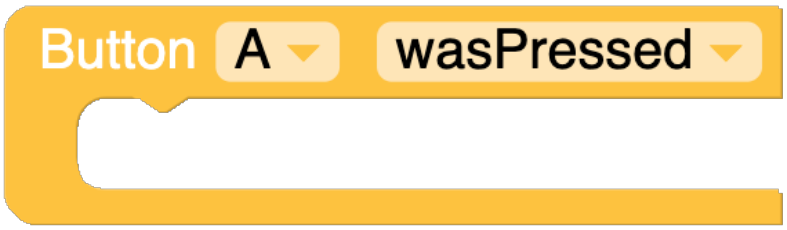


```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

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

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

Event Blocks

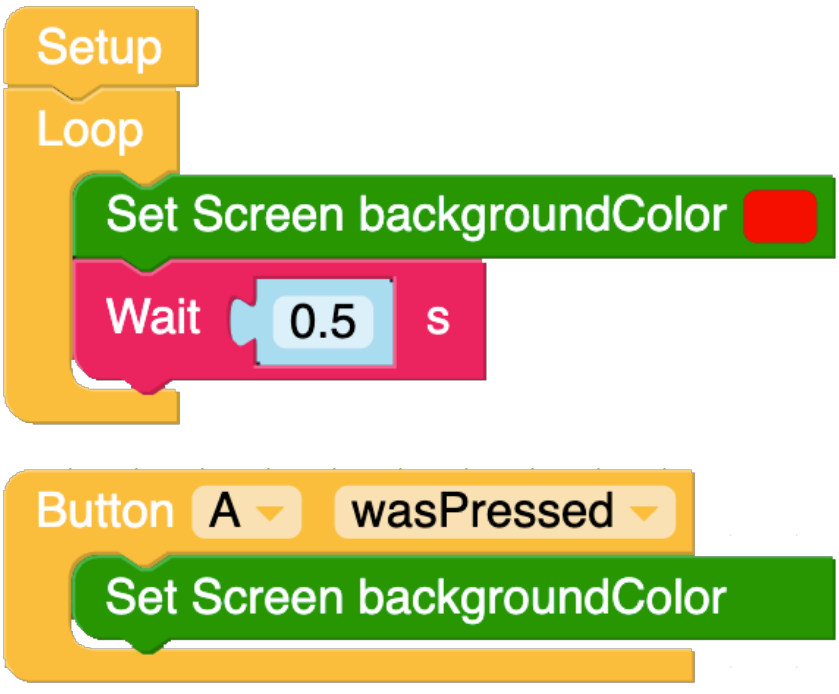
Button Loop



As well as the main loop, we have the button loop that continuously watches the three main buttons on the front of the M5Stacks and then runs the code inside it when it detects a triggering even.
The four events the button loop waits for 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.

Example

In this example I have two loops. The main loop set the screen to red and the button loop changes the screen to green when the button is pressed. The main loop is required to reset the screen colour after the button has been released.



Event Blocks

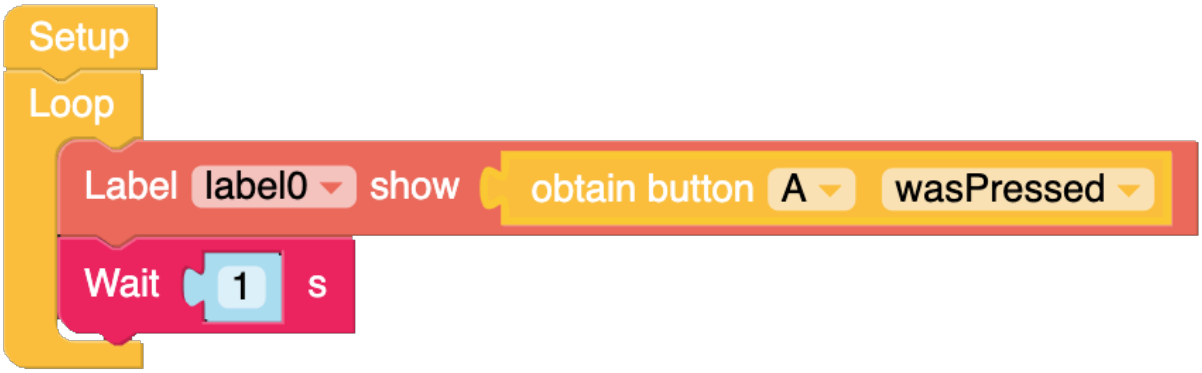
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

def buttonA_wasPressed():
  # global params
  setScreenColor(0x009900)
  pass
btnA.wasPressed(buttonA_wasPressed)

while True:
  setScreenColor(0xff0000)
  wait(0.5)
  wait_ms(2)
```

Event Blocks

Obtain Button Value,

The obtain button block works in a similar mode to the Button loop but instead of running code placed inside it, it is a value block that returns true or false when one of the button events have been triggered.
The four events the button value block waits for 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.

Example



In this example I am using a Label block to show the status of the buttons.

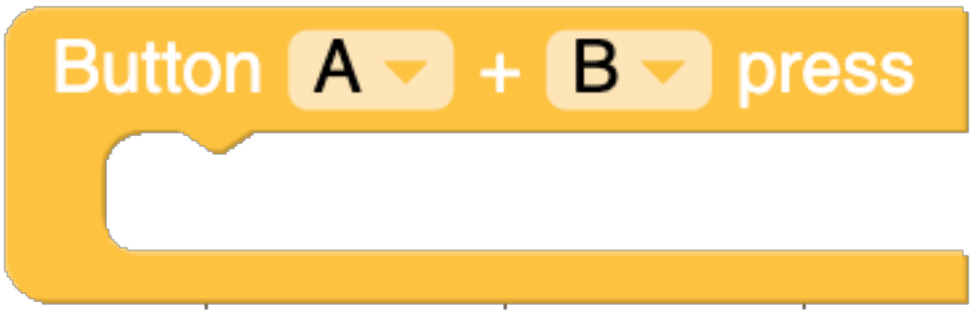
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

label0 = M5TextBox(95, 100, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(btnA.wasPressed()))
  wait(1)
  wait_ms(2)
```

```
def buttonA_wasPressed():
  global active, counter1
  active = False
  counter1 = 0
  label0.setColor(0xffffff)
  label1.setColor(0xffffff)
  setScreenColor(0x000000)
  hat_spk0.sing(831, 1)
  label0.setText(str(counter1))
  label1.setText('Clear')
  pass
btnA.wasPressed(buttonA_wasPressed)

@timerSch.event('timer1')
def ttimer1():
  global active, counter1
  if active == True:
    counter1 = (counter1 if isinstance(counter1, int) else 0) + 1
  label0.setText(str(counter1))
  pass

@timerSch.event('timer2')
def ttimer2():
  global active, counter1
  if counter1 > 60 and counter1 < 120:
    label1.setText('stage 1 alarm')
    setScreenColor(0xffff00)
    hat_spk0.sing(220, 1)
    label0.setColor(0x000000)
    label1.setColor(0x000000)
  if counter1 > 120 and counter1 < 180:
    label1.setText('stage 2 alarm')
    setScreenColor(0xff6600)
    hat_spk0.sing(448, 1)
    label0.setColor(0x000000)
    label1.setColor(0x000000)
  if counter1 > 180:
    label1.setText('stage 3 alarm')
    setScreenColor(0xff0000)
    hat_spk0.sing(889, 1)
    label0.setColor(0x000000)
    label1.setColor(0x000000)
  pass

@timerSch.event('timer3')
```

```
def ttimer3():
  global active, counter1
  if (imu0.acceleration[0]) > 1.9:
    if active == False:
      active = True
      label1.setText('Impact')
  pass


timerSch.run('timer1', 1000, 0x00)
timerSch.run('timer2', 1000, 0x00)
timerSch.run('timer3', 15, 0x00)
hat_spk0.sing(220, 1)
counter1 = 0
setScreenColor(0x000000)
label0.setColor(0xffffff)
label1.setColor(0xffffff)
label0.setText(str(counter1))
label1.setText('Clear')
active = False
```

Button A+B Press Loop

Button A+B press block expands on the function of the button loop by waiting until it detects that any two buttons on the front of the m5stack that have been selected in the drop down boxes

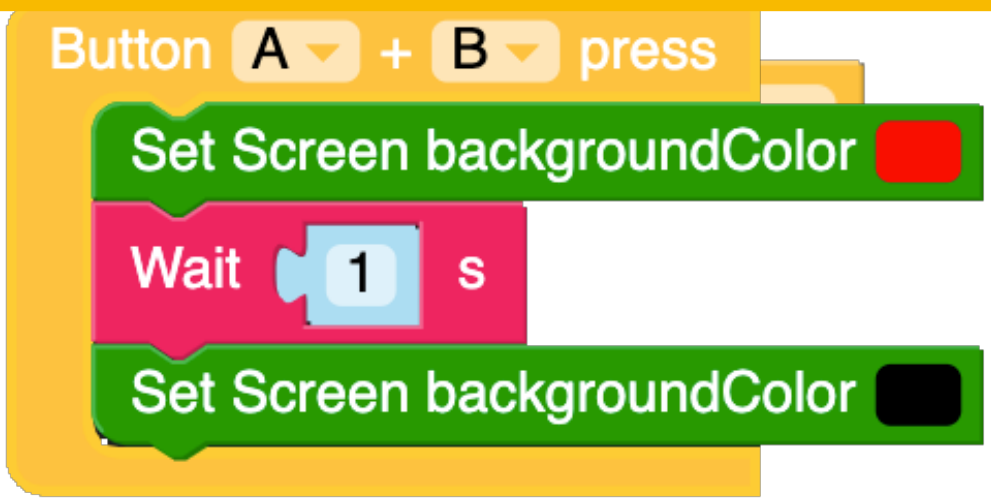


have been pressed at the same time.

Example

In the following example the screen will turn red when both A and B is pressed and then if the buttons have been released for more then a second, will turn black.

Event Blocks



Button Value

Similar to the Obtain Button function, this block also monitors buttons A, B, and C on the front panel of the M5Stacks but is simplified to only having options for Pressed and Released.

Example 1,


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

setScreenColor(0x222222)

label0 = M5TextBox(109, 88, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

def multiBtnCb_AB():
  # global params
  pass
btn.multiBtnCb(btnA,btnB,multiBtnCb_AB)
```

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

setScreenColor(0x222222)

label0 = M5TextBox(95, 100, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(btnA.isPressed()))
  wait_ms(2)
```

In the following example a label with show false until button A is pressed, the label will show true until A button is released.

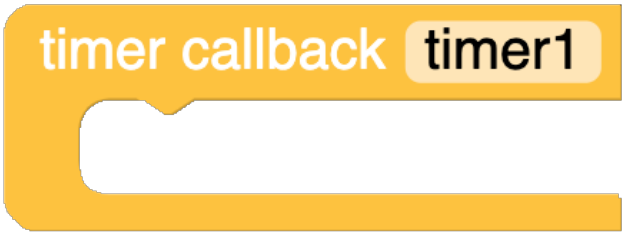
Example 2,


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

setScreenColor(0x222222)

label0 = M5TextBox(95, 100, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(btnA.isReleased()))
  wait_ms(2)
```

In this following example a label with show false while button A is pressed, the label will show true when A button is released.

Event Blocks.

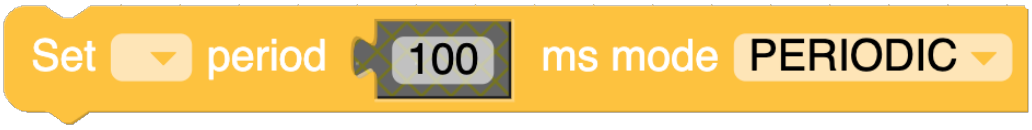
```
timerSch.run('', 100, 0x00)
```



Timers (Part 1)

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

Timer Callback Loop.



The timer callback look creates a timer that can contain actions that are set to run under certain time critical events.

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

The Micropython code for the loop is as follows:

Set timer Period

Defines the time the timer will run for.

Event Blocks.

The Micropython code for set timer is

Start Timer

Starts the timer for a defined period of time.
The Micropython code for start timer is:

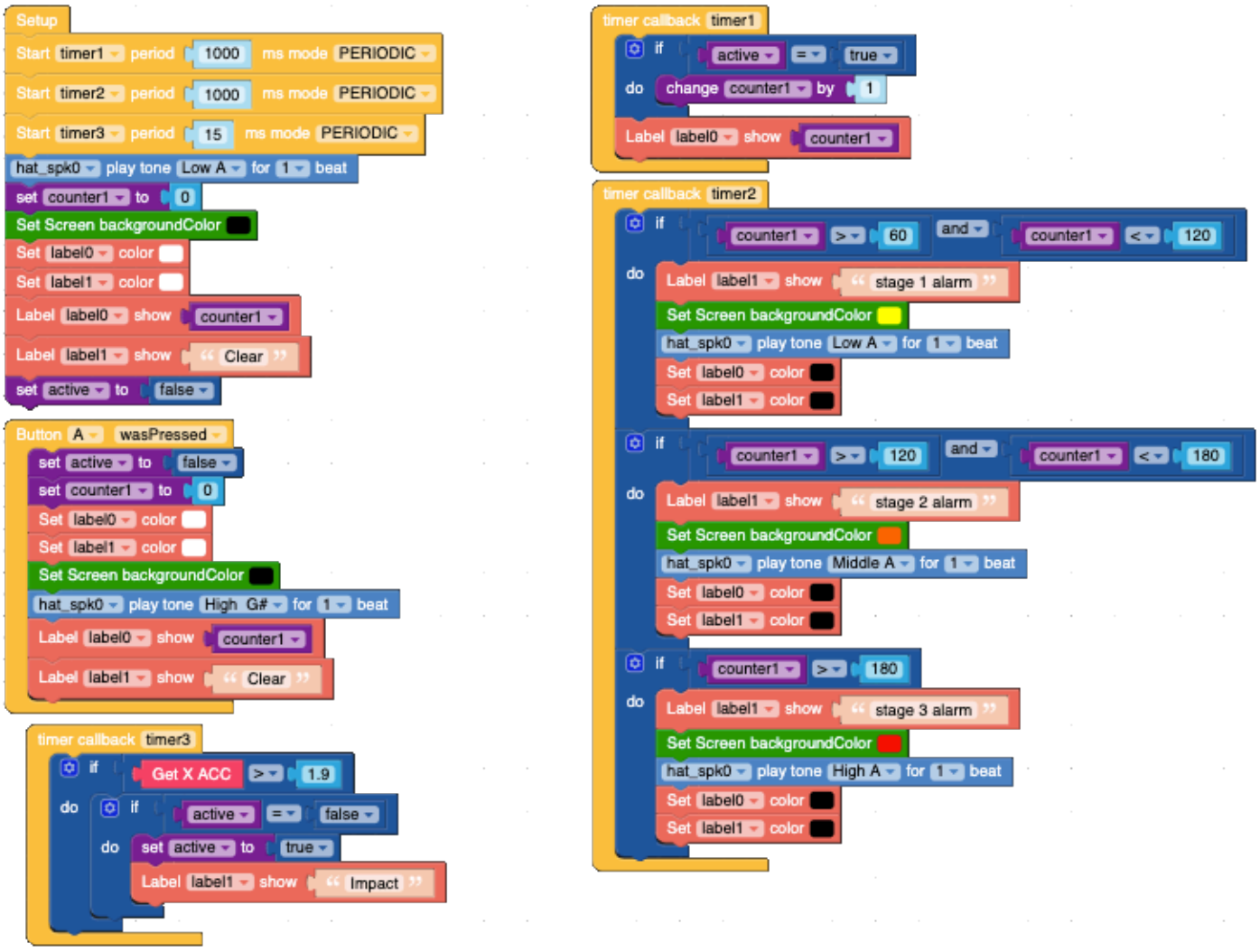
Stop Timer

Stops the timer selected in the drop down box.
The Micropython code for the stop block is:

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

Example.
The following example is taken from my M5Stick challenge entry and was developed with help from Herbert Stiebretz.
In this example, three timers have been created and set to start and the beginning of the program. with timer three also getting triggered by signals from the M5Sticks internal IMU. Timer 2 contains three conditions that are triggered when the timer reaches set conditions. Button A is used to reset the timer.

Event Blocks.



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 imu
import hat

setScreenColor(0x111111)

hat_spk0 = hat.get(hat.SPEAKER)
imu0 = imu.IMU()
label0 = M5TextBox(10, 91, "Text", lcd.FONT_Default,0xFFFFFF, rotate=270)
label1 = M5TextBox(45, 133, "Text", lcd.FONT_Default,0xFFFFFF, rotate=270)

active = None
counter1 = None
```

Event Blocks.

Virtual User interface Designer and Graphic Functions.

The M5Stack and M5Stick models have a selection of text and graphic functions defined for us. Some of these functions share the same code blocks and some have their own unique blocks. In this chapter I will explore the blocks available and teach you how to use them.

User Interface Elements

Virtual User interface Designer and Graphic Functions.

The M5Stack and M5Stick core models have a selection of text and graphic functions defined for us. Some of these functions share the same code blocks and some have their own unique blocks. In this chapter I will explore the blocks available and teach you how to use them.

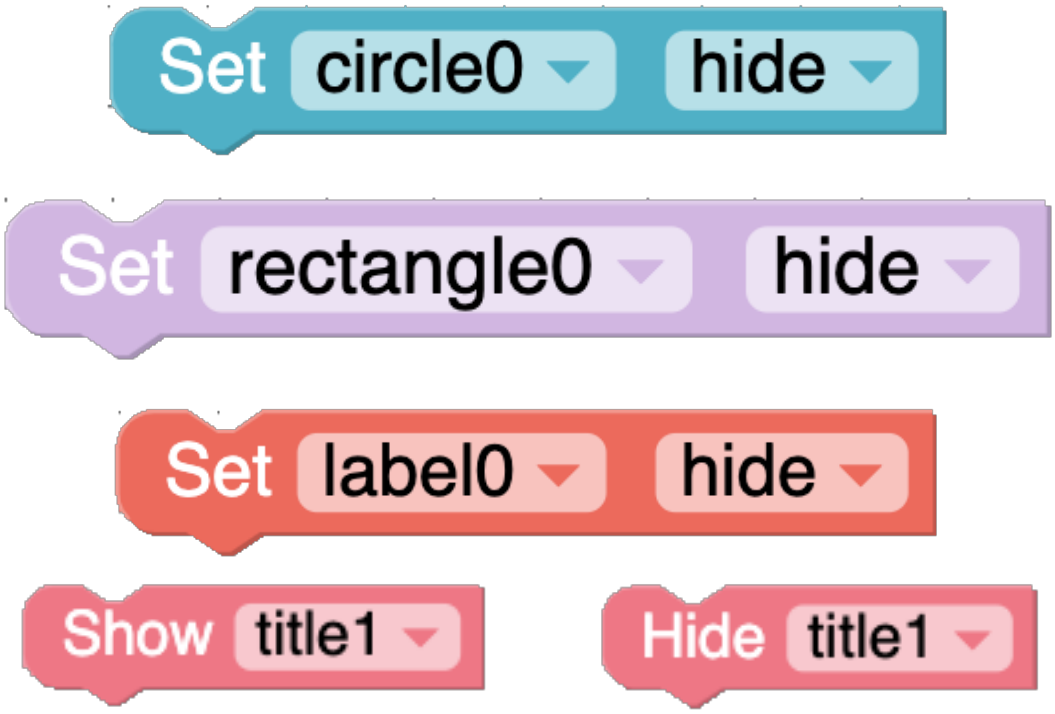
Functions are not the only things to share code. some of the hardware like the WS2812bs and RGB LEDs also share some of the code blocks.

Code Block	Title	Label	Rectangle	Circle	Image
Show Text	Yes	Yes	Yes	Yes	Yes
Show	Yes	Yes	Yes	Yes	Yes
Hide	Yes	Yes	Yes	Yes	Yes
Set Colour	Yes	Yes	Yes	Yes	Yes
Set Colour RGB	Yes	Yes	Yes	Yes	Yes
Set Background Colour	Yes	Yes	Yes	Yes	Yes
Set Background Colour RGB	Yes	Yes	Yes	Yes	Yes
Set Width and Height	No	No	Yes	No	Yes
Set Width	No	No	Yes	No	Yes
Set Height	No	No	Yes	No	Yes
Set X and Y Position	No	No	Yes	Yes	Yes
Set X Position	No	No	Yes	Yes	Yes
Set Y Position	No	No	Yes	Yes	Yes
Set Radius	No	No	No	Yes	No
Available to M5 Stack	Yes	Yes	Yes	Yes	Yes
Available to M5 Stick	No	Yes	Yes	No	No

The table above shows some of the User Interface (UI) functions and which code blocks they share.
When used in conjunction with other functions we can create functions that can be as simple as displaying the value of a sensor or as complicated as you can imagine.
There are two Show blocks, the first will display the UI element defined on UI builder while the second takes a value from one off the sensors and changes the text to show that value.
Hide, hides the onscreen text.

User Interface Elements

The Show/Hide Blocks.



The Show/Hide Micropython code for the blocks

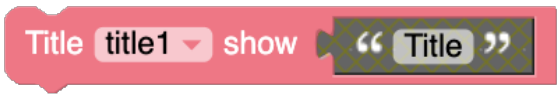
As you can see in the images above the Show/Hide block is only available to the Label,

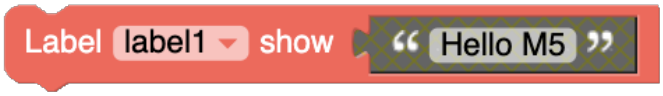
```
circle0.hide()
circle0.show()
rectangle0.hide()
rectangle0.show()
label0.hide()
label0.show()
title0.show()
title0.hide()
```

Rectangle and Circle UI elements. If you have more than one element on the screen you can use the block to control the individual elements by clicking on the arrow beside the element number. The second block is where you choose Hide or Show for the elements visibility. You can use this block to hide elements until certain event or conditions to happen and then set the individual elements to true in the condition or event function.
The Title UI element is slightly different in that its show and hide blocks are separate.

User Interface Elements

The Show Text Blocks





These two block are used to show or print a string of text onto the lcd screen. You can see that it has a puzzle shaped space in it that allows you to use the title and label blocks for showing values returned from sensors.
The Micropython code for these blocks are

```
title0 = M5Title(title="Title", x=3 , fgcolor=0xFFFFFF, bgcolor=0x0000FF)
label0 = M5TextBox(26, 85, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
label0.setText('Hello M5')
```

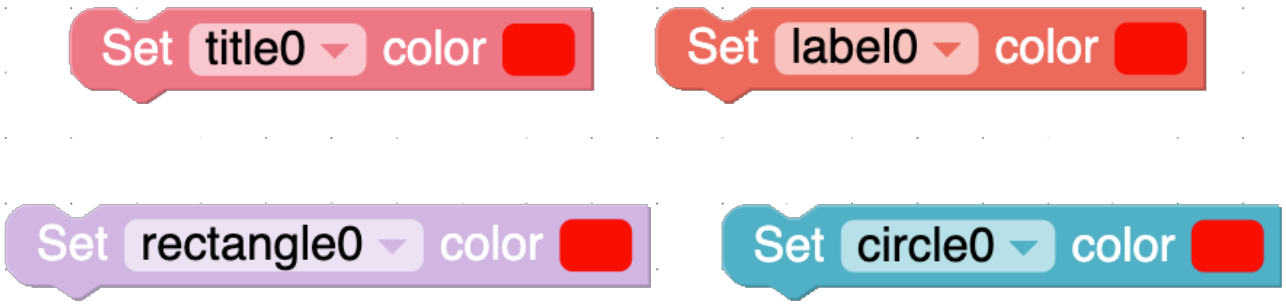
Set Screen Rotate Mode

Allows you to rotate the screen using one of the four preset modes shown below.
Mode 0 turns the screen 90 degrees clockwise,
Mode 1 is normal direction,
Mode 2 is 90 degrees anticlockwise,
Mode 3 is 180 degrees clockwise.
The Micropython code for these blocks are

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

User Interface Elements

Set Color



The Set Colour block allows you to set each elements colour using the colour picker. Set Screen backgroundColor set the screen or "canvas" colour independent of the other U.I Elements.
The Micropython code for these blocks are:

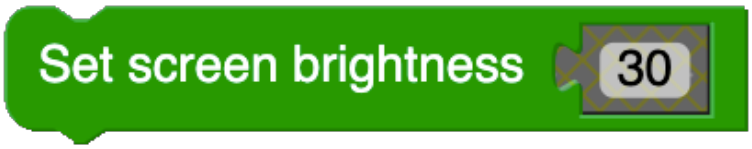
```
title0.setFgColor(0xff0000)
label0.setColor(0xff0000)
rectangle0.setBgColor(0xff0000)
circle0.setBgColor(0xff0000)
```

Set Screen Background Colour

Sets the background colour of the screen.
The Micropython code for this block is

```
setScreenColor(0x111111)
```

User Interface Elements



Set Screen Brightness

Sets the screens brightness using a value block.
The Micropython code for this blocks is

```
lcd.setBrightness(30)
```

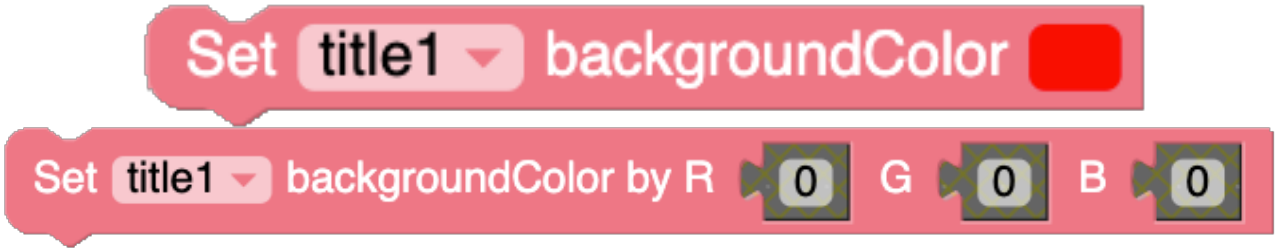
Set Colour R G B.

This Set Colour block allows you to set each elements colour by specifying individual value for each of the separate colour channels.
The Micropython code for these blocks are:

```
rectangle0.setBgColor(0x000000)
circle0.setBgColor(0x000000)
label0.setColor(0x000000)
title0.setFgColor(0x000000)
```

User Interface Elements

Set Title Background Colour



Only available to the Title UI element, the **Set backgroundColor** and **Set backgroundColor RGB** block allows us to change the colour off the title bar that appears at the top of the screen when the title element is visible. For more information on colour functions available in UIFlow read chapter 4.3.3.
The Micropython code for these blocks are:

```
title0.setBgColor(0xff0000)
title0.setBgColor(0x000000)
```

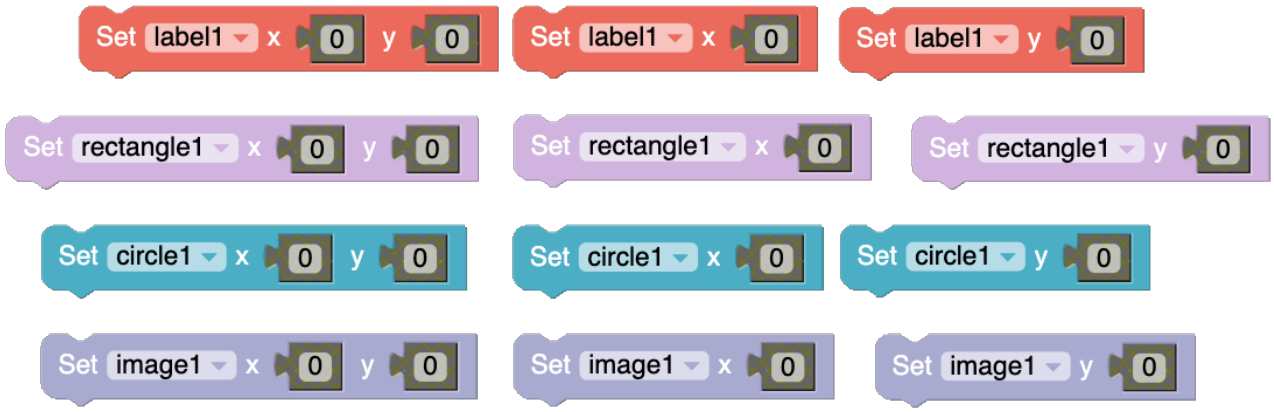
As you can see, while the blocks operate differently, they produce the same code in Micropython.

Set Width and Height

Available only to the Rectangle UI element, the set width and set height blocks are used to control the size.
The Micropython code for these blocks are:

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

Set X and Y



The set X and Y blocks are used to set the position of the UI element on the M5Stacks canvas using math blocks or variable blocks. By using variables as the positions we can move elements around like in games or animations.
The Micropython code for these blocks are as follows:

