UIFlow Handbook

Written by Adam Bryant

Table of Contents

Special thanks for Contributions to this book.

Adam "AJB2K3" Bryant - This is me, the author,

Jimmie Lau - For creating the M5Stack.

Luke "Lukasmaximus" Henderson - For answering my constant questions and being the bridge between myself and M5Stack and for donating an M5Go.

Ben " World101" Stein for his work on porting Blynk to the M5Stack UIFlow system and for trying to teach me how it works.

Forum Member Pattiuak for his great servo tester code.

Preface

This book is a collection of written up notes that I have made while Learning how to use and program the M5Stack range of Programmable controllers. My hope is that by the end I will have produced a book that will be usable by other beginners to the M5Stack range of products. I have always struggled to learn languages and programming was always beyond me if I couldn't find anyone who's work I could copy and learn from.

About this book

At the time of writing (winter of 2018), the documentation for the M5Stack family of products was incomplete or incorrect making learning how to use the hardware difficult. With the introduction of the UIFlow blocky programming language, programming this device is finally within my grasp.

Because of the way that I am learning how to use and program the M5Stack, I have divided the book into the following chapters.

Chapter 1 *Welcome to the M5Stack and its family*, introduces the the M5Stack system and how it is divided into the various hardware category's.

Chapter 2 *M5Go!* The M5Go is available in a starter kit that includes the M5Go version of the M5Stack core Module, a selection of sensors and out put devices to interface with the world, the leads necessary to connect it all together and, a selection of *LEGO® Technic™* compatible parts.

Chapter 3 *M5Stack Hardware*. In this chapter I will give a more detailed explanation of the individual parts that make up the M5Stack Family including links to technical data-sheets where available.

Chapter 4 *UIFlow.* In this chapter I will introduce you to M5Stack Integrated Development Environment which has the user friendly Blocky programming environment and the more advanced Micropython programming environment that make up the core of all firmware on the M5Stack range of devices.

Chapter 5 *Micropython*. I will show you how to programming the M5Stack family using the Micro python programming language which is the other programming environment available in UIFlow.

Chapter 6 *Examples and Demos*. This chapter will show you how to move beyond the basic hardware and sample code in order to to combine them to make things more interesting.

Chapter 7 *Moving beyond (Part 1)* This chapter is the first that will show what can be created if we use additional hardware along with the M5Stack Products.

Chapter 8 *Moving beyond (Part 2)* In this chapter I will show you how to build a basic tracked *LEGO® Technic™* platform that we can connect hardware to and use as a basic robot.

Appendix A - Data-Sheet Catalogue. This section will contain links to locations where technical document for various components can be downloaded.

Downloading the code.
All code provided in this book is provided "As is" and is only meant as a bases for your own projects. Code produced by myself along with additional documents and updates is available from

https://github.com/Ajb2k3/UIFlowHandbook

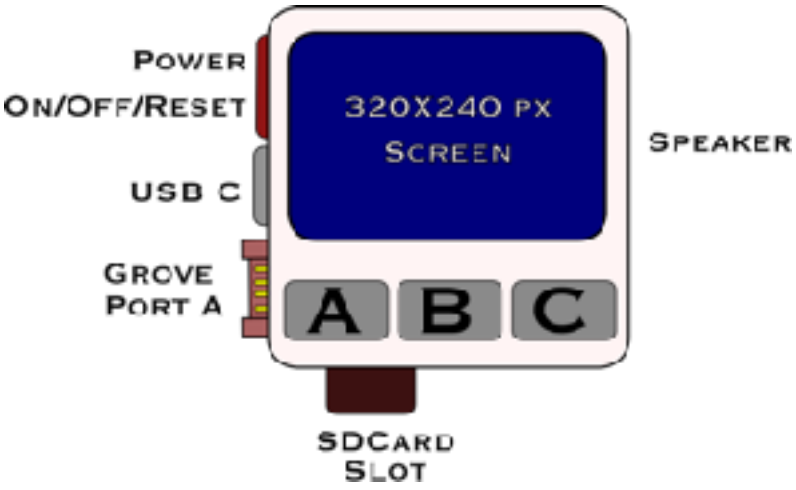
1

Welcome to the M5Stack.

The M5Stack is a range of Programmable devices measuring just 50mm X 50mm (2"x 2") and built around the ESP32 IoT micro controller created by Esprissif. The main body contains the ESP32, a screen, three buttons, a SD micro card slot, a digital (I2C) Grove connection, a USB C port, a power/reset button and a mono speaker.
The range of add-on's consist of sensor units and output units that connect to the m5stack using the Grove Connection.Stackable modules that connect between the M5Core and the Base Plates and the baseplates that contain batteries and other addition hardware.

Created by the M5Stack Company that shares it name, the various modules and add ons have been created in order to decrease the time it take to create a working product sample when developing new products.

All M5Stacks share the same common features.



The following table show the different version available of the M5Stack modules as of writing.

M5 Core

M5Core Gray

M5 Go

M5 Fire

	M5Core	M5Core Gray	M5Core White (Go)	M5 Fire
Mems	None	MPU9250	MPU9250	MPU9250
Ram	520KB	520KB	520KB	520KB+4M
Flash	4MB	4MB	4MB	16MB
Screen	320x240 Colour	320x240 Colour	320x240 Colour	320x240 Colour
Speaker	Yes	Yes	Yes	Yes
Base Plate	I/O	I/O or Faces Base.	Go Base	Go Base
Grove Ports	1	1	3	3
RGB LED	None	None	10x SK6812	10x SK6812
Battery	150mAh	150mAh	550mAh	550mAh

DISCLAIMER, THESE SPECIFICATION ARE SUBJECT TO CHANGE BY M5STACK.
PLEASE NOTE THAT THE ABOVE TABLE ALSO CONTAINS THE M5STICK'S FOR COMPARISON.

This table has been reduced to just the main core units but, there are a few more versions of the M5Stack that are available.

2

M5 Go!

The M5Go is the beginners introduction set to the M5Stack and come is two sizes.

M5Go Lite is the smallest version and contains the White M5Stack, M5Go Base, one Environmental unit, a grove lead to connect the environmental sensor the the M5Stack, and the USB leads to connect to the computer.

M5Go IoT Starter Kit contains the White M5Stack, M5Go Base, one Environmental unit, an I.R. Sensor, R.G.B Light unit, a P.I.R Sensor, POT/Rotation Sensor, a water sensor, a selection of grove leads to connect the sensors the the M5Stack,a POGO lead, the USB lead to connect to the computer, and a selection of *LEGO® Technic™ compatible parts*

Insert Image of M5go starter kit!

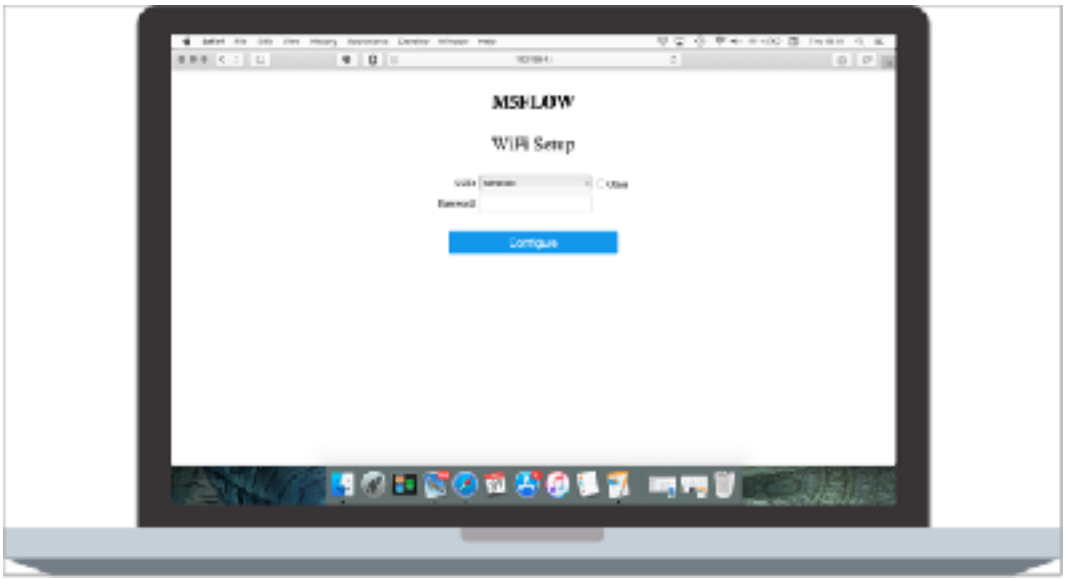
On first start up the M5Go demo will run taking you through various functions that are available to the M5Go and the included units. If the power button is pressed, the M5Go will reset and run the connection program in an attempt to connect to flow.m5stack.com.

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).

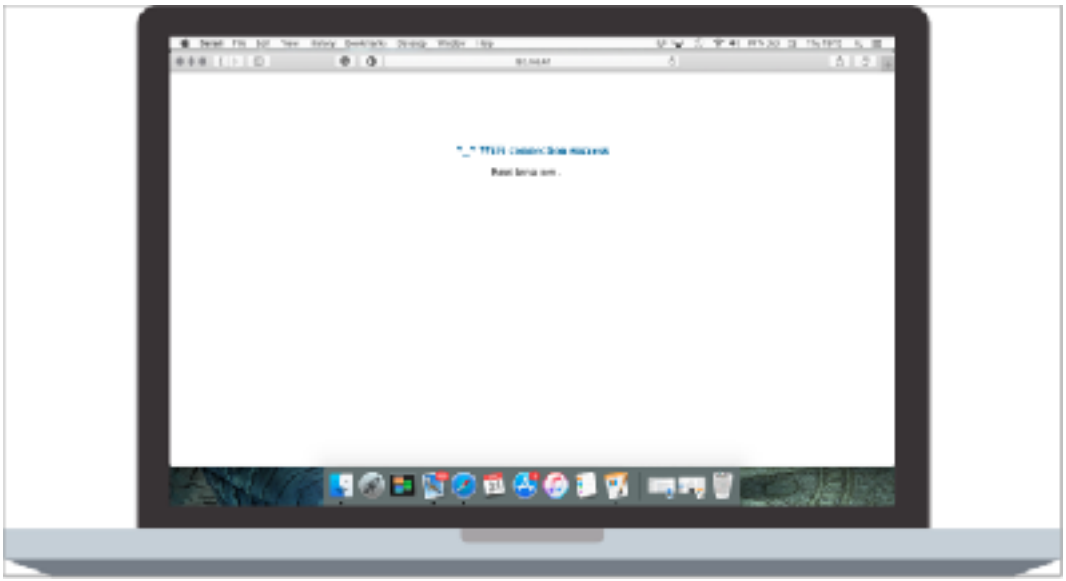
On your computer you need to go to the wifi connection, click on it to bring up the list of available wifi devices and click on the name that matches the screen of the M5Stack.

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



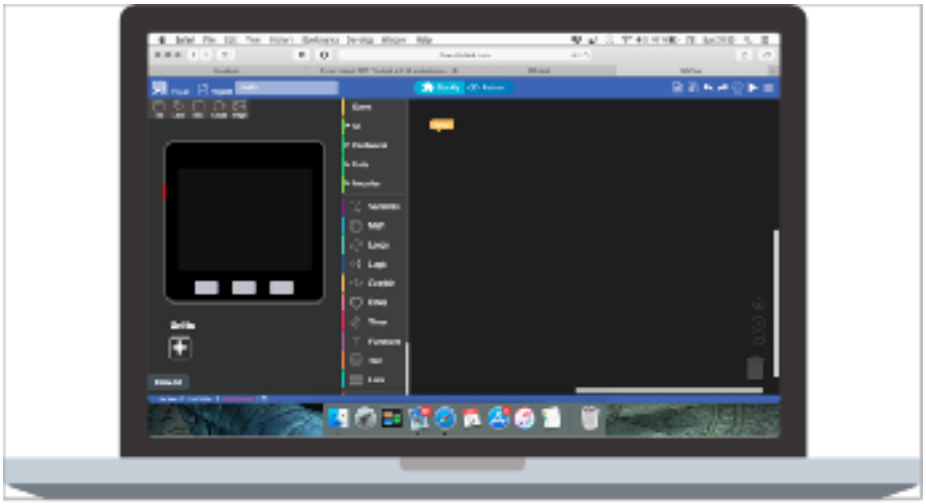
Select your wifi network that your computer normally connects to, type in the password and wait for it to connect.

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.

I**mportant note: Once in a while my own M5Stack refuses to connect if there are too many devices active on the network..**

2.1

M5Go's Demo and the Units.

As mentioned previously, there is a function demo that is preinstalled in the M5Stacks firmware. When run you will be treated with a Hello. The three buttons on the front of the M5Stack are used to navigate through the demo.

Pressing button "C" (the right hand button) will take you to the next screen that will show you how to switch the M5Stack on or off and reset the M5stack.

Next is the speaker demo, if you press button "B" (the middle button) a tune will play from the built in speaker.

Press button "C" and this takes us to the microphone demo. You will see a jagged line moving in M5os (Our avatar), if you speak into the bottom of the M5Stack you will see the jagged line get bigger and smaller.

Pressing "C" takes us to the Gyro demo. There is a small red spot that will change to black. once it turns black, if you move and tilt the M5Stack, the black dot will move.

Pressing "C" again now takes us to the RGB demo which lights the red and blue LED bars on the M5Go base plate.

The next screen is an explanation of what the ports on the M5Go can do. Pressing next takes us onto the first of the Unit demos.

Turn off the M5Go and plug the Environmental unit into the RED Port (Port A) of the M5Go using one of the four wire Grove leads. When you turn the M5Go back on and navigate to the environmental demo screen the ENV SENSOR box on screen will now show the temperature, Humidity and the air pressure.