```
rectangle0.setPosition(0, 0)
rectangle0.setPosition(x=0)
rectangle0.setPosition(y=0)
```

As you can see, there are two different ways to set positions of objects in UIFlow. The first line combines the X and Y positions into one command whereas the next two lines have separate X and Y commands.

Set Circle Radius.

The Set Circle Radius block is only available to the circle UI element and can be set using maths block or variable blocks.
The Micropython code for this block is

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

Images.

You may have noticed from the demos shown previously that the M5Stack family of core units and sticks can display images on their screen. This is all well and good but there is a catch.
For images to be used they must be created to the specifications listed below.

- **.JPG** files must have the .**JPG** extension.
- **.JPG** file must have "Baseline" formatting.
- Progressive and Lossless JPEG format is not supported.
- Image size: Up to 65520 x 65520 pixels.
- Colour space: YCbCr three components only.
- Gray scale image are not supported.
- Sampling factor: 4:4:4, 4:2:2 or 4:2:0.

.**BMP** images are supported.

- Uncompressed only.
- RGB 24-bit.
- No colour space information.

These images can be stored on the ESP32 internal memory or onto and loaded from a microSD card placed in the built in card reader however, images stored on the SD card will take longer to be loaded and displayed due to the slower read time off the SPI bus.

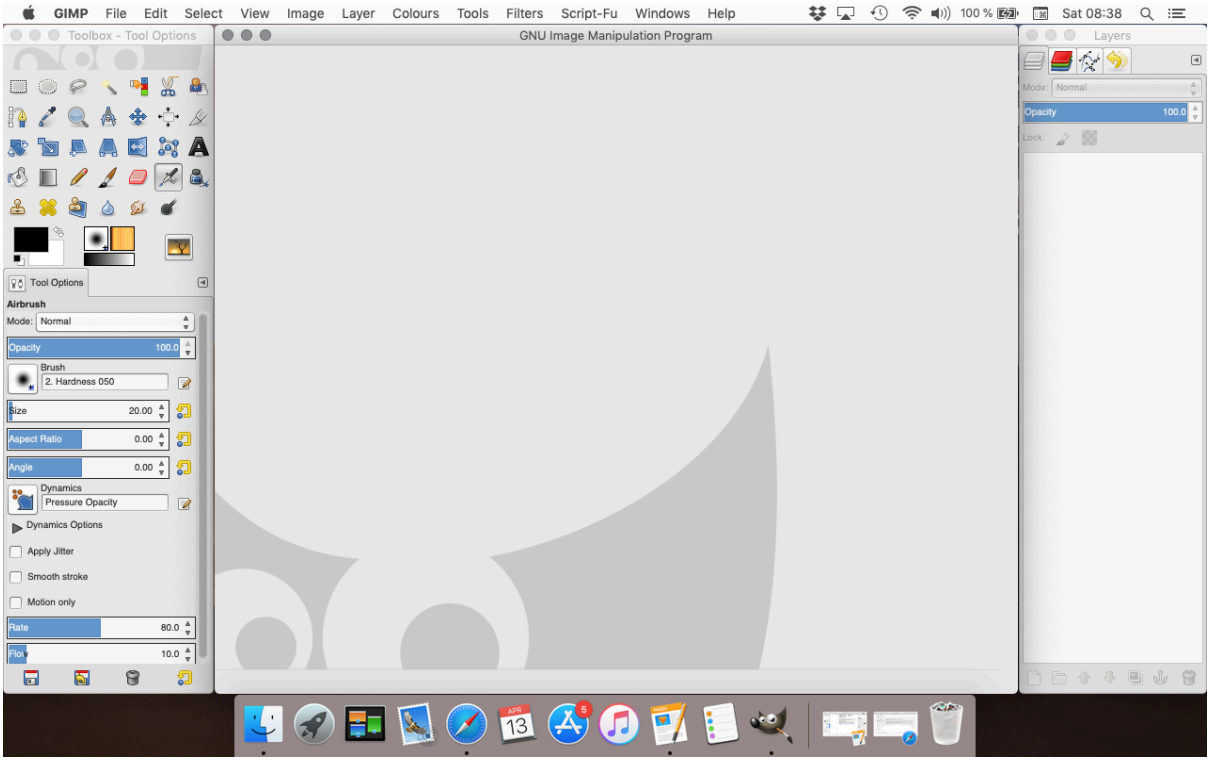
How do we create these files?

.

There are many image programs that can be used to create images but, not all of them can give us control over the more advance file formatting functions that are kept hidden in basic art programs.
Throughout this book I will be using the open source programs G.I.M.P and Inkscape (in fact, most of the images used in this book have been created and/or edited using G.I.M.P and Inkscape).

User Interface Elements

.BMP Files.



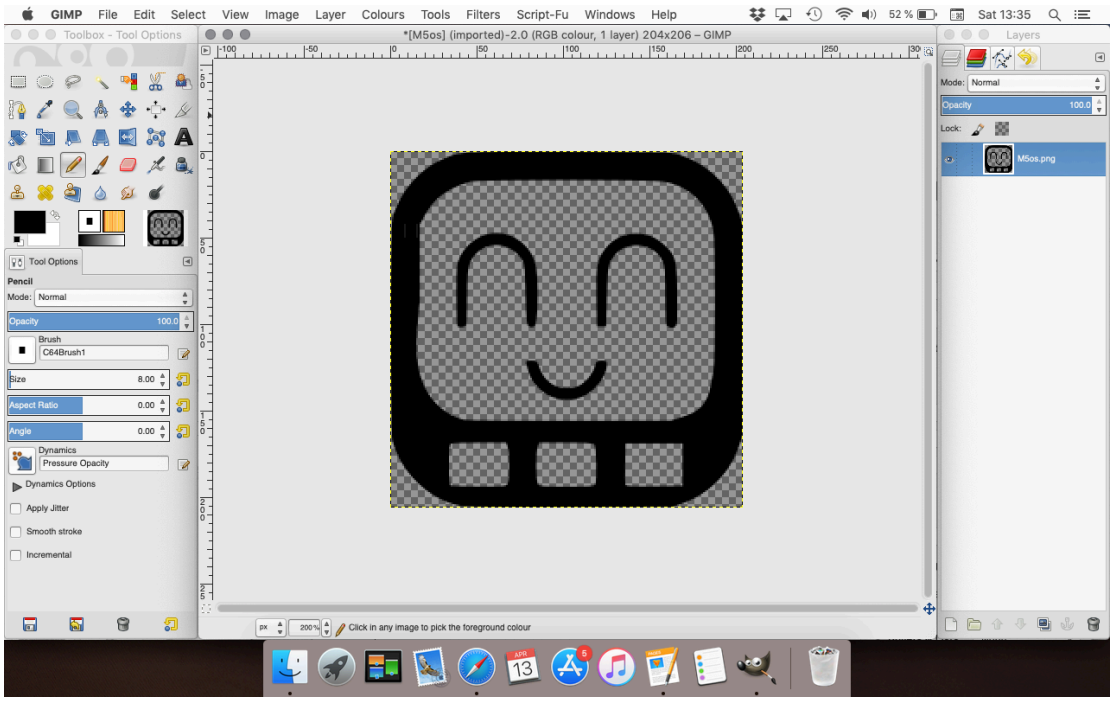
Screenshot of how G.I.M.P's windows are configured on OSX 10.14.2

I first became aware of G.I.M.P back in 1998 when it was still in Beta test and in the last twenty years has changed greatly into a program that rivals Photoshop. Being open source, G.I.M.P is free to download from https://www.gimp.org

The first time G.I.M.P is loaded up it may not look like the above screen shoot as it is made of three movable windows, I will not go deeply into the operation of G.I.M.P beyond the functions we need for this chapter as that is beyond the scope of this book.

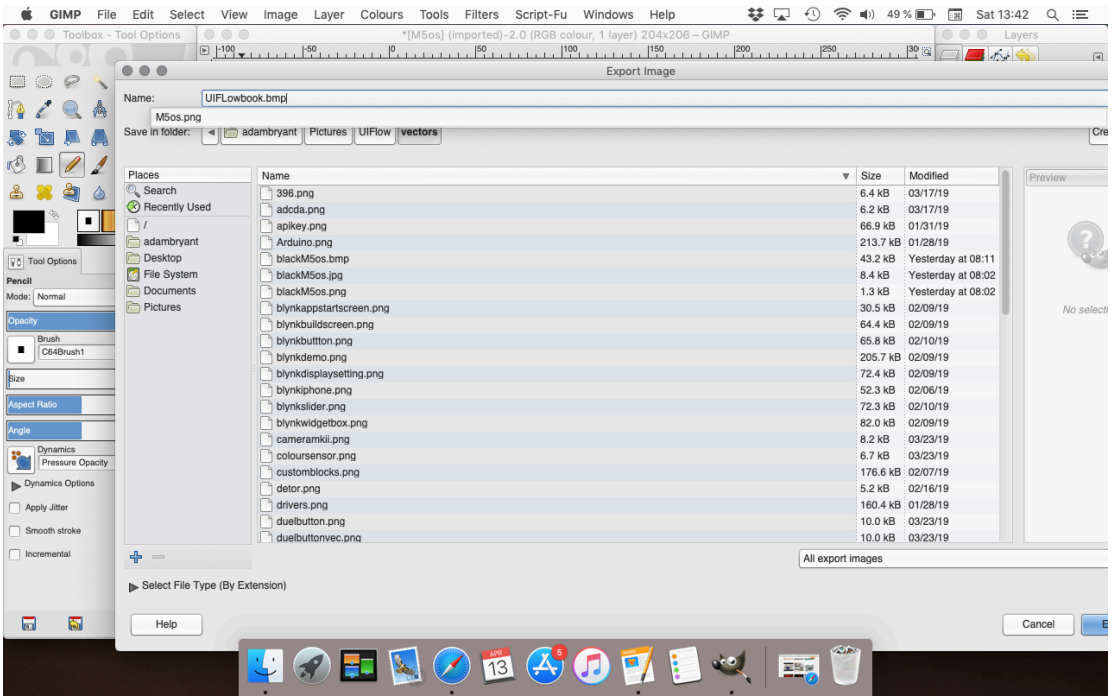
To get started, first we need an image.

User Interface Elements

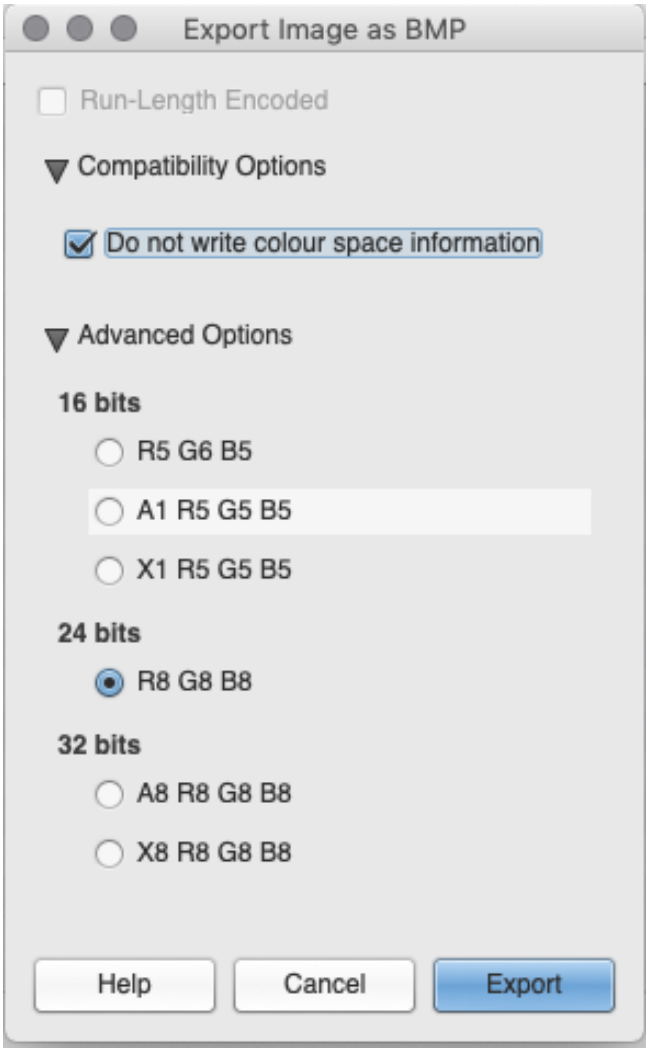


The Image I am using above has been provided by Luke from M5Stack.
When loaded up we can see that our image has a transparent background. The available file formats that Micropython can read do not at present support transparency and when we save the image the transparent background will turn white.

We don't need to do anything here except save the image. Goto **"File>Export As"** to bring up the export dialog.
If we just use **"File>Save AS"** the image will just be saved to G.I.M.P's native file format of **.XCF**.



The the Export Image dialogue at the top of this window is the Name: box. Type in a filename under ten characters long and put **.BMP** after the name. Hit return and the file format dialog appears.

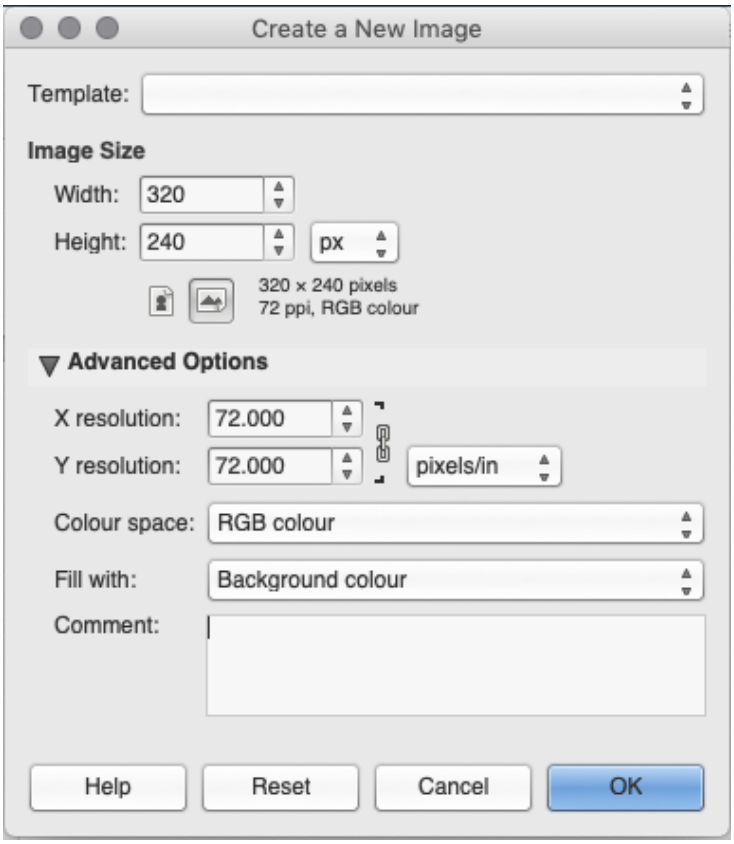


In this window click on the arrow next to **Compatibility Options** and a box will appear with a tick box next to **Do not Write colour space information.**
Click the box to make an arrow appear in it and then click the **Advanced Options** arrow and the above options will appear. Click on the circle under **24 bits** and then click on export to save the file.
If you look at the file in the file manager it will show a file size of **124Kb** this is way over the limit we have of **24Kb** that Micropython can read.
To make it smaller we need to to some editing of the file.
For this go too, **Image>Scale image** and change the image size from **204x206px** to **102x103px** and click Ok. Export the file again and check the file size.
This step has reduced the size from 124Kb down to 32Kb but, it is still too big for Micropython.
Goto **Image>Mode** and select **Grayscale,** Save it again and we can see that the file size is now only 12kb.

If we upload the file new file to the M5Stack and try to load it noting will happen because grayscale mode just doesn't work. Go back to I**mage>Mode** and select and change it back to **24bit mode**. The only thing we have left is to scale the image down a little bit more and report it until we can get the file size low enough.

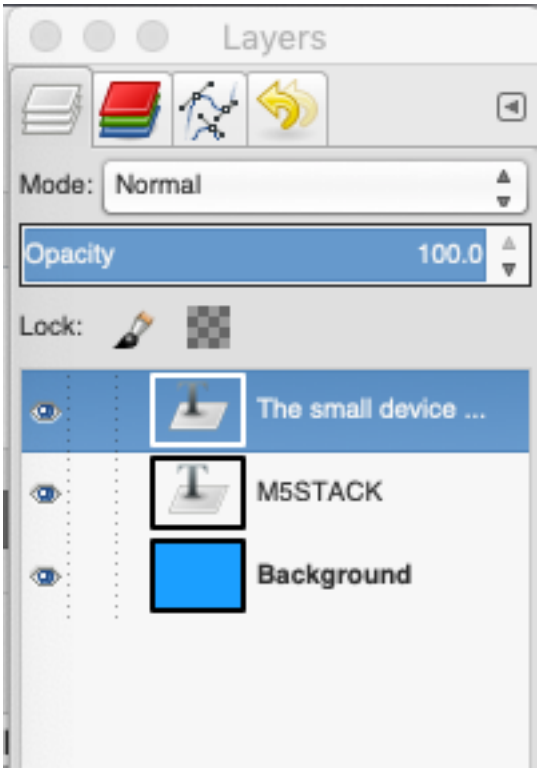
.jpg files.

Creating a .jpg file is almost the same as creating the .bmp files but we have a few more steps to follow. In G.I.M.P go to **File>New Image** and set the option to the same as in the screen shot



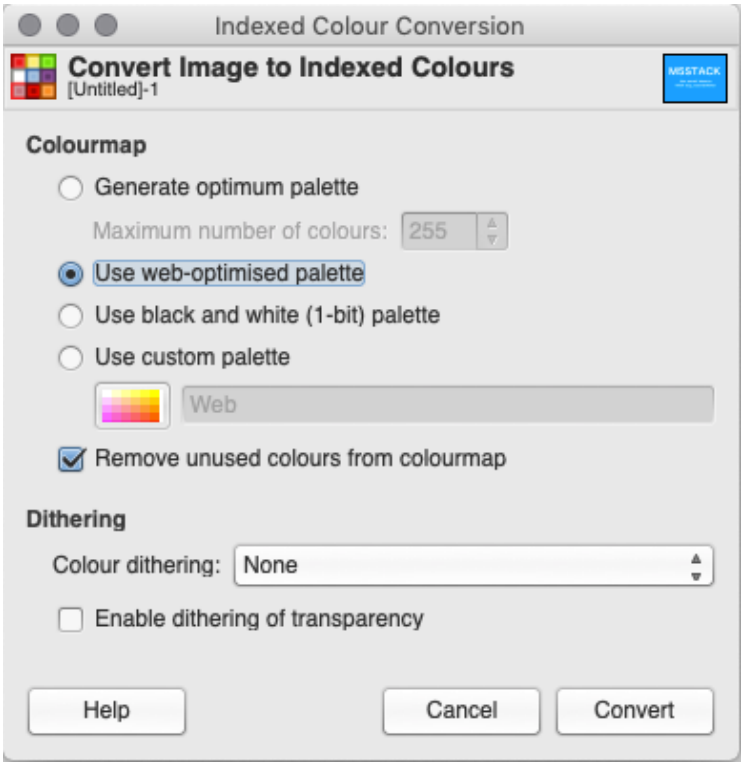
below.

Next fill the image with a design. In the next image I filled the background in blue and just added the text in white.



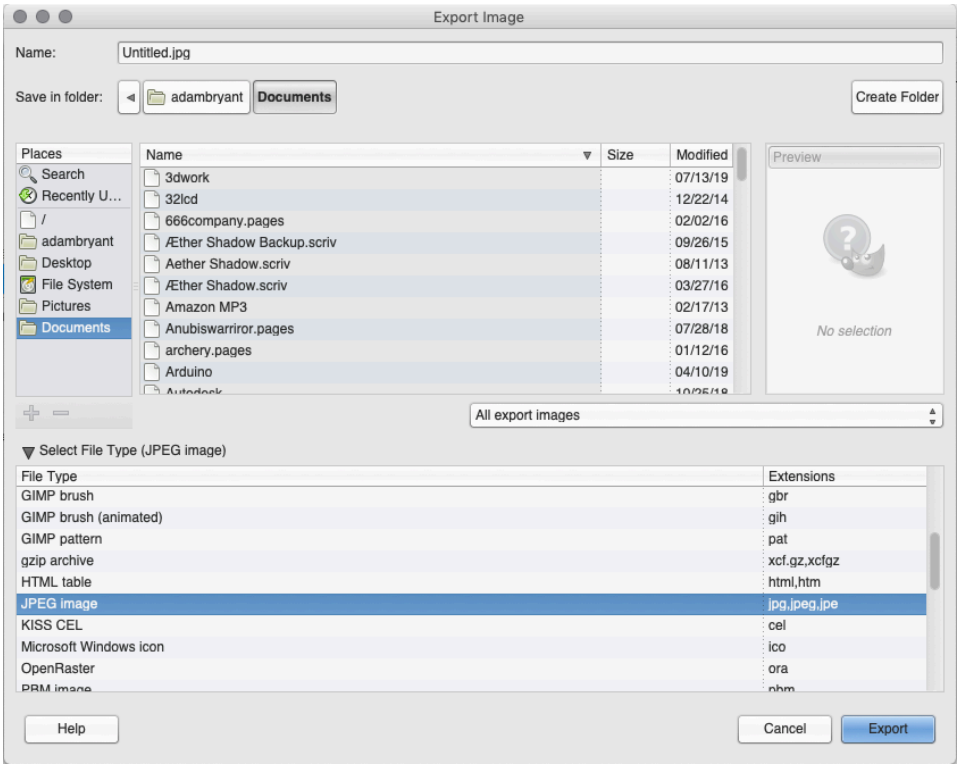
Next look at the Layers panel and you will see that we have three layers.

The .jpg file format does not support layers and so we need to flatten the image into one layer. Go to **Layer>Merger Down** to merge the top layer with the one below it. Repeat the merge down so that we only have one layer.

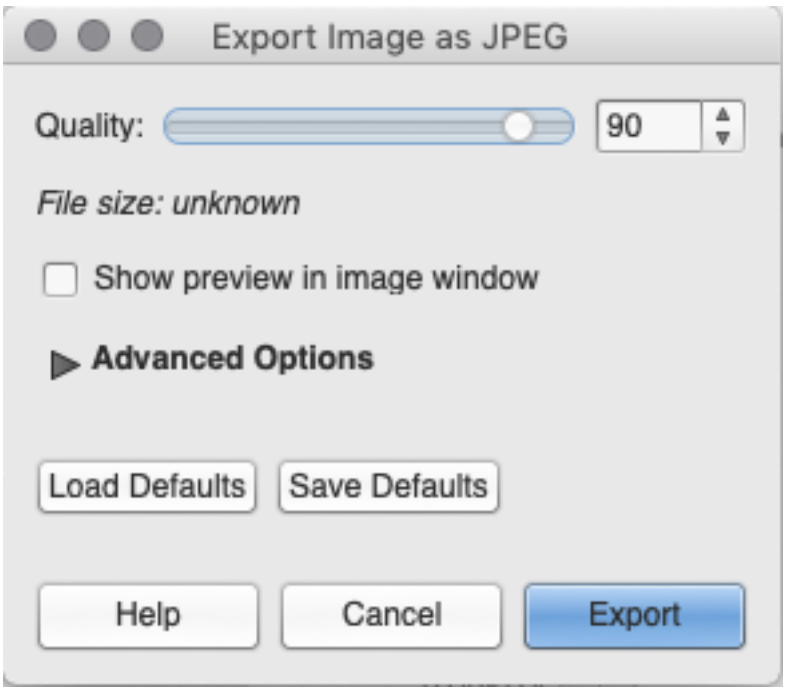


Next click on **Image>Mode>Index** and set the options as follows
This reduces the images colour data in the colour map saving us some data and reducing the file size as well as making the file easer for the ESP32 to read.
Click on **Convert** and the window will apply the changes and close.

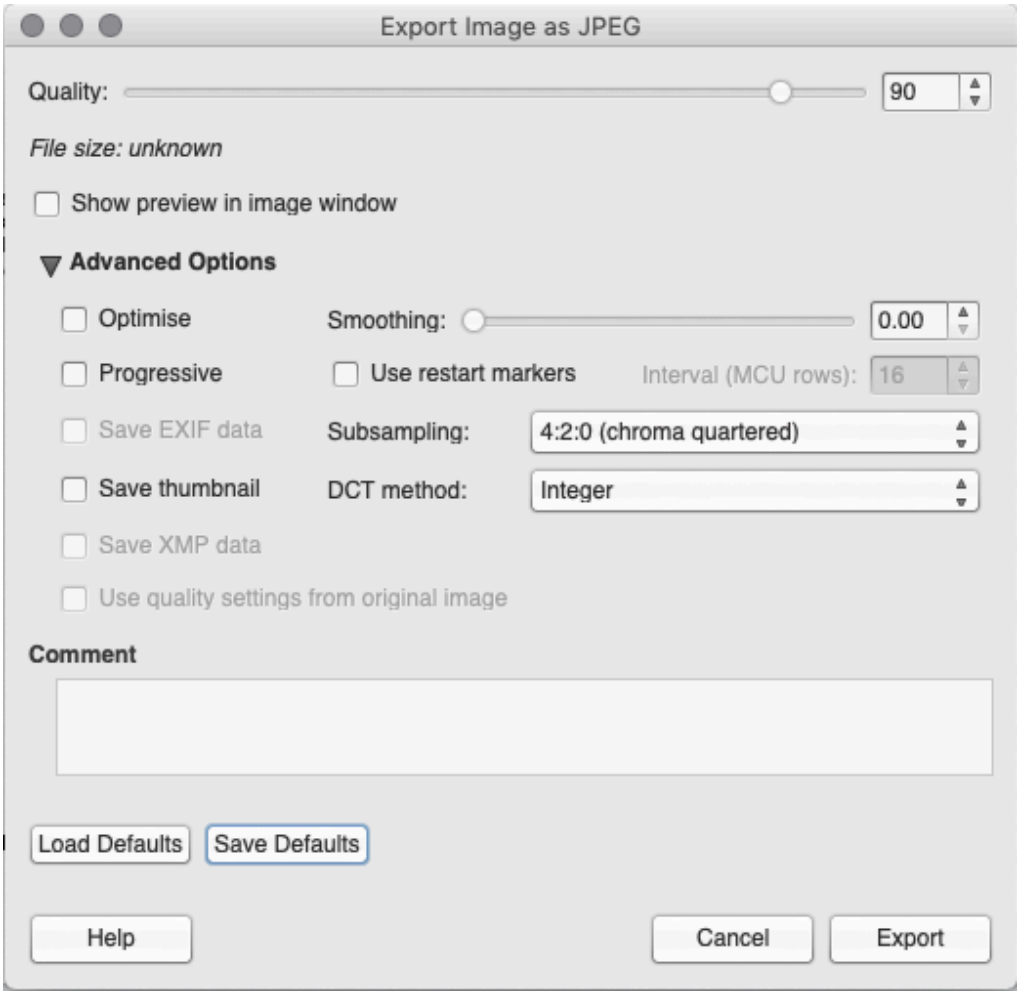
Now we have our image repaired it is time to save and create a .jpg file. The programmers changed the wayG.I.M.P saves files so that Save and Save As only save to G.I.M.P's .xcf file



format. To create the .jpg file we ned to click on **File>Export** to bring up the export dialog.

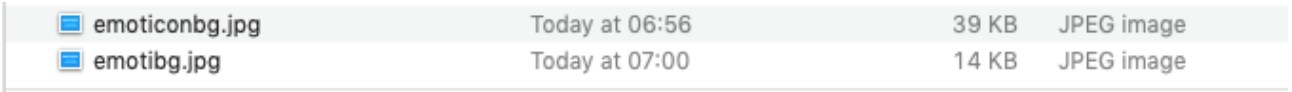


The file type box may be collapsed, click on the black arrow next to **Select File Type** to open the box and scroll down to find the JPEG Image Click on JPEG Image to highlight it and the file name at the box will change to Untitled.jpg.
Give the file a name under seven characters long and click **Export**.
In the next dialog, click on the arrow next to **Advanced Options** and change the setting to match



those in the screen shot below and click on **Export** to save the file.

The M5Stack firmware can only handle images up to 24kb in size. The following screen shot shows the same file (saved with a different name to make them stand out) the first has been saved with 90% quality and is to big for the M5Stacks memory. to reduce the file size, slide the Quality slider to the left to reduce the quality of the image. Because my image is simple, there is no loss in the quality of the images appearance after I reduced the quality to 80%.



Displaying Images on screen.

Now that we have our images how do we get them to show up on the screen?
The first method to get the image to display is to directly uploaded it to the M5Stacks internal memory.

User Interface Elements

At the top of the screen you will find UIFlows icon menu

This menu contains several useful functions.

Starting from the left icon we have:

IDE Switcher - This icon when clicked shows a menu that allows us to switch between versions of the UIFlow IDE. Some times we will encounter code that may work with one version but not with another.

Forum link - This icon when clicked will take you to the M5Stack forums.

The Documentation - This will take you to the section of the website where the online documentation is currently stored.

Code Examples - Clicking this icon will cause a window to appear containing demo and sample code.

Undo and Redo allow us to undo changes we made to code or redo the changes if we change our mind of the previous change didn't work.

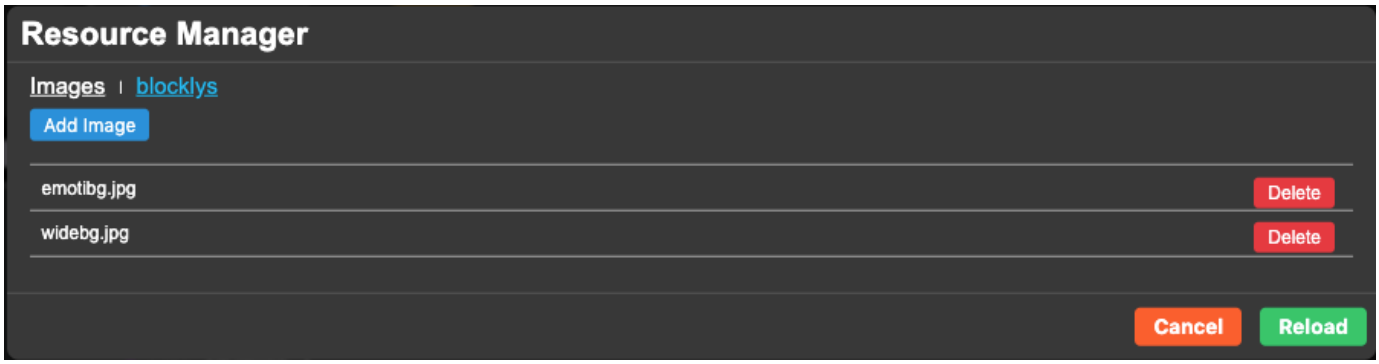
M5 Resource Manager - this button allows us to connect to the M5Stack or stick and upload or download files from the internal memory.

Play/Test - This button temperorqrly uploads code onto M5Stack or sticks to test however, the code is lost if the device is reset.

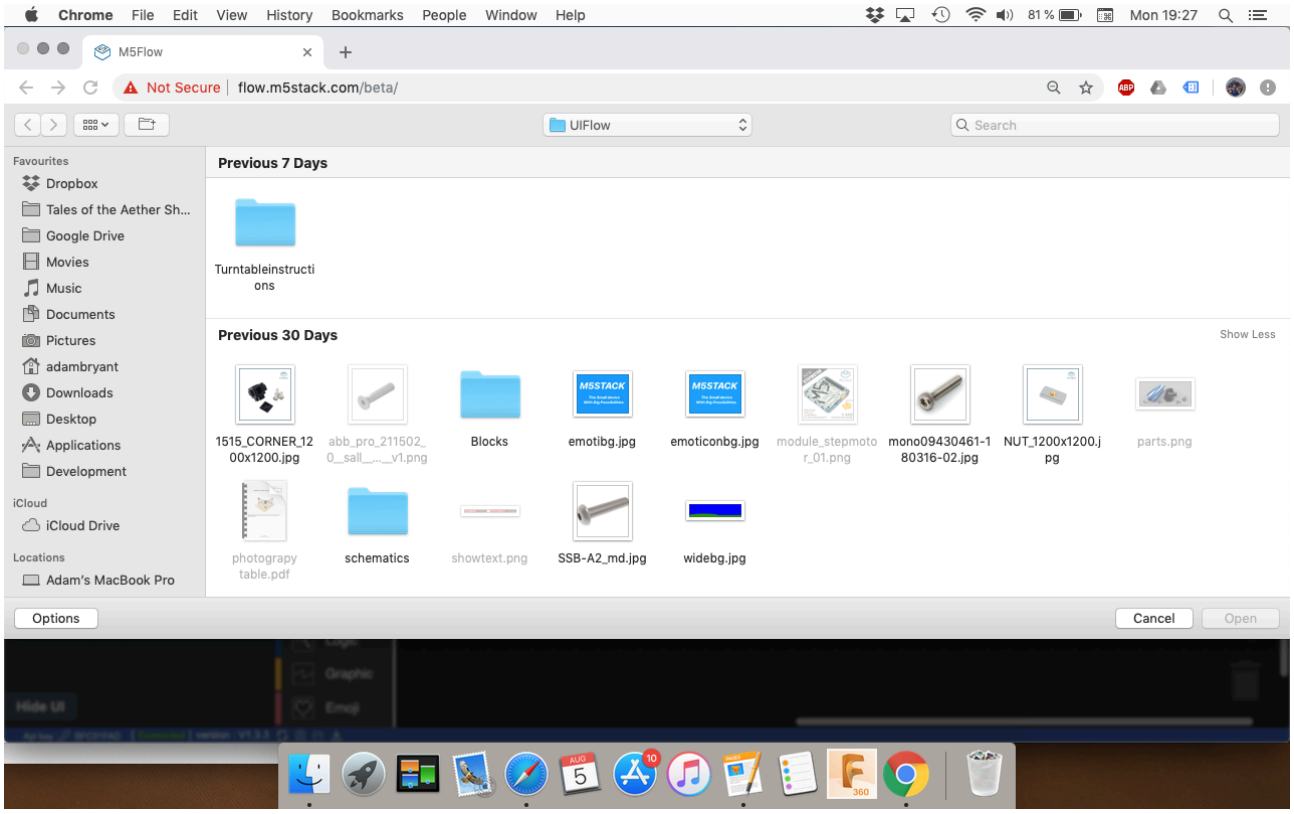
UIFlows Setting Menu - This button brings up the setting menu of UIFlows IDE allowing us to change the settings used by UIFlow to compile code for the M5Stack or the M5Stick. I will explain more about this menu later in the book.

For uploading images we will need to use the M5 File Manager.

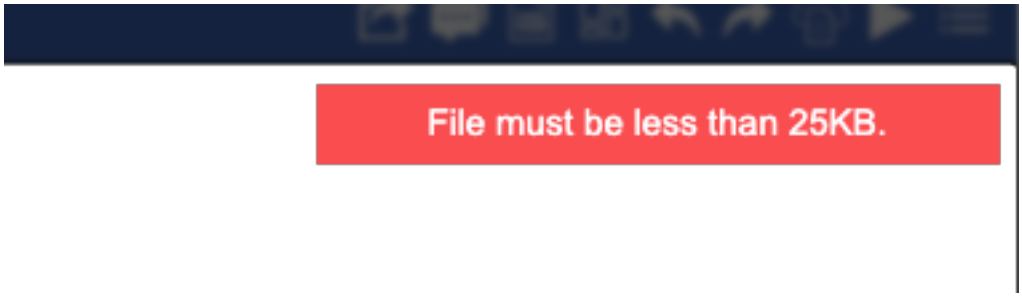
User Interface Elements



Click on the icon shown above and the resource manager panel will be displayed.
In this screen shot you can see that I have two images already loaded onto the M5Stack. To delete a file from the internal storage you just need to click on the "Delete" button next to the file. To add a file, click on "Add Images and the computers operating system file manager will open.



Click on a .jpg image and hit ok. If the image is below the file size of 25kb it will be added to the list, if not you will see the following pop in from the right hand edge of the screen.

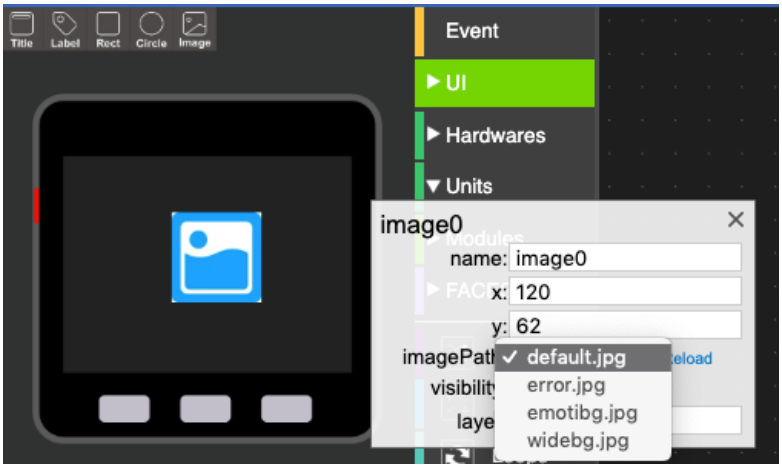


If the file is the correct size then the gray bar under the file name will start to brighten from the left end to the right end showing that it is in the process of being uploaded. If the image doesn't show up you can try clicking on reload.

Now we have images on the M5Stack we can start writing code to display the stored images.
There are several ways we can do this.
The easiest way to display an image on screen is to use the image loader found in the U.I Builder

panel above the virtual M5Stack.
Click on it and drag it onto the screen.

User Interface Elements



Now we click on the blue and white icon to access the image options and then click on "ImagePath" to drop down a list of files stored on the m5stack
In the following image we can see that the picture is not aligned properly on the screen.

To correct this we need to click on the image again to access the setting and set both X and Y to zero. You will notice that as you type the numbers the image will move around the screen until it reaches the position shown below.
All we need to do now is to press the play/test button and the image will now be shown on screen.

Internal Hardware

The M5Stack and M5Stick family have some differences on the inside due to configuration options available. The blocks in this section allow you to access those functions available in the various hardware

Speaker

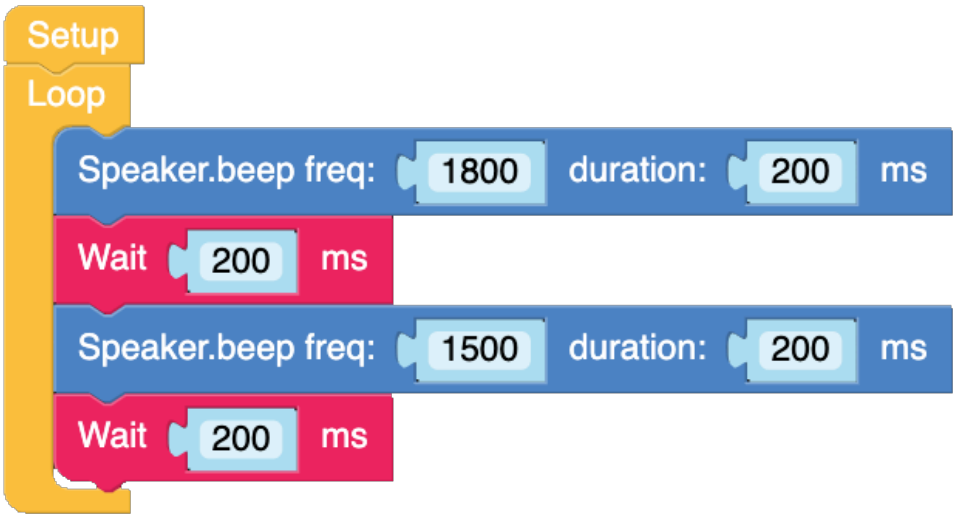
Speaker,

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(1800, 200)
```

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

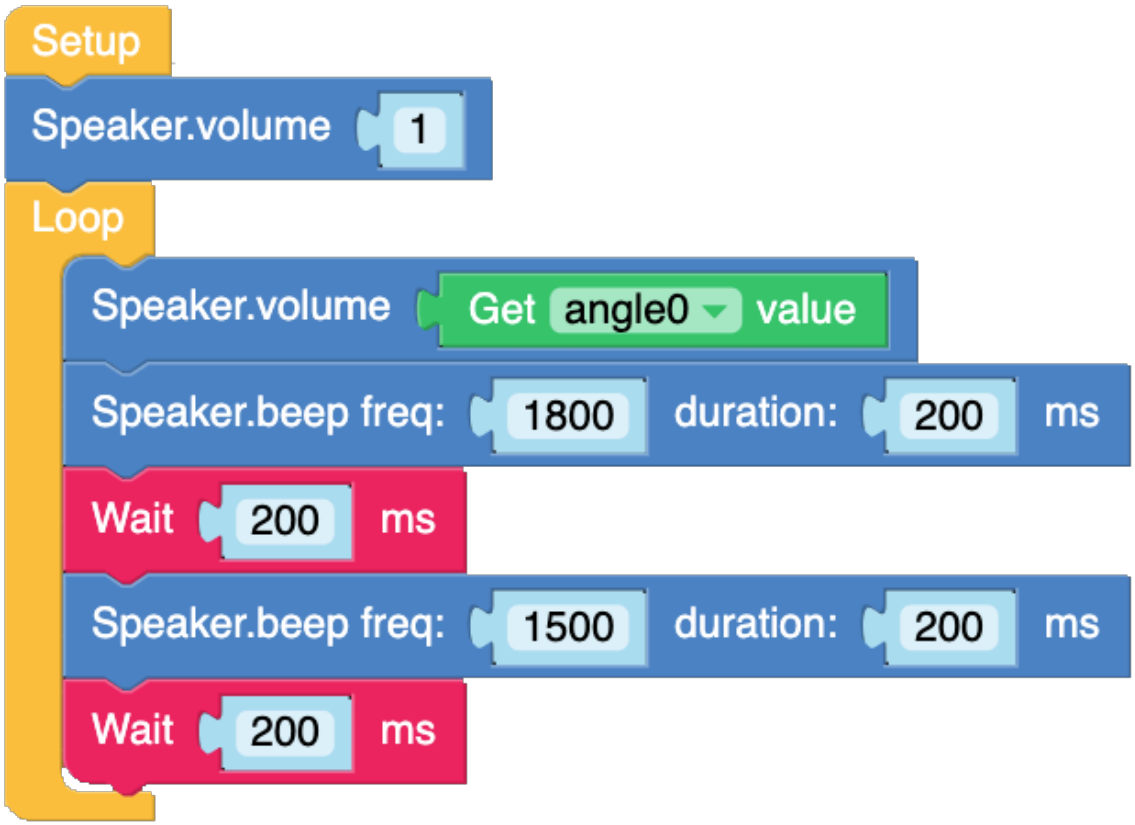
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(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(1)
```

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



Speaker

The Micropython code for this example is as follows.

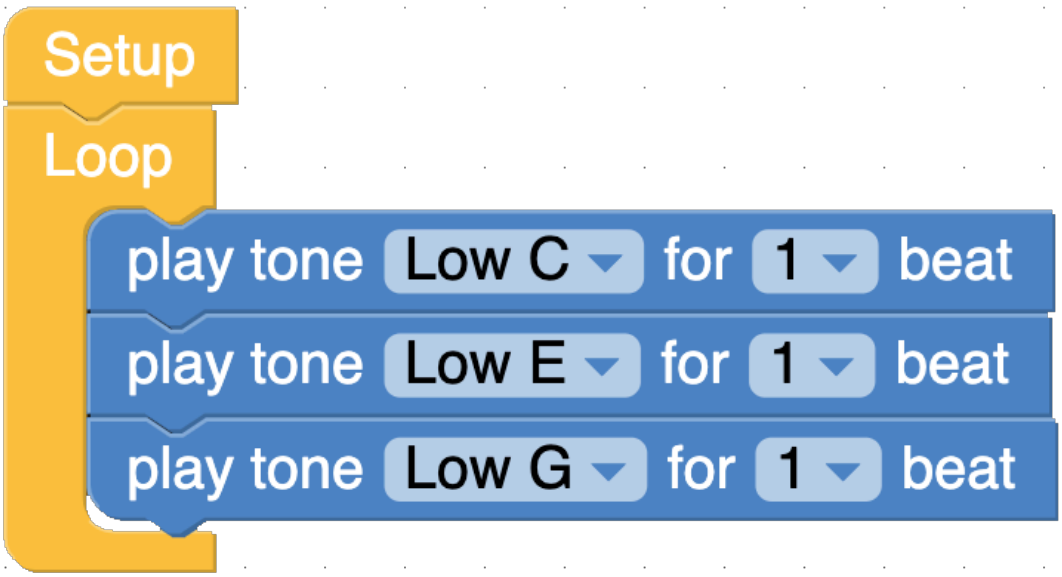
```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(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.
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 *

setScreenColor(0x222222)

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

RGB Bar Colour

RGB,
Set RGB Bar Colour,

Sets all the WS2812b LED colour in the M5Go base to the same colour using the colour selector.
The Micropython code for this block is:

```
rgb.setColorAll(0xff0000)
```

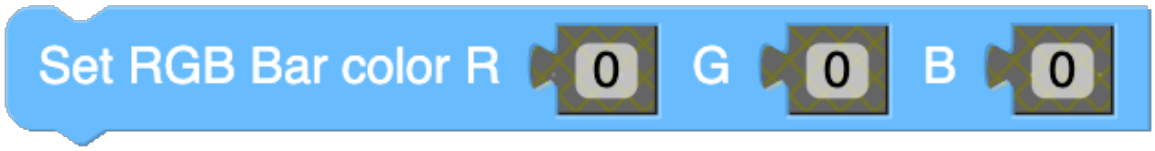
Example
In this example, all the RGB LED's in the bases left and right bars are lit up in red.

The Micropython code for this is:

```
from m5stack import *
from m5ui import *
from uiflow import *
setScreenColor(0x222222)
rgb.setColorAll(0xff0000)
```

Set RGB Bar Colour R,G,B,

Sets all the WS2812b LED colour using separate number defined using maths number blocks or variables.
The Micropython code for this block is:




```
rgb.setColorAll(0xff0000)
```

Example:
In this example, all the RGB LED's in the bases left and right bars are lit up by defining the values for Red, Green and, Blue using numbers.

The Micropython code for this demo is:

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

setScreenColor(0x222222)
rgb.setColorAll(0xff0000)
```

As you can see, while the values are entered into the blocks using numbers, the Micropython code values are generated in hexadecimal values.

RGB Bar Colour

Set (Side) RGB Bar Colour,

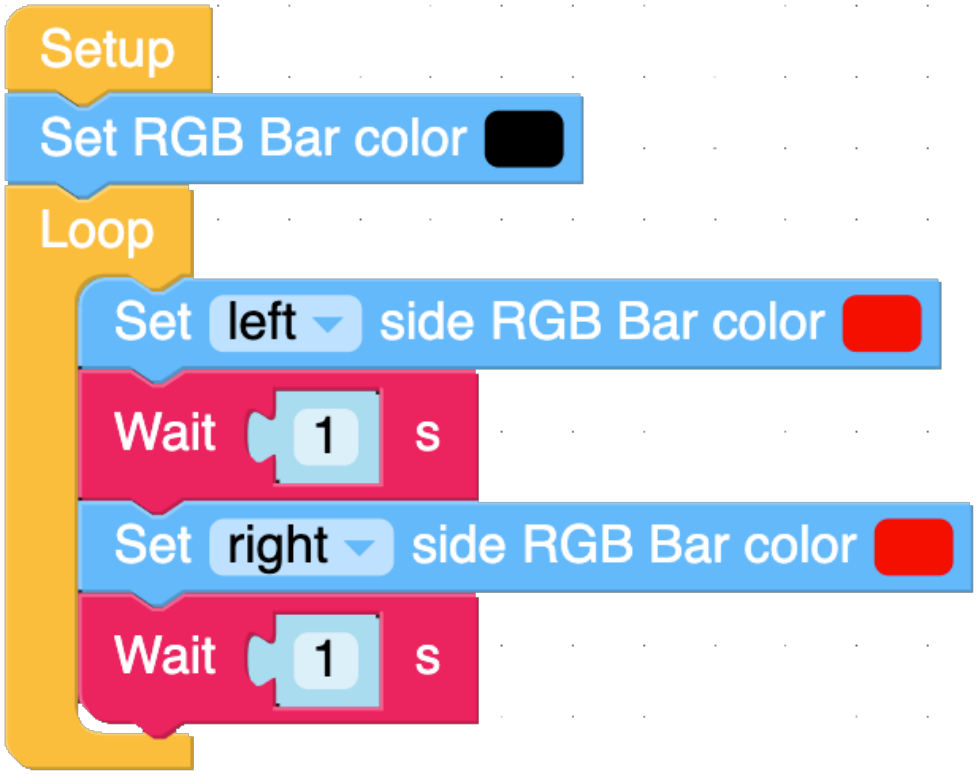
Sets all the WS2812b LED colour on the left or the right side of the M5Go base to the same colour using the colour selector.
The Micropython code for this block is:

As you can see, the Micropython code doesn't have a left or right option instead, it uses a version

```
rgb.setColorFrom(6 , 10 ,0xff0000)
```

of the RGB Range block. For more information on the RGB Range block, please view the RGB LED section later in the book.

Example.
In this example I alternate which side lights up.

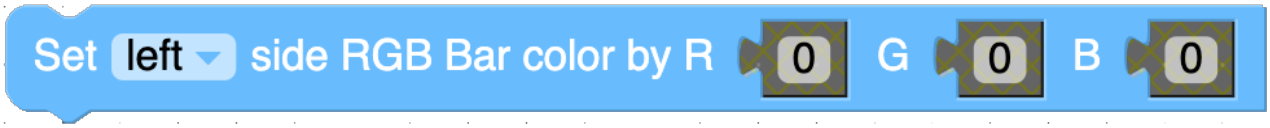


RGB Bar Colour

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

setScreenColor(0x222222)

rgb.setColorAll(0x000000)
while True:
  rgb.setColorFrom(6 , 10 ,0xff0000)
  wait(1)
  rgb.setColorFrom(1 , 5 ,0xff0000)
  wait(1)
  wait_ms(2)
```

RGB Bar Colour

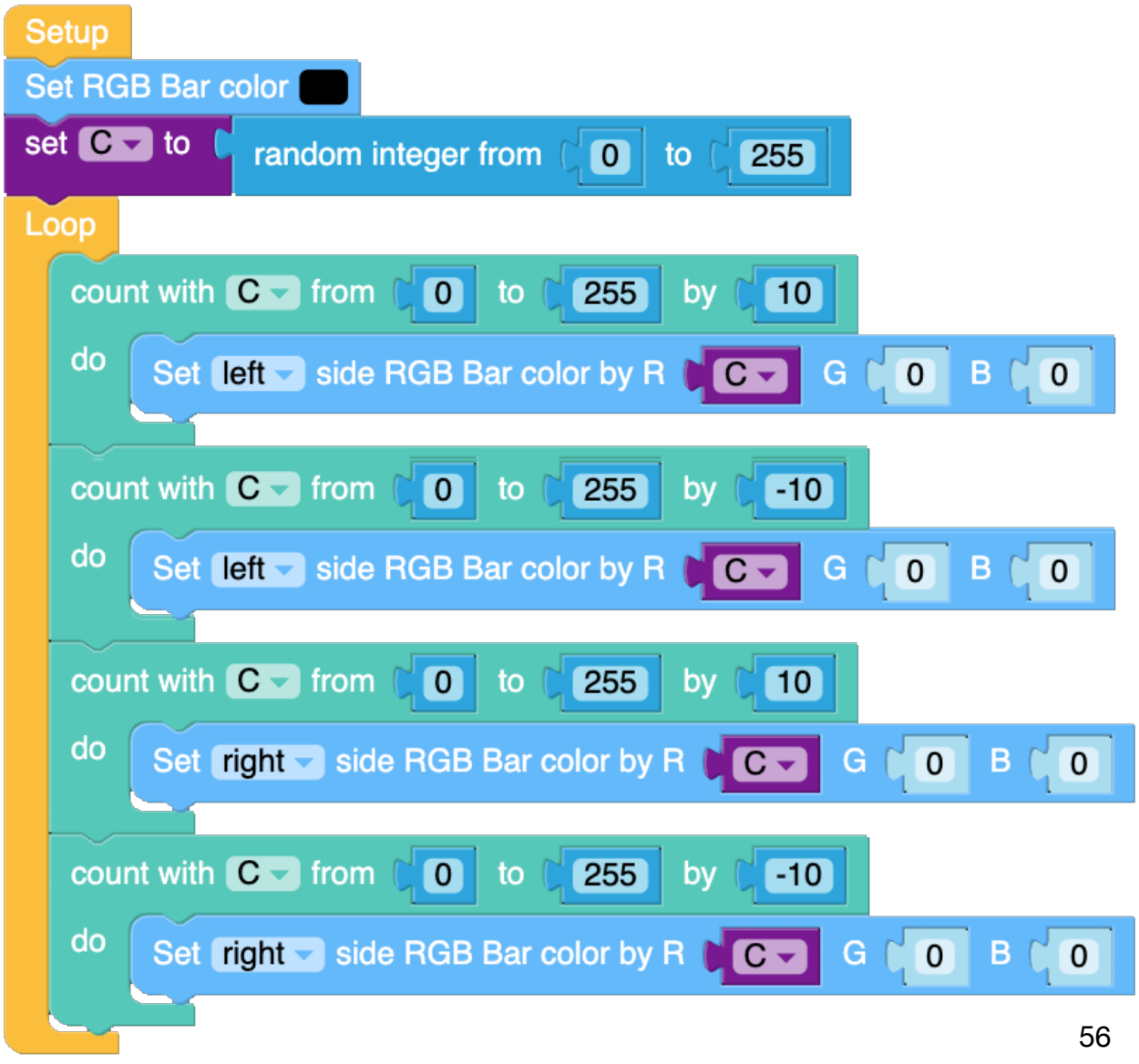
Set (Side) RGB Bar Colour R,G,B,



Sets all the WS2812b LED colour on the left or the right side of the M5Go base to the same colour using individual Red, Green and Blue colour values set by maths number blocks or variables..
As you can see, the Micropython code for this block is the same as the previous block:

```
rgb.setColorFrom(6 , 10 ,0xff0000)
```

Example.



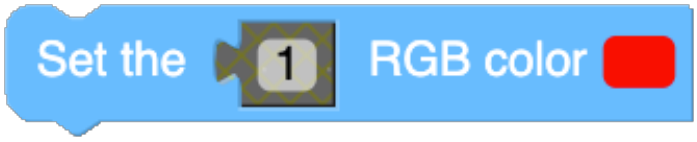
RGB Bar Colour

In this example I am increasing and decreasing the colour of the bars one side at a time.

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

setScreenColor(0x222222)

import random

C = None

rgb.setColorAll(0x000000)
C = random.randint(0, 255)
while True:
  for C in range(0, 256, 10):
    rgb.setColorFrom(6 , 10 ,(C << 16) | (0 << 8) | 0)
  for C in range(0, 256, 10):
    rgb.setColorFrom(6 , 10 ,(C << 16) | (0 << 8) | 0)
  for C in range(0, 256, 10):
    rgb.setColorFrom(1 , 5 ,(C << 16) | (0 << 8) | 0)
  for C in range(0, 256, 10):
    rgb.setColorFrom(1 , 5 ,(C << 16) | (0 << 8) | 0)
  wait_ms(2)
```

RGB Bar Colour

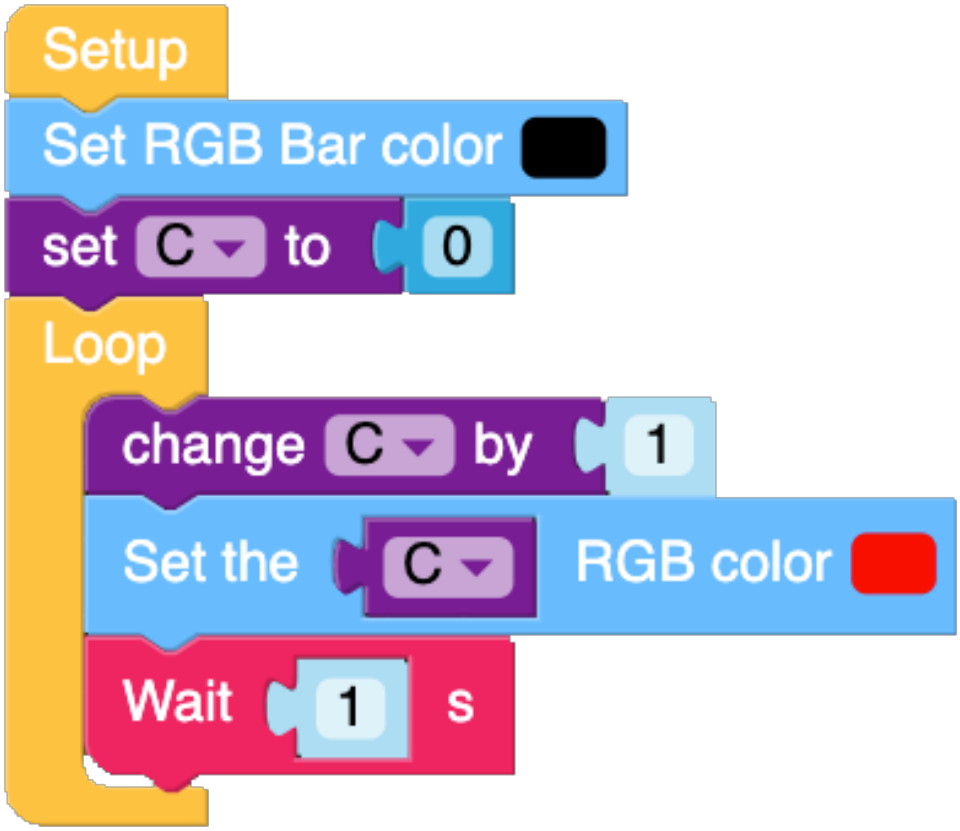
Set (individual) RGB Colour,



Sets one of the ten individual WS2812B LEDs colour on the M5Go base using the colour Picker. The position can be set using a maths block or a variable block.

rgb.setColor(1, 0xff0000)

The Micropython code for this is:



Example
In this example I am lighting each of the ten WS2812B LEDs in the Go base one after the other using a variable to chose which WS2812b should light up.

RGB Bar Colour

.

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

setScreenColor(0x222222)

C = None

rgb.setColorAll(0x000000)
C = 0
while True:
  C = (C if isinstance(C, int) else 0) + 1
  rgb.setColor(C, 0xff0000)
  wait(1)
  wait_ms(2)
```

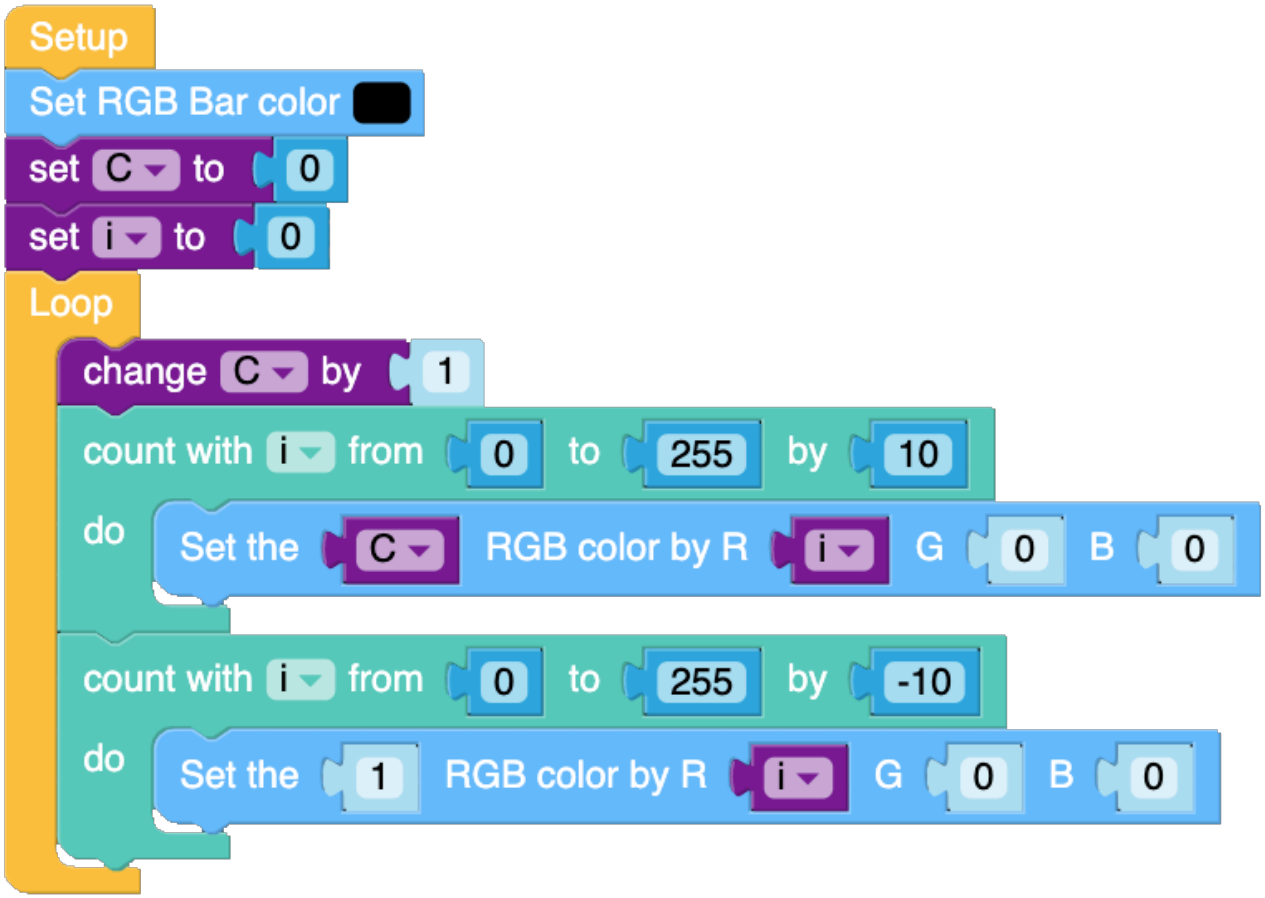
RGB Bar Colour

Set (individual) RGB Bar Colour R,G,B,

Sets one of the ten individual WS2812B LEDs colour on the M5Go base using individual Red, Green and Blue colour values using maths blocks or variable blocks.
The Micropython code for this is:

```
rgb.setColor(1, 0x000000)
```

Example

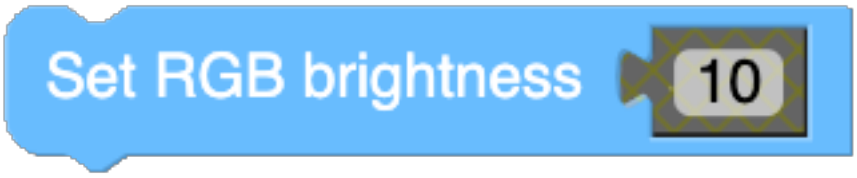


RGB Bar Colour

Using the previous example, we can expand it to also alter the colours as well as which WS2812b LED illuminates.

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

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

C = None
i = None

rgb.setColorAll(0x000000)
C = 0
i = 0
while True:
  C = (C if isinstance(C, int) else 0) + 1
  for i in range(0, 256, 10):
    rgb.setColor(C,(i << 16) | (0 << 8) | 0)
  for i in range(0, 256, 10):
    rgb.setColor(1,(i << 16) | (0 << 8) | 0)
  wait_ms(2)
```

Set RGB Brightness,



Sets the brightness level of all the WS2812B LEDs in the M5Go baseplate. We can use Maths blocks or variables to control the brightness of the WS2812b LED's and if you have the Angle sensor, we can also use that as a "dimmer" like control.
The Micropython code for this is:

```
rgb.setBrightness(10)
```

Example

In this example I have created a simple dimmer control for the LEDs

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

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

rgb.setColorAll(0xff0000)
while True:
  rgb.setBrightness((angle0.read()))
  wait_ms(2)
```

IMU.

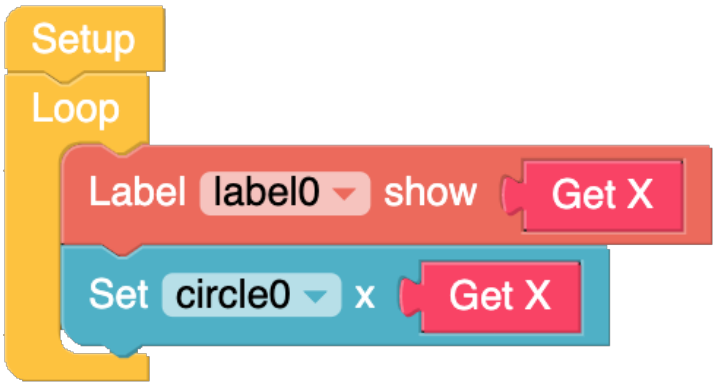
The M5StickC has a built in gyro which can be used for movement tracking. In order to control the IMU (Inertial Measurement Unit) we use the following blocks which return numbered strings.

Get X,

Gets the X Value from the IMU and returns it as a number that can be used instead of a maths value block or a variable block.
The Micropython code for this block is:

```
(imu0.ypr[1])
```

Example



In this example I am using the GetX block to set the X position of a circle while a Label show the numerical value of X.
The Micropython code for this example is.

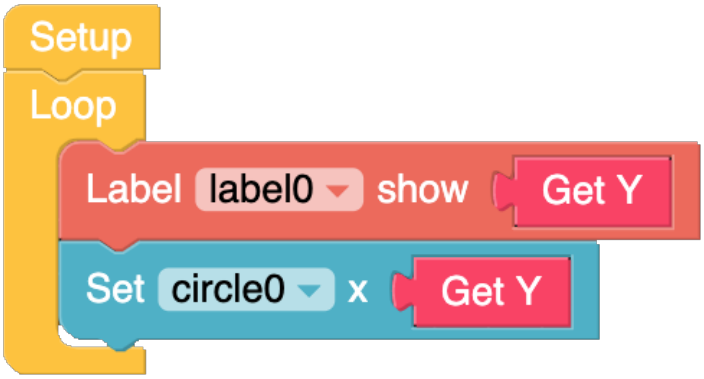
```
from m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x222222)

Imu0 = imu.IMU()
label0 = M5TextBox(93, 223, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
circle0 = M5Circle(164, 115, 15, 0xFFFFFF, 0xFFFFFF)

while True:
  label0.setText(str(imu0.ypr[1]))
  circle0.setPosition(x=(imu0.ypr[1]))
  wait_ms(2)
```

Get Y,

Gets the Y Value from the IMU and returns it as a number that can be used instead of a maths value block or a variable block.
The Micropython code for this block is

```
(imu0.ypr[2])
```

Example



In this example I am using the GetY block to set the Y position of a circle while a Label show the numerical value of Y.
The Micropython code for this example is.

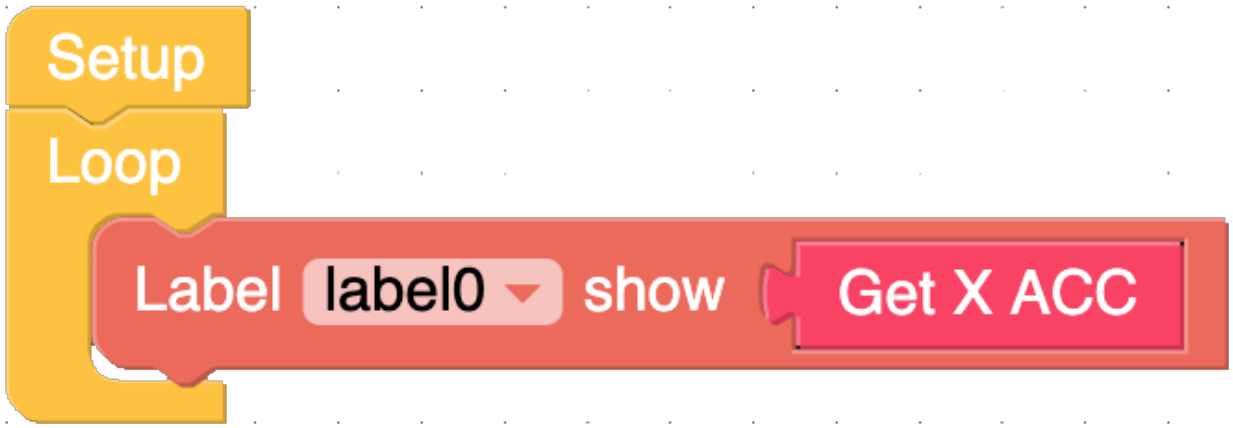
```
from m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(93, 223, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
circle0 = M5Circle(164, 115, 15, 0xFFFFFF, 0xFFFFFF)


while True:
  label0.setText(str(imu0.ypr[2]))
  circle0.setPosition(x=(imu0.ypr[2]))
  wait_ms(2)
```

Get X ACC,

Gets the X acceleration value from the IMU.
The Micropython code for this block is:

```
imu0.acceleration[0]
```

Example



In this example I have just used a label to show the X accelerations reading from the M5Sticks IMU.
The Micropython code for this demo is as follows.

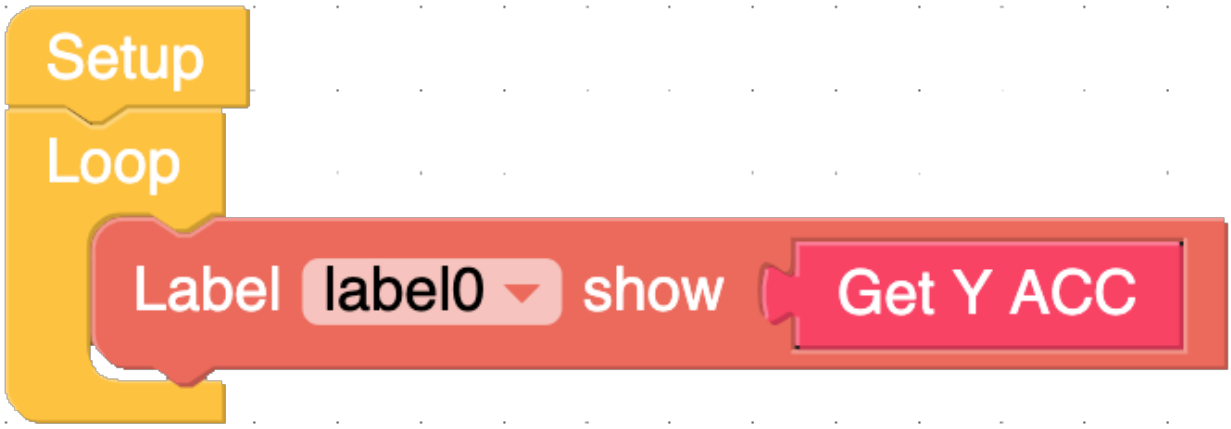
```
from m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(imu0.acceleration[0]))
  wait_ms(2)
```

Get Y ACC,

Gets the Y acceleration value from the IMU.
The Micropython code for this block is:

```
imu0.acceleration[1]
```

Example



In this example I have just used a label to show the Y accelerations reading from the M5Sticks IMU.
The Micropython code for this demo is as follows.

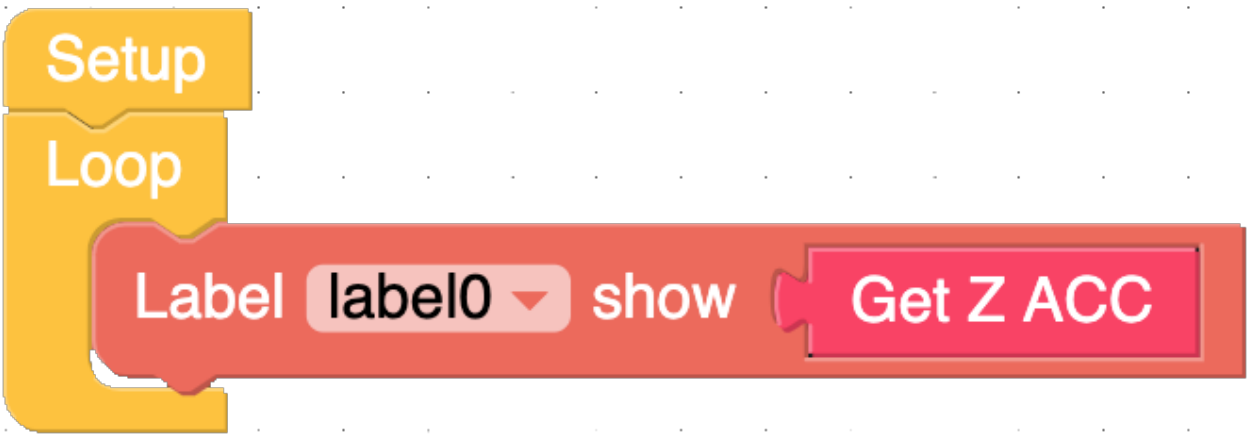
```
ffrom m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(imu0.acceleration[1]))
  wait_ms(2)
```

Get Z ACC,

Gets the Z acceleration value from the IMU.
The Micropython code for this block is:

```
imu0.acceleration[2]
```

Example
In this example I have just used a label to show the Z accelerations reading from the M5Sticks IMU.



The Micropython code for this demo is as follows.

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

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
 label0.setText(str(imu0.acceleration[2]))
 wait_ms(2)
```

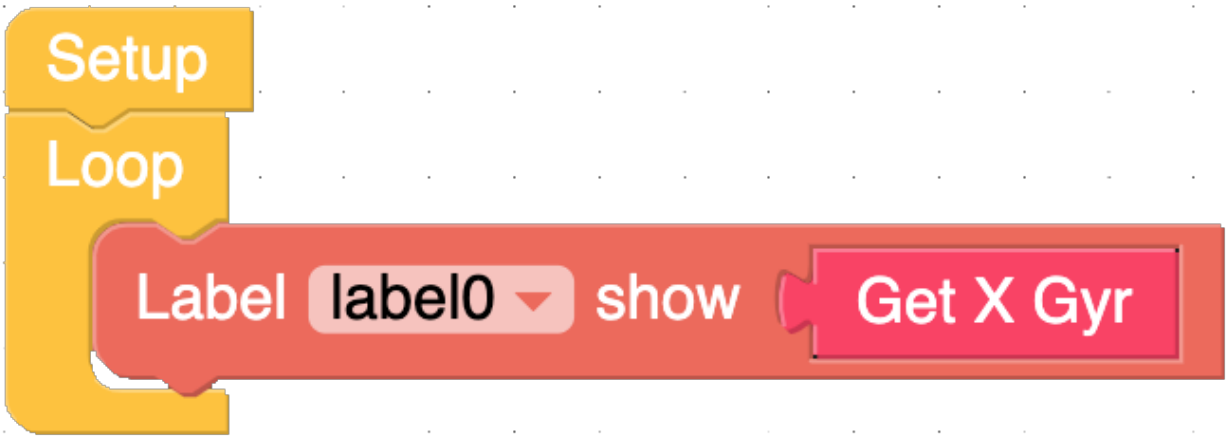
It is worth noting that X, Y and, Z are not used in the code but instead use 0, 1 and, 2 respectively to represent X, Y, and, Z.

Get X Gyro,

Gets the X value from the IMU's Gyro.
The Micropython code for this block is:

```
imu0.gyro[0]
```

Example
In this example I have just used a label to show the X rotation reading from the M5Sticks IMU.



The Micropython code for this demo is as follows.

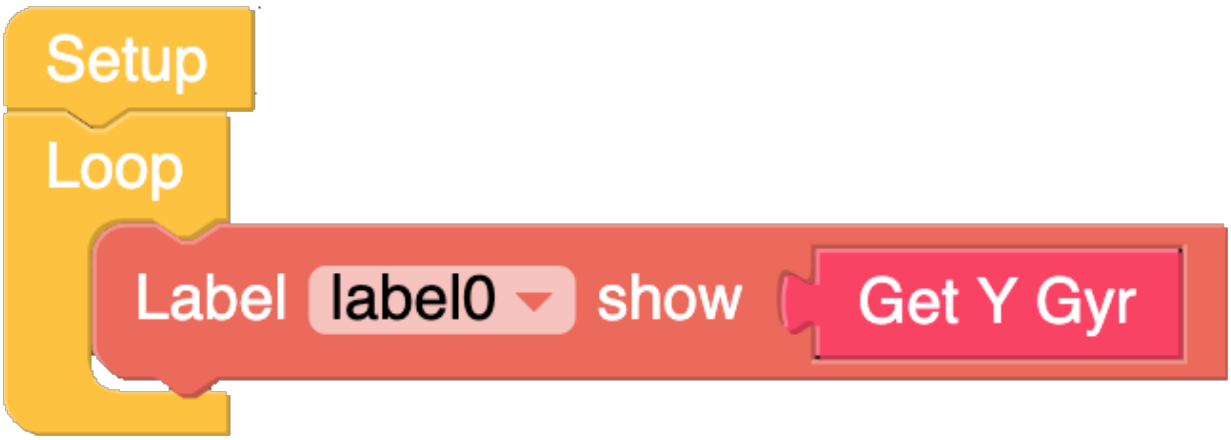
```
from m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(imu0.gyro[0]))
  wait_ms(2)
```

Get Y Gyro,

Gets the Y value from the IMU's Gyro.
The Micropython code for this block is:

```
imu0.gyro[1]
```

Example
In this example I have just used a label to show the X rotation reading from the M5Sticks IMU.



The Micropython code for this demo is as follows.

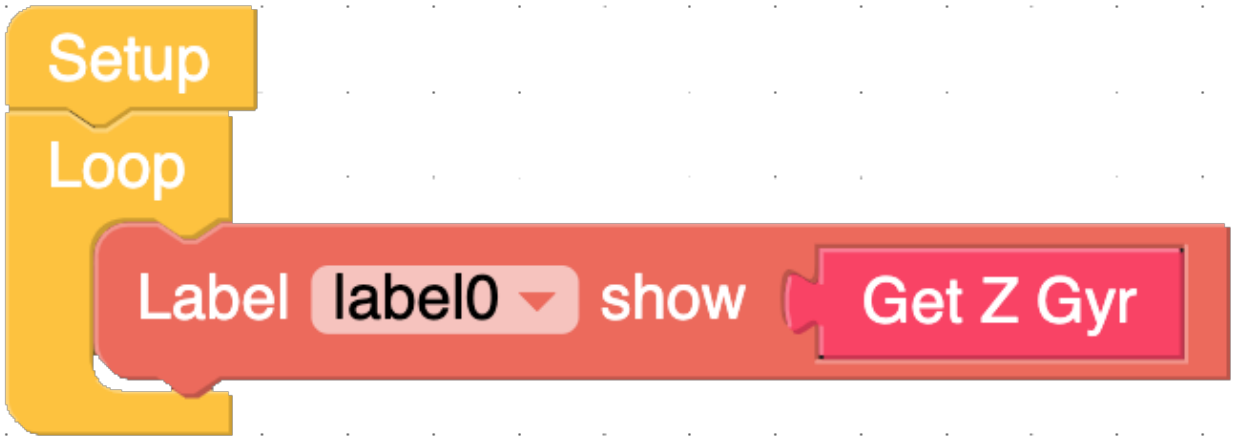
```
from m5stack import *
from m5ui import *
from uiflow import *
import imu

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(imu0.gyro[1]))
  wait_ms(2)
```

Get Z Gyro,

Gets the Z value from the IMU's Gyro.
The Micropython code for this block is:

```
imu0.gyro[2]
```

Example
In this example I have just used a label to show the X rotation reading from the M5Sticks IMU.



The Micropython code for this demo is as follows.

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

setScreenColor(0x111111)

imu0 = imu.IMU()
label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(imu0.gyro[2]))
  wait_ms(2)
```

Power P1 - AXP192

Power (StickC)

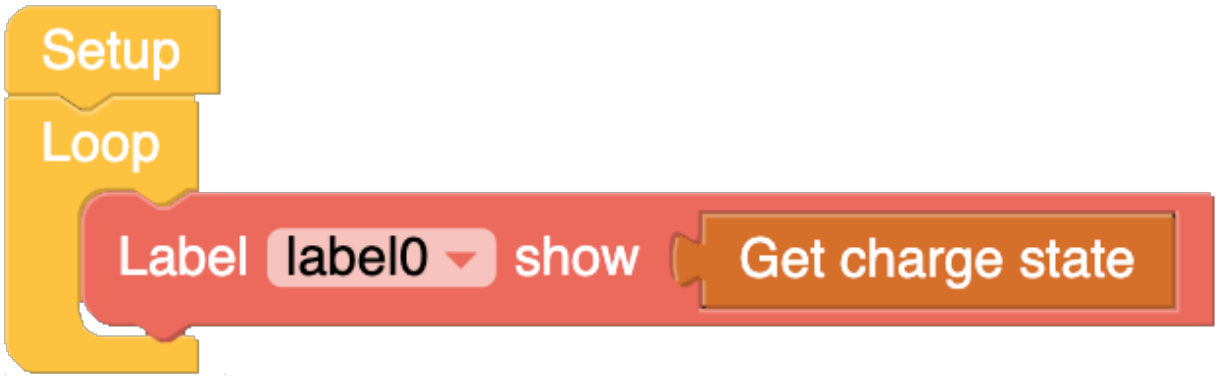
The following power blocks are only available to units that are fitted with a modified version of the power control chip which allows the ESP32 to communicate with the power management chip.

Get Charge State

The Get Charge State block is is a value block which returns the status from the power management chip. The block reports false if there is no USB power connected or True if a USB cable is plugged in and connected to a supply.
The Micropython code for this is.

```
axp.getChargeState()
```

Example



In this example I have just created a label that show if a USB cable is plugged in and the M5Stick is charging.The Micropython code is as follows.

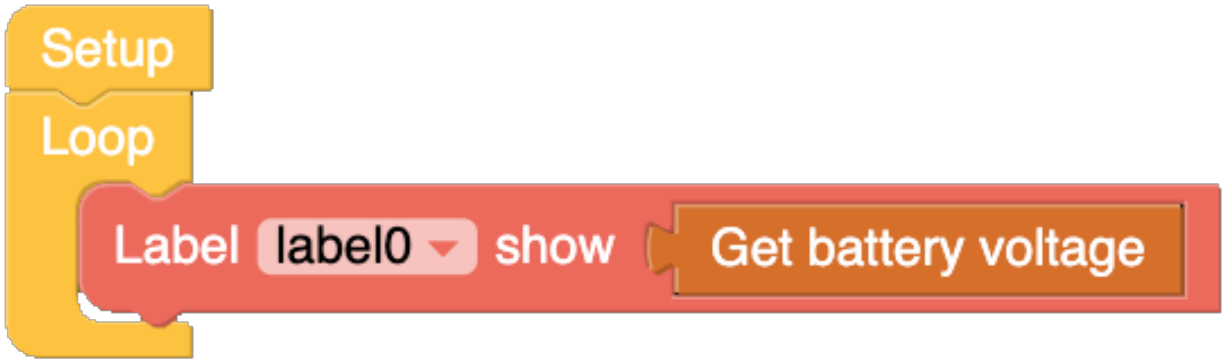
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x111111)

label0 = M5TextBox(19, 30, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getChargeState()))
  wait_ms(2)
```

Get Battery Voltage

Returns the current voltage from the internal battery.
The Micropython code for this block is.