The next demo is the Motion sensor which is plugged into the BLACK Port (port B) and uses the PIR Sensor.

Now it is time for the RGB LED unit, this is plugged into the RED Port.

The Penultimate demo is the IR Sensor.

The Last demo is the Angle sensor which show how far you have rotated the angle sensor and using the reading to brighten or dim the LED bar on the M5Go's Base.

2.2

Moving on and experimenting with the M5Go.

In the previous parts of this chapter I introduced you the the M5Go demo and connecting to UIFlow. Now that we have had a play with the demo, it time to see what else we can do with the M5Go.
Before we start using UIFlow with the M5Go, we need to play with the LEGO® Technic™ compatible parts.

2.2.1

Experiment 01- Environmental monitor.

PHOTO REQUIRED OF ASSEMBLED ENVMON

The Aim of this experiment to to make a small monitor to sense the temperature, air pressure and humidity of a room. To build our monitor we will need the following items from the M5Go Starter Kit.

- M5Go,
- ENV Sensor,
- Short Grove cable,
- Two Gray bars with nine round holes in them,
- Six black connectors,

Turn the M5Go upside down to show the base which has been designed to connect with *LEGO® Technic™.*

Take four of the black connectors and push them into the four corner holes of the base.

Take two off the longer beams and place them as show hanging out over the side with the power button and the grove connector.

Take the ENV (Environmental) Sensor, place two black connectors into the back of the sensor.

Take the short Grove lead and plug one end into the sensor and the other into the red port (Port A) of the M5Go.

Push the ENV sensor into the end of the gray beams and we are ready for our first UIFlow Experiment.

To reproduce the code shown below in the screen shoot and upload it to the M5Go we need to delve into the drawers and find pull out blocks that will make up our code.

Screen shot off the code for our environmental sensor.

The first thing to do is to click on the [+] symbol below the M5Go to bring up the Unit window. Find the ENV Sensor and make sure the port is set to "A".
Now we add a Title Bar, set the text to Environmental Sensor and, set the back ground to green. To do this we click on the Title icon and drag it to the top of the Screen of the M5Go. In the "UI" drawer we now find a series of blocks for the Title. Click on the "Title Show" block and it will be placed on the screen "Greyed Out", click on this block and drag it under the Setup block and it will turn pink. Click on the box between the quote marks and type ''Enviromental Sensor''. Now open the Title drawer again and click on the "Set Title backgroundColor" block and drag that into position under the "Title Show" block. Click on the small red square off the block and the colour picker will pop up, click on green and the colour picker will disappear leaving a green square where the red had been.
To add text to the screen we can use the "Label" function. In this example I have added six labels and set their "X" position to 120px to get them close to the middle of the screen. Click on the label icon and drag six of them onto the M5 screen. The order I placed them was

Label0
Label3
Label1
Label4
Label2
Label5

You don't need to place them in the order I have, you just need to remember which order the label's are in.
Open the "UI" drawer, open the "Label" drawer and click on the "Label Show" block to add it to the screen under the "Set title0 backgroundColor" block.
Right click on the "Label Show" block and a new menu pops up. this menu allows us to duplicate or delete blocks on screen, click on "Duplicate" and a second "Label Show" block will appear, move this under the first "Label Show" block and duplicate the block a third time and we will now have three "Label Show" blocks. These three labels will be the titles for the readings we will be collecting from our sensor, click on "Label0" and change them to different Labels and then fill in the text as shown in the code screenshot above.

Now that we have made our UI or User Interface, it is time to access the sensor and get some readings. We will want the reading to keep updating the value on screen to show the changes. To do this we need to make a loop that will constantly read the sensor and update the screen values. Click on the "Event" drawer and click on the "Loop" block to add it to the screen. drag it under the Label blocks to add it to our code and then duplicate the First three Label block but this time placing them inside the loop block. Change the Label0 to number different to the previous three blocks.
To get the values from the sensor, we need to open the "Units" drawer and then open the "ENV" drawer. The three blocks here are "Value" blocks, all these do is read the sensor and put the value into the code. To display it on screen we need to add each of the three blocks to the three "Label Show" block inside the loop.
If we click on the "Play" near the top right of the UIFlow screen, the screen of the M5Go should change to show the following (The numbers should be slightly different depending on where you are running your M5Go.


Environmental Sensor
Temperature
25.34
Pressure
1014.28
Humidity
1014.28

2.2.2

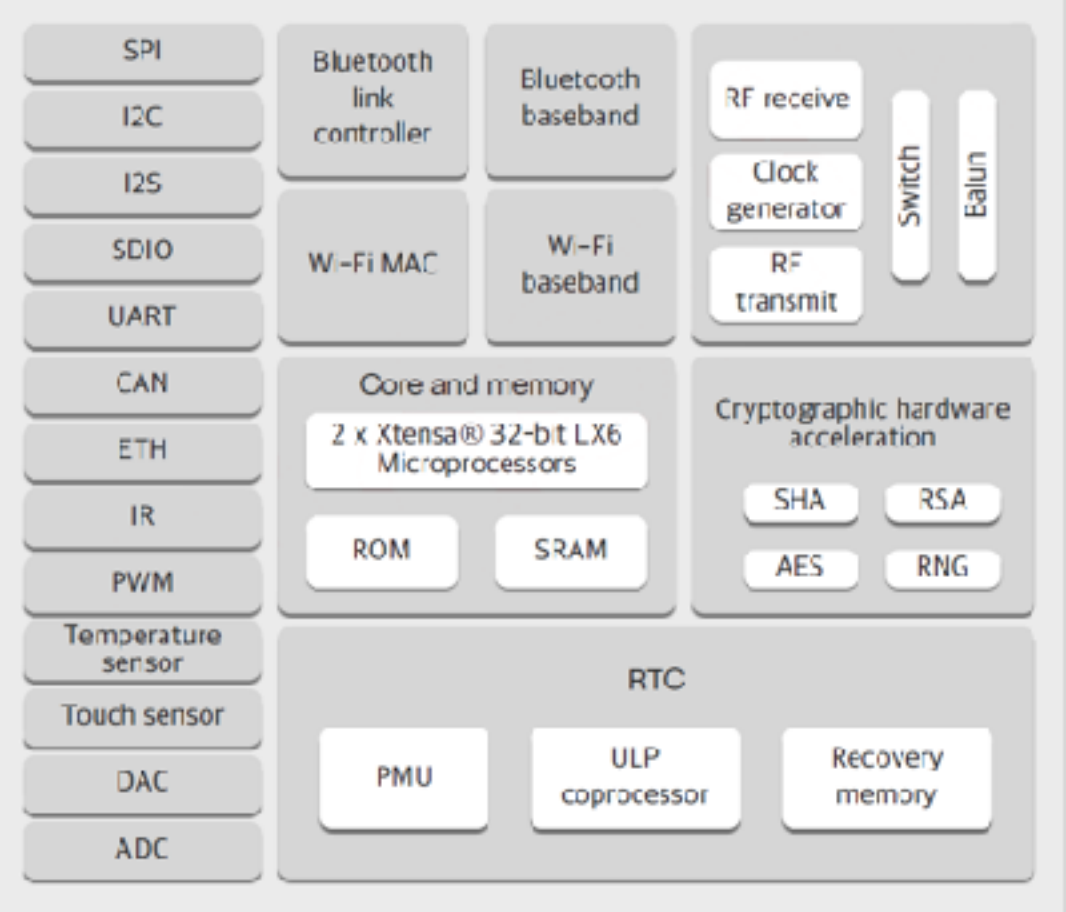
Motion Sensor (PIR)

3

M5Stack Hardware.

In a previous chapter I mentioned that the M5Stack range of products are based around a 50mm X 50mm core unit but, what I neglected to mention is that there is also a range of mini units that are around 25mm X 50mm in size. While each of the core units differ slightly, all of the core units are built around the 240MHz duel core ESP32 microprocessor by Espressif.

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



As you can see in the above block diagram taken from the espressif data sheet, the ESP32 is a little special. Not only does it contain the two 240MHz Xtensa 32-bit LX6 microprocessors, the chip also has an Ultra Low Power co-processor that can be used when the two Xtensa microprocessor are in sleep mode. I wont be going much further beyond this information on the ESP32 but if you would like to learn more about its capabilities I recommend Kolban's book on the ESP32 by Neil Kolban https://leanpub.com/kolban-ESP32

Currently there are five M5Stack cores and two M5Stick units.
The M5Stack cores make use of a 320x240px colour screen while the M5Sticks make use of a 64x128px mono OLED screen, a duel function power button, a USB C port and a single digital I2C Grove port.
Where they differ is that the M5Stacks are made to be upgraded by stack modules between the Core and the baseplate via the 2.54mm 2X15 pin expansion bus. The M5Stack cores also feature a Micro SD card slot that can be written to and read from.

3.1

M5Stack Cores

While all the M5Stack cores look the same there are some slight differences in them.

The basic version of the M5Stack in the black model.

While this has the lowest specifications of all the models it does come with the I/O base that allows direct connection to the M5Stack Internal Bus.

The next of the basic versions is the Faces version, this came in a gray case and is the same as the black but came with the Faces adapter base.

The third of the basic models is the Special gray. While mostly like the Black in that it came with an I/O base, it was the first to come with the MPU9250 IMU.

It is worth noting that while they now come with 16MB of flash, the older versions of these three only came with 4MB of flash.

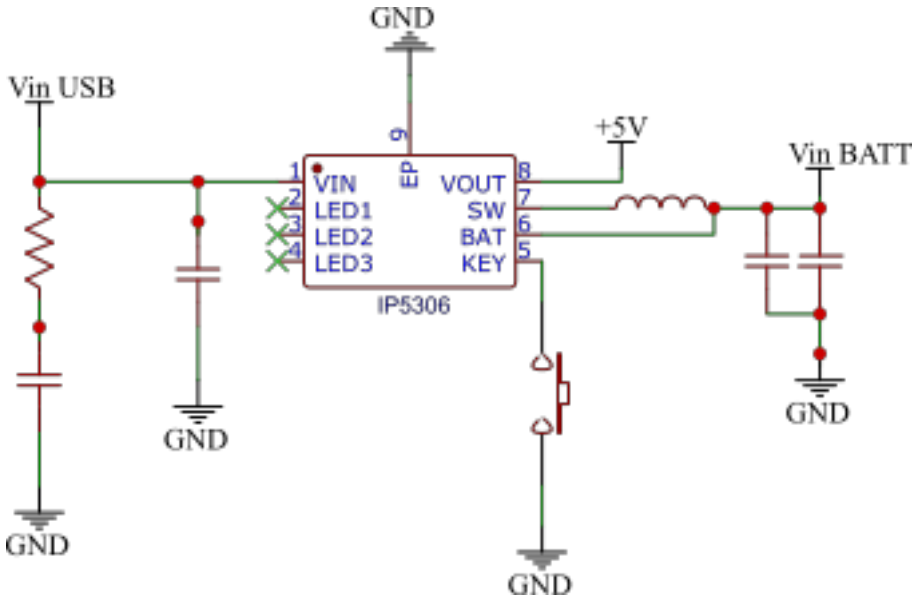
The mid level version aimed at S.T.E.M learning came in a white case and was the first to come with the Go Base.

The top level version and current model is the M5Fire that comes with the Go Base but comes with 16MB flash and 4MB of PSRAM as well as the built in IMU.

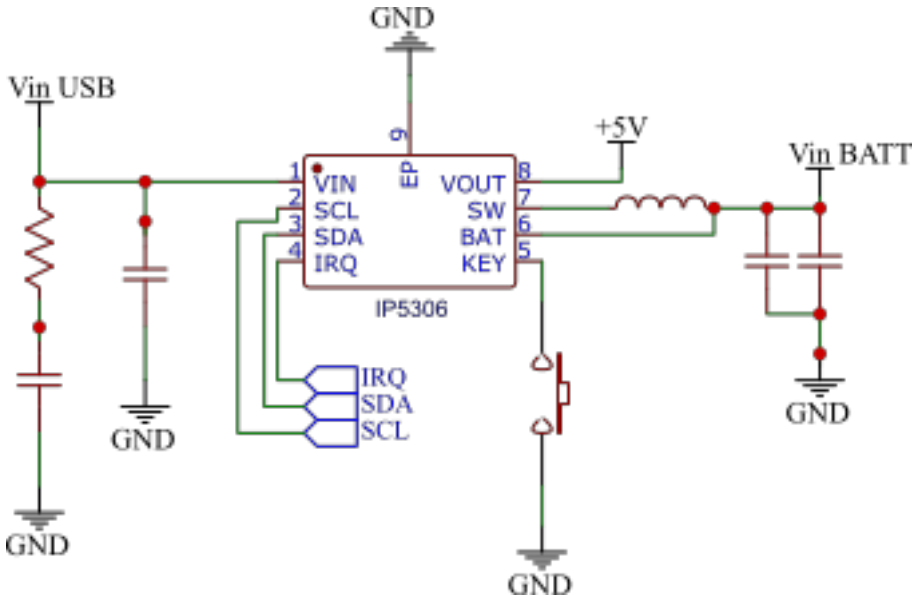
3.1.1

Power Management.

While all M5Stacks make use of the IP5306 power management chip, there are two version of it used. Prior to the M5Fire, M5Stack used the normal version shown below in the schematic diagram.



However, with the M5Fire and customised version was produced that communicated with the M5Fire over the I2C connection and allowed it to communicate power management and battery data.



The difference in the two simplified schematic diagrams is that in the first, the LED pins are used to show the charge status of a connected battery while in the second the LED pins are replaced with SDA, SCL, and IRQ. Again, before I go any further I must remind you that the above circuits

are not complete and not those used by M5Stack. These diagrams are just simplified circuits I pulled together to show the difference in the two versions.