```
AXP.GETBATVOLTAGE()
```

Example
In the following example I am just using a label to show the voltage of the internal battery.



The Micropython code for this is

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

setScreenColor(0x111111)

label0 = M5TextBox(10, 88, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getBatVoltage()))
  wait_ms(2)
```

Power P1 - AXP192

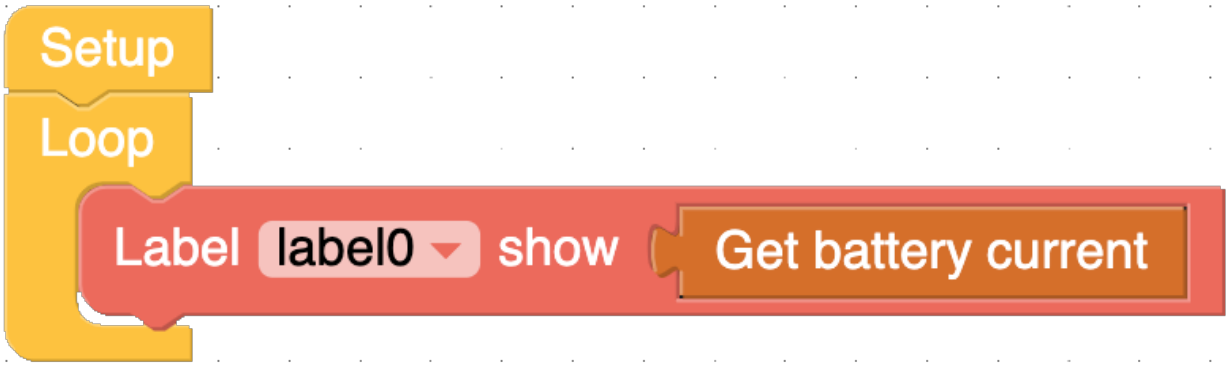
Get Battery Current

Returns the current in milliamps from the internal battery.

The Micropython code for this block is

```
axp.getBatCurrent()
```

Example
In this example I have just created a label that show internal battery current use on the screen.



The Micropython code is as follows.

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

setScreenColor(0x111111)

label0 = M5TextBox(21, 57, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getBatCurrent()))
  wait_ms(2)
```

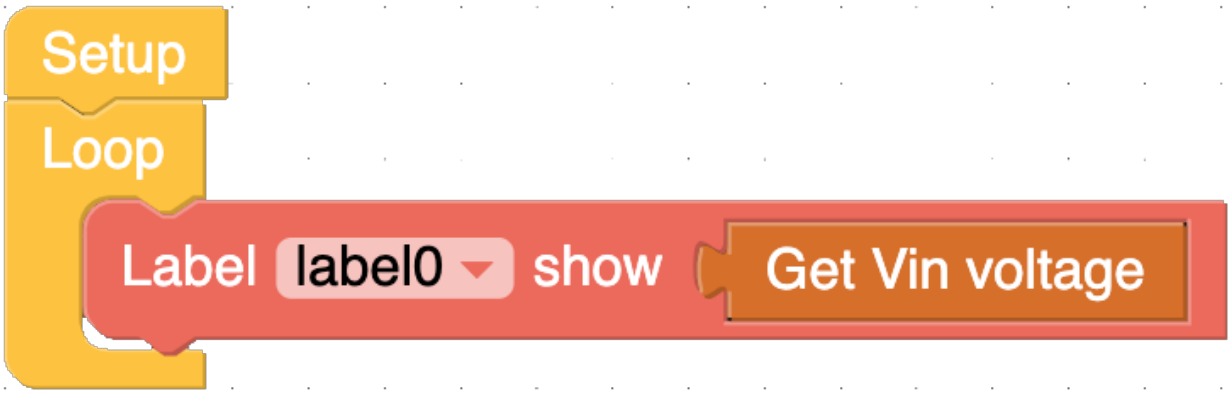
Get Vin Voltage

Returns the voltage on the input pin of the HAT connector on the top of the M5StickC

The Micropython code for this block is

```
axp.getVinVoltage()
```

Example
The following example uses a label show the voltage on the screen.



The Micropython code is as follows.

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

setScreenColor(0x111111)

label0 = M5TextBox(14, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getVinVoltage()))
  wait_ms(2)
```

Power P1 - AXP192

Get Vin Current

Returns the current on the input pin of the HAT connector on the top of the M5StickC

The Micropython code for this block is

```
axp.getVinCurrent()
```

Example
This example is the same as the Vin Voltage demo but with the voltage block changed for the Vin Current Block.

The Micropython code for this example is

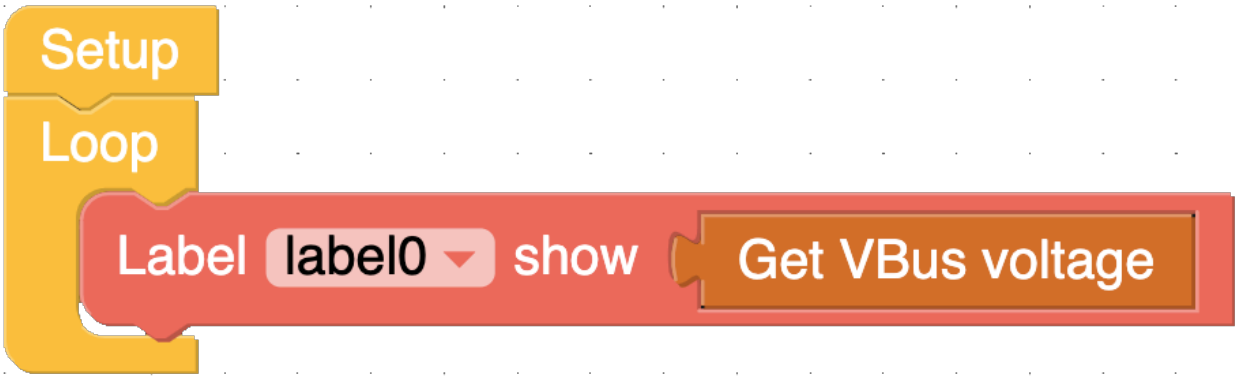
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x111111)

label0 = M5TextBox(5, 40, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getVinCurrent()))
  wait_ms(2)
```

Get VBus Voltage

The Micropython code for this block is

```
axp.getVBusVoltage()
```

Example
This example is the same as the Vin Voltage demo but with the voltage block changed for the VBus Voltage Block.



The Micropython code for this example is

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

setScreenColor(0x111111)

label0 = M5TextBox(5, 36, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getVBusVoltage()))
  wait_ms(2)
```

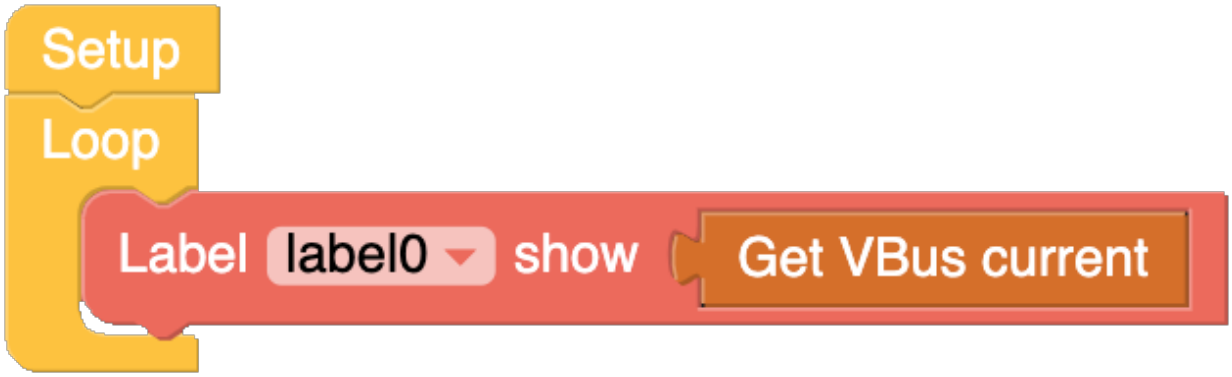
Power P1 - AXP192

Get VBus Current

The Micropython code for this block is

```
axp.getVBusCurrent()
```

Example



The Micropython code for this block is

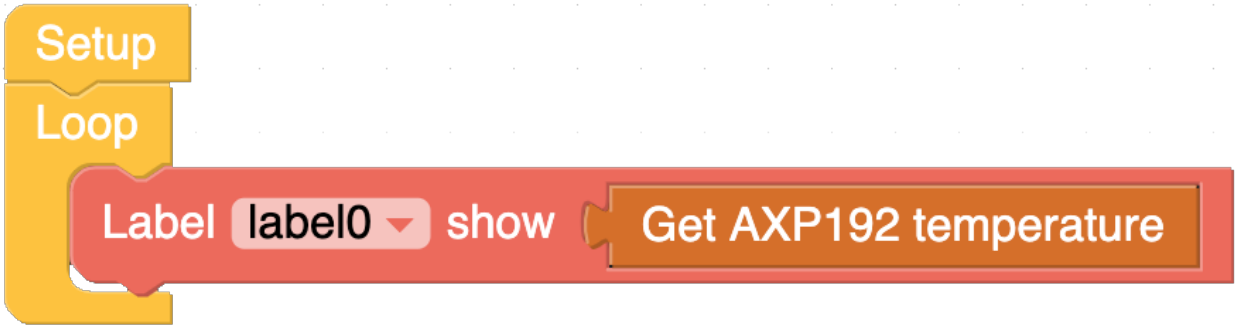
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x111111)

label0 = M5TextBox(11, 26, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getVBusCurrent()))
  wait_ms(2)
```

Get AXP192 Temperature

Returns the temperature from the AXP192's internal temperature sensor.
The Micropython code for this block is

```
axp.getTempInAXP192()
```

Example



The Micropython code for this block is

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

setScreenColor(0x111111)

label0 = M5TextBox(16, 40, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(axp.getTempInAXP192()))
  wait_ms(2)
```

Power P1 - AXP192

Power Off

Can be used to power down the AXP192 and the M5Stack or M5Stick Connected to it.
The Micropython code for this block is

```
axp.powerOff()
```

Example.

This demo when run will show hello M5 on an M5StickC screen and then switch off when the button on the from tis pressed.
To switch back on, you have to use the power/reset button on the side of the M5StickC.

The Micropython code for this demo is shown below.

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

setScreenColor(0x111111)

label0 = M5TextBox(20, 33, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

def buttonA_wasPressed():
  # global params
  axp.powerOff()
  pass
btnA.wasPressed(buttonA_wasPressed)

while True:
  label0.setText('Hello M5')
  wait_ms(2)
```

Power P1 - AXP192

The next two blocks are for advanced control of the AXP192's functions and not recommended for normal use.
Incorrect use of these two blocks could result in damage of your M5Stack or M5Stick.

Set Battery Charge Current

The Set Battery Charge Current block is used to control how much current if fed to the battery for charging. The current available options are 100mA, 190mA, 280mA, 360mA, 450mA, 550mA, 630mA and 700mA.

Again, I must mention that playing with these setting can be dangerous as the batteries used are lithium based and could explode or catch fire if incorrectly charged.

The Micropython code for this block is

```
axp.setChargeCurrent(axp.CURRENT_100MA)
```

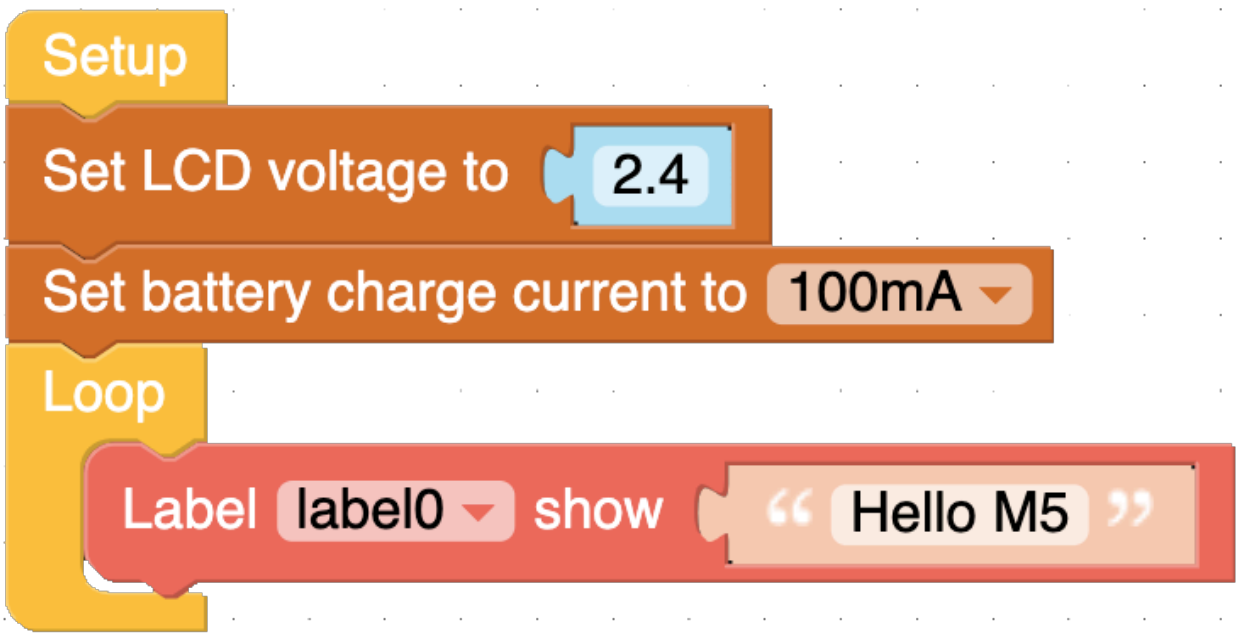
Set LCD Voltage

The Set LCD Voltage block can be used to control the voltage that the LCD runs at. Voltage is used using a variable block or a mathematical value block. Setting the voltage lower will reduce the brightness but if set too low then the screen will stop working. Setting the voltage higher runs the risk of damaging the screen.

The Micropython code for this block is

```
axp.setLDO2Volt(0)
```

Example
In this example I have used the blocks in the setup section of the program but the Set LCD Screen can be used in a loop if you know how to use it correctly.



The Micropython code for this demo is as follows.

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

setScreenColor(0x111111)

label0 = M5TextBox(20, 33, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

axp.setLDO2Volt(2.4)
axp.setChargeCurrent(axp.CURRENT_100MA)
while True:
  label0.setText('Hello M5')
  wait_ms(2)
```

Power P2 - IP5306

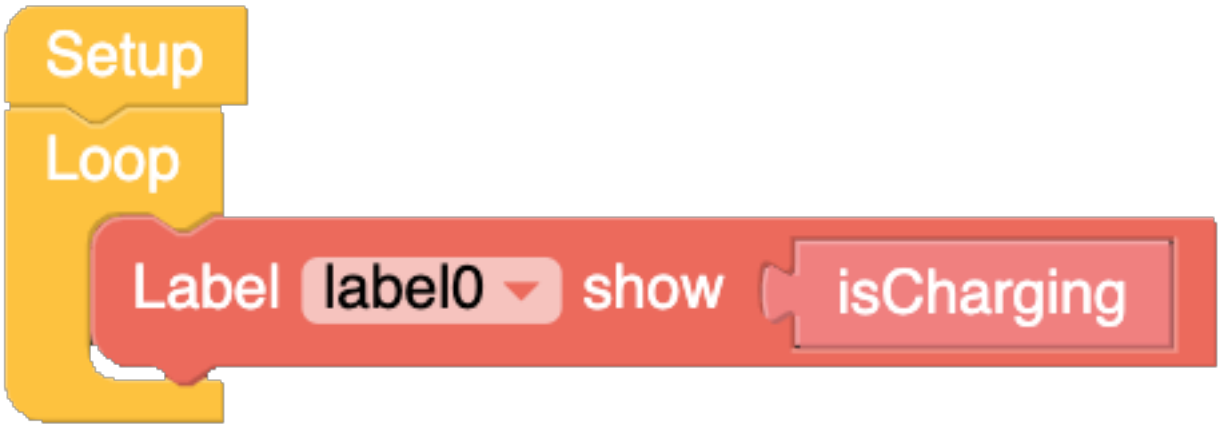
The following power blocks are only available on certain M5Stack models that if the customised version of the IP5306 chip.

Is Charging

The Is Charging block is a value block which returns the status from the power management chip. The block reports false if there is no USB power connected or True if a USB cable is plugged in and connected to a supply.
The Micropython code for this block is

```
power.isCharging()
```

Example



As with previous examples, here I am just using a label to show the status returned from the Is Charging Block.
The Micropython code for this example is as follows.

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

setScreenColor(0x222222)

label0 = M5TextBox(108, 74, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(power.isCharging()))
  wait_ms(2)
```

Is Charge Full

The Is Charge Full block is a value block which returns the status from the power management chip. The block reports false if there is the battery is not fully charged or True if the battery is fully charged.
The Micropython code for this block is

```
power.isChargeFull()
```

Example

As with previous examples, here I am just using a label to show the status returned from the Is Charge Full Block.
The Micropython code for this example is as follows.

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

setScreenColor(0x222222)

label0 = M5TextBox(108, 74, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(power.isChargeFull()))
  wait_ms(2)
```

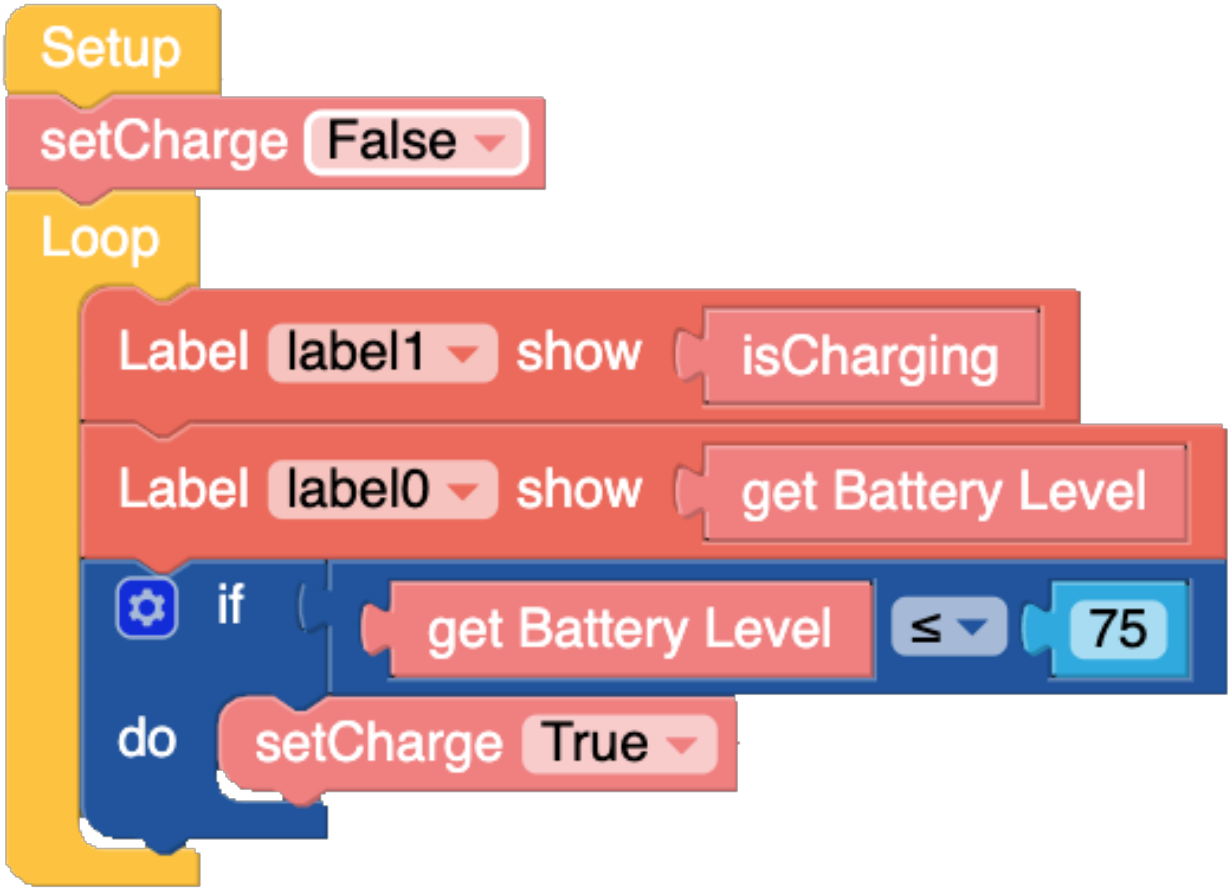
Power P2 - IP5306

Set Charge

The Set Charge block allows you to control wether the battery is charging or not. Unlike the other blocks in this group, the Set Charge Block is a function block and not a value block.
The Micropython code for this block is

```
power.setCharge(True)
```

Example



In this example I have initially set the charging to false. In the loop the two labels check the status of the the battery to see the charge level and charge status. I have then added a logic loop that checks if the battery is under 75 and if so set the battery charging. I have done this because my battery is flat as I forgot to charge it.
The Micropython code for this example is as follows.

Power P2 - IP5306

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

setScreenColor(0x222222)

label0 = M5TextBox(108, 74, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
label1 = M5TextBox(115, 38, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

power.setCharge(False)
while True:
  label1.setText(str(power.isCharging()))
  label0.setText(str(power.getBatteryLevel()))
  if (power.getBatteryLevel()) <= 75:
    power.setCharge(True)
  wait_ms(2)
```

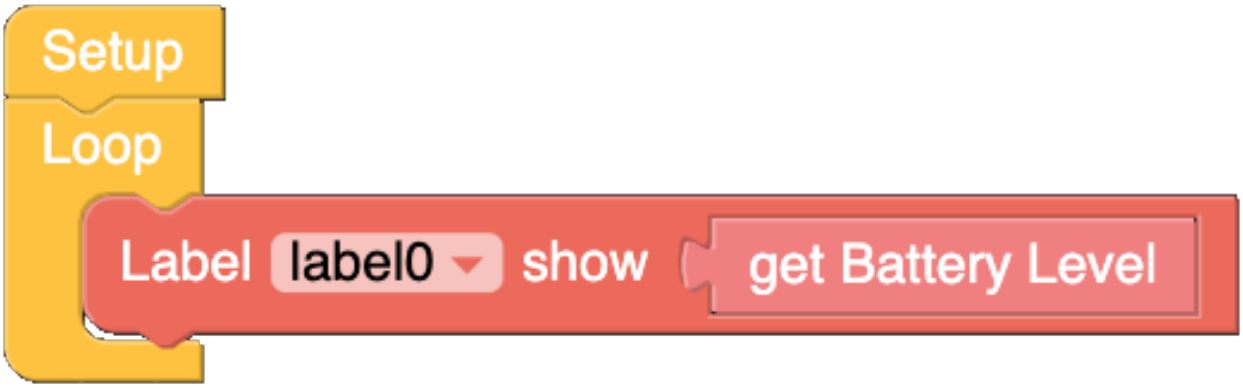
Power P2 - IP5306

Get Battery Level

TheGet Battery Lever block is a value block that returns the charge level of the battery as read by the IP5306.
The Micropython code for this block is

```
power.getBatteryLevel()
```

Example



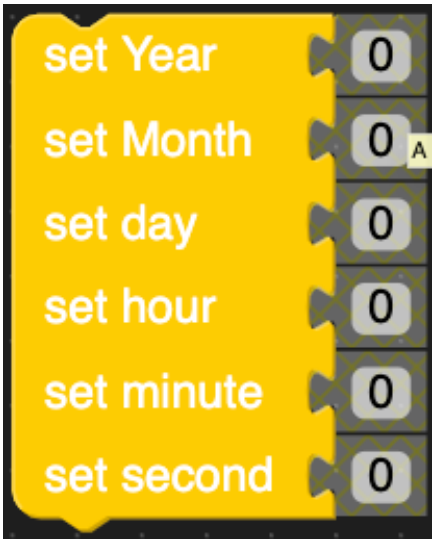
As with previous examples, here I am just using a label to show the status returned from the Get Battery Level Block.
The Micropython code for this example is as follows.

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

setScreenColor(0x222222)

label0 = M5TextBox(108, 74, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(power.getBatteryLevel()))
  wait_ms(2)
```

Real Time Clock

At Present, the real time clock is only available to M5Sticks. The blocks in this section allow you to configure and make use of the RTC for time and date critical function.

Time and Date configuration block.



Used to configure the initial time and date settings for the internal clock.

Get Year,

Gets the year from the RTC.

Get Month,

Gets the month from the RTC.

Get Day,

Gets the day from the RTC.

Get Hour,

Gets the hour from the RTC.

Get Minute,

Gets the minute from the RTC.

Get Second,

Gets the seconds from the RTC.

Get Local Time.

Gets the local from the RTC.

LED

LED

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,

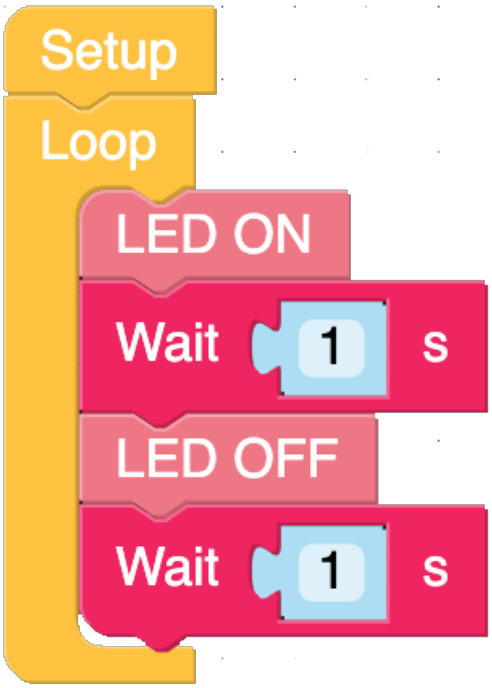
```
M5LED.ON()
```

LED Off

Turns the red led on the top of the M5Stick C off.
The Micropython code for this is,

```
M5LED.OFF()
```

Example.
The following example turns the LED on for one second and then off for one second.



The Micropython code for this example is

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

setScreenColor(0x111111)
while True:
  M5Led.on()
  wait(1)
  M5Led.off()
  wait(1)
  wait_ms(2)
```

HATS

First coined by the Raspberry Pi community, H.A.T. stands for Hardware Attached on Top and was a looses specification for solder less hardware add-ons that are connected on top of development boards. Due to the small size of the M5Sticks, M5Stack modules and only a limited number of units can be used. To get around the hardware limitations of the M5Sticks, a dedicated range of H.A.Ts have been developed to expand the capabilities of the M5Stick.

ENV Hat

Get Pressure,

Gets the current air pressure.

Get Temperature,

Gets the current temperature.

Get Humidity,

Gets the current Humidity.

Example

In this example I am using three labels to show the Pressure, temperature and, humidity from the environmental sensor.

The Micropython code for this is as follows.

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

setScreenColor(0x222222)
env0 = unit.get(unit.ENV, unit.PORTA)

label0 = M5TextBox(74, 52, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
label1 = M5TextBox(67, 85, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
label2 = M5TextBox(89, 122, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(env0.pressure))
  label1.setText(str(env0.temperature))
  label2.setText(str(env0.humidity))
  wait_ms(2)
```

P.I.R Hat,
Get hat_pir Status,

Returns True or False if the PIR detects anything.

Example
In the following example I am using a label to show the status of the P.I.R on the Stick C's screen.

The Micropython code for this example is as follows:

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

setScreenColor(0x111111)

hat_pir0 = hat.get(hat.PIR)

label0 = M5TextBox(28, 85, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(hat_pir0.state))
  wait_ms(2)
```

Speaker

Hat_spk0 Play Tone

The Play Tone block allows users to play musical notes and set the duration in beats or fractions of Beats.

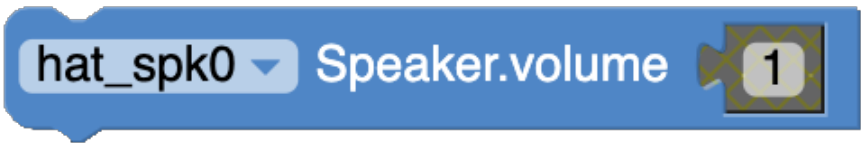
Hat_spk0 Speaker.beep

The hat_spk0 speaker Beep block allows us to make sounds by setting the frequency of the

sound and the duration of the sound.

Hat_spk0 Speaker Volume

The hat_spk0 Speaker volume allows us to set the volume of the speaker sounds.



Example.

NCIR
NCIR Read

Returns a value from the N.C.I.R Hat Sensor.

Example
In this example I have added a label that gets its text value from the NCIR unit.


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

setScreenColor(0x222222)
ncir0 = unit.get(unit.NCIR, unit.PORTA)

label0 = M5TextBox(132, 61, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(ncir0.temperature))
  wait_ms(2)
```

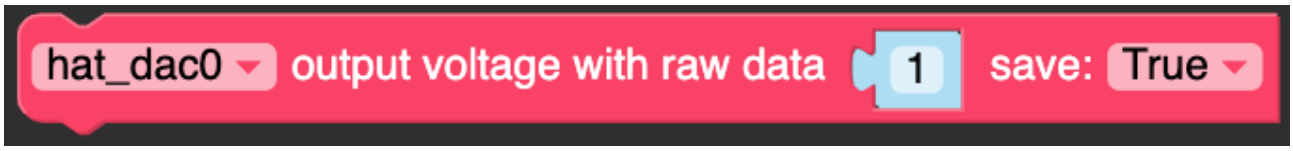
DAC

hat_dac0 Output Voltage

Gets a voltage reading from 0 to 3.3 volts If save it set to true then the data will be written to the M5Sticks E.E.P.R.O.M.

hat_dac0 Output Voltage with Raw Data.

Gets a raw data reading of 0 to 4096. If save it set to true then the data will be written to the



M5Sticks E.E.P.R.O.M.

ADC

Returns an analogue reading from the Analogue to Digital Converter Hat.

Example

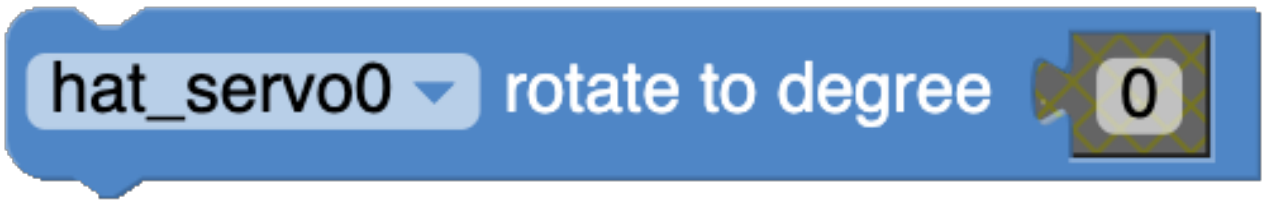
In this example I am using a label to display the voltage measured from the ADC.

The Micropython code for the sensor is

```
adc0 = unit.get(unit.ADC, unit.PORTA)

label0 = M5TextBox(90, 88, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

label0.setText(str(adc0.voltage))
```

Hat Servo

Rotate Servo to Position.



Rotates the servo to a position defined with a maths number block.

Write Milliseconds

Not all servos are made the same and there are times where the servos rotation position will not match the angle set in the "Rotate to Degree" block. To fine tune this position we can use this block as it sets the servo position in microseconds with a maths number block. In order to find the position in milliseconds we will need to look at the individual servo specifications and data sheets for the numbers.

8 Servos Hat.

The eight Servos Hat is a robot chase that allows you to control up to eight servos at the same time.

Rotate Servo Position

Write Servo Millisecond

Set RGB Bar Colour

Set RGB Bar Colour (R,G,B,)

Puppy C

The Puppy C chase is a walking robot chase designed for the M5StickC and comes with four SG90 servos and a 16340 battery.

Set Servo position

Set servo position 0 legs 0, 1, 2, 3

Joystick

BugC

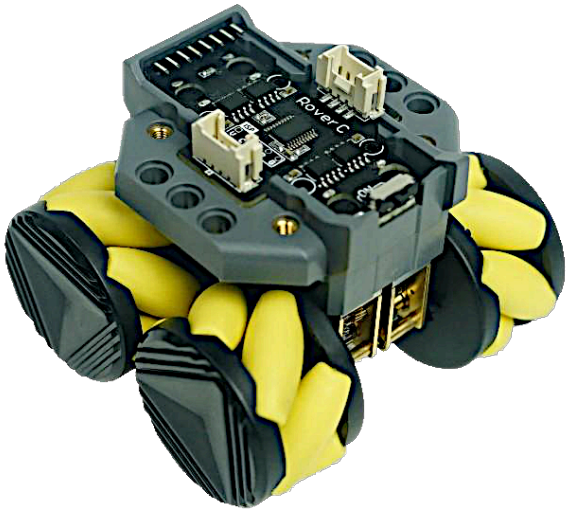
Finger

BeetleC

YUN

JoyC

RoverC



The RoverC is a mobile omnidirectional chassis designed to connect to the M5StickC using the eight pin HAT connector. The Chassis communicates with the M5StickC over I2C and is control by its own STM32F030F4 controller.

The chassis features four independently controlled N20 motors connected to Mecanum wheels which can provide up to twelve different modes of movement, a 750mah 16340 battery to provide additional power for the motors, two I2C ports, four M3 screw points, and six mounting hole for adding LEGO compatible parts.

UIFlow Blocks.

RoverC

When programming the RoverC in UIFlow there are only three blocks provided to control the chassis. These three blocks are as follows.

Set Wheel Pulse

Set Wheel Pulse can be used to set a pulse length between 20 and 100 using a maths block or with a variable to only one wheel turning the wheel in a clockwise rotation. Sending a pulse length between -20 and -100 will result in the wheel turning in an anticlockwise rotation. Sending a pulse to only one wheel will result in a normal turning action.

The Micropython code for this block is as follows.

```
hat_roverc0.SetPulse(0, 0)
```

Set Wheel Pulse (four wheels)

Set Wheel Pulse can be used to control all four wheels at once which give a greater range of movement including turning on the spot and sliding to a direction. As with the previous block, the pulse length can be set between 20 and 100 using a maths block or with a variable turning the wheel in a clockwise rotation. Sending a pulse length between -20 and -100 will result in the wheel turning in an anticlockwise rotation.

The Micropython code for this block is as follows.

```
hat_roverc0.SetAllPulse(0, 0, 0, 0)
```

RoverC

Set RoverC Speed

The Set RoverC Speed block is used to control the speed that the RoverC move at. The speed length is between 20 and 100 using a maths block or with a variable.
The Micropython code for this block is as follows.

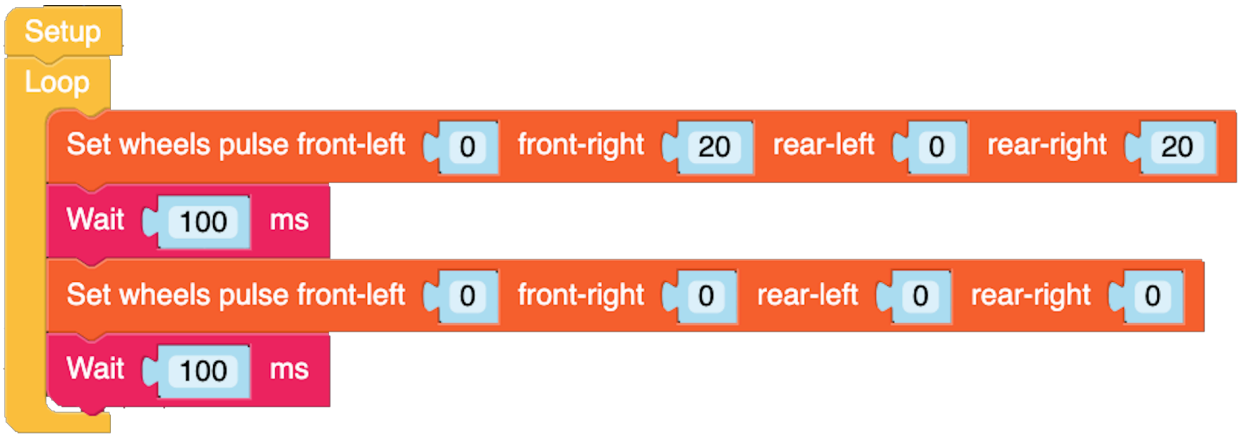
```
hat_roverc0.SetSpeed(0, 0, 0)
```

Movement

Because the RoverC uses Mecanum wheels instead of normal wheels movement needs to be programmed in what would to some look like a peculiar way.

Turn Left.

To make the RoverC turn left or anticlockwise you would use the code blocks as follows.