3.2

M5Sticks

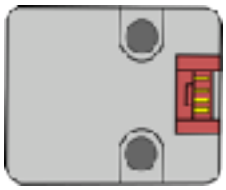
As of writing there are three M5Sticks (the orange MK C has just arrived to testers.)
The white stick is the basic model and is the equivalent of the M5Stack Core Black, the dark gray comes with the IMU fitted and is the equivalent of the M5Go.
The new model which come in an orange case has a colour screen and an I/O port on the top along with the digital Grove connector on the bottom.

3.3

Units

The M5Stack system has a series of input and output units that connect to the various cores and sticks via one of the three Grove ports available on the core or the M5Go base plate. Designed by Seeedstudio, the Grove system is a modular four wire system that uses a four pin connector that can only be connected one way around.

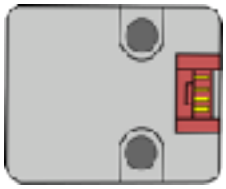
While the Grove connector only has four wires which are,



- Pin1 - Yellow,
- Pin2 - White,
- Pin3 - Red
- Pin4 - Black

There are currently four modes that these wire can be used in

Digital Mode.



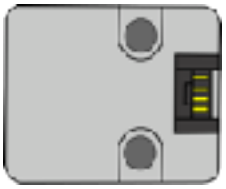
In Digital mode the connections are as follows:

Pin 1 = D0
Pin 2 = D1
Pin 3 = VCC
Pin 4 = GND

Most Digital units use Pin1, D0, but some may use D1. In a later chapter I will show you how to make an adapter that will allow you to switch the pins around so that one device uses D0 and a second identical unit will be able to use D1 at the same time.

In Digital mode, the devices will Pull the pin high causing the M5Stack to read it as a 1 or pull it low causing the M5Stack to read it as a 0.On the M5Stack and M5stick Port A, Red Port is the Digital port and the I2C port.

Analogue Mode.

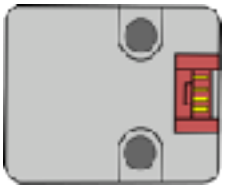


In analogue mode the connections are as follows:

Pin 1 = A0
Pin 2 = A1
Pin 3 = VCC
Pin 4 = GND

As with Digital mode, most units will use Pin1 (A0) however some may use Pin2(A1).
In Digital mode the pins only have two values High (1) or low (0). In analogue mode the M5Stack can read a value between 0 and 4095. On the M5Stack Port B, Black Port of the Go base is the analogue port. Units designed to work with the I/O port will normally have a black grove connector on them instead of a red grove port found on I2C devices.

I2C Mode.

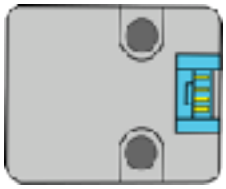


In I2C Mode the connections are as Follows.

Pin 1 = SCL
Pin 2 = SDA
Pin 3 = VCC
Pin 4 = GND

In I2C mode We can connect many devices to the same pins as long as each device has a different address.
Like digital mode, I2C devices are connected to Port A, the Red Port.

UART Mode.



The forth mode that units can use is **UART** mode. **UART** stands for **U**niversal **A**synchronous **R**eceiver-**T**ransmitter and in this mode the pins are as follows,

Pin 1 = RX
Pin 2 = TX
Pin 3 = VCC
Pin 4 = GND

In UART mode, devices communicate with the M5Stack using both the TX and RX pins.
UART mode uses Port C (Blue port) of the Go base and units designed for use with the UART will normally have a blue grove connector.

Normally Units will have colour connector shrouds matching the port colours but some units will come with white shrouds marking them as generic units.

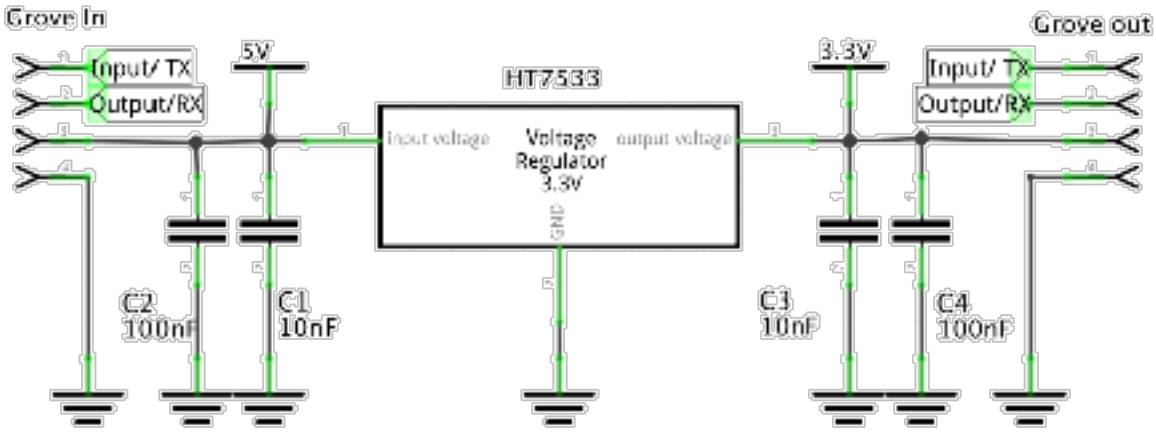
Over the following pages, I will give you a guide to the various units that are currently available along with the normal modes that they operate in. Some times you will see the same unit appear in more than one column as this is because they have more then a single mode of operation.

Digital Units	***I2C Units***	***Analog Units***	***UART Units***	***Generic Units***
CatEar, NeoHex, Neopixel Strip, Relay, Weight,	ADC, CardKB, Colour Sensor DAC, Environment, EXT I/O, Heart, I2C Expander, Joystick, Makey Makey, NeoFlash*, NCIR, RFID Thermal, TOF, Trace *(1)	Angle Sensor, Button (Single and Duel), I/O Expander, IR Sensor, Light Sensor, PIR, NeoFlash*,	Fingerprint, GPS, RS485 adapter,	3.96 Module, Butterfly, Camera, Grove T, Grove Hub, Grove Connector, Proto, Servo,

* The Neoflash has both an I2C port and and Analogue Port.
*(1) The Trace unit is for the LIDAR Bot.
Not all analogue devices use the same GPIO pin to communicate, Some use A0 and some use A1. This will prove useful as it allows us to connect more then one analogue device to the same port through one of the adapters i.e the P.I.R. and Relay units used is the Ambush Sentry chapter found later in this volume.
In theory, as long as we can program the M5Stack to work with devices, any devices using the grove system should be usable.

Unit Power supply.

The Grove system supplies 5V to devices connected to them however most devices require a 3.3V supply. For these devices the units have their own power regulator based around the HT7533 Low Dropout regulator made by Holtek. Below is the schematic diagram of the circuit found in most units.

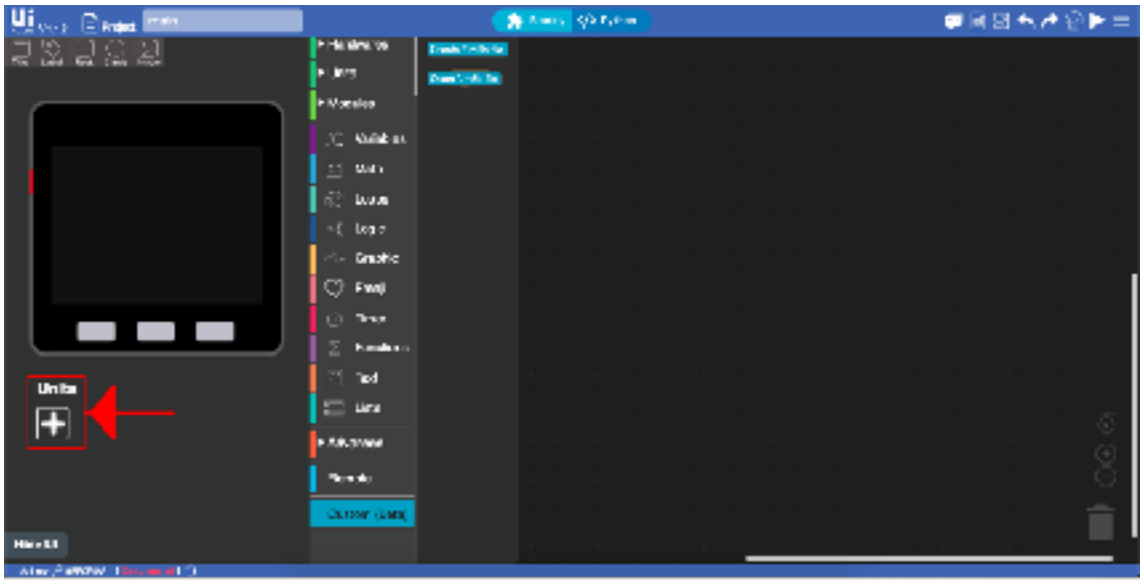


Grove In is the connector found on the Units but, Grove out is only there to represent internal connections.

3.3.0

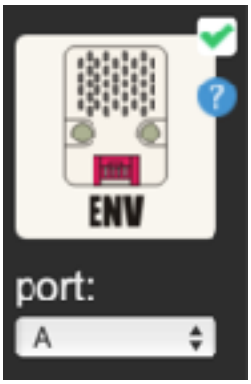
Using Units in UIFlow.

To use a unit in UIFlow and access the code blocks connected to the unit, we first need to add the unit to UIFlow. To add a unit, click on the "+" below the virtual M5Stack or M5Stick.

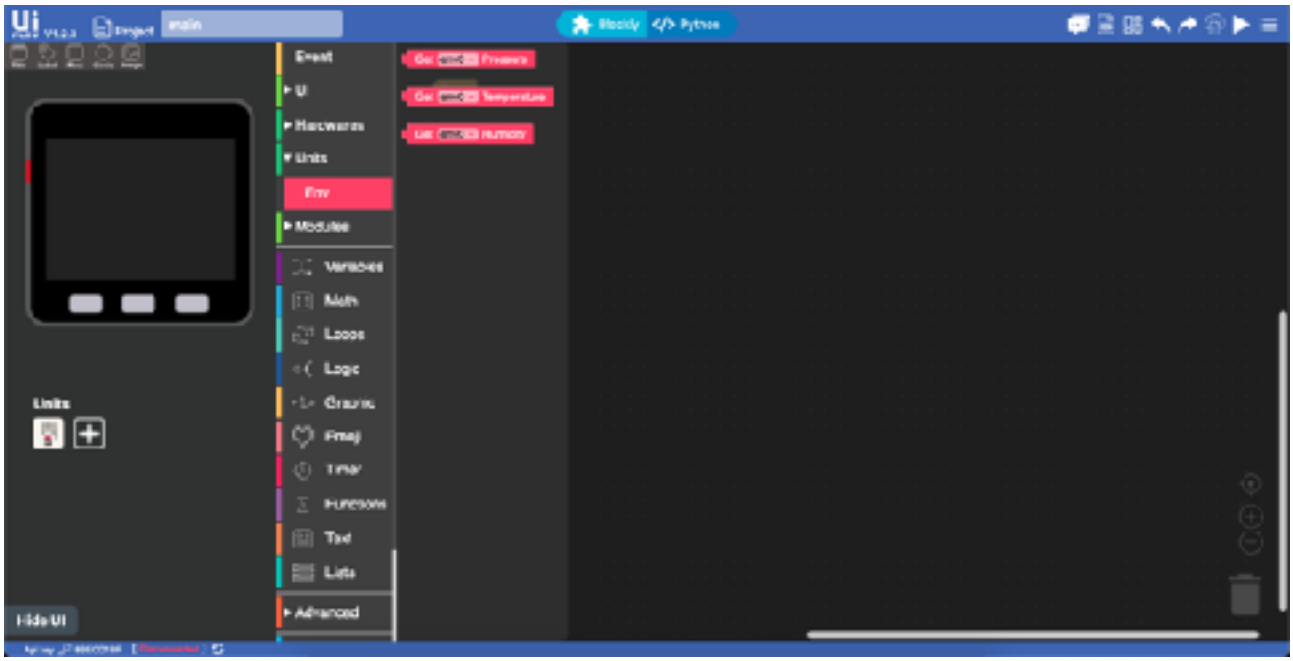


This brings up the Unit Selection window.

To add a unit to UIFlow click on it and a tick will appear in the top right corner of the units picture.



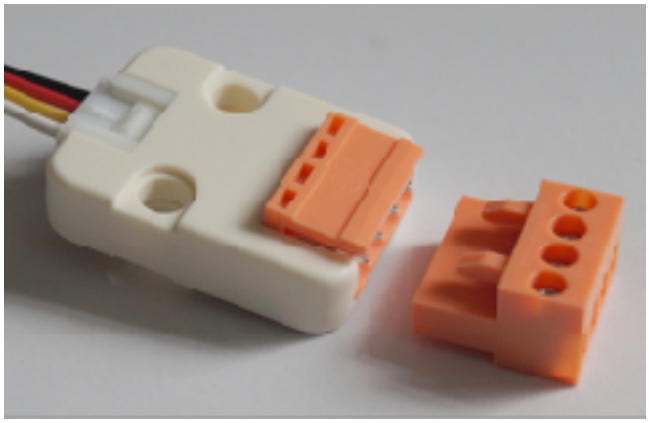
And below it port will appear with a drop down box.(I have describe which port is which earlier in the chapter.) Normally the port will already be set to the default port for the M5Stacks and M5Sticks but, sometimes you made need to change the port letter. Click on the two arrows to open the port dropdown list and select the port you will be using. Once finished click the green "OK" button and you will be returned to the main screen with the unit now appearing below the virtual M5Stack or M5Stick.

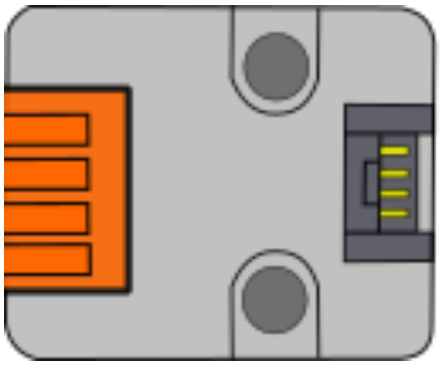


If you click on Units in the Block menu the unit will now appear and if you click the unit, the units code blocks will now be available.

3.3.1

3.96 Module.



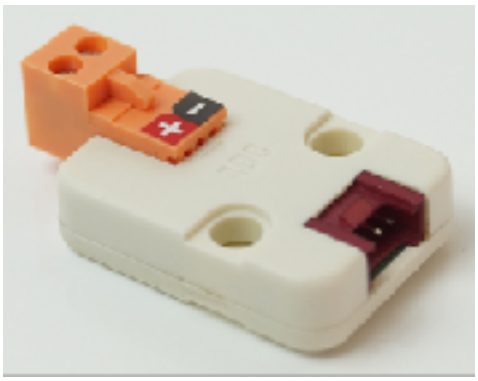


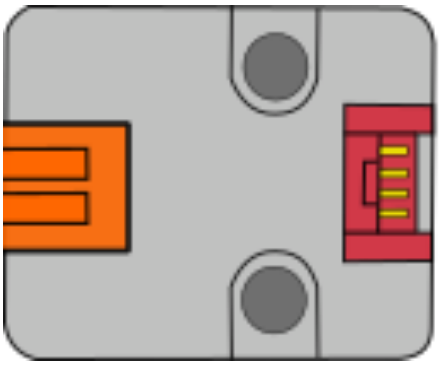
The 3.96 module is a simple module that has a female grove connector on one side and a VH3.96 Screw terminal plug with matching socket on the other. The module has been designed to allow you to connect non grove connector accessories to the M5 system by screwing the wires into the screw terminal plug.

When using none Grove devices you need to be careful that the signal lines are compatible with the M5Stack. According to the documents it means that the signal voltage must be no higher than 1.0V.

3.3.2

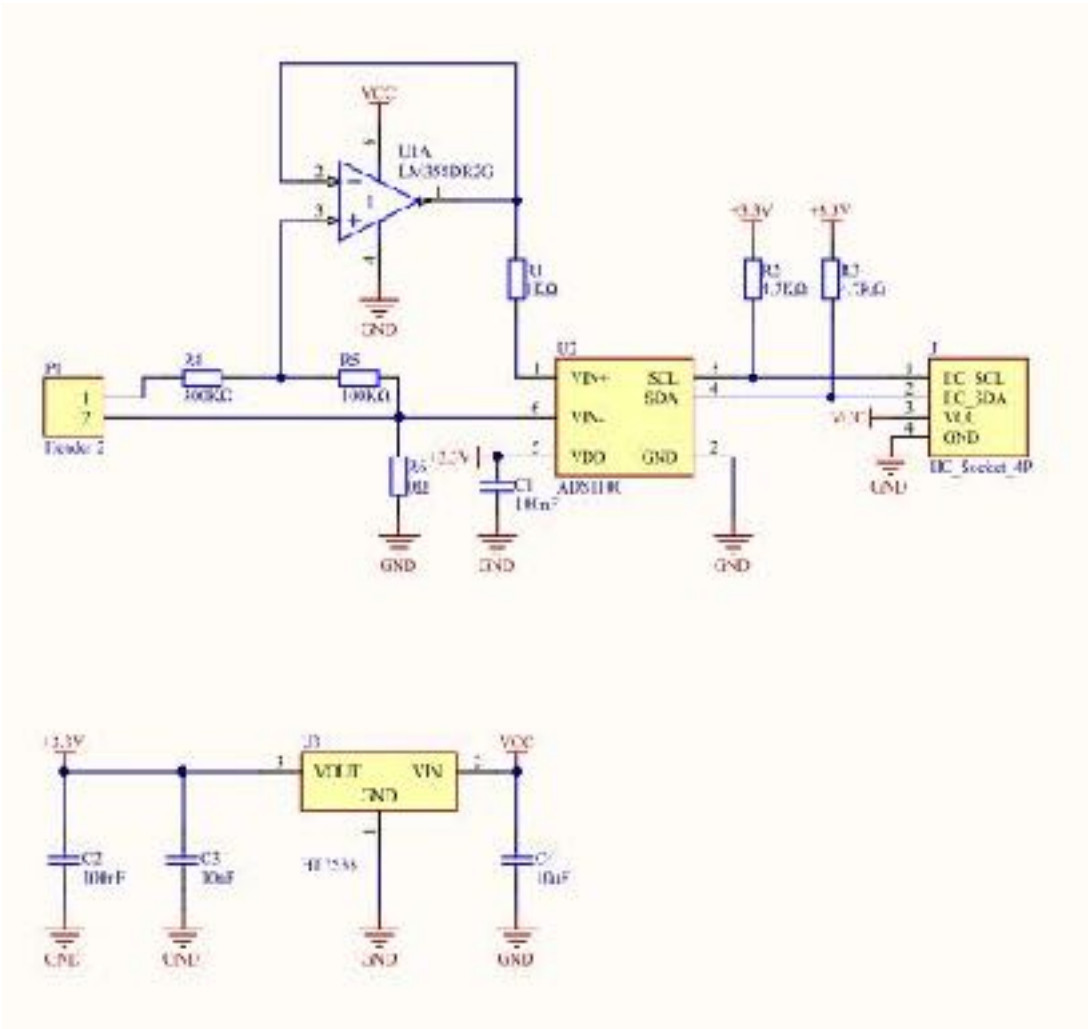
ADC (Analogue to Digital Converter.)





I2C Address = 0x48
Connect to Port A (**Red Port**).

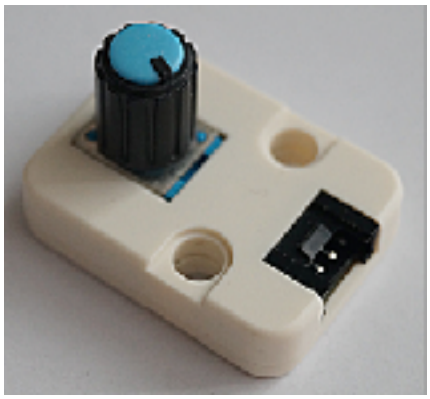
The ADC Has a two pin VH3.96 plug and socket on one side and connects to Port A (I2C Port) of the M5Stack. The unit uses an ADS1100 self-calibrating Burr-Brown converter made by Texas Instruments with a 16 bit resolution.
While the ADS1100 only has a detection voltage of 0 to 5V, the additional of the LM358 give the ADS1100 the ability to read from 0 to 12V.
Below is the circuit diagram as provided in the M5Stack documents.

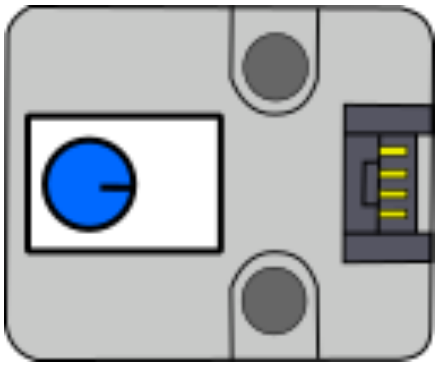


Here we see that the sensor also has its own power adapter to produce 3.3V from the Grove ports 5V VCC supply.
The ADS1100 is connected in another potential divider configuration but, this time it is closer to a Wheatstone bridge configuration than the normal two resistor configuration of a normal potential divider.
The ADC has been designed to provide analogue reading over the I2C port and is hard wired to use address 0x48 and can not be changed without replacing the A/D chip marked AD0.

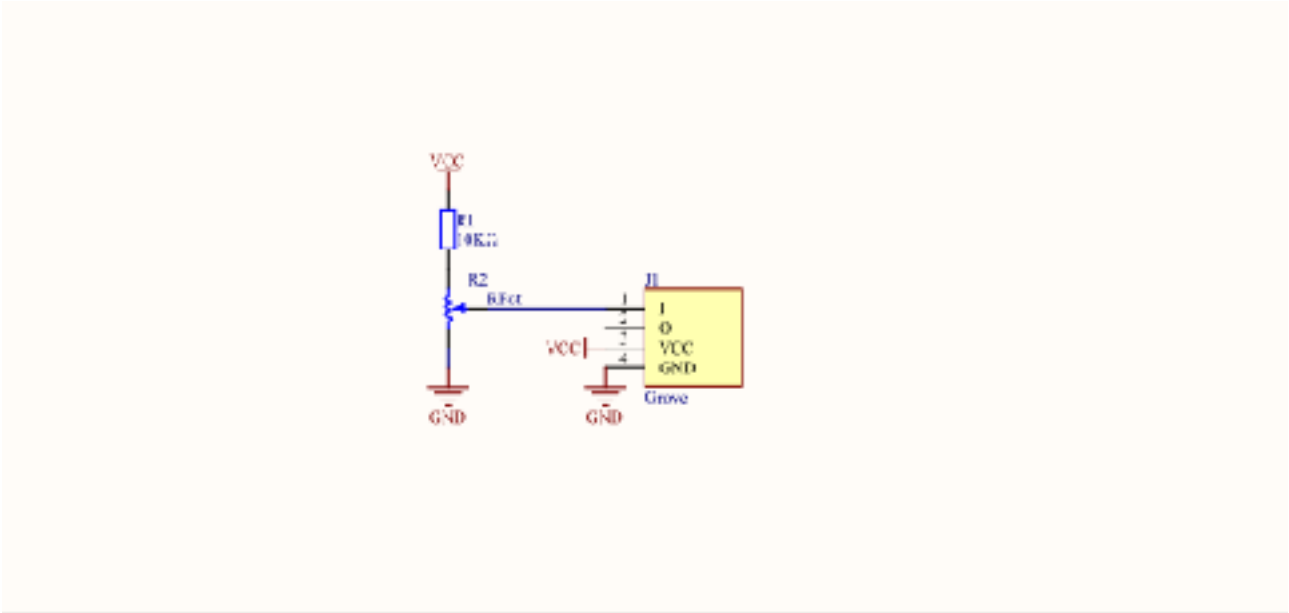
3.3.3

Angle Sensor





Connect to Port B (**Black Port**)
The angle sensor is a simple analogue sensor that uses a variable resistor to vary the voltage read by the I/O port on the M5Go Base on pin A0.
In the simple circuit diagram below we can see that the sensor only consist of a 10K Ohm resistor and a .10K ohm variable resistor in a potential or voltage divider circuit.



What is a Potential Divider?

A **Potential** or **voltage divider** is a simple circuit which turns a large voltage into a smaller one. Using just two series resistors and an input voltage, we can create an output voltage that is a fraction of the input. Voltage dividers are one of the most fundamental circuits in electronics and you will see it used in multiple units that require an analogue input voltage.

Reading the Angle sensor.

To read the angle sensor we connect it to port b of the go base and use the following UIFlow code.

If we want to use the sensor on the M5Stack Core Black, we have to connect it to the I/O base plate using some jumper leads or a grove four pin to Dupont lead.

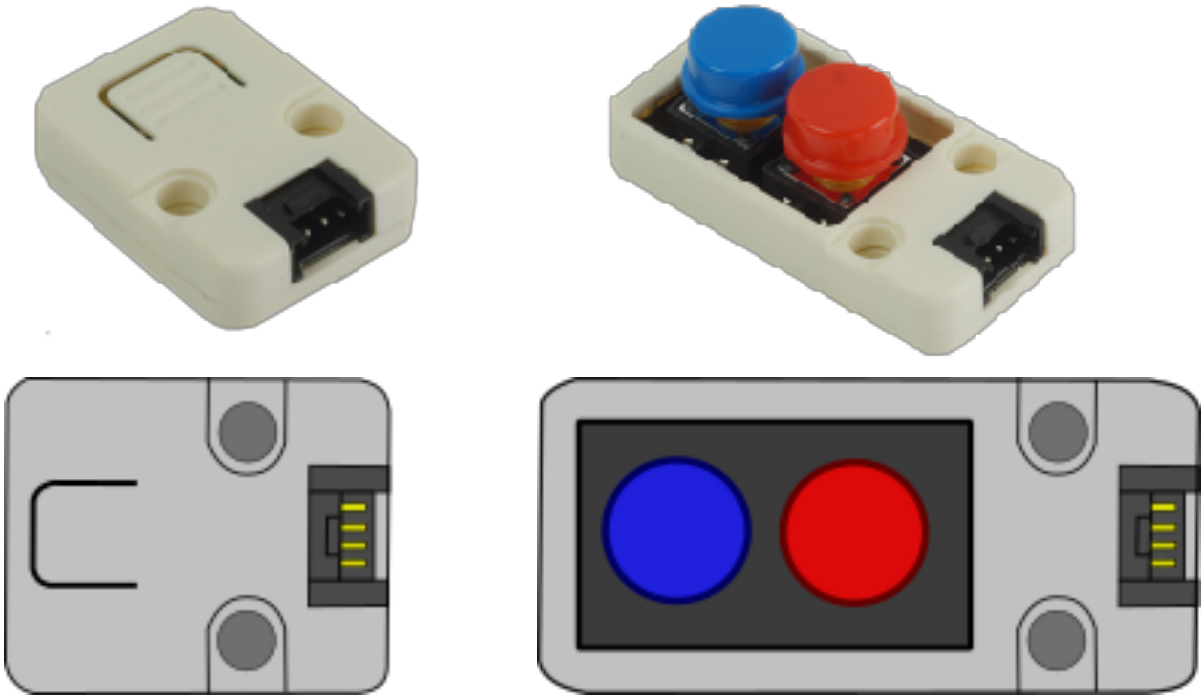
To use it on the I/O baseplate we need to connect the Red wire to +5V, Black to G or ground pin, The yellow wire plugs into pin 35 and the white wire will plug into pin 36 even though we wont be using pin 36.