RoverC

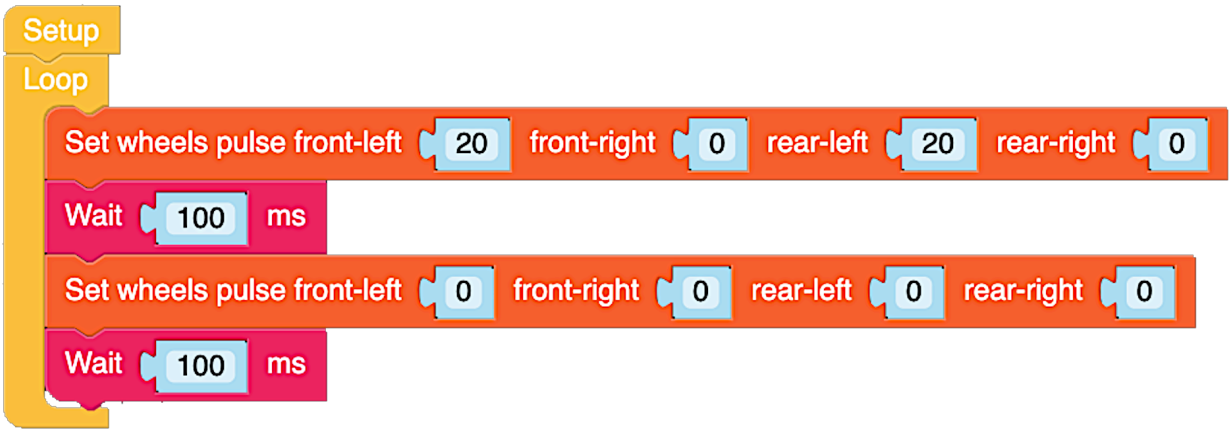
The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(0, 20, 0, 20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

In this example I send a value of 20 to both the right hand motors and then after 100ms send a value of 0 to stop the motors.

Please note: If the motors do not receive instructions to se the motors to 0 before being switched off, the RoverC will start moving as soo as it is switched on again using the onboard power switch.

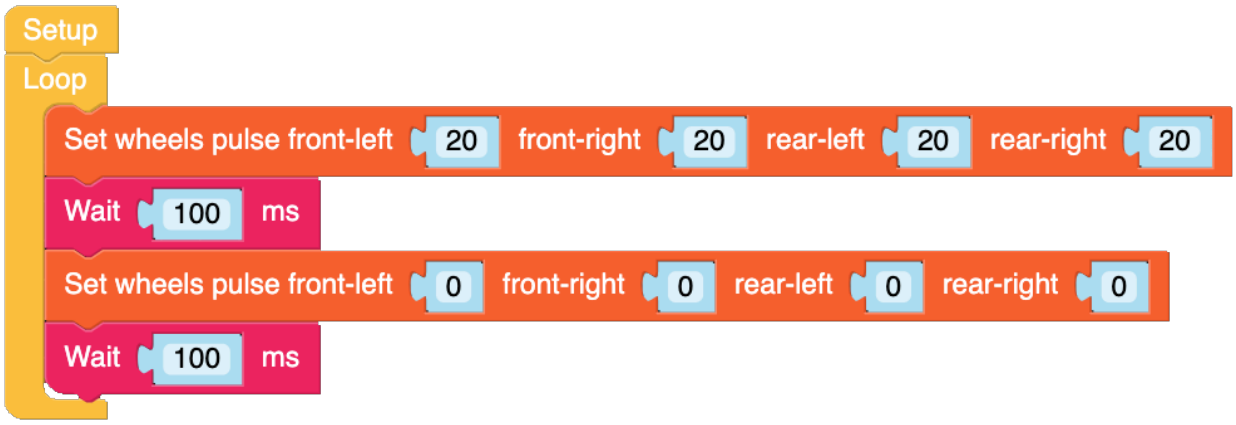
Turn Right



To make the RoverC turn right or clockwise you send value to the left hand motors.
In this example I send a value of 20 to both the left hand motors and then after 100ms send a value of 0 to stop the motors.
The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(20, 0, 20, 0)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

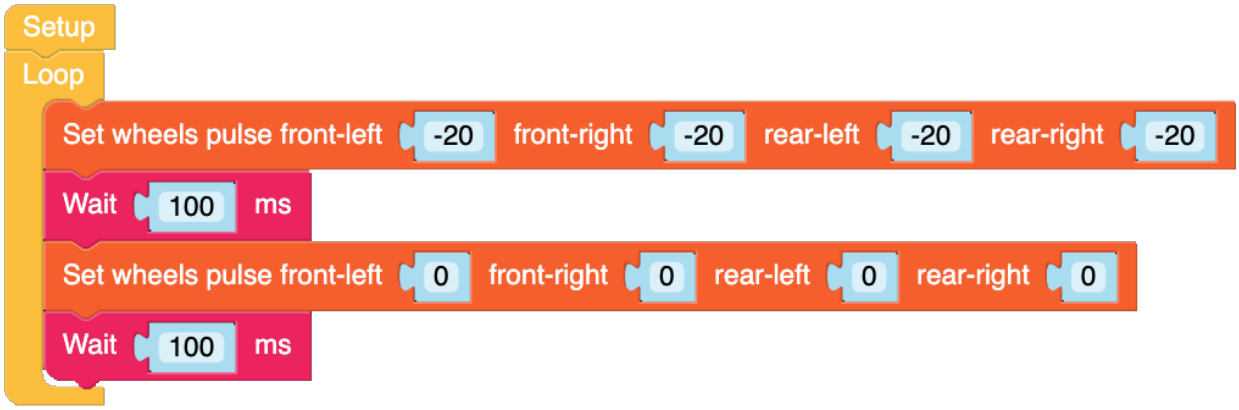
Move Forwards



In order to make the RoverC drive forwards all we need to do is set the same value to all for wheels at the same time.
The Micropython code for this is.

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(20, 20, 20, 20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

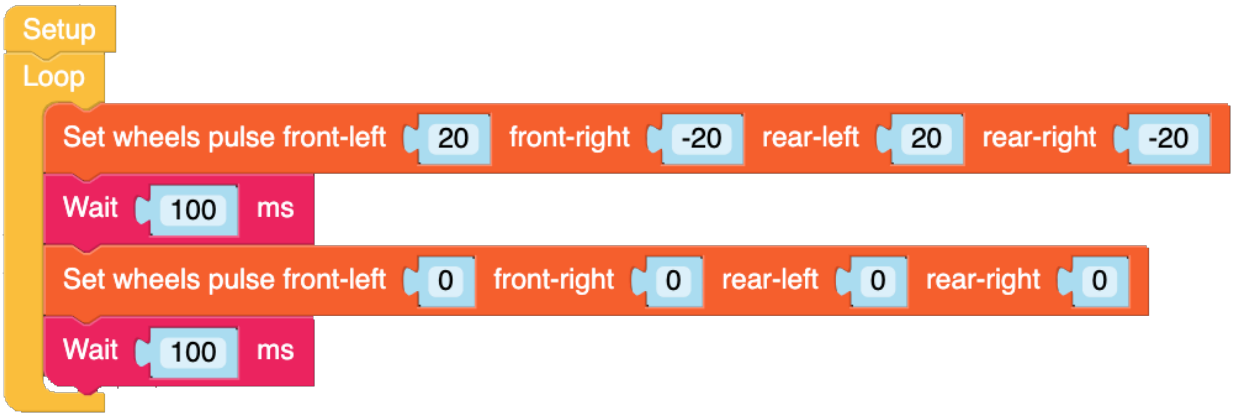
Move Backwards



To make the RoverC reverse we just need to send a minus value to all four motors at once. The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(-20, -20, -20, -20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

Rotate Clockwise

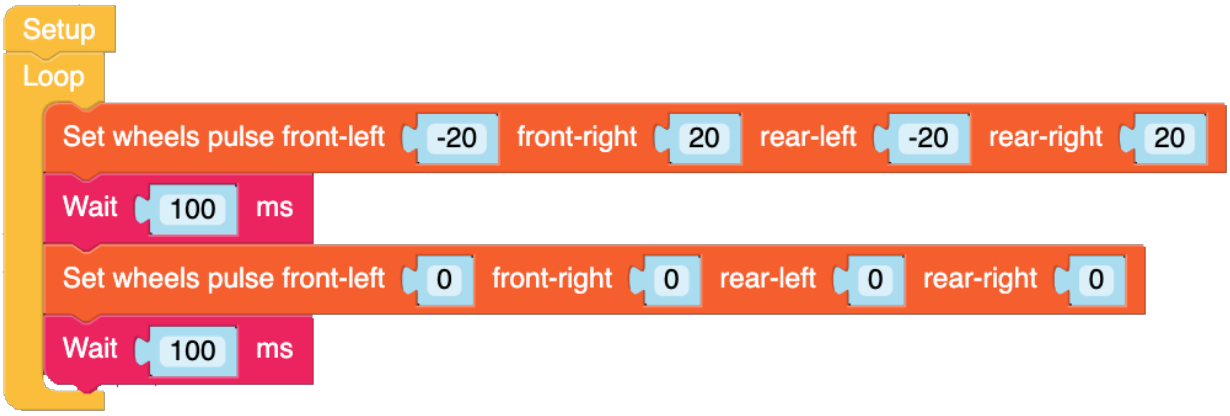


Because the RoverC has four independently motorised wheels the RoverC can rotate on the spot like a tracked vehicle. In order to make the RoverC Rotate clockwise we set the left wheels with a positive value and the right wheels with a negative value.Rotate AntiClockwise
The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(-20, 20, -20, 20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

RoverC

Rotate Anti-Clockwise



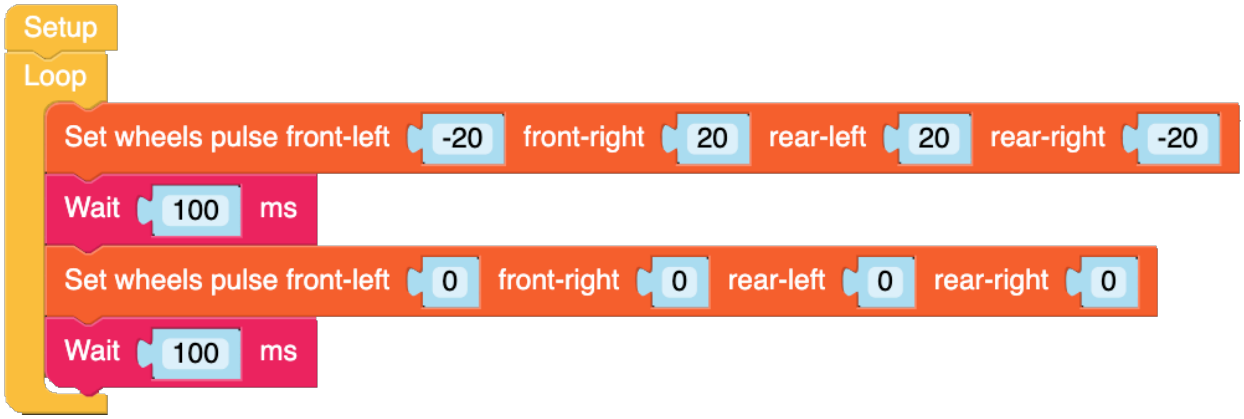
The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(20, -20, 20, -20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

RoverC

As well as rotating clockwise or anticlockwise, the Mecanum wheels also allow the RoverC to Slide around while still facing the same direction.

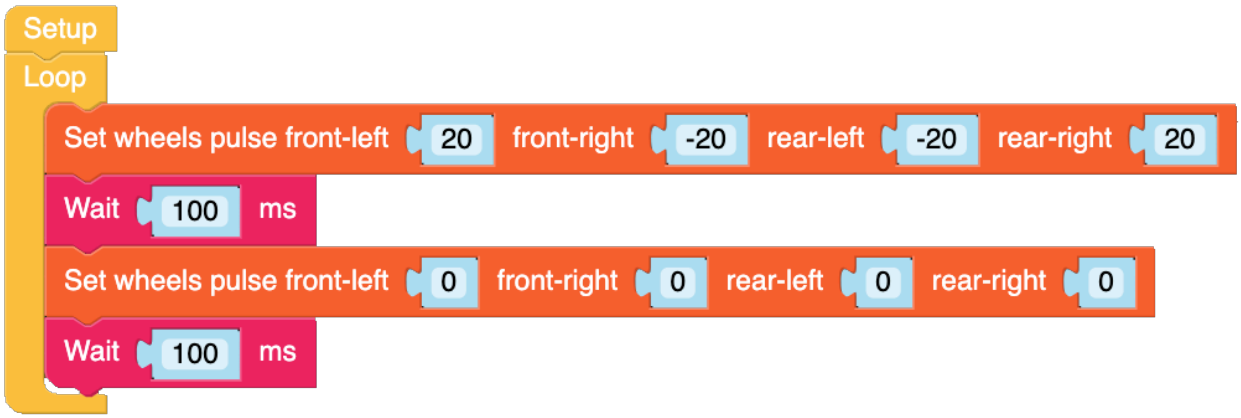
Slide Left



The Micropython code for this is

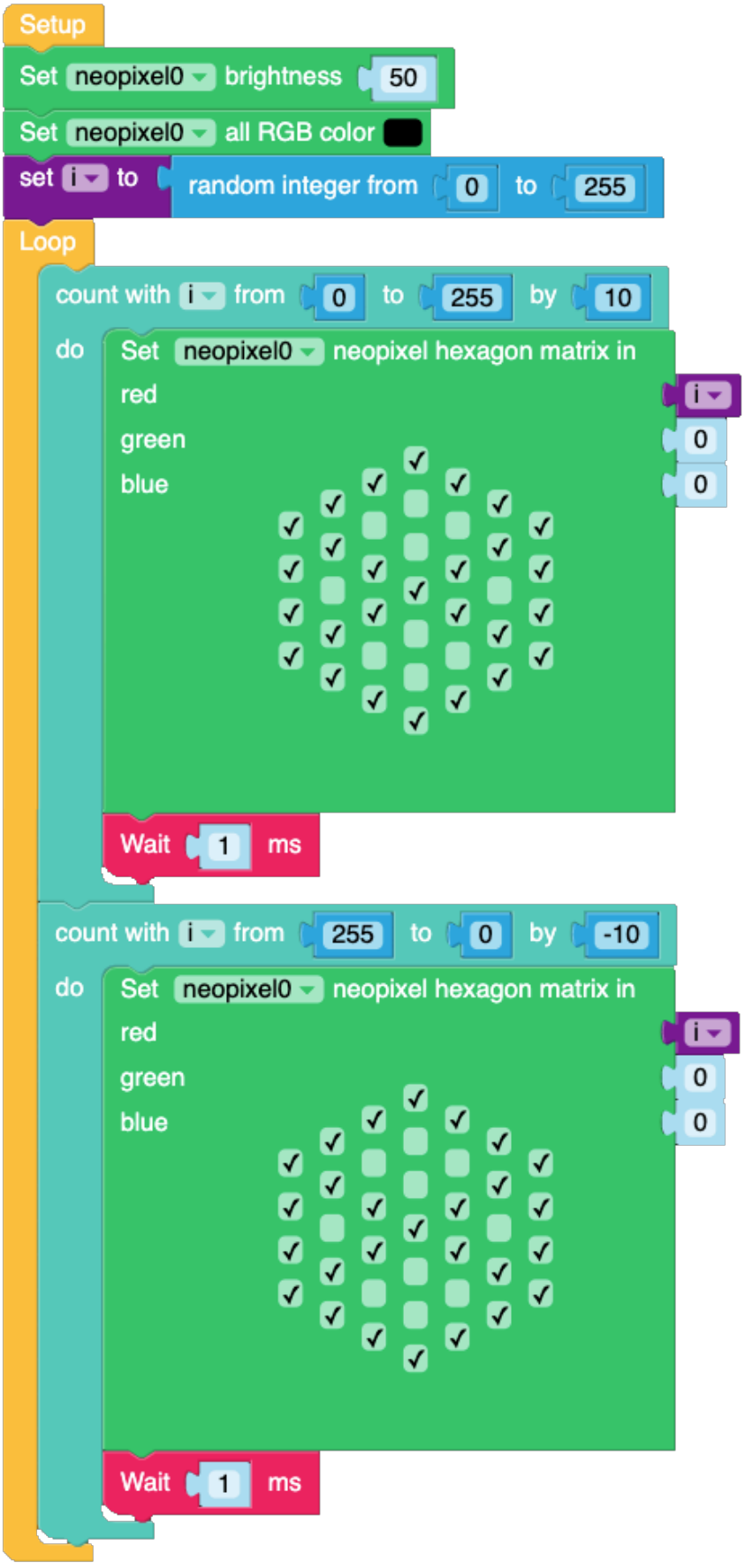
```
from m5stack import *
from m5ui import *
from uiflow import *
import hat

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(-20, 20, 20, -20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

Slide Right

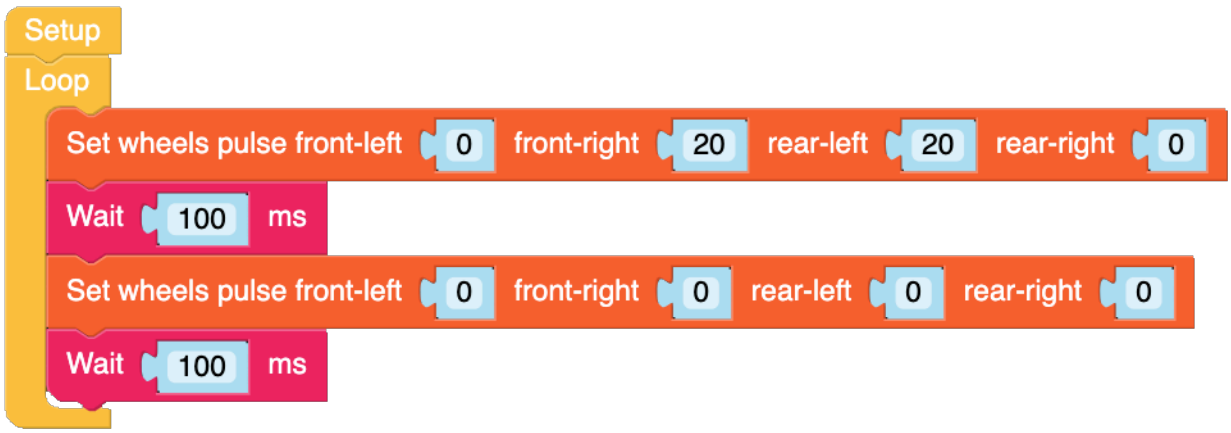


The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(20, -20, -20, 20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```


Setup
Set neopixel0 brightness 50
Set neopixel0 all RGB color
set i to random integer from 0 to 255
Loop
count with i from 0 to 255 by 10
do
Set neopixel0 neopixel hexagon matrix in
red
green
blue
i
0
0
Wait 1 ms
count with i from 255 to 0 by -10
do
Set neopixel0 neopixel hexagon matrix in
red
green
blue
i
0
0
Wait 1 ms

Slide Forward Left

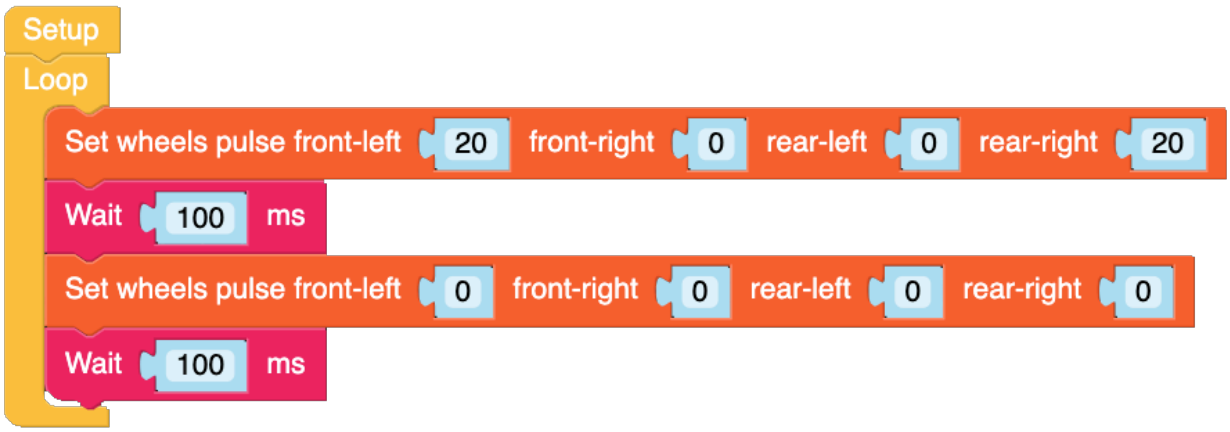


The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(0, 20, 0, 20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

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

setScreenColor(0x111111)
neopixel0 = unit.get(unit.NEOPIXEL, unit.PORTA, 38)
import random
i = None

neopixel0.setBrightness(50)
neopixel0.setColorAll(0x000000)
i = random.randint(0, 255)
while True:
  for i in range(0, 256, 10):
    neopixel0.setColorFrom(1, 6, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(8, 10, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(12, 13, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(15, 16, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColor(19, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(22, 23, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(25, 26, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(28, 30, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(32, 37, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    wait_ms(1)
  for i in range(255, -1, -10):
    neopixel0.setColorFrom(1, 6, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(8, 10, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
```

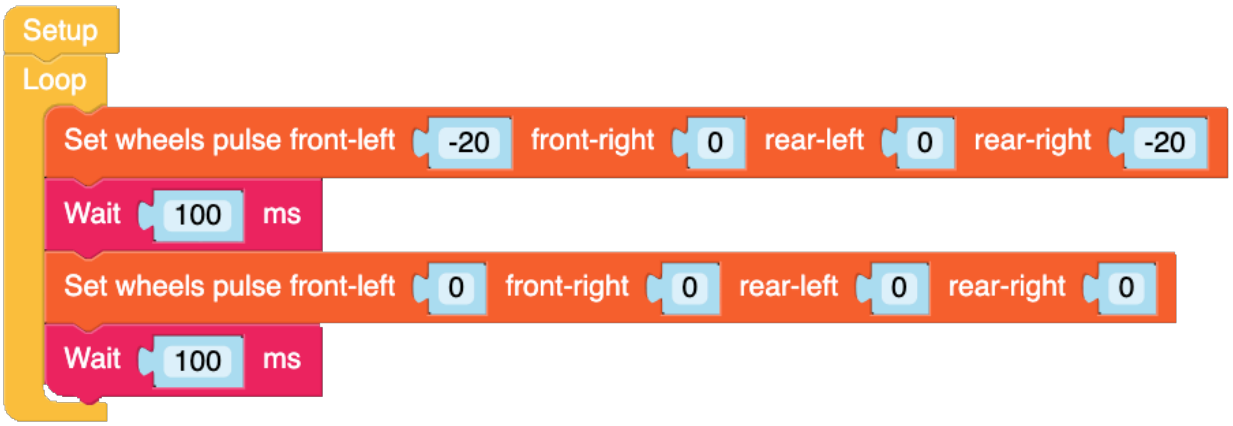
Slide Forward Right



The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(0, 20, 20, 0)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

Slide Backwards Left



The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(-20, 0, 0, -20)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

Slide Backwards Right

The Micropython code for this is

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

setScreenColor(0x111111)

hat_roverc0 = hat.get(hat.ROVERC)

while True:
  hat_roverc0.SetAllPulse(0, -20, -20, 0)
  wait_ms(100)
  hat_roverc0.SetAllPulse(0, 0, 0, 0)
  wait_ms(100)
  wait_ms(2)
```

TOF

Thermal Camera

Units

The M5Stack and M5Stick family share a series of add ons called Units. These units connect to the four pi
I2C connector the M5Stack core or M5Stick.

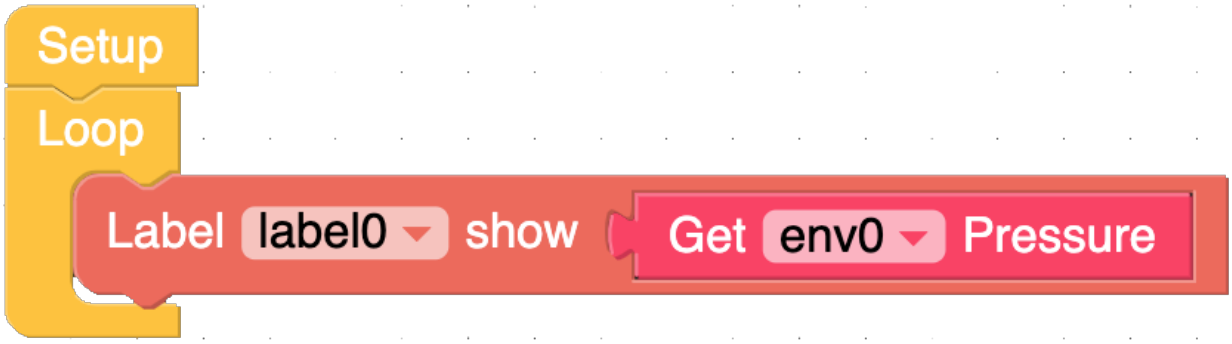
In this chapter I will show you the blocks dedicated to specific units and give basic examples on how to
work with them.

Environmental,
Get Pressure,

The Get Pressure block is a value block that returns the air pressure value read by the sensor. In order to use the value, we need to combine this block with function blocks.

```
env0.pressure
```

The Micropython code for this block is:
Example.



In this example I am just using a text block to display the pressure on the M5Stack or M5Sticks screen.

The Micropython code for this example is as follows.

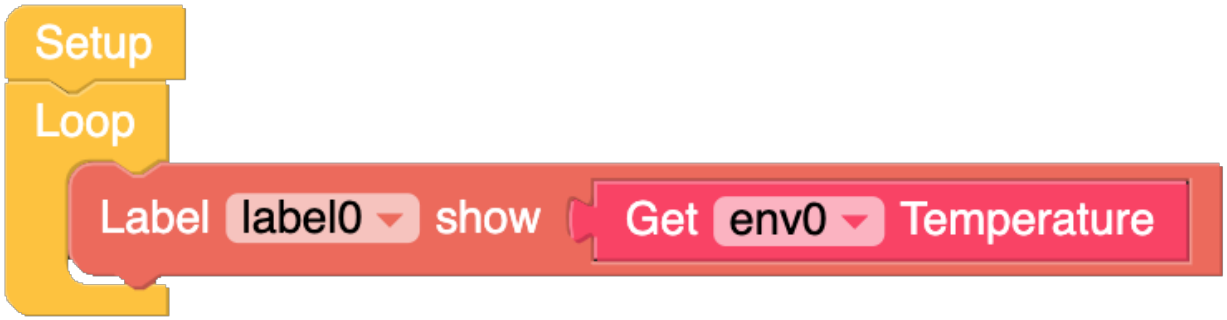
```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(0x222222)
env0 = unit.get(unit.ENV, unit.PORTA)

label0 = M5TextBox(152, 65, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(env0.pressure))
  wait_ms(2)
```

Get Temperature,

The Get Temperature block is a value block that returns the air temperature value read by the sensor. In order to use the value, we need to combine this block with function blocks.
The Micropython code for this block is as follows.

```
env0.temperature
```

Example
The following example replaces the Pressure block with the temperature block to show the current air temp read by the sensor.



The Micropython code is as follows.

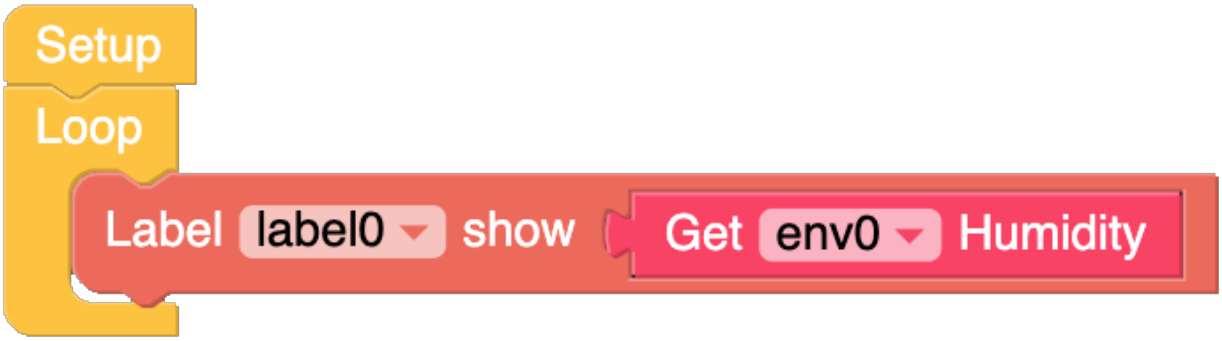
```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(0x222222)
env0 = unit.get(unit.ENV, unit.PORTA)

label0 = M5TextBox(103, 67, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(env0.temperature))
  wait_ms(2)
```

Get Humidity,

The Get Humidity block is a value block that returns the humidity value read by the sensor. In order to use the value, we need to combine this block with function blocks.
The Micropython code for this block is as follows.

```
env0.humidity
```

Example
As with the previous example I have just replace the Pressure block with the humidity block.



The Micropython code is as follows.

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

setScreenColor(0x222222)
env0 = unit.get(unit.ENV, unit.PORTA)

label0 = M5TextBox(103, 67, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(env0.humidity))
  wait_ms(2)
```

Angle

Angle,
Get Angle,

Returns the value generated by the angle sensor. Values are from 0 to 4096.

```
angle0.read()
```

Example

In this example I have used a label to show the value read from the sensor.

And the Micropython code for this is:

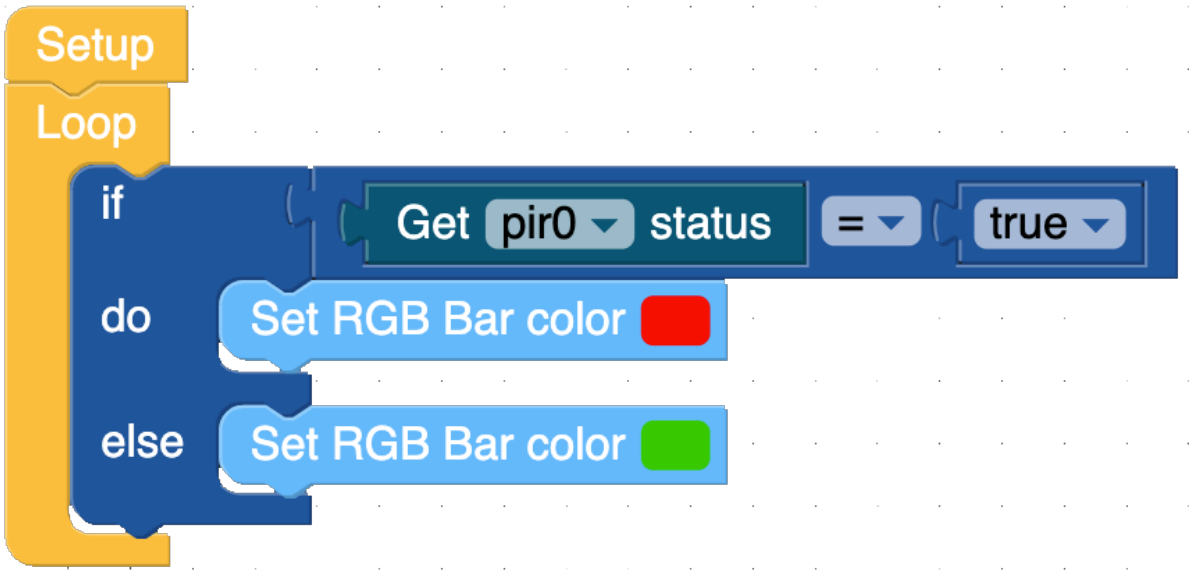
```
angle0 = unit.get(unit.ANGLE, unit.PORTB)

label0 = M5TextBox(90, 88, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

label0.setText(str(angle0.read()))
```

PIR,
Get PIR Status,

Get PIR Status block is a value block that returns true if the sensor detects I.R light given off of people or returns false if noting gets detected.
The Micropython code for this block is:

```
pir0.state
```

Example



In this demo the PIR is connected to Port B of the M5Go. When it detects a human, the lights on the base turn red and only clear when a human is no longer detected.
The Micropython code for this is as follows:

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

setScreenColor(0x222222)
pir0 = unit.get(unit.PIR, unit.PORTB)
while True:
  if (pir0.state) == True:
    rgb.setColorAll(0xff0000)
  else:
    rgb.setColorAll(0x33cc00)
  wait_ms(2)
```

RGB LED

RGB LED,

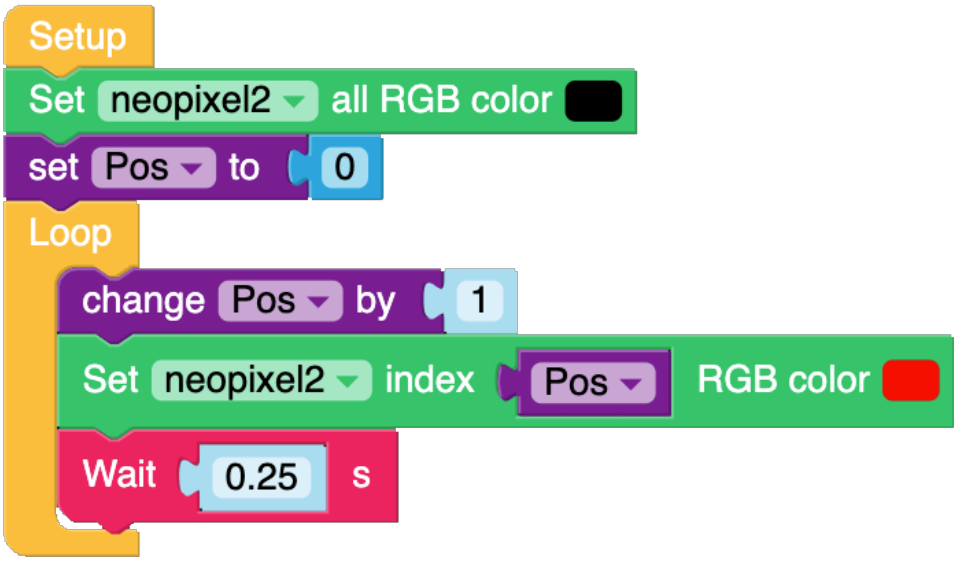
There are several units produced by M5Stack that fall into the RGB LED unit category, but also any RGB product that uses the WS2812b or the SK6812 RGB LEDS can be controlled with the blocks from this category.
The current available products that use these blocks are as follows

Hex RGB Unit
The HEX RGB unit contains 37 SK6812 LEDs.

RGB LED Unit
The RGB LED Unit contains three SK6812 LEDs.

RGB Flex Strip
The RGB Flex strip is a self adhesive strip of SK6812 LEDs that is available in various lengths from M5Stack and has a grove connector on each end.
RGB Cat Ear
The RGB Cat Ear is a hand band containing One Hundred and Eighteen SK6812 LEDs.

NeoFlash Light Box.
The Neoflash Light Box contains One Hundred and Ninety two SK6812 LEDs in a grid of eight rows twenty fours LED



In order to understand how the RGB strips and arrays are wired, I created the following UIFlow program.
This program lights and led in a string then moves on to light the next led in sequence showing how they are arranged. This program is useful for testing if arrays have even rows reversed. In

the case that they are reversed, the lens will light from left to right then on the next row they will start on the right and move left.

Set Index RGB Colour,

Sets an individual WS2812b LED colour using the Colour picker. Index is a value that can be set

```
neopixel2.setColor(1, 0xff0000)
```

with a defined number or a variable and is a reference to the Ws2812b led is a position on a array or strip.
The Micropython code for this block is:


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

setScreenColor(0x222222)
neopixel2 = unit.get(unit.NEOPIXEL, unit.PORTA, 192)

neopixel2.setColor(1, 0xff0000)
```

Example
In this example I have set the first WS2812b led to red.
And the Micropython code for this is as follows:

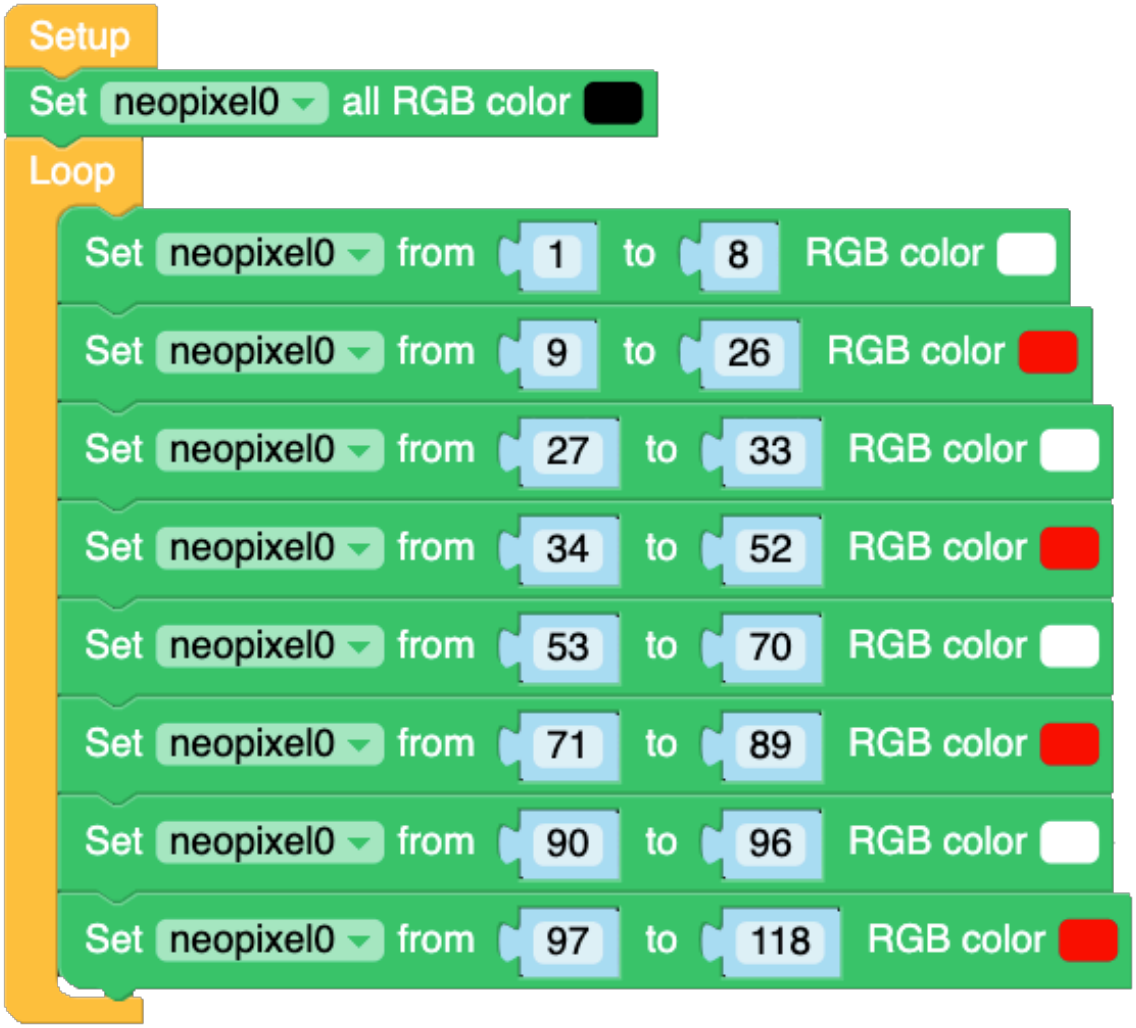
Set RGB Range to colour,

Sets the colour of a range of WS2812b led's using the colour picker. Values can be defined with maths blocks or with variables to specify the led's.

```
neopixel0.setColorFrom(1, 5, 0xff0000)
```

The Micropython code for this is as follows:

Example



In this example I have used the range blocks to just make the caters glow red.
The Micropython code for this is as follows:

RGB LED

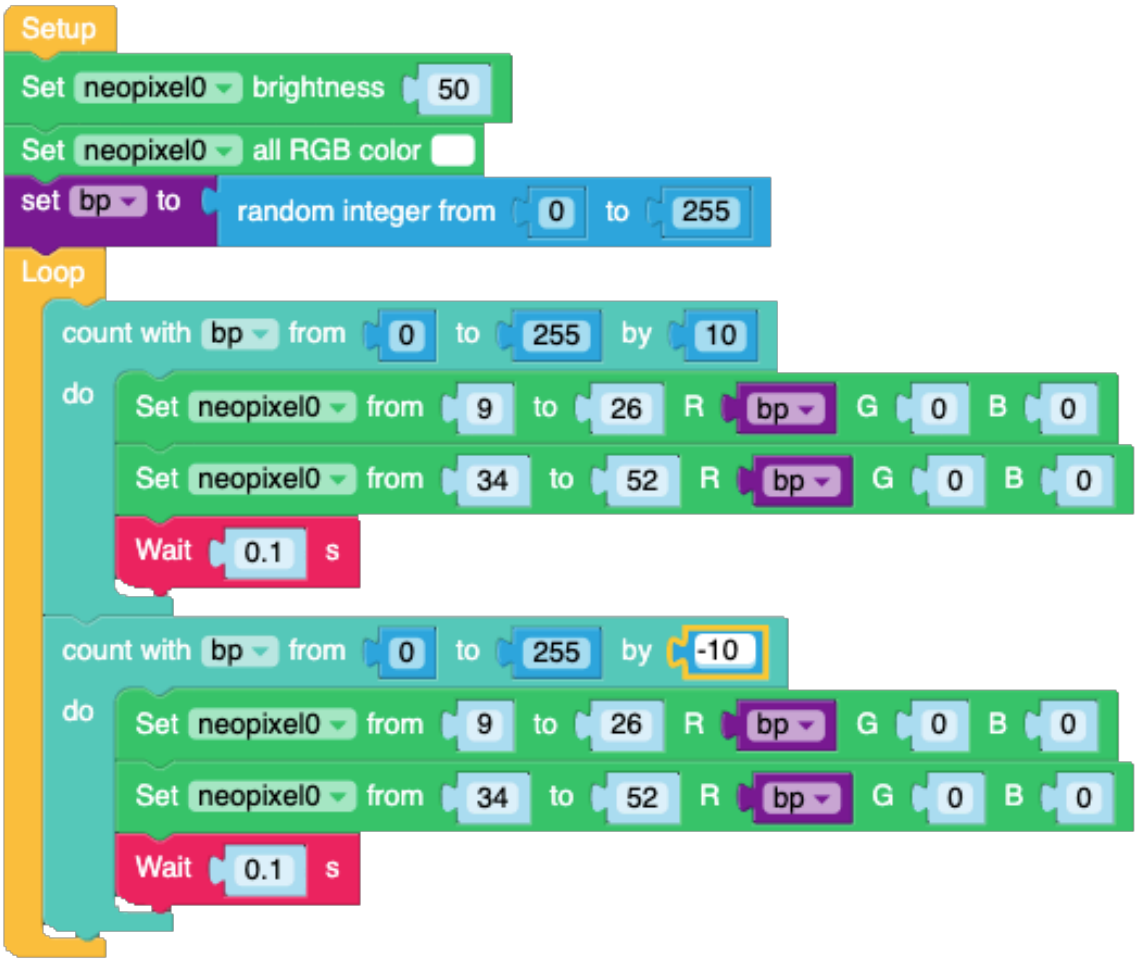
```
from m5stack import *
from m5ui import *
import units

clear_bg(0x111111)
neopixel0 = units.RGB_Multi(units.PORTA, 118)

neopixel0.setColorAll(0x000000)
while True:
  neopixel0.setColorFrom(1, 8, 0xffffff)
  neopixel0.setColorFrom(9, 26, 0xff0000)
  neopixel0.setColorFrom(27, 33, 0xffffff)
  neopixel0.setColorFrom(34, 52, 0xff0000)
  neopixel0.setColorFrom(53, 70, 0xffffff)
  neopixel0.setColorFrom(71, 89, 0xff0000)
  neopixel0.setColorFrom(90, 96, 0xffffff)
  neopixel0.setColorFrom(97, 118, 0xff0000)
  wait(0.001)
```

Set WS2812b Range to RGB colour,

Sets the colour of a range of WS2812b led's using individual Red, Green, and Blue values.Values can be defined with maths blocks or with variables to specify the led's.
The Micropython code for this is as follows.

```
neopixel0.setColorFrom(9,26,(bp << 16) | (0 << 8) | 0)
```

Example.



In this example I have set all the RGB LEDs to white and am making the ears fade in and out between red and white.

RGB LED

The Micropython code for this is:

```
from m5stack import *
from m5ui import *
import units

clear_bg(0x111111)
neopixel0 = units.RGB_Multi(units.PORTA, 118)
import random
bp = None
neopixel0.setBrightness(50)
neopixel0.setColorAll(0xffffff)
bp = random.randint(0, 255)
while True:
  for bp in range(0, 256, 10):
    neopixel0.setColorFrom(9,26,(bp << 16) | (0 << 8) | 0)
    neopixel0.setColorFrom(34,52,(bp << 16) | (0 << 8) | 0)
    wait(0.1)
  for bp in range(0, 256, 10):
    neopixel0.setColorFrom(9,26,(bp << 16) | (0 << 8) | 0)
    neopixel0.setColorFrom(34,52,(bp << 16) | (0 << 8) | 0)
    wait(0.1)
  wait(0.001)
```

Set all WS2812b Colour,

Sets all WS2812b LEDs to the same colour using the colour picker. We can use this block in the setup phase before the loop to set all the pixels in an array to black or off.
The Micropython code for this is:

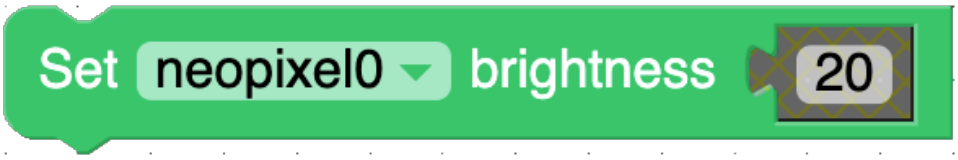
```
neopixel0.setColorAll(0xff0000)
```

Example

Here I have used the Set All block in the setup section to set all the RGB led's to black or off.
The Micropython code for this demo code is:

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

setScreenColor(0x111111)
neopixel0 = unit.get(unit.NEOPIXEL, unit.PORTA, 192)

neopixel0.setColorAll(0x000000)
```

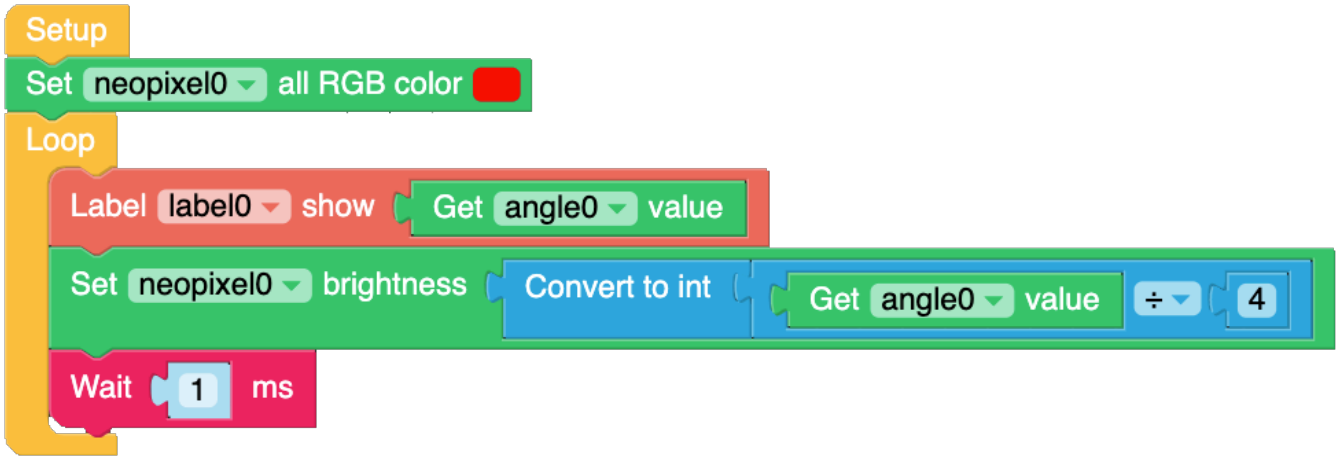
Set WS2812b Brightness,



Set the brightness levels of all WS2812b LEDs Brightness can only have a value between 0 and 100.
The Micropython code for this block is:

```
neopixel0.setBrightness(20)
```

Example.
In this example I have use the Angle Unit to control the brightness of the RGB Led's. The Get Angle block returns values as a string with a value from 0 to 1024. In order to use this value as to control the brightness which only has values from 0 to 256 we have to divide the value by four and then use the **convert to int** block to generate the value that the **set brightness** block can use.



RGB LED

And the Micropython code for this is:

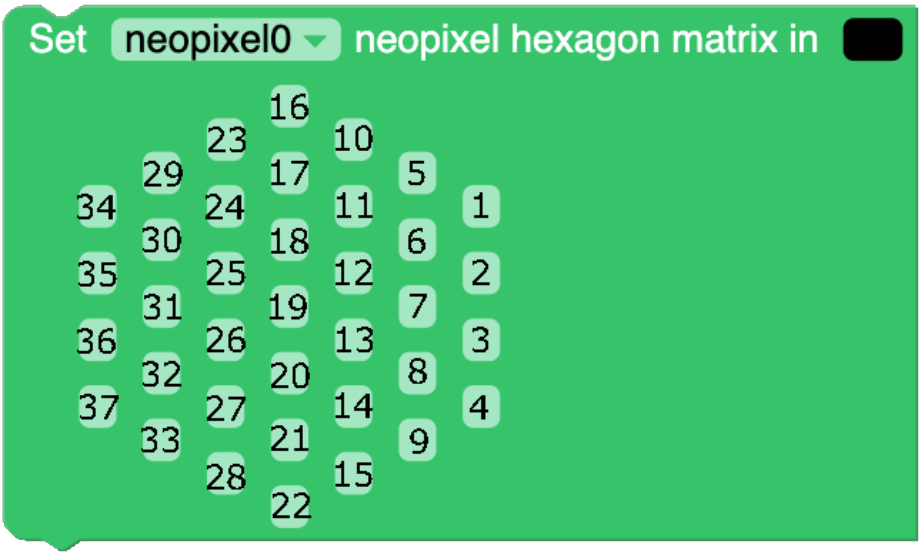
```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(0x222222)
neopixel0 = unit.get(unit.NEOPIXEL, unit.PORTA, 256)
angle0 = unit.get(unit.ANGLE, unit.PORTB)

label0 = M5TextBox(79, 63, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

neopixel0.setColorAll(0xff0000)
while True:
  label0.setText(str(angle0.read()))
  neopixel0.setBrightness(int(((angle0.read()) / 4)))
  wait_ms(1)
  wait_ms(2)
```

Set WS2812b Hex Colour,

Used to set the colour of the individual WS2812b LEDs on the Neohex using the colour picker. To select an LED to set to the selected colour, click on a light green pad and a tic will appear to show that the position is now selected.
The Micropython code for this block is:

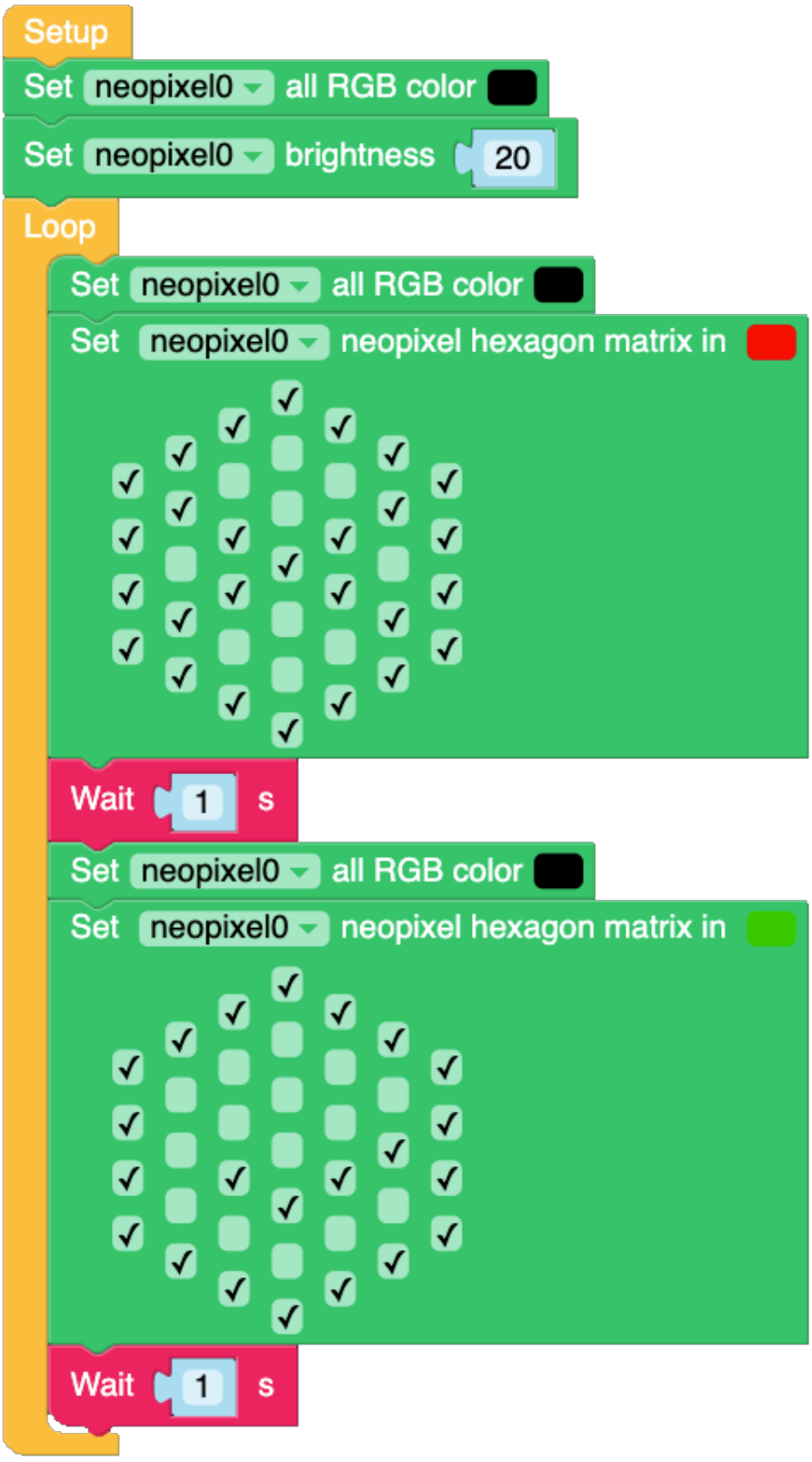
```
neopixel0.setColor(0, 0x000000)
```

The numbers in the brackets are the LED position followed by the hex colour code. The LED positions are shown in the following image:



RGB LED

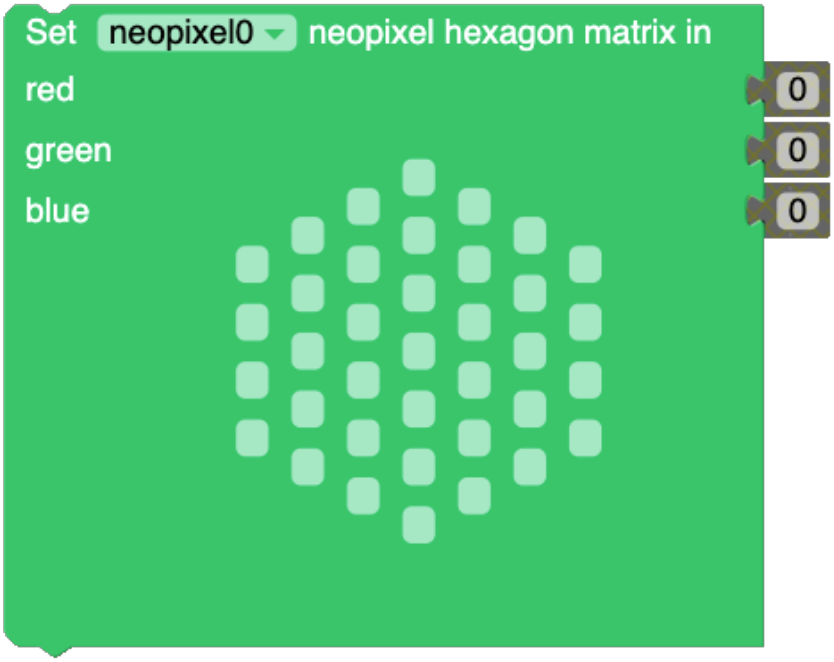
Example
In this example, when run the Neohex flashes between a red X and a green tick.



And here is the Micropython code for this example:

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

setScreenColor(0x111111)
neopixel0 = unit.get(unit.NEOPIXEL, unit.PORTA, 38)

neopixel0.setColorAll(0x000000)
neopixel0.setBrightness(20)
while True:
  neopixel0.setColorAll(0x000000)
  neopixel0.setColorFrom(1, 6, 0xFF0000)
  neopixel0.setColorFrom(8, 10, 0xFF0000)
  neopixel0.setColorFrom(12, 13, 0xFF0000)
  neopixel0.setColorFrom(15, 16, 0xFF0000)
  neopixel0.setColor(19, 0xFF0000)
  neopixel0.setColorFrom(22, 23, 0xFF0000)
  neopixel0.setColorFrom(25, 26, 0xFF0000)
  neopixel0.setColorFrom(28, 30, 0xFF0000)
  neopixel0.setColorFrom(32, 37, 0xFF0000)
  wait(1)
  neopixel0.setColorAll(0x000000)
  neopixel0.setColorFrom(1, 5, 0x33cc00)
  neopixel0.setColor(7, 0x33cc00)
  neopixel0.setColorFrom(9, 10, 0x33cc00)
  neopixel0.setColor(13, 0x33cc00)
  neopixel0.setColorFrom(15, 16, 0x33cc00)
  neopixel0.setColor(20, 0x33cc00)
  neopixel0.setColorFrom(22, 23, 0x33cc00)
  neopixel0.setColor(26, 0x33cc00)
  neopixel0.setColorFrom(28, 29, 0x33cc00)
  neopixel0.setColorFrom(33, 37, 0x33cc00)
  wait(1)
  wait_ms(2)
```

Set WS2812b Hex (Variable) Colour,



Used to set the colour of the individual WS2812b LEDs on the Neohex using individual Red, Green and Blue values.

The Micropython code is the same as in the previous example however this block allows us to control the values with numbers.

```
neopixel0.setColor(0, 0x000000)
```

Example

In the following example I have used the fading demo from the RGB range block to brighten and darken the red cross.

```
    neopixel0.setColorFrom(12, 13, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(15, 16, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColor(19, int('0x%02x%02x%02x' % (round(min(100, max(0, i)) * 2.55),
round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(22, 23, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(25, 26, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(28, 30, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    neopixel0.setColorFrom(32, 37, int('0x%02x%02x%02x' % (round(min(100, max(0,
i)) * 2.55), round(min(100, max(0, 0)) * 2.55), round(min(100, max(0, 0)) * 2.55))))
    wait_ms(1)
  wait_ms(2)
```

The Micropython code for this is:

Joystick,

Get X,

Returns the joysticks X value.

Get Y,

Returns the joysticks Y Value.

Is Pressed,

Returns the state of the joysticks button when pressed.

Reverse X

Returns an inverted value of the X position.

Reverse Y

Returns an inverted value of the Y position.

Light,
Get Light Analogue Value,

Returns the analogue value from the light sensor.

Get Light Digital Value,

Returns the digital value from the light sensor.

Earth,

Get Earth Analogue Value,

Returns the analogue value from the earth sensor.

Get Earth Digital Value,

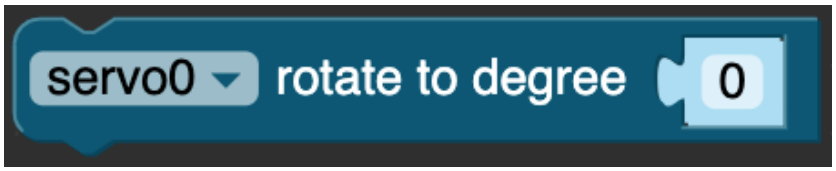
Returns the digital value from the earth sensor.

Makey,

Get Value,

Get all Value,

Servo,



Servo Rotate to Degree,
Tells the servo to move to a specified position.
Servo Speed,

Sets the move speed of the servo.

Weight,

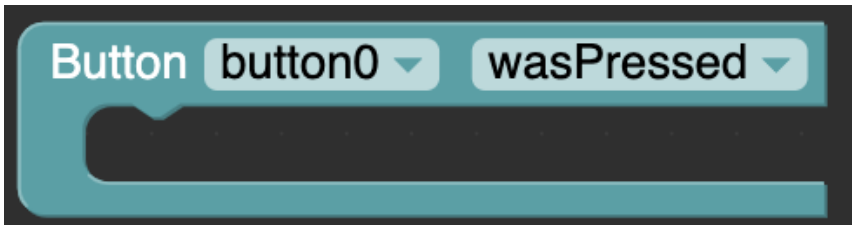
Zero Weight,
Sets the values coming from the sensor to zero.
Get Weight,

Get the weight from the sensor.

Get Raw Data
Gets the raw 24 bit value data from the sensor.

Button,

Button Loop,

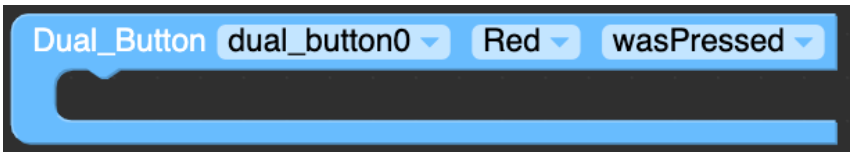


The button loop continuously watches the button unit and then runs the code inside it when it detects a triggering even.
The four events the button loop waits for are as follows.
Was Pressed - This is the normal mode of operation and waits until a single short press of the button is detected before running 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.

Button Action,

The obtain button block works in a similar mode to the Button loop but instead of running code placed inside it, it is a value block that tells other code if a button event has been triggered.

Duel Button,
Duel_Button Loop,

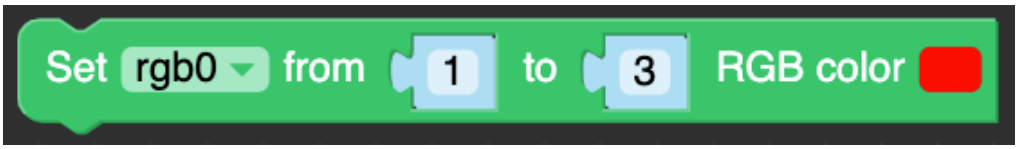


The duel button loop continuously watches the two buttons on the duel button unit and then runs the code inside it when it detects a triggering event on either of the buttons.
The four events the button loop waits for are as follows.
Was Pressed - This is the normal mode of operation and waits until 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.
Button Action,

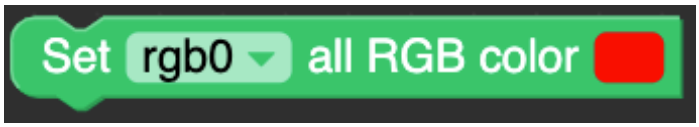
The obtain duel button block works in a similar mode to the Button loop but instead of running code placed inside it, it is a value block that tells other code if a one of the two buttons have been triggered.

RGB,

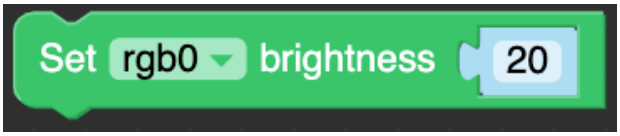
Set RGB Bar Colour,
Sets one of the three R.G.B LED colours using the colour picker.



Set RGB Bar Colour R,G,B,
Sets more than one R.G.B LED colours using the colour picker.



Set RGB all Colour,
Sets all the R.G.B LED colours using the colour picker.



Set RGB Brightness,
Sets the Brightness of the R.G.B LEDs.

Relay,

Set Relay On,

Sets the relay unit in to the on position.

Set Relay Off,

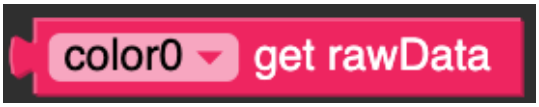
Sets the relay unit in to the off position.

Heart Unavailable,
Not available in UIFlow at present.

ADC,
ADC Read,

Returns an analogue reading from the Analogue to Digital Converter Unit.

Colour,



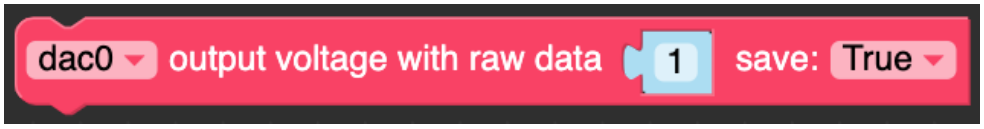
Get Raw data.
Returns the three raw values for Red, Green and blue from the sensor.

Get Red
Returns the red value from the sensor.
Get Green

Returns therein value from the sensor.

Get Blue
Returns the blue value from the sensor.
DAC,
dac0 Output Voltage

Gets a voltage reading from 0 to 3.3 volts If save it set to true then the data will be written to the M5Sticks E.E.P.R.O.M.



dac0 Output Voltage with Raw Data.
Gets a raw data reading of 0 to 4096. If save it set to true then the data will be written to the M5Sticks E.E.P.R.O.M.

IR,
Get IR State,

Returns the state of the IR Unit.

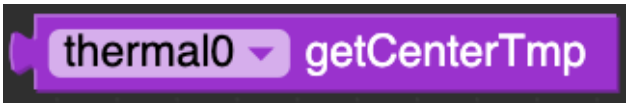
Set IR On,
Turns the IR unit on.
Set IR Off,

Turns the IR unit off.
NCIR,
NCIR Read,

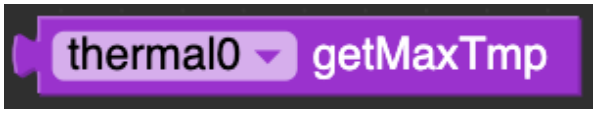
Returns values from the NCIR Unit.

Thermal,

Get Temp X, Y,
Returns the temperature detected at the specified coordinates.

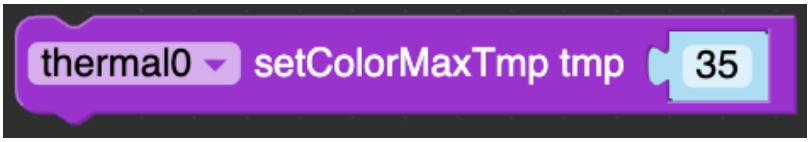


Get Centre Temp
Returns the temperature detected in the middle of the sensor.



Get Max Temp,
Returns the maximum temperature detected by the unit.

Get Min Temp,
Returns the minimum temperature detected by the unit.



Set Colour Max Temp,
Sets the upper level that the sensor will use.

Set Colour Min Temp,
Sets the lower level that the sensor will use.
Update X, Y, show centre,

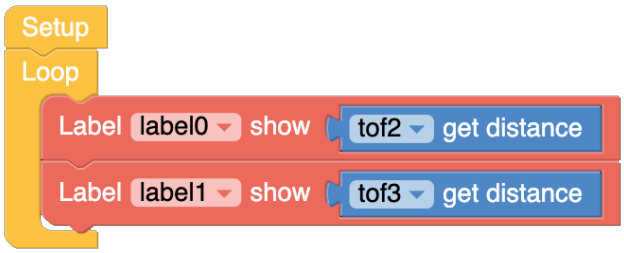
Changes the X, Y temperature read position while displaying the centre cross hair.

TOF,
Get Distance,

Returns the distance value from the T.O.F Unit.

Example

In the following example I am using two labels to show readings from two separate but identical T.O.F units.

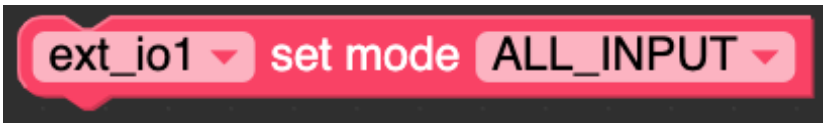



```
from m5stack import *
```

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

setScreenColor(0x222222)
pahub0 = unit.get(unit.PAHUB,
unit.PORTA)
tof2 = unit.get(unit.TOF, unit.PAHUB0)
tof3 = unit.get(unit.TOF, unit.PAHUB1)

label0 = M5TextBox(29, 90, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)
label1 = M5TextBox(177, 99, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(tof2.distance))
  label1.setText(str(tof3.distance))
  wait_ms(2)
```

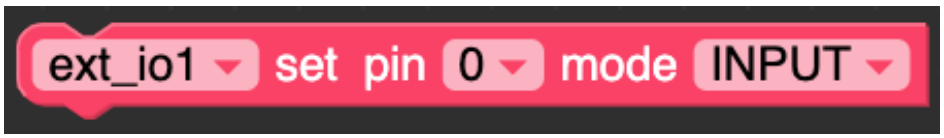
EXT I/O,

Set I/O Port Mode,



Set the mode of all the pins to Input or output.

Set I/O Pin Mode,



Sets the individual pins to input or output.

Digital WritePort,

Writes a hex value to the I/O unit.

Digital Write Pin,

Set each individual pin as high or low.

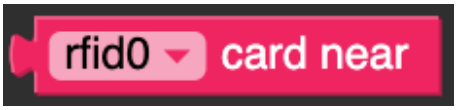
Digital Read Port,

Reads the digital state of the I/O Unit.

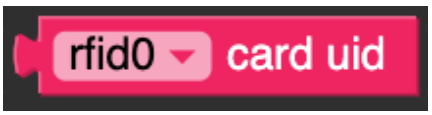
RFID,

Read String,
Reads a string of information stored at the specified address on the RFID card or tag.
Write String,

Writes a string of date to the specified address.
Card Near,



Returns true of false if an RFID tag or card is in proximity of the unit.
Card UID,



Returns the UID stored on the RFID tag or card.

PAHub

While the PAHub has some blocks to use, the hub is designed to allow the connection of multiple identical units that have the same I2C address.
The blocks provided are as follows.

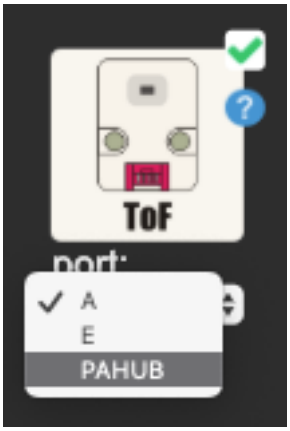
Set position state

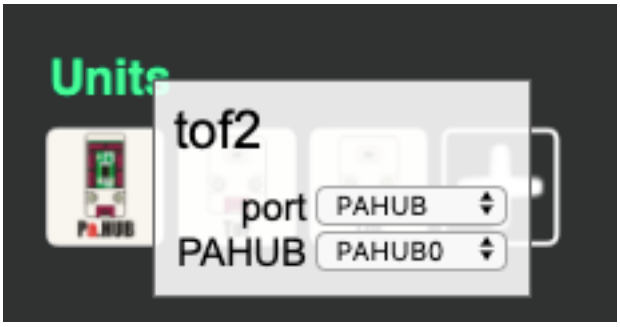
Set position

Set port value

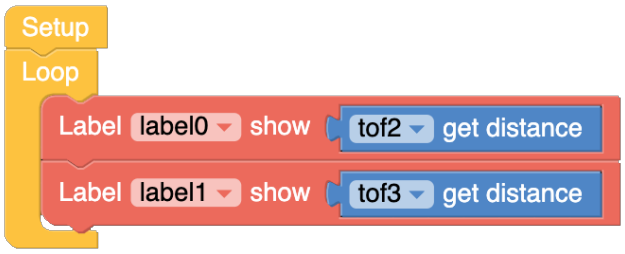
Example

In the following example I am using two labels to show readings from two separate but identical T.O.F units.
Two use the Pahub unit we first need to add it by clicking on the "+" sign and then we need to add the T.O.F units. Click on the "+" to add the T.O.F but then click on the "Port" drop down box and select PAHUB and the OK to add it. Repeat the previous steps to add a second T.O.F.





Now click on the T.O.F under the virtual M5Stack and another window will appear.
This is how we configure the PaHub's ports The Pahub has six ports labeled from 0 to 5 with 0 in the bottom right and 5 on the bottom left. set the shown window to the locations on the physical pahub where you plugged in the two T.O.F sensors.




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

setScreenColor(0x222222)
pahub0 = unit.get(unit.PAHUB,
unit.PORTA)
tof2 = unit.get(unit.TOF, unit.PAHUB0)
tof3 = unit.get(unit.TOF, unit.PAHUB1)

label0 = M5TextBox(29, 90, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)
label1 = M5TextBox(177, 99, "Text",
lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  label0.setText(str(tof2.distance))
  label1.setText(str(tof3.distance))
  wait_ms(2)
```

To view an introduction video on the PAHub, you can check out Lukes video on the M5Stack youtube channel https://www.youtube.com/watch?v=nMsCwqCE5c8

Modules

M5Stack cores have additional expansion units that share the sam 50mm X 50mm footprint. These modules can be stacked between the M5Stack cores and M5Stack bases.

Stepper Motor,

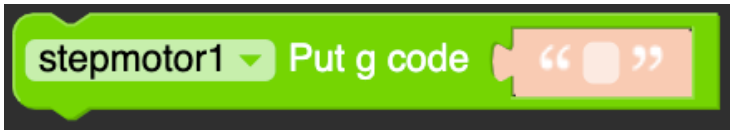
Stepper Motor Module Address,

Sets the Hex address to use for the stepper motor driver.

Stepper Motor Position,

Sets the position and speed that the stepper motor is to move to.

Run "G" Code,



For running G-Code in the modules g-code interpreter.

Stepper Motor Movement Mode,

Sets how the stepper motor is to move. Distance is used to make it move a defined distance, Absolute is use to make the motor more to a set of coordinates.

Lock Stepper Motor,

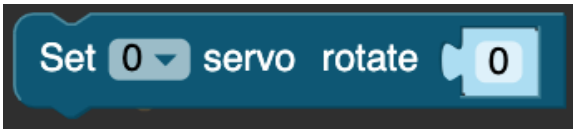
Locks the stepper motor by powering the cores preventing the motor from being mechanically turned.