The code for the I/O base is very similar to the first code example with the exception that instead of the "Get Angle" block we need to use the "Analog read pin" block and set it to pin 35.

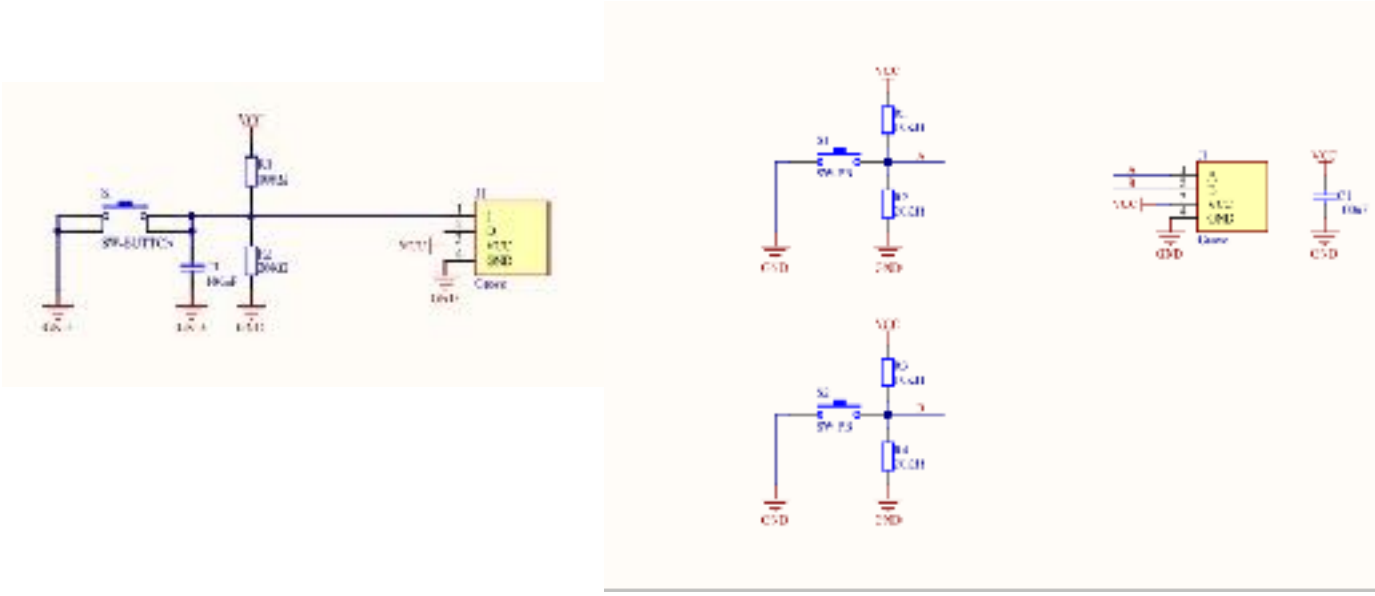
Further reading.
In chapter 7.1 I will show you how to build your own version of the Rotation Sensor.
In chapter 3.5.2 we will examine the I/O base plate.

3.3.4

Button and Single Button

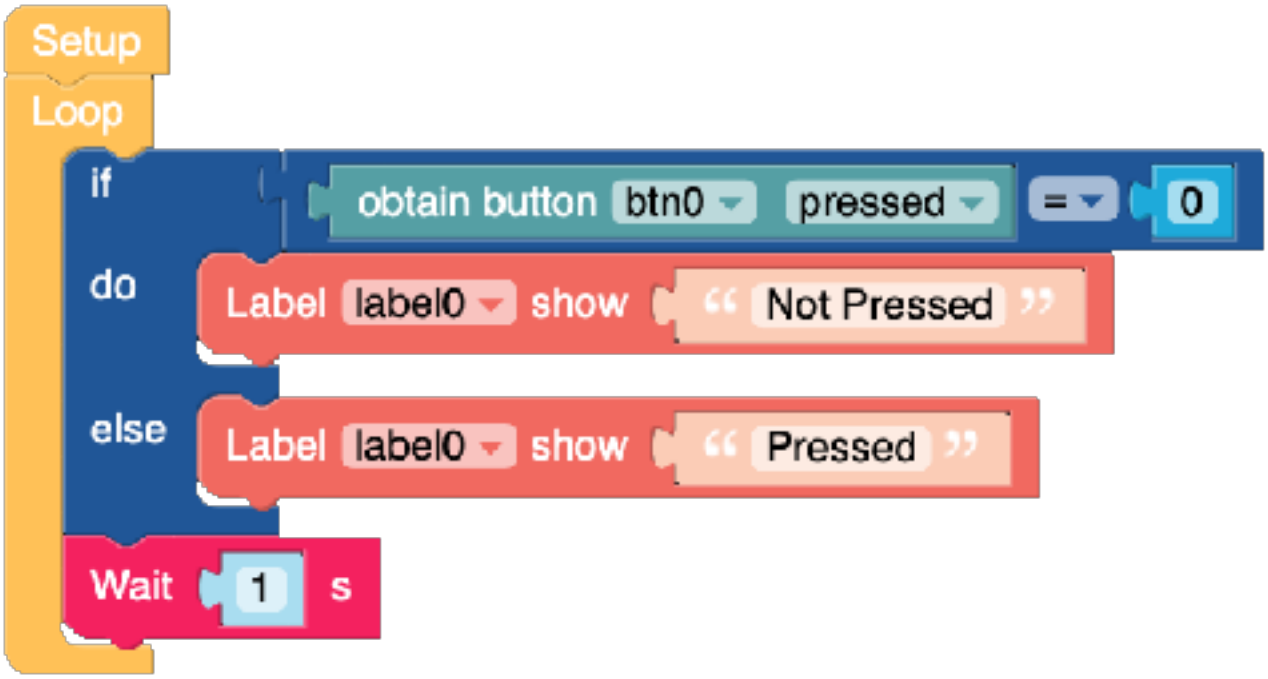


Connect to Port B **(Black Port)**
In the angle sensor I showed you that the sensor works using a potential divider. The Button and Duel button also make use of the voltage divider to provide signals to the M5Stack.



The above two schematic diagrams are of the single and duel buttons. on the left is the single button and on the right is the duel button. On both version A0 is used as the primary button however, on the duel button, A1 is used for the second signal. The capacitor between the signal

line and ground is used to Debounce the signal in an attempt to reduce repeated false signals triggered by the metal contacts inside the button bouncing around.



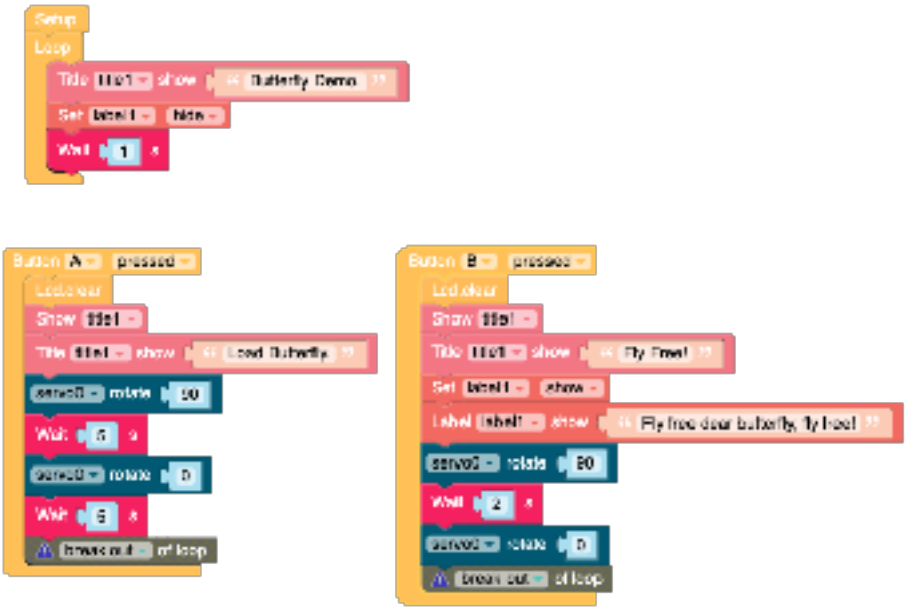
The above overly simple code uses the "obtain Button" value block to read the button connected to port B and change the text on the screen to "Pressed" or "Not Pressed".

There are two different blocks that the button units can use, One is a **Loop** block and the other is a **Value** block. (*For more information on the difference in the different types of blocks please read Chapter 4 - Blocky's Blocks.)*

3.3.5

The Butterfly.

Connect to Port A (**Red Port**).
The Butterfly module is an interesting unit that uses a servo to launch a rubber band powered butterfly across a room.
The Butterfly uses the Servo unit in UIFlow along with the servo blocks. To use the butterfly you wind up the rubber band, place, set the code to position the servo at 30 degrees to open the launcher, place the butterfly in and set the servo to 0 degrees to lock the butterfly in place.
To release the butterfly allowing it to fly across the room you set the servo to 30 degrees.



For more information on controlling servos in UIFlow you can read chapter 3.3.35.

3.3.6

Camera

There are currently three different versions of the camera available. The first version was uncased and based on the early ESP32 camera. The second came in a dark gray case and is built around the ESP32 Wroover module which has the additional 4MB of PSRAM. the third and current version comes in a white case and is fitted with a fisheye lens. As of writing there are no Blocky blocks or micro python code samples for using the cameras in projects however, they do contain their own micro server allowing wifi devices to connect to them and record video of photo's. This doesn't mean that they are not programmable, code examples and libraries exist for the Arduino environment that allow you to program them C.

3.3.7

CardKB

Connect to Port A (**Red Port**).
I2C address is 0x5F
The CardKB is a small credit card sized keyboard that uses an ATMEGA 328 to handle the key codes as well as the I2C communications with the M5Stack.
There is a single Neopixel next to the ATMEGA328 which is used to show the status of the function keys. In "Symbol" mode the Neopixel flashes green, in "Function" mode is flashes blue and in "Shift" mode it flashes red.

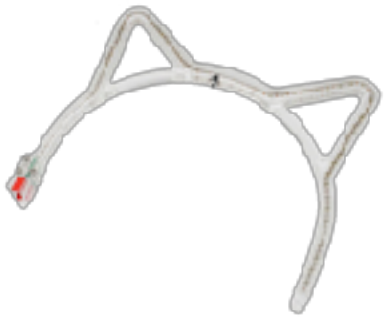
The Card KB has forty six individual buttons for the keyboard which when used in connection with the "Fn" and "Sym" keys is able to be used as a full keyboard. By default the standard out put is in lower case.

Pressing "Shift" (Up Arrow)+ a key will capitalise the letters,
Pressing "Sym" + a key will display the second character programmed to the key,
Pressing "Fn" + a key will print the third programmed symbol in the key.
There are also four direction buttons under the ATMEGA328 which act as the direction keys on a normal keyboard.

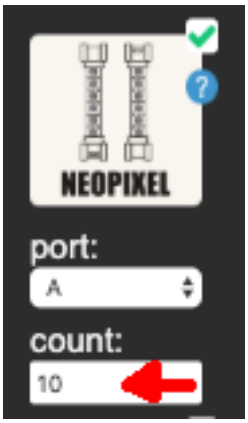
For a full listing of the symbols, please see the Appendix near the back of this book.

3.3.8

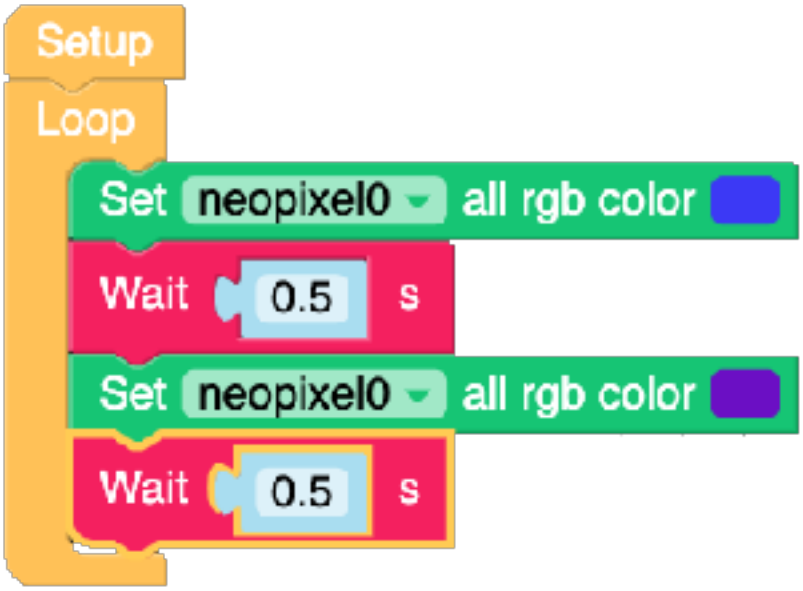
CatEar/Neko Headband.



Connect to Port A (**Red Port**).
The Neko/CatEar accessory is a Neopixal headband in the shape of cat ears that contain one hundred and eighteen individual Neopixal L.E.Ds. Because it uses Neopixels we can make use of the Neopixel blocks.
In order to use all one hundred and eighteen individual Neopixal L.E.Ds, you need to change the "Count" box when adding Neopixals to our project.