Unlock Stepper Motor,

Unlocks the stepper motor by removing the power to the coils allowing them to be mechanically turned.

Servo,

Set Servo rotate,



Tells the servo on the defined channel to rotate to a specified position.

Set Servo Speed,

Sets the move speed of the servo on the defined channel.

Bala,

Bala Move distance,
Moves the Bala motors forward or backward by a user defined value.

Turn Wheel
Turns the left or the right wheel wheel by a user defined value.
Rotate

Tells the bala to rotate.

Get Angle.
Returns the angle that the Bala unit has turned.

See Angle Offset.
Used to set an angle offset to handle inaccuracies between the motors.

Balance Loop

Tells the M5Stack to update its internal position in memory.

Bala Motor,

Set Motor direction and speed.
Sets the motors rotation direction and speed.
Run forward distance and speed.

Commands the bala to move in a direction a user defined distance at a user defined speed.
Go to Position

Commands the motor to move to a user defined position at a user defined speed.

Stop Motor
Commands the motor on the selected channel to stop.

Read encoder
Returns the values from the Bala motors built in encoder.

Lego Motor,

Set Lego Motor Rotate PWM,
Sets the Lego motor on channel 1 to rotate clockwise with a value.

Stop Lego Motor,
Stops the motor o channel 1

Clear encoder,
Clears decoder reading from memory.
Read encoder,

Returns the values from the lego motors built in encoder.

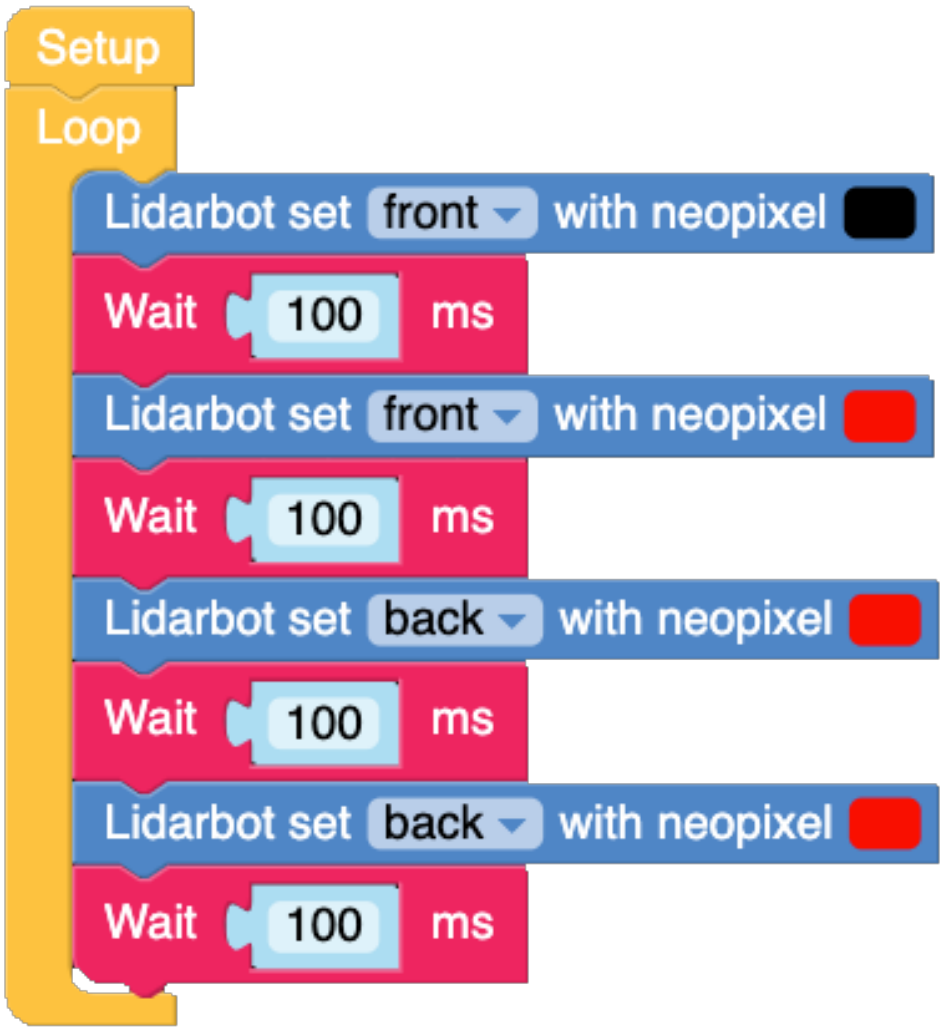
Lidarbot

Lidarbot set front with neopixel colour,

Sets the colour of all the RGB LED's using the colour picker. You can use this block to set the front bar, the rear bar or, both of the RGB LED bars.
The Micropython code for this is.

```
lidarbot0.setRgb(0xff0000,'front')
```

In the following example I turn all the RGB LED's on on the from bar then the rear bar and then both before turning them all off.



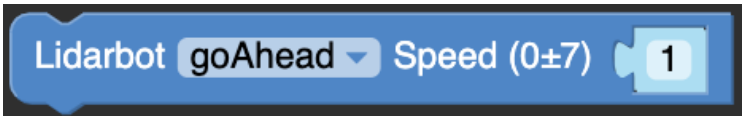
Lidar bot Set Individual LED Colour,

Sets the colour of the individual RGB LED's in the front and rear light bars using the colour picker.

Lidarbot Move,

Moves the Lidar bot in the direction at a user defined speed.

Lidarbot Set Speed,



Sets the speed of each of the individual four wheels.

Lidarbot Set Servo,

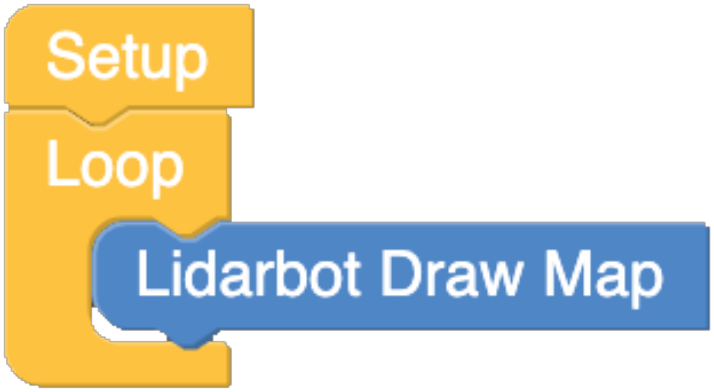
Set the angle of a servo connected to the Lidarbot.
Lidarbot Axis Speed,
Set the X and Y axis speed.

Lidarbot Draw Map,

Fetches information from the LIDAR scanner on the top of the robot and displays it on the screen as a map.
The Micropython code for this block is

```
lidarbot0.lidar.draw_map()
```

Example.

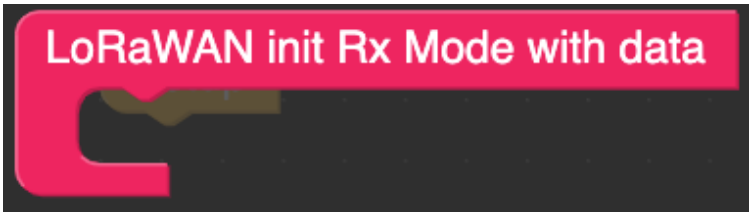


This small example is all that is needed to get the LIDAR scanner to show a map of the robots surroundings on the screen. Please note that the loop is required in order for the screen to constantly update or a complete map does not get drawn.
The Micropython code for this example is as follows.

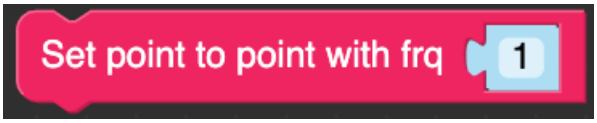
```
from m5stack import *
from m5ui import *
from uiflow import *
import module

setScreenColor(0x222222)

lidarbot0 = module.get(module.LIDARBOT)

while True:
  lidarbot0.lidar.draw_map()
  wait_ms(2)
```

Lidarbot Get Distance,

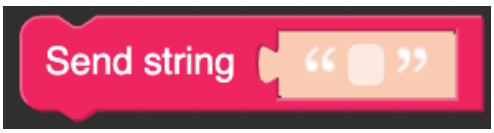
LoRaWan,
Init LoRaWan Rx Mode



Set Point to Point Frequency.



Send String.



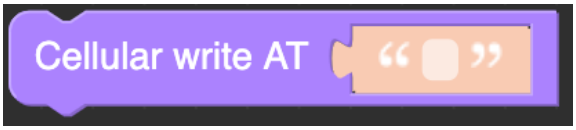
Get Data,

PM2.5

Get PM2.5 Value,
Returns particle count of selected size particles in SPM or APM value.
Get Particle Count,

Returns the amount of particles detected above the selected size.

Cellular,



Cellular Write AT,
Used to send AT commands to the Cellular module.
Cellular Wait,

Sets the time that the cellular module waits before responding to AT commands.

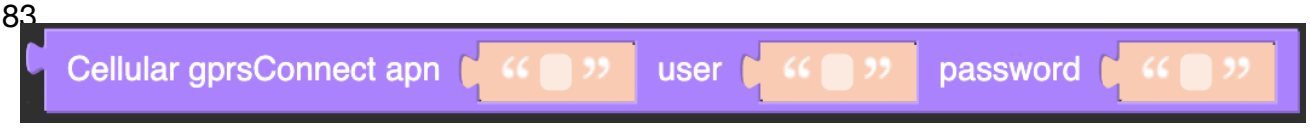
Cellular Get Response,
Immediately returns a response from the cellular module after an AT command is sent.

Cellular Test AT

Cellular Restart
Forces the cellular module to restart independent of the M5Stack.
Cellular Get SimState

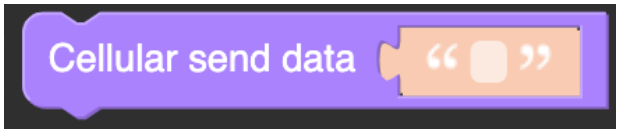
Returns the status of a Sim card plugged into the module.
Cellular Init,

Initialises the Cellular module.
Cellular GPRS Connect,



Returns the module status after ordering the module to connect to a GPRS network with the user defined credentials.

Cellular Connect host,
Returns the status of the module after trying to connect to a host with the following user defined credentials.



Cellular Send Data,
Sends data over the connection to the host device.
Cellular Send Line Data,

Cellular Get Available

Reports if Cellular networks are available.

Cellular Read

Returns available Cellular networks.
Cellular Connected

Returns the status if the cellular module has connected to a network or not.
Cellular Get Local IP

Get a local IP from a network that the module has connected to.

Cellular GPRS Disconnect,

Returns the module status after being commanded to disconnect from a network.
Cellular Get Network State,

Reports the status of the Cellular network.

Faces Modules

Gameboy Face,

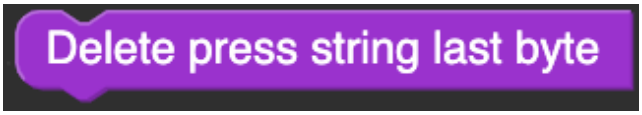
Get Direction Is Pressed,
Get Direction Was Pressed/Released,

Calculator Face,

Get Press String,

Get Press Button Int Value,
Clear Press String,

Delete Press String,



Is New Button Pressed,

Keyboard Face,

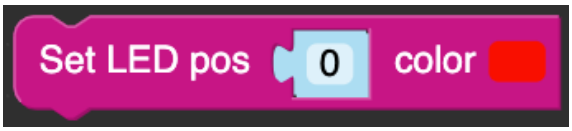
Get Press String,
Clear Press String,

Delete Press String,

Get Press Button Int Value,

Is New Button Pressed,

Encoder Face,
Set LED Pos Colour,



Clear Encode Value to Zero,
Get Encode Value,

Get Encode Direction,

Is Encode Pressed,

Joystick Face,
Get X Value,

Get Y Value,

Is Pressed,

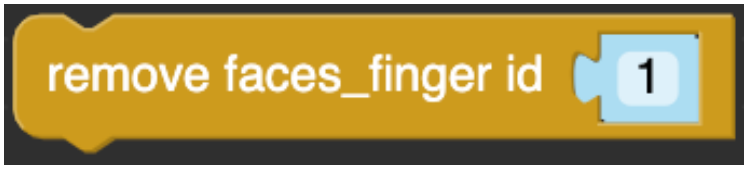
Get Reverse X Value,
Get Reverse Y Value,

Set LED Pos Colour,

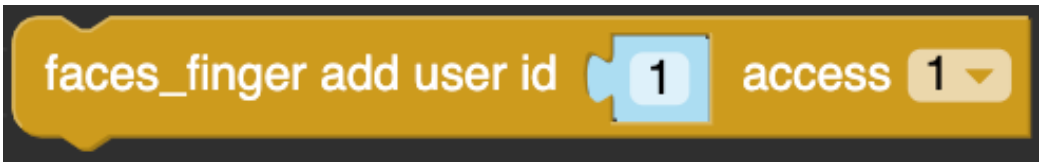
Finger Face,
Get State,

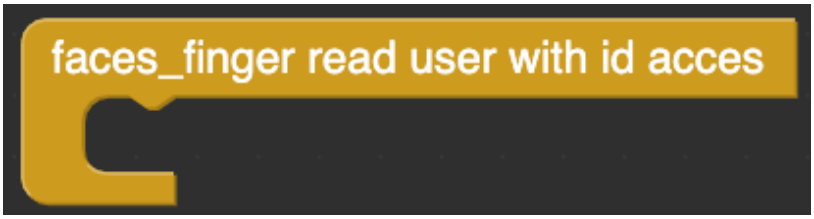
Get access,
Get ID,

Remove all,
Remover Faces Finger ID ( ),

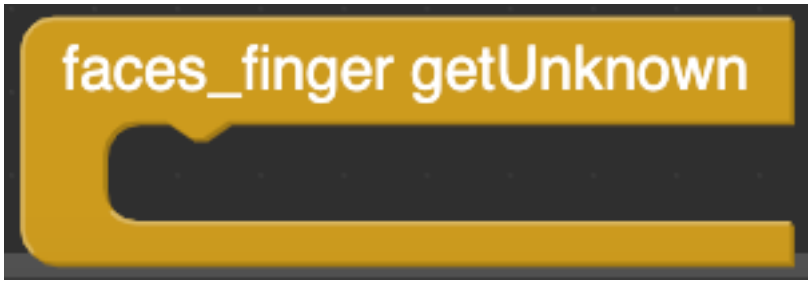


Add User Faces Finger ID ( ),





Faces Finger read User,
Faces Finger Get Unknown.



RFID Face,

Read String,
Reads a string from an RFID card or Tag.

Write String,
Writes a string to the RFID card or tags storage space.
Card Near,

Returns true if an RFID card or tag is detected.
Card UID,

Gets the Card or Tag UID.

Logic Blocks

This section contains the various blocks that allow us to preform maths and other tasks that ties the blocks in the earlier sections together.

Variables,

Variables are containers for values that can be changed by our code. We have already seen some variable in use as the hardware units and modules use variable to pass the values from their sensors to be used by our programs in order to set actions based on their values.

Create Variable,

Creates a Variable
By clicking this block to create a variable, we are presented with a message to give our variable a name. Once a name is set and OK is pressed, the following blocks will be created.

Set Variable,

Sets the initial value of the variable to the value of the following connected mathematical block or a sensor block..
The Micropython code for this block is
V1=None

Change Variable,

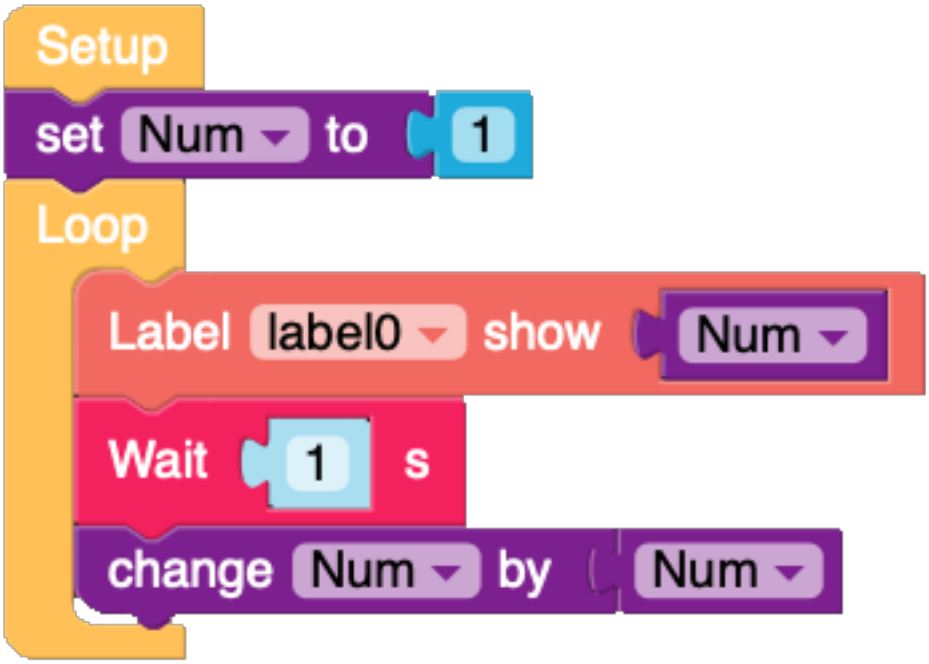
This function changes the value of the variable to that of a value from a mathematical block or a sensor block.
The Micropython code for this is
V1 = (V1 if isinstance(V1, int) else 0) + 1

Variable,

The variable block that other functions can query.
The Micropython code for this block is
(V1)

Example

In this example I have used all three of the variable blocks. The program calculates a value and then multiples that value buy the calculated value.



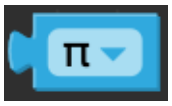
And the Micropython code for the example.

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

setScreenColor(0x222222)

label0 = M5TextBox(77, 69, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

num = None

num = 1
while True:
  label0.setText(str(num))
  wait(1)
  num = (num if isinstance(num, int) else 0) + num
  wait_ms(2)
```

Maths,

Value,
Used to hold a set value

Calculation,
Returns the sum of the equation. Options are Plus, Minus, Multiply, Divide, To the power of.
Pi,



Allows calculations to use Pi (3.14), Eulars Number (2.718), Golden Ration (1.618), Square root 2 (1.414) Square Root 1/2 (0.707), and infinity.

Remainder of,
Returns the remainder of one value divided by another value.

Condition is even,
Returns true of false if value is equal to even, odd, prime, whole, positive, negative or divisible buy.

Sum of list,
Returns the sum of a list of values.
Random Fraction,

Returns a random fraction.

Random Integer,
Returns a random integer from a range of user defined numbers.

Round,
Rounds the following value up or down.
Square Root,

Returns the square root value, absolute value, the negation of a value, the natural logarithm of a number, a base10 logarithm of a number, e to the power of a number, or 10 to the power of a number or value connected to it.
Sin,

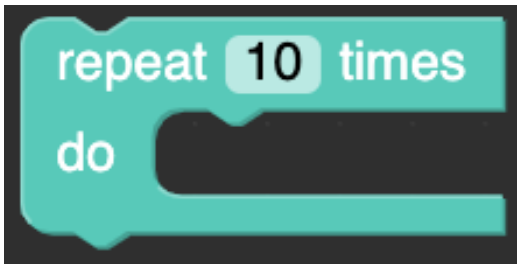
Returns the Sine, Cosine, Tangent, Arcsine, Arccosine, or Arctangent of a number.

Convert to int,
Converts a value to an integer.
Convert to Float,
Loops,

Converts a value to a footing point number.
Reserve Decimal Fraction.

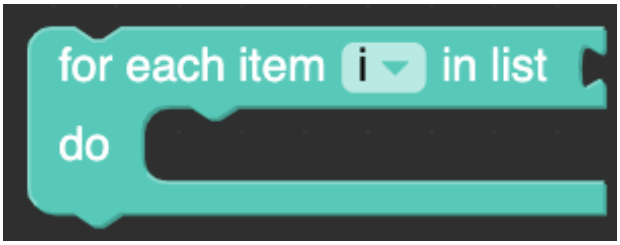
Reserves the user defined number as a decimal fraction.

Loops

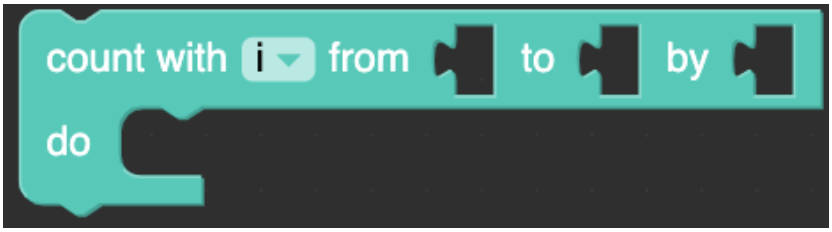


Repeat,
Repeats the code in side it ten times.

Repeat while Condition,
Repeats the code inside while the user defined condition is set.
For Each,



Repeats the code inside for each item in a list.



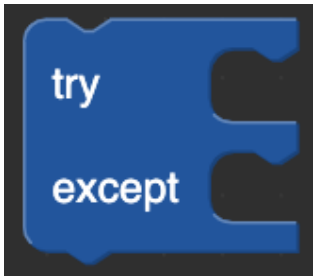
Count with,
Runs the code inside on selected items in a range using a user defined variable.

Break out,
Used to exit from a loop of code.

Logic,
If - Do,

If a defined condition is met the code inside is run.
If - Else - Do,

If a defined condition is met the code inside is run otherwise another set of code is run..



Condition True,

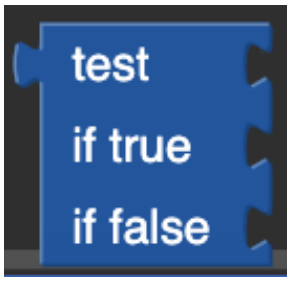
Returns true or false.

Condition Equals Condition,
Returns true or false if a sum two conditions are met.

Not,
Inverts or reverses the command.
Null,

Condition and Condition,

Used to define if two conditions are required.



Test - True/False

Graphic,

Lcd.clear

Lcd.clear turns all the pixels on the screen to black.

Lcd.fill

Lcd.fill will change all of the pixels on the screen to the same colour. The colour is set in this block by using the colour picker.

lcd.print

Lcd.print will place text on the screen at the specified coordinates and in the specified colour using the colour picker.

Font

Font specifies a font to be used for the text that will be shown on screen.

Lcd.pixel

Lcd.pixel will colour specific pixel at the specified coordinate and in the specified colour using the colour picker.

Lcd.line,

Lcd.line draws a line starting at the specified coordinates and ending in X1 and Y1 in the specified colour using the colour picker. By specifying X1 and Y1 we can have angled lines.

Lcd.rectangle,

Lcd.rectangle draws a rectangle starting at the specified coordinate with a user defined height and width in the specified colour using the colour picker.

Lcd.triangle,

Lcd.triangle draws a triangle by specifying the coordinates of the three individual points of a triangle in the specified colour using the colour picker.

Lcd.circle,

Lcd.circle draws a circle with the centre point the specified coordinate with a radius defined by the user and in the specified colour using the colour picker.

Lcd.ellipse,

Lcd.ellipse draws a circle with the centre point the specified coordinate with an X radius and separate Y radius defined by the user and in the specified colour using the colour picker.

Lcd.arc,

Lcd.arc draws an arc with the centre at the specified coordinates with a user defined radius, thickness, start point and end point and in the specified colour using the colour picker.
Please Note that there is a bug in the code that keeps deleting the Radius value.

Lcd.polygon,

Lcd.polygon draws a polygon with the centre at the specified coordinates with a user definable radius, number of sides, thickness, rotation and in the specified colour using the colour picker. Please not that when Radius and thickness are the same, we can get a snowflake like appearance.

Emoji,
The Emoji menu contains blocks that allow us to draw Emojis and control the background shown behind them.
There are three blocks available here with the biggest being the emoji map

Emoji Map

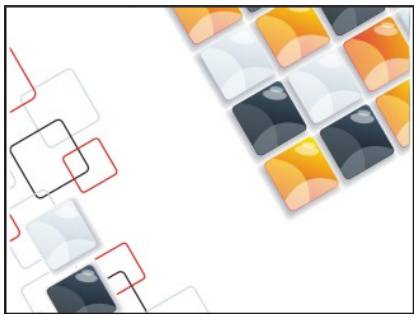
The emoji map has a 7 X 7 grid which you click on each cell to make it light up in the specified colour using the colour picker on the right of the text. To make a cell active, all you have to do is to click on each of the lighter blocks and a tick will appear to show that you have made that cell active.

Set line 1 row 1 in colour.
This block allows you to set the colour of each individual cell.

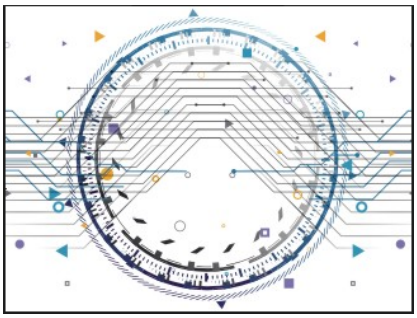
Change Background Image.

Allows you to chose one of the six built in backgrounds that show behind the emoji. The six background images are shown below.

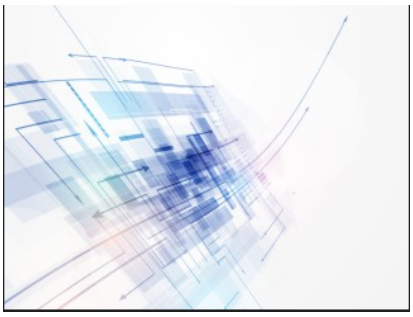
Background 0



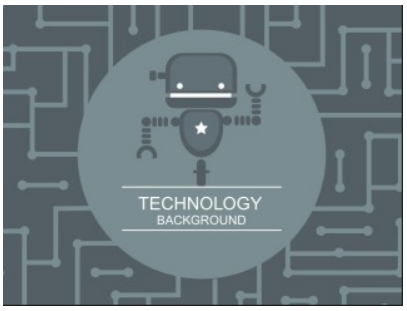
Background 1



Background 2



Background 3



Background 4

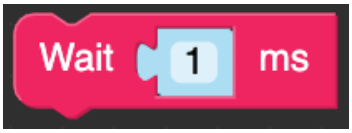
Background 5

Timers (Part 2),

Wait Seconds,

Delays a program from moving on to the next block by a user defined pause in seconds.

Wait Milliseconds,

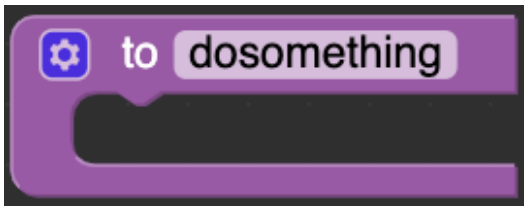


Delays a program from moving on to the next block by a user defined pause in seconds.

Get Ticks ms

Returns current runtime of esp32..

Function,
Do Something,



Creates a function with no output.
Do Something and Return Condition,

Creates a function with an output.

If Condition Return Condition,
If true, returns the condition defined in the space.

Text,

Text block
Used to display a letter, word or sting of text on the screen.

To UPPER Case
Converts the following sting to upper case.
In Text Get Letter

Returns one letter from the string.
Count In.

Returns how many time a string appears in another string.

Is Empty.
Returns true if string is empty.
Length Of,

Returns the length (including spaces) of the string that is connected.
Print

Prints the string or number connected to it on screen.

Replace With In
Finds and replaces all occurrences of a string in a string with another string.
Trim Spaces,

Returns the connected sting of txt with some or all of the spaces removed.
Prompt For Text With Message

Prints a message on screen prompting a user to insert some text.

Prompt For Text With Message ()
Prints a message on screen prompting a user to insert some text or value connected to it.

Convert To String,
Converts the values in the following blocks in to a string.
Message Plus (Concat String.)

Used to add a value to the text for example batch file naming (ABCD + 001/002/003)
Decode

Used to decode a sting of text.

Encode,

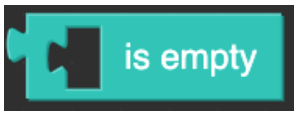
Used to encode a string of text.

Examples

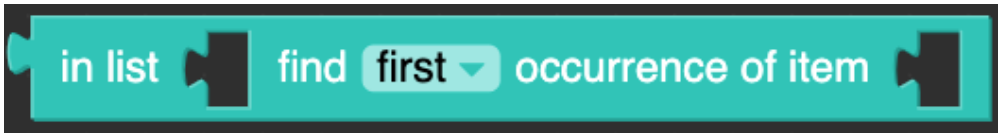
The following examples are used to edit the text in labels or to log data.

Lists,

Length Of,
Returns the length of a list.
Is Empty,

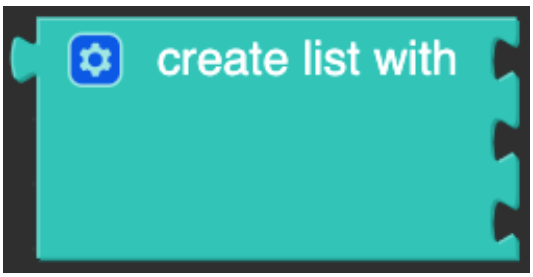


Returns true if list is empty.



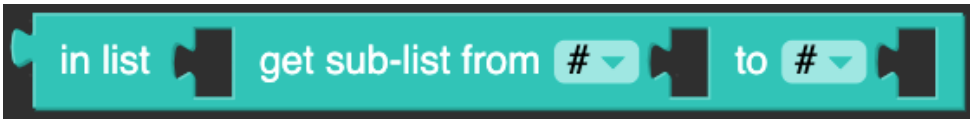
In List Find Occurrence
Returns index of first or last occurrence of a specified item or returns zero if an item is not found.

Create Empty List,
Creates an empty list.



Create List With,
Creates a list with specified items attached to it. Clicking the gear in the top left corner allows users to add additional items.

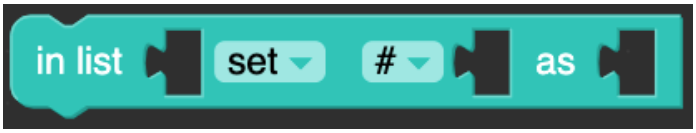
In List Get
Returns the item in the specified position of a list.
In List Get Sub List,



Creates a list from a specified section of another list.

Create List With Item Repeated.
Creates a list with a used defined item repeated a user defined amount of times.

Reverse,
Reverses a copy of a list.
In List Set,



Allows users to define an item at a specific position in a list.

Make List From List With Delimiter.

Allows users to split a list into smaller list using delimiters.

Map
Create Map,

Map Clear,
Map Contain Key,

Get Key In Map,

In Map Add Key
In Map Set Key,

In Map Delete Key,

JSON,

Dump to JSON
Dumps a sting or value to json.

Load JSON
Used to load a JSON value or string.

Advanced Blocks.

The Advanced blocks available in UIFlow provide lower level access to hardware as well as access to hardware that oes now have dedicated blocks provided. This section also contains blocks required for accessing the SDCard and files stored on it.

SDCard

Open SDCard File

Open SDCard File block is a loop that opens a file stored on the cards, completes the actions placed inside and then closes the file. By opening and closing the file after each pass, the chance of file corruption is reduced.

The argument mode points to a string beginning with one of the following sequences (Additional characters may follow these sequences.):

``r'' Open text file for reading. The stream is positioned at the beginning of the file.

``r+'' Open for reading and writing. The stream is positioned at the beginning of the file.

``w'' Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

``w+'' Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

``a'' Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file. Subsequent writes to the file will always end up at the then current end of file, irrespective of any intervening fseek(3) or similar.

The Micropython code for the loop is as follows:

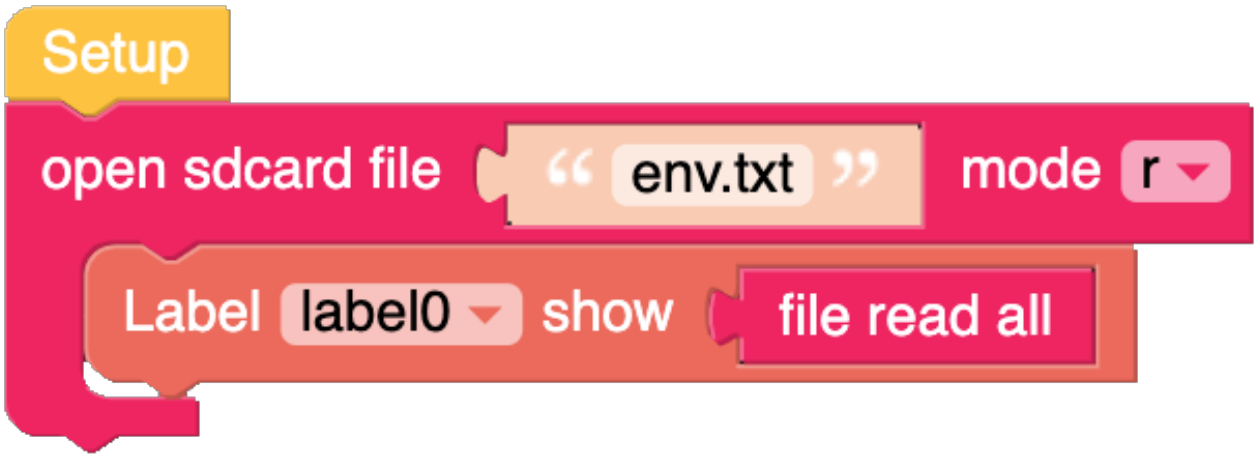
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

label0 = M5TextBox(21, 5, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

with open('/sd/', 'r')
  pass
```

File Read all

Returns the contents of the file opened in the SD File open block.

The Micropython code for this is
(fs.read())

In the following example the file env.txt (created for my data logging demo), is opened and by using a Label Show block we ca display the contents on the M5Stacks screen. Please note that for this to work the mode has to be set to "r" or "r+"



And the Micropython code for this is:

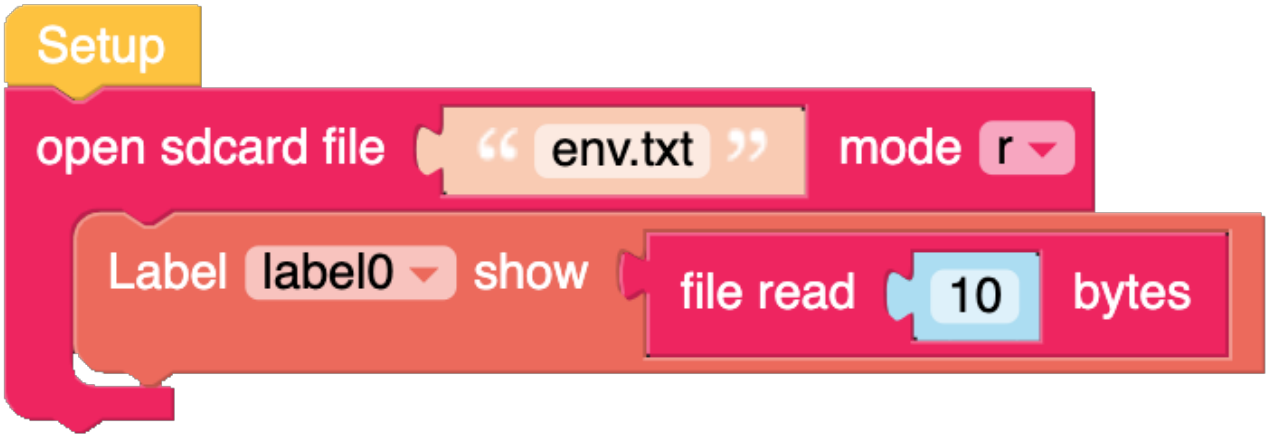
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

label0 = M5TextBox(21, 5, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

with open('/sd/env.txt', 'a') as fs:
  label0.setText(str(fs.read()))
```

File Read Bytes

Returns the first number of bytes of a file declared in a maths number block.
The Micropython code for this is

(fs.read(0))

Example
In the following example the first ten bytes of the file env.txt are display on the M5Stacks screen.



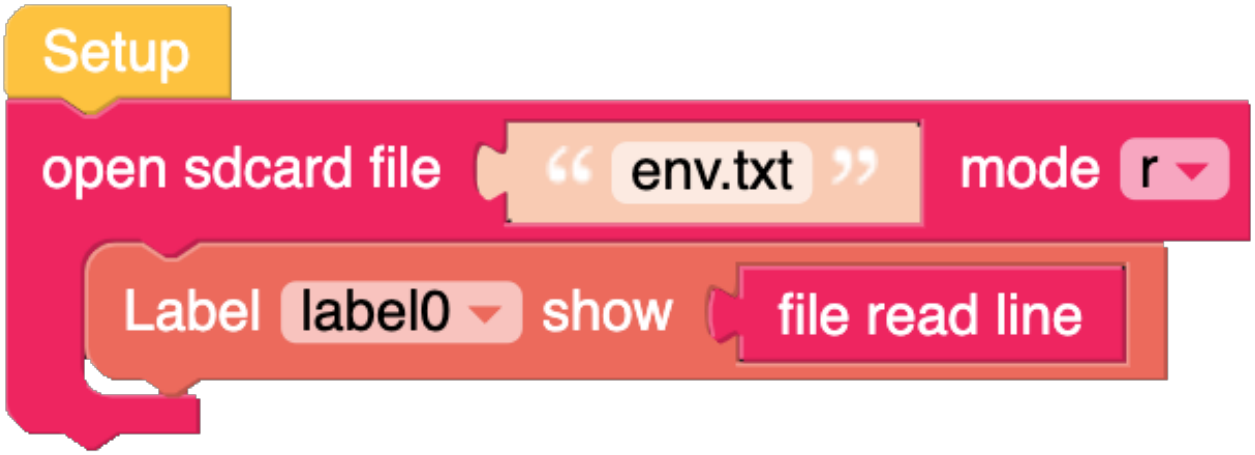
The Micropython code for this is

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

setScreenColor(0x222222)

label0 = M5TextBox(21, 5, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

with open('/sd/env.txt', 'a') as fs:
  label0.setText(str(fs.read(0)))
```

File Read Line

File Read Line is a value block that returns a line of text from a file.
The Micropython code for this is
fs.readline()

Example



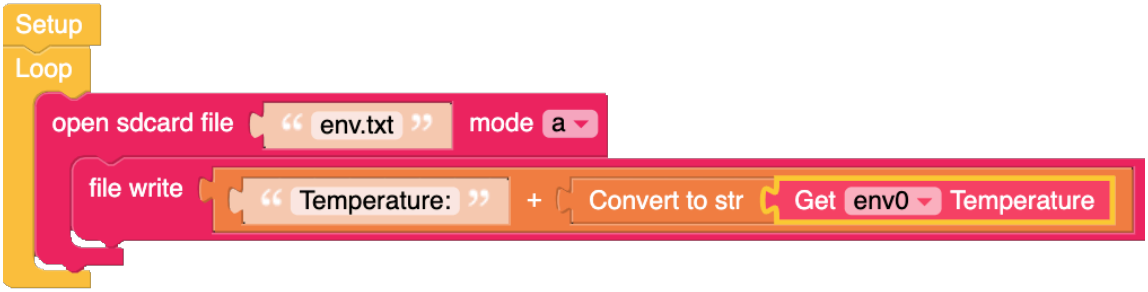
In this example the first line of the text file env.txt is displayed on the M5Stacks screen.
The Micropython code for this is

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

setScreenColor(0x222222)

label0 = M5TextBox(131, 47, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

with open('/sd/env.txt', 'r') as fs:
  label0.setText(str(fs.readline()))
```

File Write

The File Write block is used to write data into a file.
The Micropython code for this is
fs.write('')

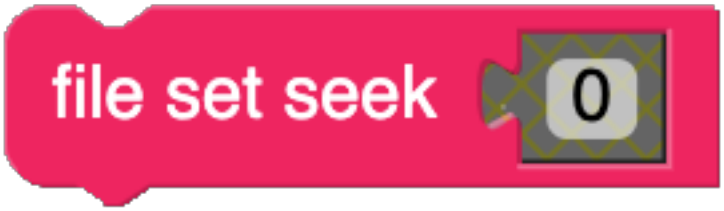
Example



In this example I have created a simple data logging device that records the temperature recorded to the text file env.txt.
The Micropython code for this is as follows.

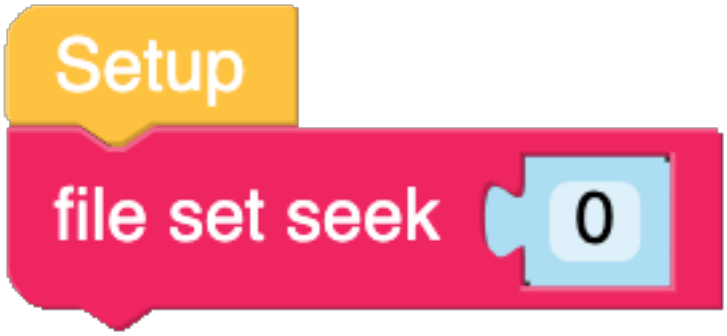
```
from m5stack import *
from m5ui import *
from uiflow import *
import unit

setScreenColor(0x222222)
env0 = unit.get(unit.ENV, unit.PORTA)

label0 = M5TextBox(21, 5, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

while True:
  with open('/sd/env.txt', 'a') as fs:
    fs.write(str(('Temperature:' + str((env0.temperature)))))
  wait_ms(2)
```

File Seek



File Seek looks for the file in the position set with a mathematical value block.
The Micropython code for this is
fs.seek(0)

Example



In this demo the file is position 0 is read from the SD card.
The Micropython code for this is

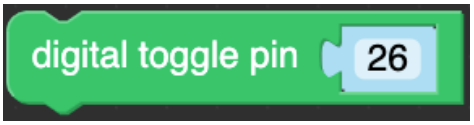
```
from m5stack import *
from m5ui import *
from uiflow import *

setScreenColor(0x222222)

label0 = M5TextBox(131, 47, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

fs.seek(0)
```

Easy I/O,
Analog Read Pin,

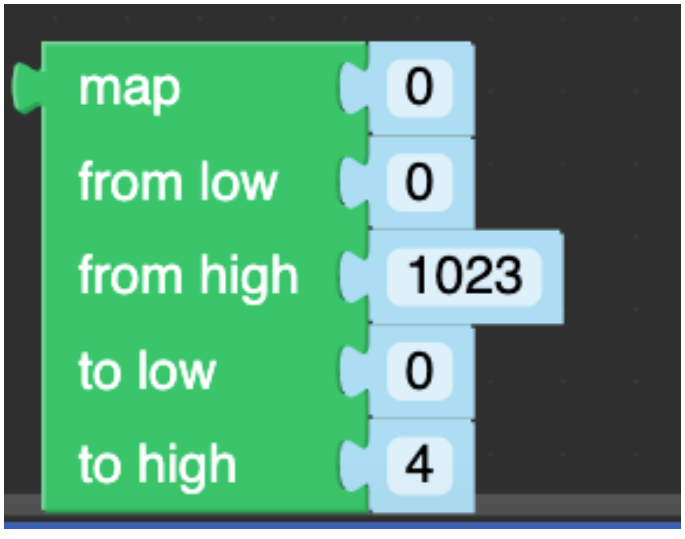
Analogue Read Pin,

Digital Read,

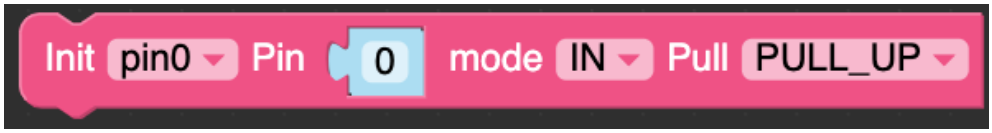
Digital Write Pin,



Digital Toggle Pin,



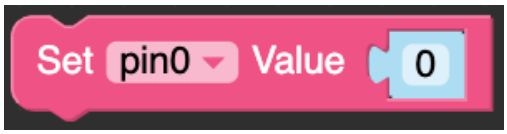
Map,
GPIO,



Init Pin Mode,
Set Pin High,

Set Pin Low,
Get Pin Value,

Set Pin Value,

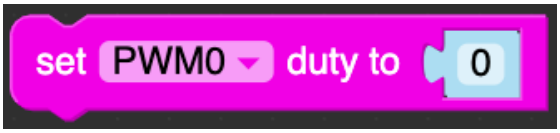


PWM,

Init PWM of Pin,

Set PWM Frequency,

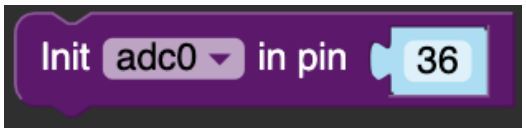
Set PWM Duty Cycle,



Pause PWM

Resume PWM

ADC,



Init ADC Pin,

Set ADC Pin Bit Width,
Set ADC Atten,

Read ADC,

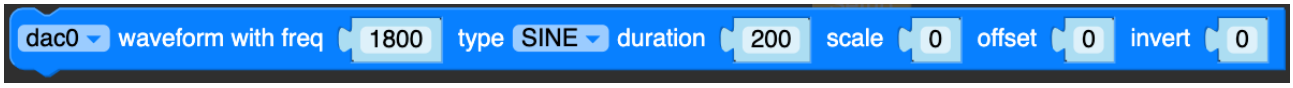
DAC,

Init DAC

Set DAC Frequency,

Write Value to DAC,

DAC Beep With Frequency,
DAC Waveform,



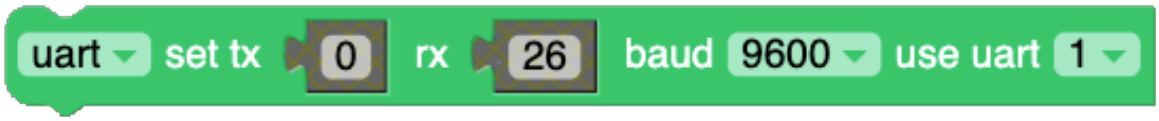
DAC Stop Wave

UART,

UART stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a micro controller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data. The UART allows us to communicate with devices over RS232 or RS485.

Set Uart,

When using the RS485 Unit the above settings are used however, when using the RS485 Hat, the following setting need to be used.
The Micropython code for this block is




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

setScreenColor(0x222222)

label0 = M5TextBox(21, 5, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

uart = None

uart = machine.UART(1, tx=0, rx=26)
uart.init(9600, bits=8, parity=None, stop=1)
```

Read Uart,

Returns data that is sent from the slave to the master unit.

Read Uart (0) Characters,
Read a Line of Uart,

Get Remain Cache,

Write Value to UART

Write a Line to UART
Write String to UART,

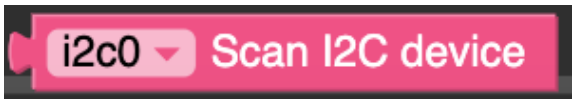
I2C,
Set I2C Port,

Starts communication with an I2C device with the address.

Set I2c SDA, SCL,

Sets the pins that the I2C port will use for SDA and SCL.

I2C Scan,



Returns a list of I2C addresses detected on the I2C bus. This block returns the addresses as binary instead of hexadecimal.

I2C Available Address in List,

I2C Read Reg One Byte,

I2C Read Reg One Short,

I2C Read Reg One Byte

I2C Write Reg One Byte,

I2C Read Byte,

I2C Write Reg One Short,

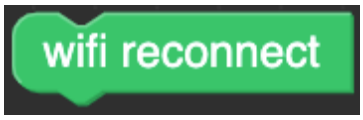
EXECUTE

Execute,
Execute Code,

Execute code is a block that is provided for the programmer to add a line of code that is not already in a library or block.
There is no dedicated Micropython code because this block is a placeholder for adding your own code.

Network,

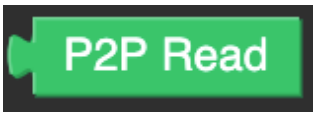
WIFI Connect,
Wifi Reconnect,



Wifi is Connected,

Connect to Wifi SSID, Password,
P2P Send API Key,

P2P Read,



ESP NOW

ESP Now is a short-range, low-power communication protocol that enables multiple devices to communicate without or without Wi-Fi. This protocol is similar to the low-power 2.4GHz wireless connection found in wireless mice and keyboards. Before communications can be initiated between devices, they need to be connected in a peer to peer, master to slave configuration using the devices M.A.C addresses. Once the connection is established, theses paired devices remain connected together without the use of a handshake protocol.

Get mac Address

To get the mac addresses of M5Stack and M5Stick devices, we need to use the **Get mac addr** block. This block returns the M5Stack or M5Stick M.A.C address that is stored in the internal memory of the device as a value to be used by function blocks.
The Micropython code for this block is :

```
espnow.get_mac_addr()
```

Example
As a variable block the **Get mac addr** requires additional blocks to be used. In the following example I have used the **Label** block to get the mac address and display it on the M5Stack or M5Sticks screen.

The Micropython code for this is:

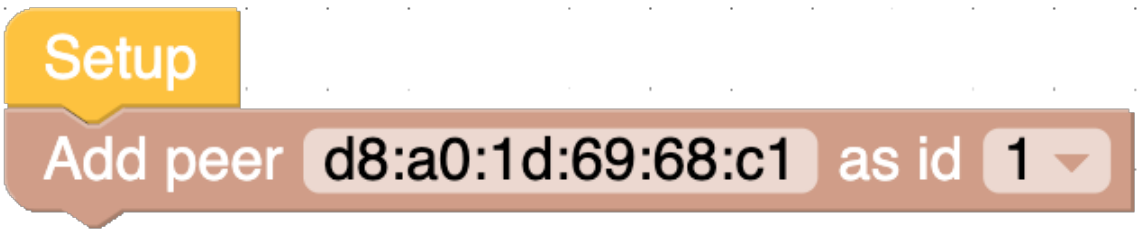
```
from m5stack import *
from m5ui import *
from uiflow import *
import espnow
import wifiCfg

setScreenColor(0x111111)

wifiCfg.wlan_ap.active(True)
espnow.init()
label0 = M5TextBox(21, 80, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

label0.setText(str(espnow.get_mac_addr()))
```

Add Peer

Now that we can see the individual device MAC address' we need to connect them together. To do this we next use the Add Peer block. To set the slave device that a primary device we will control, we replace **ff:ff:ff:ff:ff:ff** with the slave devices MAC address and then add this to master devices code setup. We can add multiple slaves to our network, but for each we need to set a different ID number. The Micropython code for this block is as follows.

```
espnow.add_peer('ff:ff:ff:ff:ff:ff', id=1)
```

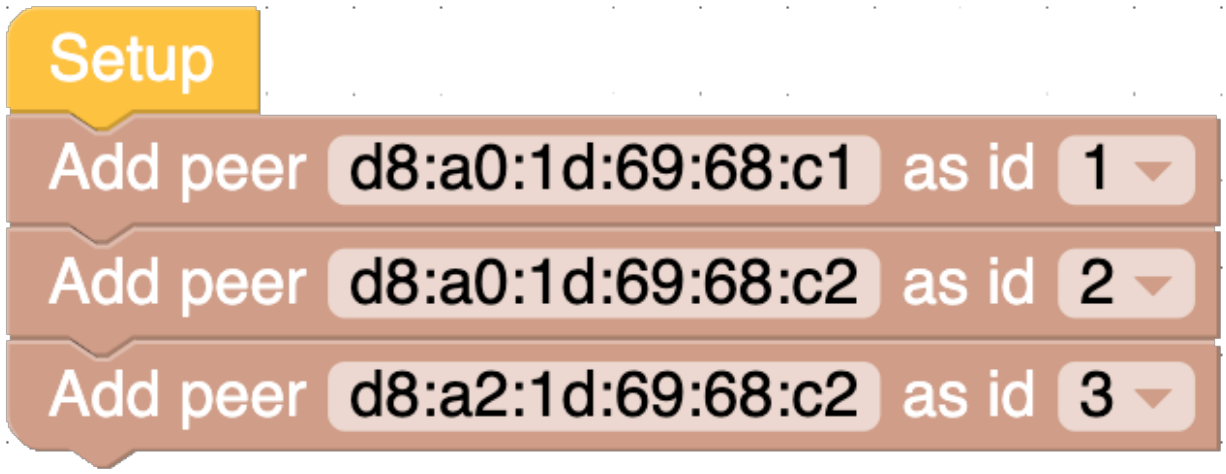
Example 1



In this example I have taken the six hexadecimal value M.A.C address from the slave device and added it to the setup of the master device and set its id value as one.
The Micropython code for this example is:
It iss worth noting that not only do these examples call the **espnow** library, they also call the **wifiCfg** library required for handling Wifi actions.

```
from m5stack import *
from m5ui import *
from uiflow import *
import espnow
import wifiCfg

setScreenColor(0x222222)

wifiCfg.wlan_ap.active(True)
espnow.init()

espnow.add_peer('ff:ff:ff:ff:ff:ff', id=1)
```

Example 2



If using more than one slave then the setup will look as follows.
When used this way we can control up to nine slave devices from one master device. The Micropython code shown on the following page show how the different MAC and ID values are changed.

```
from m5stack import *
from m5ui import *
from uiflow import *
import espnow
import wifiCfg

setScreenColor(0x222222)

wifiCfg.wlan_ap.active(True)
espnow.init()

espnow.add_peer('d8:a0:1d:69:68:c1', id=1)
espnow.add_peer('d8:a0:1d:69:68:c2', id=2)
espnow.add_peer('d8:a2:1d:69:68:c1', id=3)
```

Set PMK

The setup I have used for initiating a connection between devices so far is okay for none secure connections but, if we want to secure the connection between devices we need to use the **SET PMK** block.The **SET PMK** or **Set Primary Master Key** block that the M5Stack will use while communicating.
The Micropython code for this is

```
espnow.set_pmk('')
```

This block is used to define a custom, user defined sixteen byte value. If no custom value is defined then the ESP will use a default PMK value. The Set PMK block should be used before adding peers or slave units. For most programmers we wont need to use this block for basic work, but there will be times when this block will have its use.

Broadcast Data

Now that we have established a connection between M5Stack and M5Stick devices it time to send some information. The **Broadcast Data** block is used to send a string across the connection however, this block will send the data to all devices connected to the network.
The Micropython code for this block is:

```
espnow.broadcast(data=str(''))
```

Example
In the following code section, when uploaded to a master device will send the message "Hello M5Stack to any connected slave devices.

The Micropython code for this snippet is as follows.

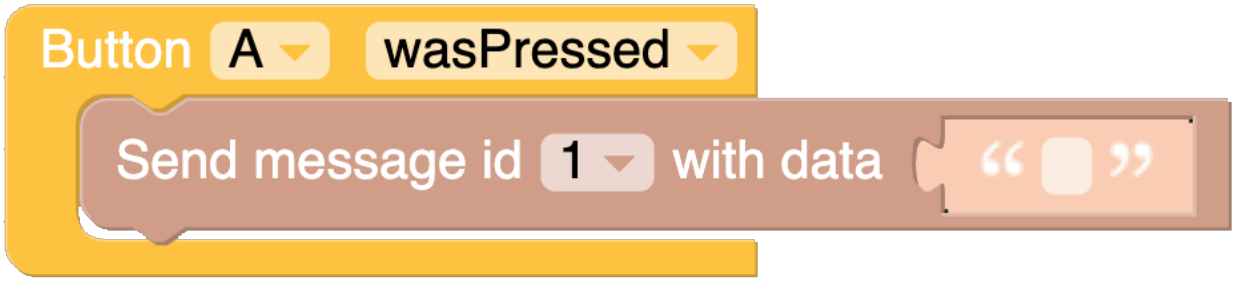
```
def buttonA_wasPressed():
  # global params
  espnow.broadcast(data=str('Hello M5Stack'))
  pass
btnA.wasPressed(buttonA_wasPressed)
```

Send Message ID with Data

To send data to a master device to a specific slave device in the network, we use the **Send Message ID with Data** block. The Micropython code for this block is

```
espnow.send(id=1, data=str(''))
```

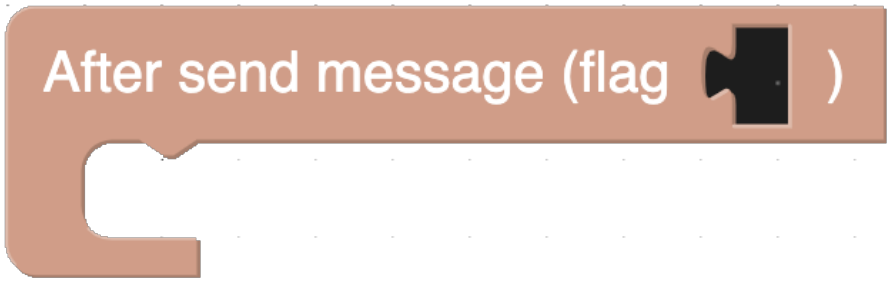
In the following example I have just replaced the **Broadcast Data block** with the **Send Message Id With Data** block in order to send messages to only one unit.



And the code sample for this block is:

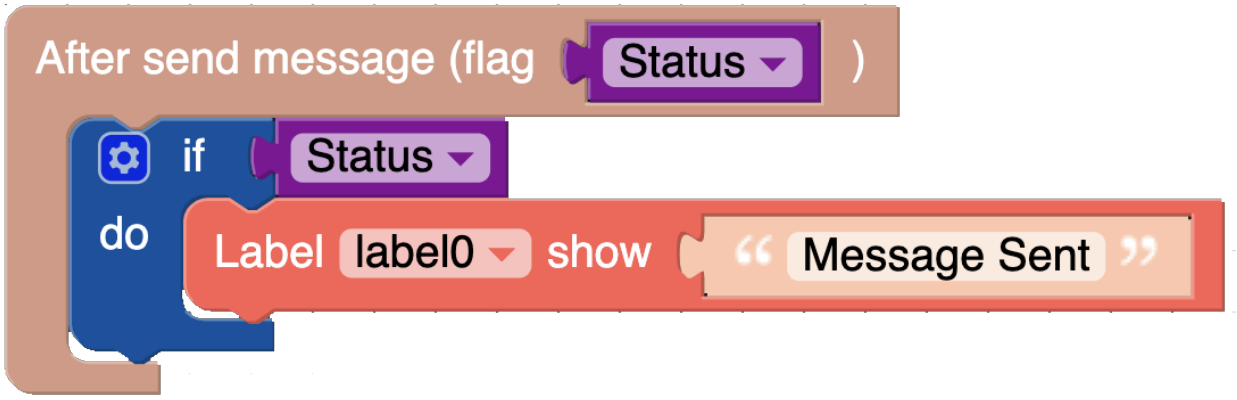
```
def buttonA_wasPressed():
  # global params
  espnow.send(id=1, data=str(''))
  pass
btnA.wasPressed(buttonA_wasPressed)
```

After Send Message Flag



So far the block give no feedback to the sender if the messages and data have been sent. In order to get feedback, we use the Send Message Flag loop block. Behind the scenes this block checks to see if a message has been sent, If the message has then the loop will receive a "True" response and run the code placed inside. If the message hasn't been sent, the block receives a "False" and waits until a true is sent to it.

Example



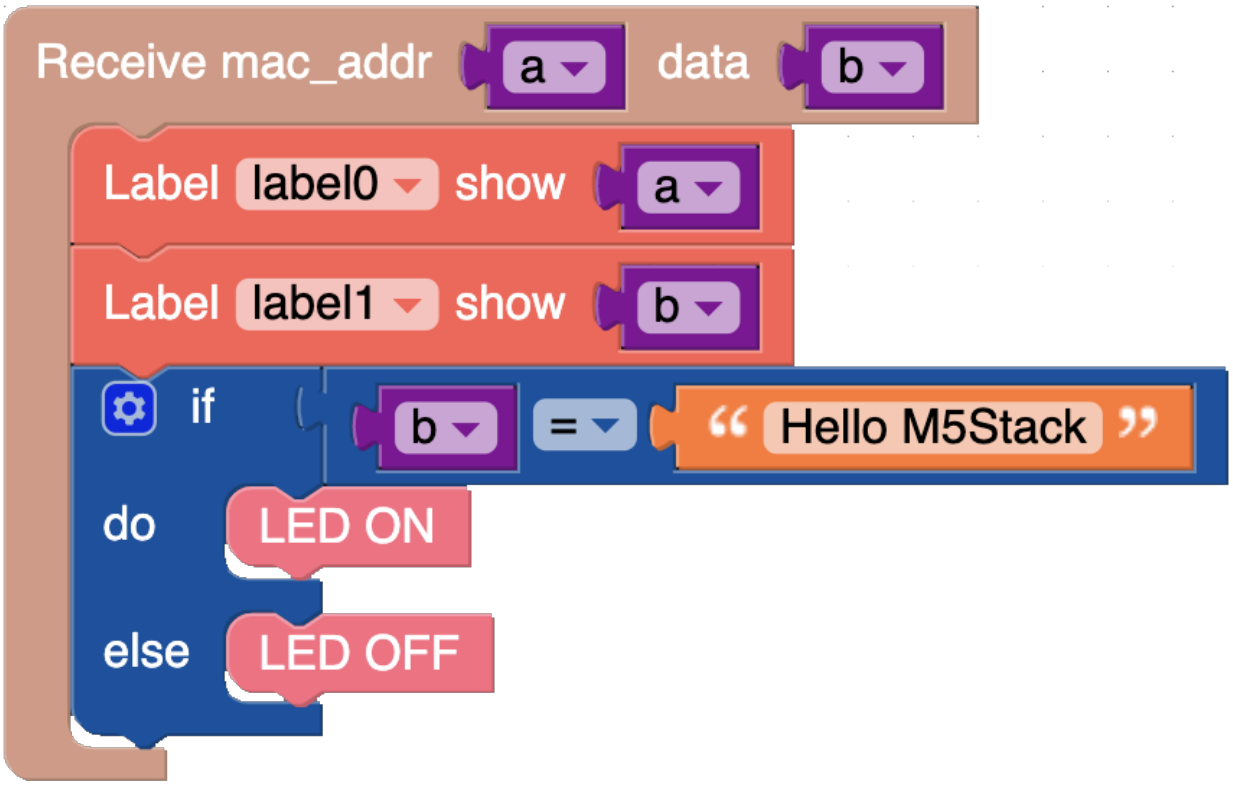
In this example I have used a Label block to show on the screen if a message has been sent or not. The Micropython code same for this loop is as follows.

```
wifiCfg.wlan_ap.active(True)
espnow.init()
label0 = M5TextBox(21, 54, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

Status = None

def send_cb(flag):
  global Status
  Status = flag
  if Status:
    label0.setText('Message Sent')

  pass
espnow.send_cb(send_cb)
```

Receive MAC Address Data

So far I have shown the blocks required to send messages and data from a master device to a slave device. The last block is used on the slave device to receive messages and preform an action based on the received message. The **Receive MAC Address Data** loop receives the sent data and also the MAC address sent by the master device.

Example

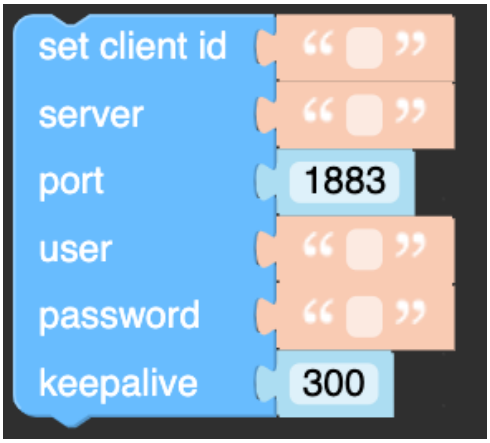


In the following example which must be run on a slave device, the LED will be lit up if the slave device receives the Hello M5Stack message sent in the earlier examples. The Micropython code for this example is as follows.

ESP NOW

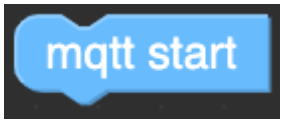
```
from m5stack import *
from m5ui import *
from uiflow import *
import espnow
import wifiCfg

setScreenColor(0x111111)

wifiCfg.wlan_ap.active(True)
espnow.init()
label0 = M5TextBox(21, 54, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)
label1 = M5TextBox(18, 86, "Text", lcd.FONT_Default,0xFFFFFF, rotate=0)

a = None
b = None

def recv_cb(_):
  global a,b
  a, _, b = espnow.recv_data(encoder='str')
  label0.setText(str(a))
  label1.setText(str(b))
  if b == 'Hello M5Stack':
    M5Led.on()
  else:
    M5Led.off()

  pass
espnow.recv_cb(recv_cb)
```

MQTT,
MQTT Setup,



This MQTT Setup block contains the credentials required to connect to a MQTT service.

MQTT Subscribe,
This loop subscribes to an MQTT topic and reruns the code inside keeping the M5Stack or Stick subscribed to the topic.



MQTT Start,
Starts the MQTT tasks.
Get Topic Data,

Returns information from the subscribed MQTT topic.
Publish Topic,

Publishes a message to the selected topic.

Remote Control

The remote folders contain blocks dedicate to allowing one M5Stack or Stick to control another or allow none M5Stack products control.

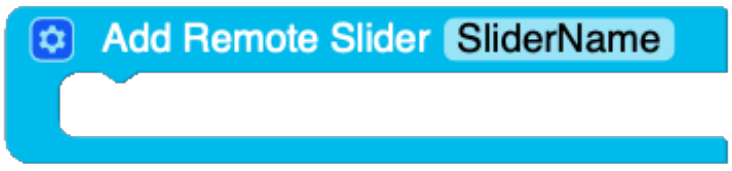
Currently there are two folders for remote access as one contains test version of the existing blocks.

Remote,

Remote QR code block display a QR code on the M5Stacks screen pointing to the remote control hosted page on flow.m5stack.com.

Add Remote Switch,

Add Remote Button,
Creates an On/Off button



Add Remote Slider,
Creates a slider which alters the value of a variable "X". This can be used as a brightness control or for controlling separate colour channels.

Add Remote Other,

Custom Blocks.

The custom block menu does not contain blocks by default but can be used to load any blocks created in the custom block editor.

Appendix 1
Data-sheet Catalogue.

This page is a catalogue of the known data sheets pertaining to modules used throughout the M5Stack Product line. 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.

ADS1100 self calibrating analogue to digital converter
http://www.ti.com/lit/ds/symlink/ads1100.pdf

AK8983C
https://www.akm.com/akm/en/file/datasheet/AK8963C.pdf

ATMEGA328P
http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

AS312 PIR Sensor
https://forum.mysensors.org/assets/uploads/files/1494013712469-pir-as312.pdf

AT6558 GPS Receiver
http://www.icofchina.com/d/file/xiazai/2016-12-05/b1be6f481cdf9d773b963ab30a2d11d8.pdf

AXP192 Single Cell Li Battery and Power Management IC
http://www.x-powers.com/en.php/Info/down/id/50

BMP280 Pressure Sensor
https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

Cadence/Tensilica LX6 Processor
https://mirrobo.ru/wp-content/uploads/2016/11/Cadence_Tensillica_Xtensa_LX6_ds.pdf

CP2102 USB to UART communications chip
https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf

DHT12 Digital Temperature and Humidity Sensor.
http://www.robototehnika.ru/file/DHT12.pdf

ESP32
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ESP32 Wrover Module
https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf

FPC1020A fingerprint Module

http://www.shenzhen2u.com/doc/Module/Fingerprint/710-FPC1020_PB3_Product-Specification.pdf

HK4100 F 5V DC relay
https://img.ozdisan.com/ETicaret_Dosya/445413_4369639.pdf

Holtek HT75XX LDO Voltage Regulator.
http://www.e-ele.net/DataSheet/HT75XX-1.pdf

L293D H-Bridge Driver
http://www.ti.com/lit/ds/symlink/l293.pdf

LM393DR2G
https://www.onsemi.com/pub/Collateral/LM393-D.PDF

MAX2359 Amplifier
https://datasheets.maximintegrated.com/en/ds/MAX2659.pdf

MAX30100 Pulse-Oximeter.
https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

MLX90614 N.C.I.R Sensor
https://www.sparkfun.com/datasheets/Sensors/Temperature/MLX90614_rev001.pdf

MPU6050
https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

MPU9250
https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

PCA9554A
https://www.nxp.com/docs/en/data-sheet/PCA9554_9554A.pdf

SK6812 LEDs
https://cdn-shop.adafruit.com/product-files/1138/SK6812+LED+datasheet+.pdf

SPX3819 LDO Voltage Regulator.
http://www.mouser.com/ds/2/146/SPX3819_DS_R200_082312-17072.pdf

TCA5948 A 8 way I2C Switch.
http://www.ti.com/lit/ds/symlink/tca9548a.pdf

TCS3472 Colour Light to Digital Converter.
https://ams.com/documents/20143/36005/TCS3472_DS000390_2-00.pdf

VL53L0X Time Of Flight Sensor.
https://www.st.com/resource/en/datasheet/vl53l0x.pdf

WS2812b (Neopixel) LED
https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

Appendix 2
I2C Address.

The following tables contain a list of known I2C addresses used by M5Stack devices.

CORE	ADDRESS	CHIP
M5CORE	0x75	IP5306
M5CORE	0x68	MPU6886
M5CORE	0x6C	SH200Q
M5CORE	0x10	BMM150
M5STICK	0x75	IP5306
M5STICK	0x68	MPU9250
M5STICK-C	0x34	AXP192
M5STICK-C	0x6C	SH200Q
M5STICK-C	0x51	BMM8563

Table 1 - M5Stack and M5Stick internal additional hardware I2C addresses. Please not that the additional devices were optional on some versions.

Device	Address	Chip
ACCEL	0x53	ADXL345
ADC	0x48	ADS1100
Card KB	0x5F	MEGA328P
Colour Sensor	0x29	TCS3472
DAC	0x60	MCP4725
ENV	0x5C	DH12
ENV	0x76	BMP280
EXT I/O	0x27	PCA9554PW
Heart	0x57	MAX30110
Joystick	0x52	MEGA328P
Makey Makey	0x51	MEGA328P
N.C.I.R.	0x5A	MLX90614
PaHub	0x70	TCA9548A
PbHub	0x40	MEGA328P

Device	Address	Chip
R.F.I.D	0x28	MFRC522
Thermal	0x33	MLX90640
Trace (See 3.3.38)	0x5A	MEGA328P
TOF	0x29	VL53L0X
HAT-ENV	0x5C	DH12
HAT-ENV	0c77	BMP280
HAT-ENV	0x10	BMM150
HAT-THERMAL	0x33	MLX90640
HAT-NCIR	0x5A	MLX90614
HAT ADC	0x48	ADS1100
HAT-TOF	0x29	VL53L0X & VCSEL
HAT-DAC	0x60	MCP4725
HAT-JOYSTICK	0x38	STM32F030

Table 2 - Unit I2C addresses.

Camera	Address	Chips
ESP32	0x68, 0x76	MPU6050, BME280
M5CAMERA	0x68, 0x76	MPU6050, BME280
M5CAMERA - F	0x68, 0x76	MPU6050, BME280
M5CAMERA - X	0x68, 0x76	MPU6050, BME280

Table 3 - Camera hardware devices. Please not that the additional devices were optional on some versions.

MODULE	ADDRESS	IC
PLUS	0x62	MEGA328P
GO - PLUS	0x61	MEGA328P
STEP - MOTOR	0x70	MEGA328P
SERVO	0x53	MEGA328P
LEGO +	0x56	MEGA328P
FACES - ENCODER	0x5E	MEGA328P
FACES - JOYSTICK	0x5E	MEGA328P
FACES - KEYBOARD	0x78	MEGA328P

MODULE	ADDRESS	IC
FACES - CALCULATOR	0x78	MEGA328P
FACES - GAMEBOY	0x78	MEGA328P
FACES - RFID	0x28	MFRC522

Table 4 - Module and Faces I2C address.

Appendix 3
Table of symbols available on the Card KB.

The following table list the symbols available through the various Key combinations along with the raw hexadecimal values.

Key	Value	Sym+Key	Value	Shift+Key	Fn+key
Esc	0x1B	Esc	0x1B	Esc	Esc
1	0x31	!	0x21	1	1
2	0x32	@	0x40	2	2
3	0x33	#	0x23	3	3
4	0x34	$	0x24	4	4
5	0x35	%	0x25	5	5
6	0x36	^	0x5E	6	6
7	0x37	&	0x26	7	7
8	0x38	*	0x2A	8	8
9	0x39	(	0x28	9	9
0	0x30	)	0x29	0	0
Del	0x08	Del	0x08	Del	Del
Tab	0x09	Tab	0x09	Tab	Tab
q	0x71	{	0x7B	Q	Q
w	0x77	}	0x7D	W	W
e	0x65	[	0x5B	E	E
r	0x72	]	0x5D	R	R
t	0x74	/	0x2F	T	T
y	0x79	\	0x5C	Y	Y
u	0x75	|	0x7C	U	U
i	0x69	-	0x7E	I	I
o	0x6F	'	0x27	O	O
p	0x70	"	0x22	P	P

Key	Value	Sym+Key	Value	Shift+Key	Fn+key
Fn	NULL	NULL	NULL	NULL	NULL
Shift	NULL	NULL	NULL	NULL	NULL
a	0x61	;	0x3B	A	A
s	0x73	:	0x3A	S	S
d	0x64	`	0x60	D	D
f	0x66	+	0x2B	F	F
g	0x67	-	0x2D	G	G
h	0x68	_	0x5F	H	H
j	0x6A	=	0x3D	J	J
k	0x6B	?	0x3F	K	K
l	0x6C	NULL	NULL	L	L
Enter	0x0D	Enter	0x0D	Enter	Enter
Sym	NULL	NULL	NULL	NULL	NULL
z	0x7A	NULL	NULL	Z	Z
x	0x7B	NULL	NULL	X	X
c	0x63	NULL	NULL	C	C
v	0x76	NULL	NULL	V	V
b	0x62	NULL	NULL	B	B
n	0x6E	NULL	NULL	N	N
m	0x6D	NULL	NULL	M	M
,	0x2C	<	0x3C	,	,
.	0x2E	>	0x3E	.	.
Space	0x20	Space	0x20	Space	0x20
Up	0xB5	Up	OxB5	Up	Up
Down	0xB6	Down	0xB6	Down	Down
Left	0xB4	Left	0xB4	Left	Left
Right	0xB7	Right	0xB7	Right	Right