By default this is set to ten, click on it and enter the value 118 then press "OK" and we can now use them all.
Thecode sample on the following page controls all the Neopixels at the same time and is just a basic example, with a little experimentation and play with other units, we can achieve greater things.

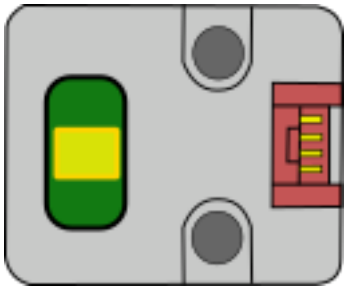


Please note: While I say to connected it to Port A (**Red Port**), Neopixels are not I2C devices and as such can be used on other ports.

For more information on Neopixels check out chapters 3.3.26 to 3.3.28.

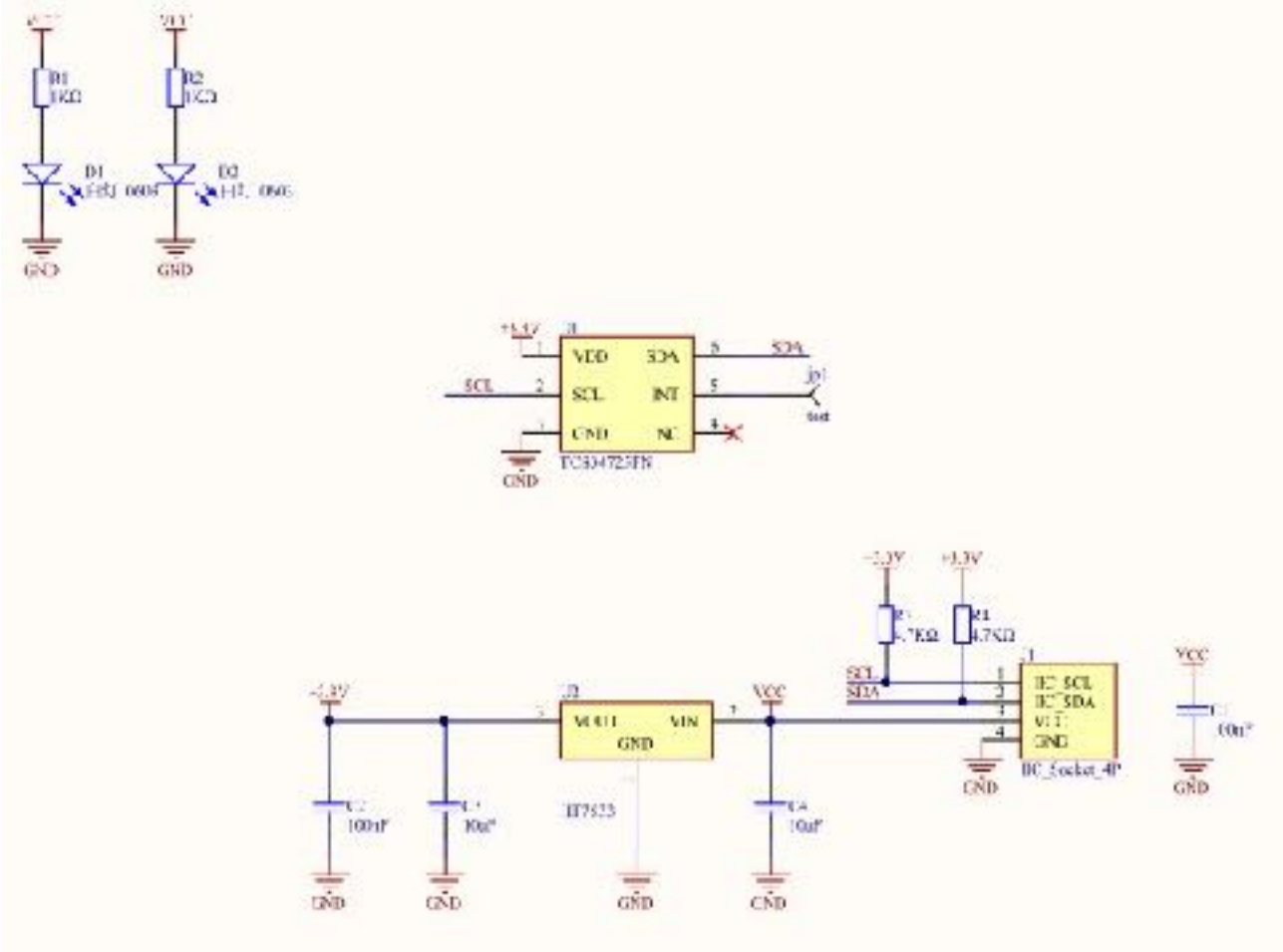
3.3.9

Colour Sensor

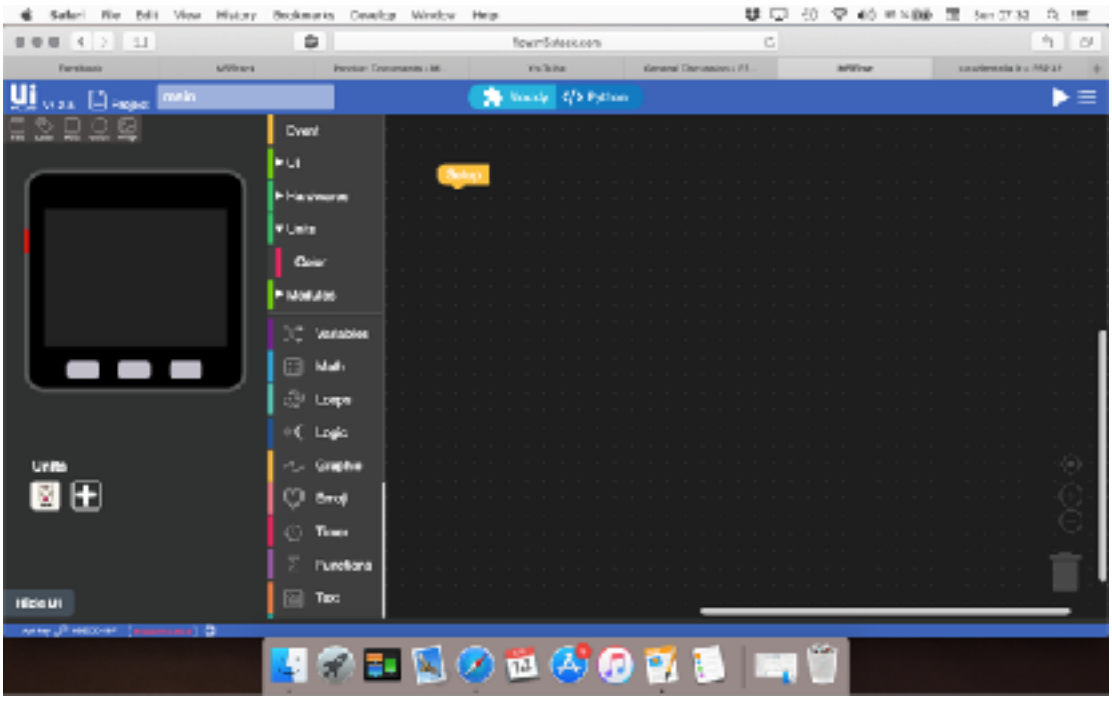


Connect to Port A (**Red Port**)
I2C Address is 0x29.
The colour sensor is built around the TCS3472 Colour light to digital converter made by AMS of Austria and features a built in I.R. filter. The unit contains two L.E.D's mounted either side of the sensor acting as the light source.

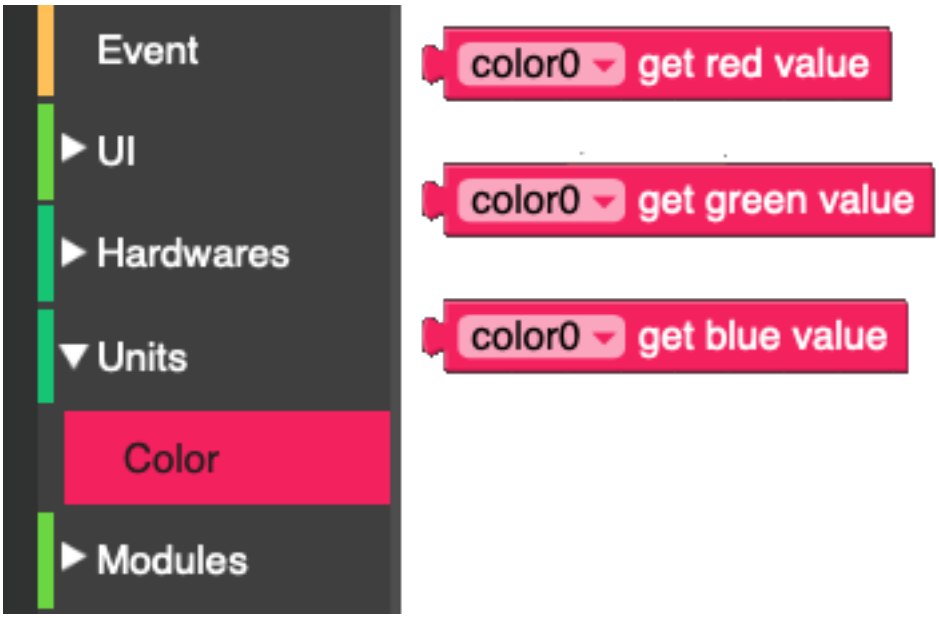
Below is the schematic diagram of the sensor showing the three main parts that make up the sensor and allow it to communicate with the M5Stack via a grove connector connected to the I2C port.



To use the colour unit in UIFlow, first we need to click the "+" sign under the virtual M5stack to bring up the units window, click on the colour unit picture to make a tick appear in the top right corner then click ok to add it to our project.



The code sensor only sends Red, Green and, Blue values. In order to access these we need to include the blocks in our code



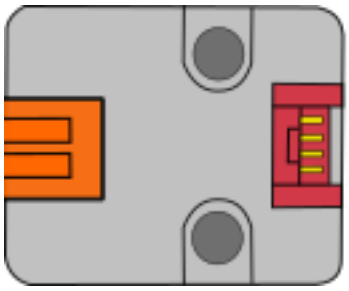
As you can see, the code blocks only return values. In order to make use of them we need additional code blocks.

In the first demo I have just added three labels that fetch the value and display the numeric values on the screen.

In this second demo, I added a rectangle to the screen, set the size to 320x240px and set the colour to the R,G,B values received from the sensor.

3.3.10

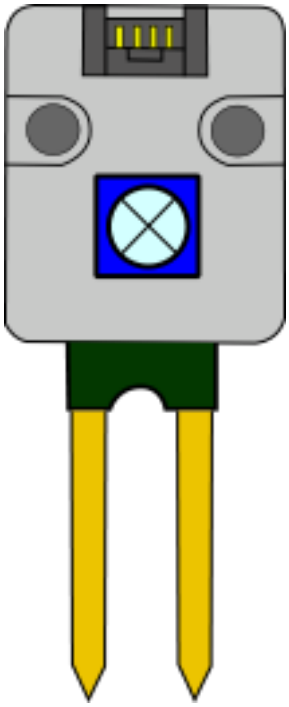
DAC (Digital to Analogue Converter)



Connect to Port A (**Red Port**)
I2C Address is 0x60

3.3.11

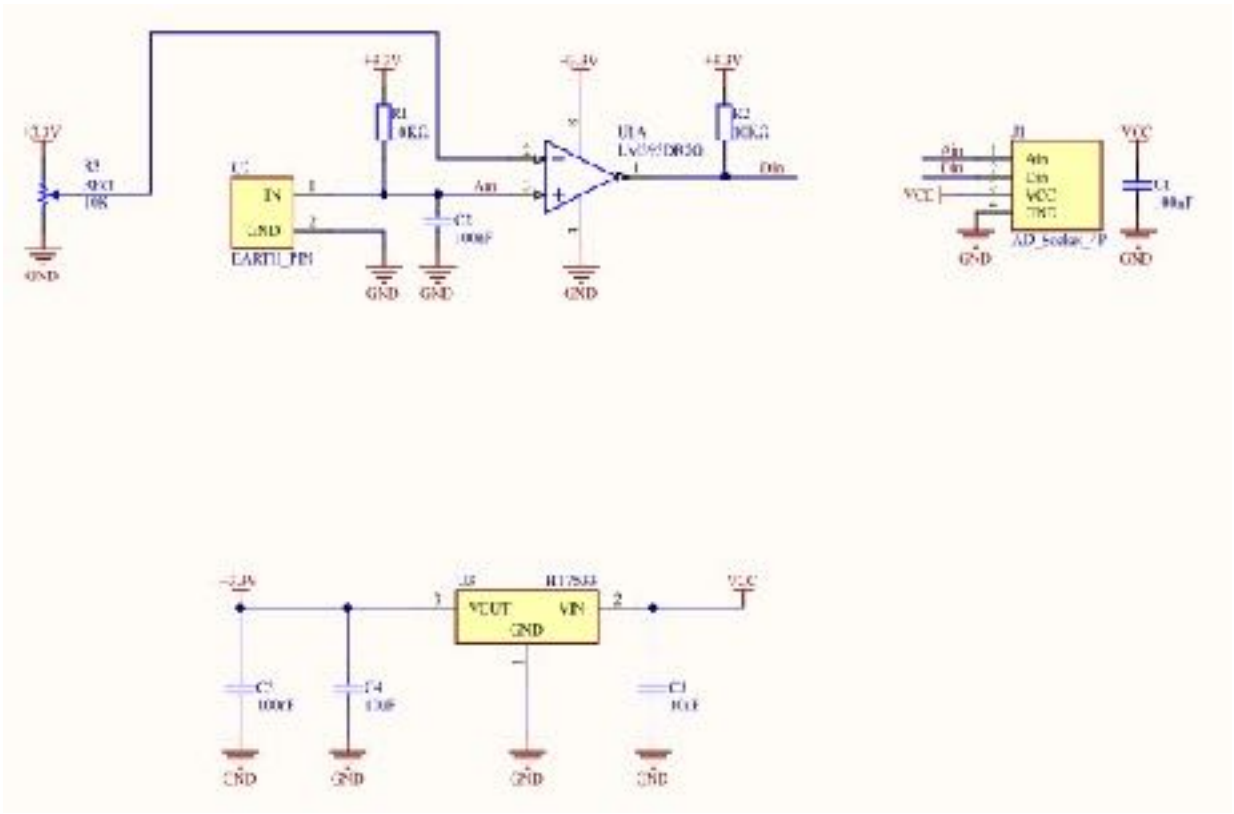
Earth Sensor (Moisture Sensor)



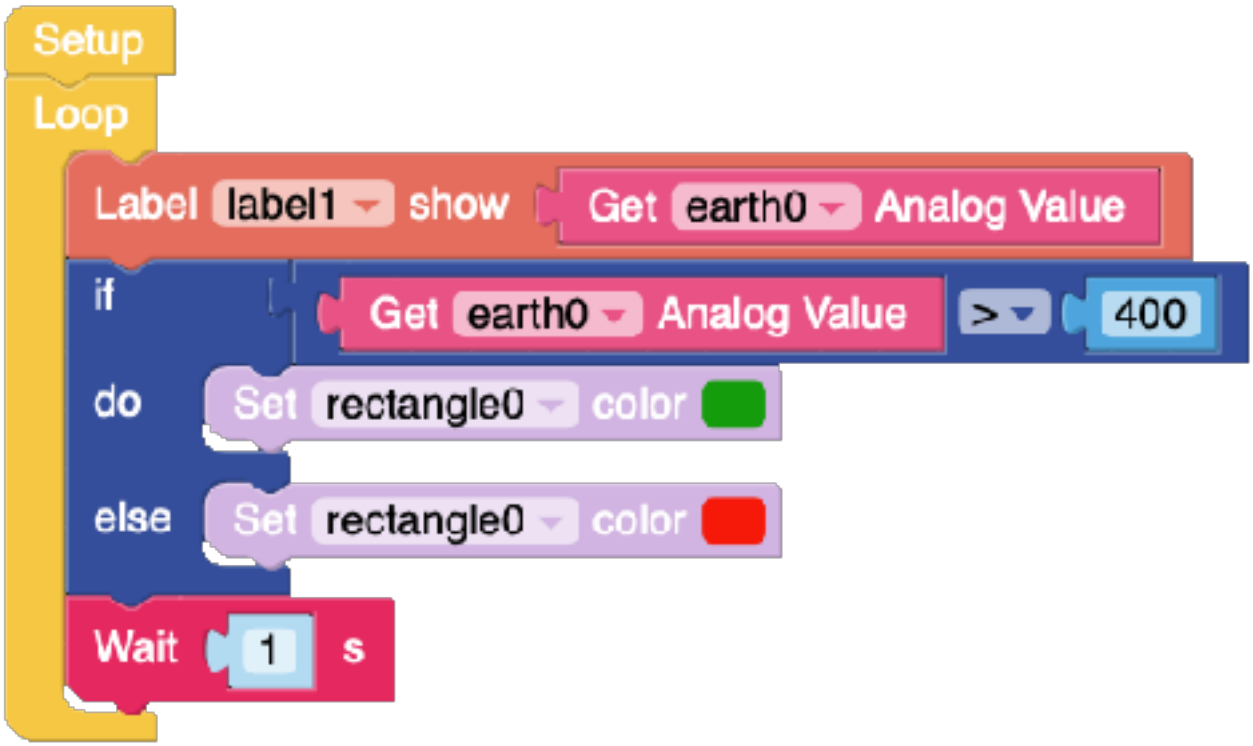
Connects to Port B (**Black Port**)
The Earth Sensor is a resistive moisture sensor that has an analogue mode and a digital mode. In analogue mode the sensor will return a value between 0 and 1024 on pin 36 (Yellow) of the grove connector but in digital mode the sensor will only return a value of 0 or 1 on pin 26 (White)of the grove connector..

In the schematic diagram on the following page we can see the familiar potential divider used in various other sensor units

.

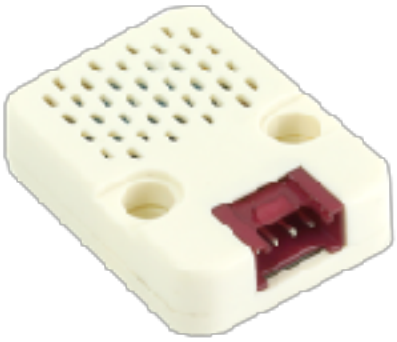


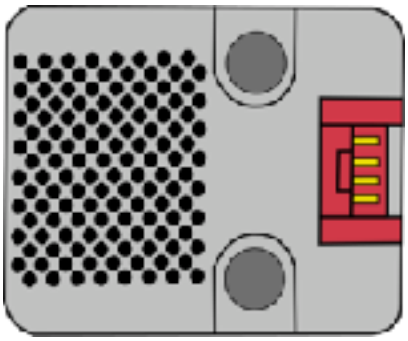
The following code demo reads the sensor in analog mode and changes the text of **Label1** to show that value and then the **"IF"** block checks the value of the sensor and if is greater than **" > " 400** (my soil was slightly damp here) it changes the box on the screen to green or red if its less then **400**.



3.3.12

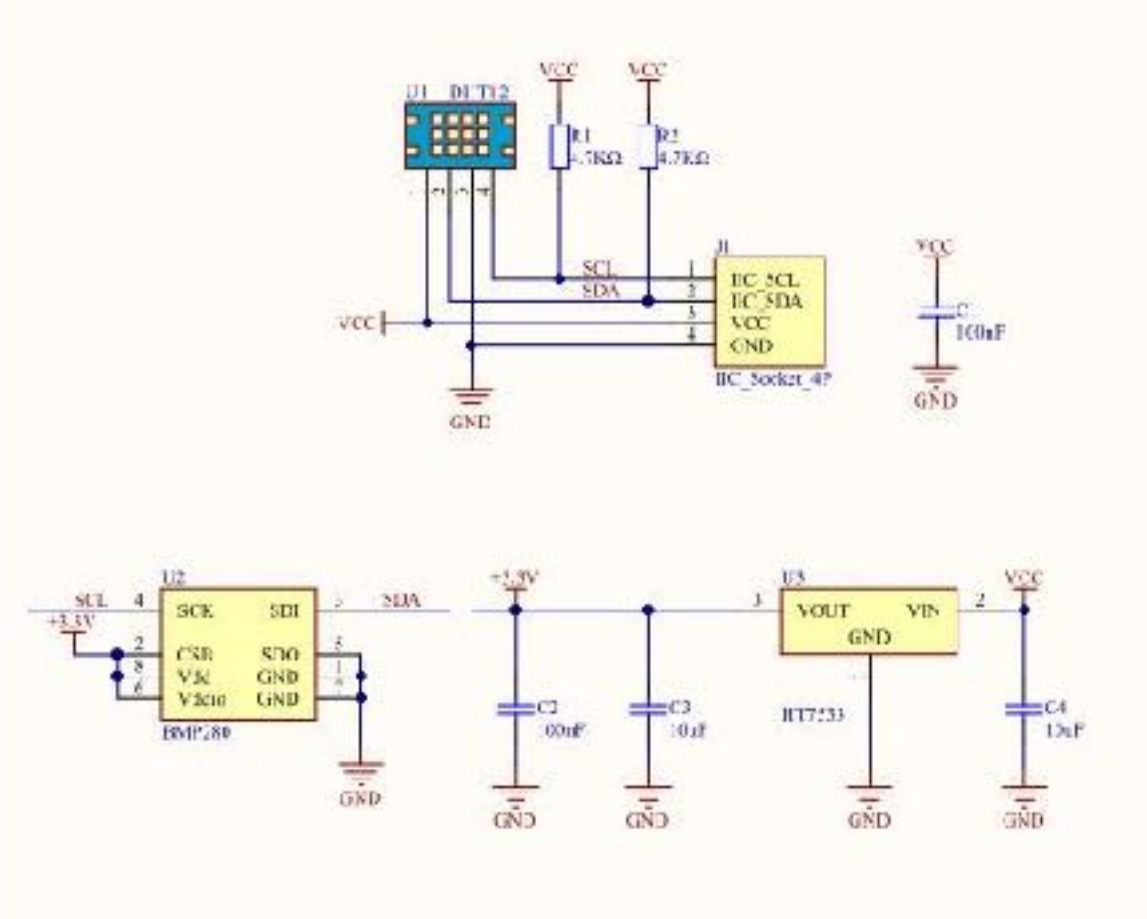
ENV Sensor (Environmental Sensor)





Connects to Port A (**Red Port**)
The ENV sensor has three sensors in to two devices squeezed into one unit. The first device, the **DHT12,** is used to measure temperature and humidity while the second device, the **BMP280** is used to measure air pressure.

The schematic diagram below shows how it is connected together..



Accessing the data from the ENV sensor is done by the use of three Variable blocks shown below.

To read and display the values, we can use a simple code identical to the colour code used previously.

The only difference between this and the colour code demo is that I have added a Title bar across the top of the screen and three headings to show which value is which.
When run on the M5Go with the ENV Sensor connected, you will see the following.


Enviromental Sensor
Pressure
1014.80
Temperature
19.13
Humidity
18.5

3.3.13

EXT I/O

Connects to Port A (**Red Port**)
The I2C address for this unit is 0x27.
The EXT I/O is an I2C device built around the **PCA9554PW** chip that adds an additional eight gpio's to the M5Stack or stick.

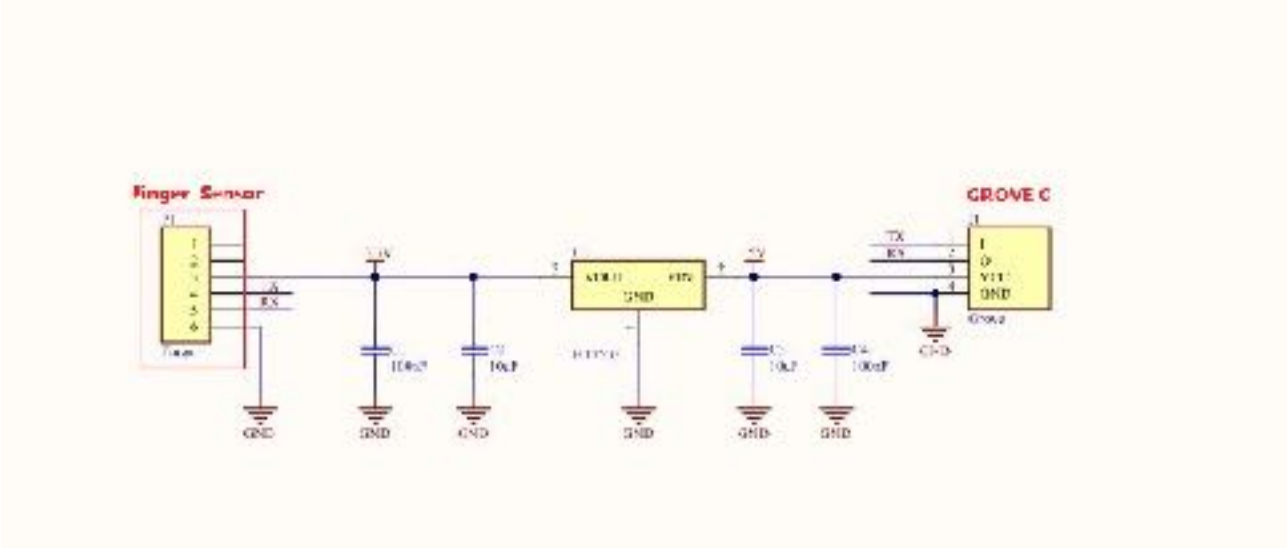
Using UIFlow we can set all eight of the pins to Input or Output at the same time or set each pin individually as an Input or Output.

For the data sheet on the PCA9554PW chip used in the module you can visit.
https://www.nxp.com/docs/en/data-sheet/PCA9554_9554A.pdf

3.3.14

Fingerprint Reader

Connects to Port C (**Blue Port**)
The fingerprint unit is built around the FPC1020A capacitive fingerprint reader and recognition chip. The FPC1020 is a 192 X 192 pixel 8 bit sensor with a 508 DPI resolution.
It communicates over UART at 19200 bps meaning that it can only be used with UART pins of the Core base or Port C of the Go base.
The schematic below for the Fingerprint unit shows quite a simple connection between the FPC1020A and the M5Stack.

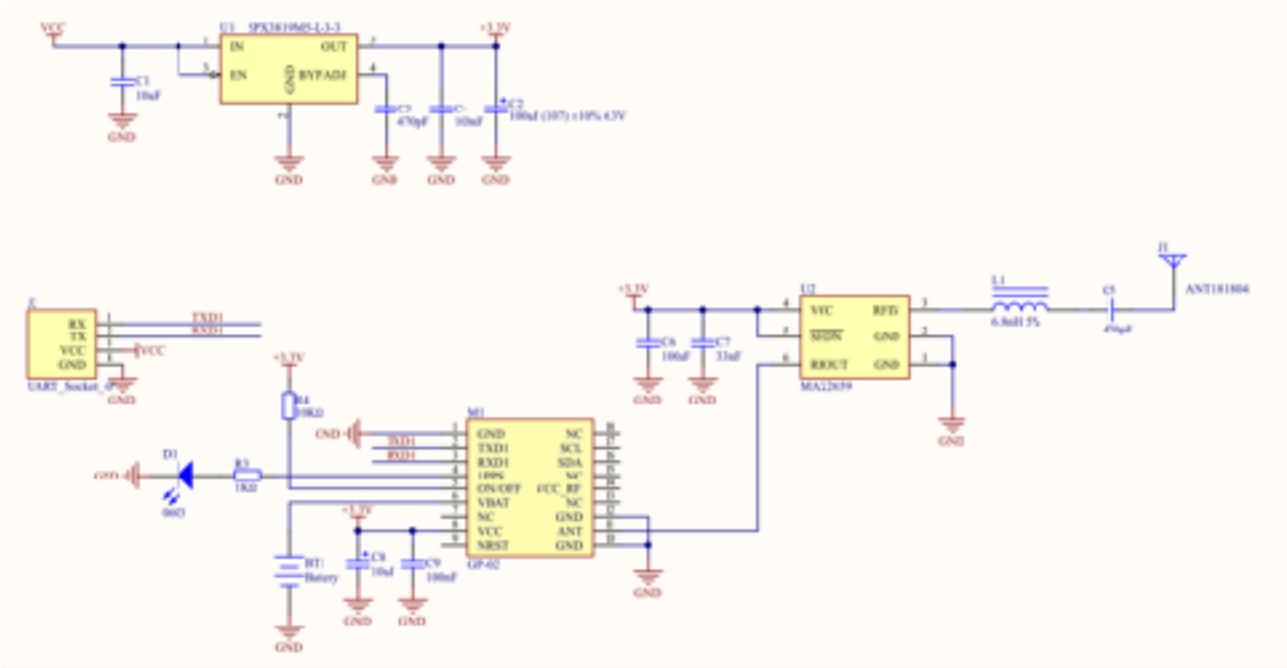


For the data sheet on the FPC1020A finger print reader used in the module you can visit.
http://www.shenzhen2u.com/doc/Module/Fingerprint/710-FPC1020_PB3_Product-Specification.pdf

3.3.15

GPS (Global Positioning Sensor),

Connects to Port C (**Blue Port**)
The GPS unit is built around the AT6558 receiver with a MAX2659 acting as a signal amplifier. Unlike most of the units, the GPS Units uses an SPX3819 LDO to supply the 3.3v at 500ma needed by the gps receiver.



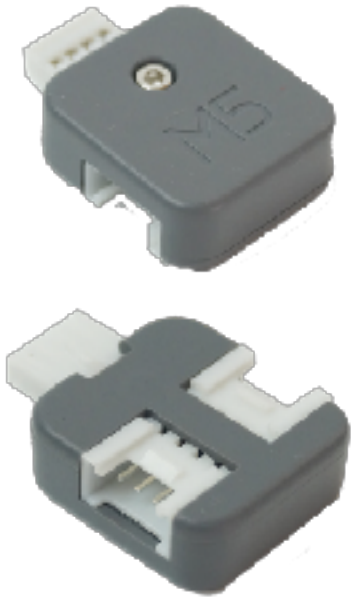
Currently the GPS unit is not compatible with UIFlow as no blocks exist for it yet.

For the data sheet on the AT6558 GPS receiver used in the module you can visit.
http://www.icofchina.com/d/file/xiazai/2016-12-05/b1be6f481cdf9d773b963ab30a2d11d8.pdf

For the data sheet on the MAX2659 amplifier used in the module you can visit.
http://www.icofchina.com/d/file/xiazai/2016-12-05/b1be6f481cdf9d773b963ab30a2d11d8.pdf

3.3.16

Grove T, Grove Hub and Grove Connector



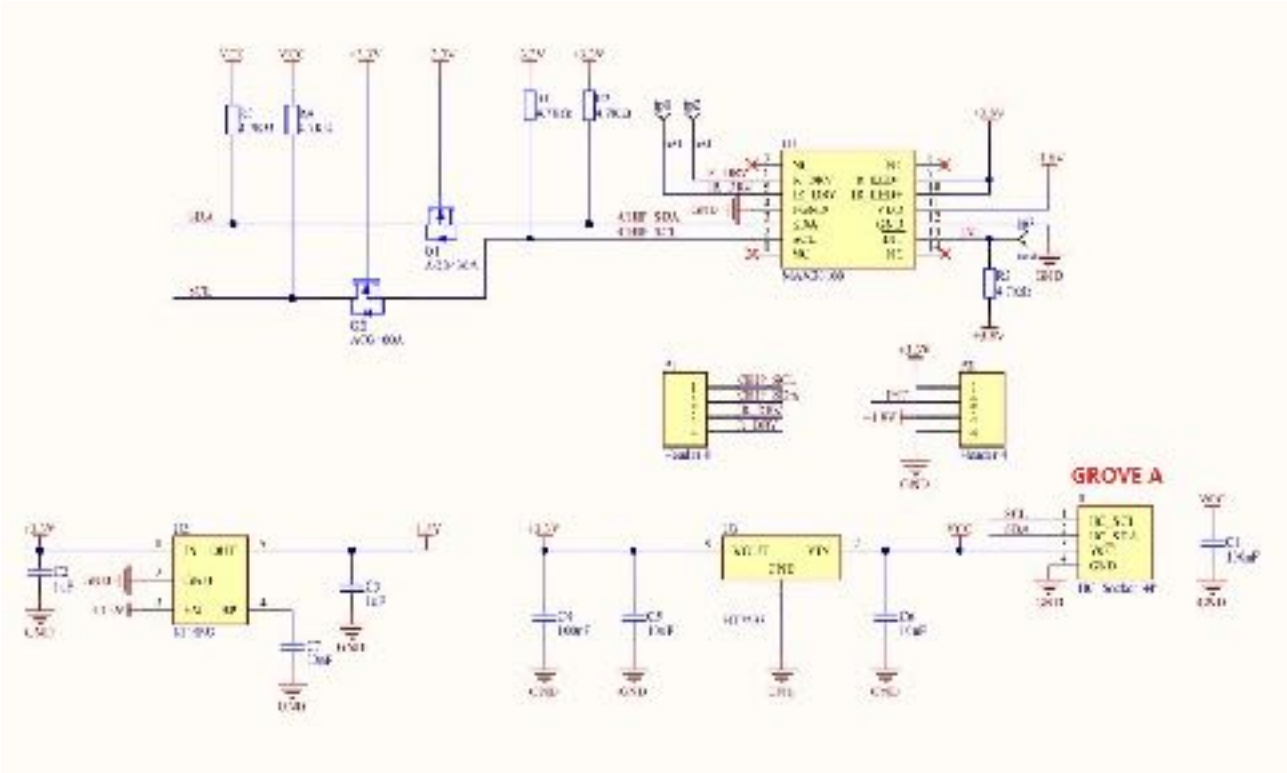
Unlike the I2C Expander, the Grove T, Grove Hub and Grove Connector are not restricted to certain ports and can be used with any of the ports.
All these three adapters do is connected all pins together so that more than one unit can be used on a port. The only exception is the connector which is required to join two Grove leads together.

3.3.17

Heart Sensor,

Connects to Port A (**Red Port**)
The I2C address for this unit is 0x57.
The Heart sensor is built around the MAX30100 Pulse Oximeter module by Maxim Integrated.

Below is the Schematic diagram of the Heart Unit.



Currently, the heart unit does not have and blocks to allow its use in UIFlow however, code is available to use this unit directly in Micropython.

3.3.18

I2C Expander

Connects to Port A (**Red Port**)
The I2C address for this unit is 0x70 to 0x77 (representing the six devices connected to it..
The I2C hub has been created around the TCA9548ALow-Voltage 8-Channel I2C Switch by Texas Instruments. This adapter allow you to connect six I2C Units (**Red Port**) to the I2C port on the M5Stack all at the same time, all with the same I2C address.

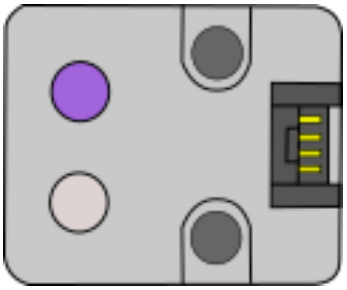
For the data sheet on the TCA9548A used in the module you can visit.
http://www.ti.com/lit/ds/symlink/tca9548a.pdf

3.3.19

I/O Expander,

3.3.20

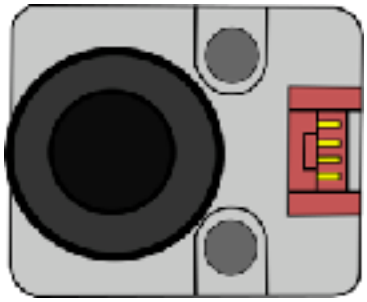
I.R. Sensor



3.3.21

Joystick

For the data sheet on the LM393DR2G comparator used in this unit, you can visit.
https://www.onsemi.com/pub/Collateral/LM393-D.PDF



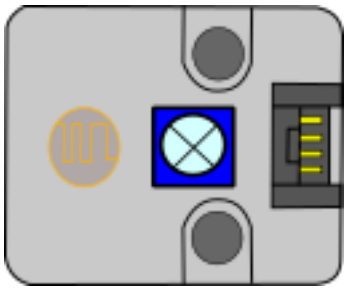
I2C Address = 0x52
Connect to Port A (**Red Port**).

The joystick unit has been designed to provide a simple joystick for projects. It uses 10K Potentiometer for the X and Y direction signal generation as has push button that can be activated by using down on the centre of the stick.
An ATMEGA328P is at the heart of the device and handles the job of reading the potentiometers and communicating with the M5Stack over the **I2C port (Red Port**)

The following code reads the joystick and moves a circle around the screen.
To create the circle we drag a circe object from above the virtual M5Stack on the stacks screen and press run.

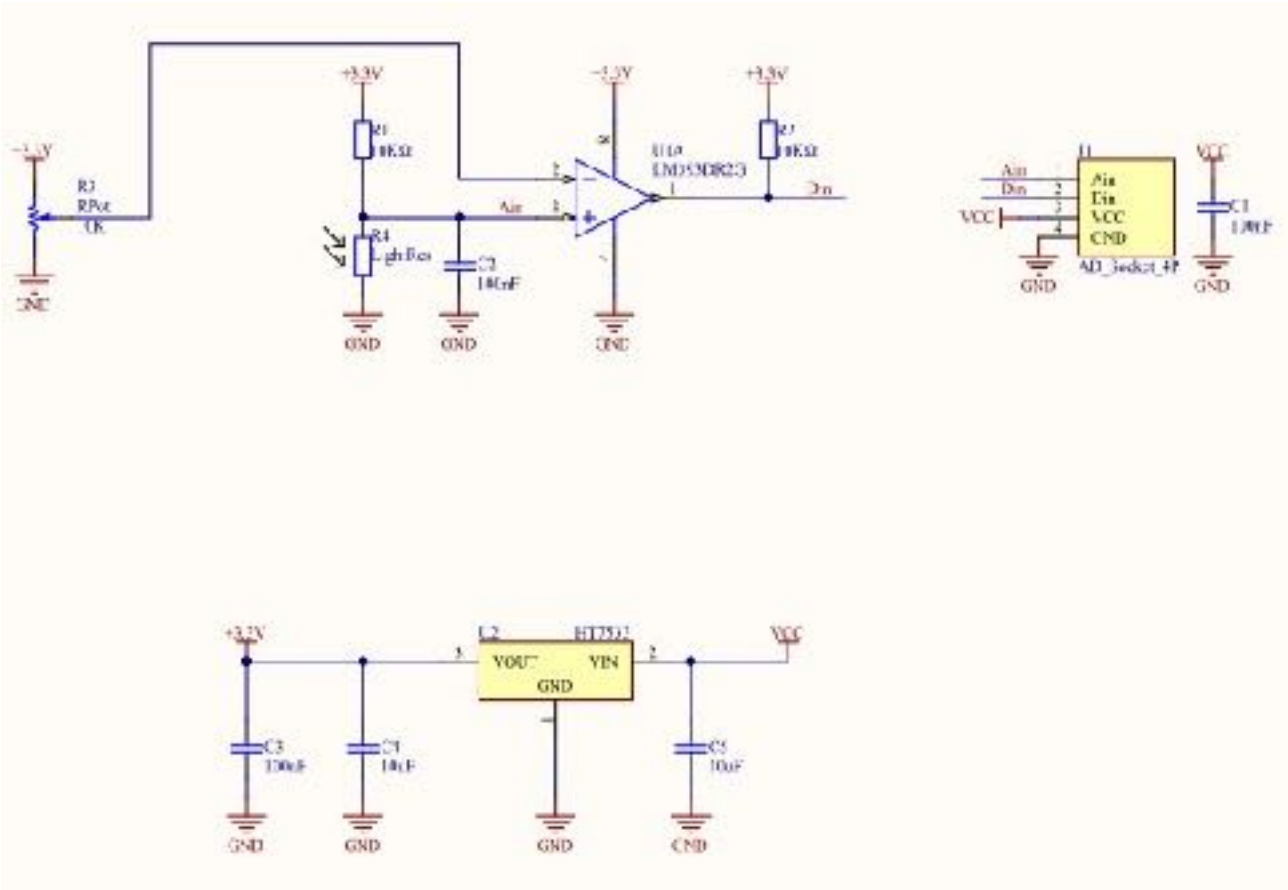
3.3.23

Light Sensor



Connect to Port B (**Black Port**)
The light sensor uses an L.D.R (Light Dependent Resistor) to read light and send an analogue value to the M5Stack.
The potentiometer on the front is used with the internal comparator to produce a high (1) or low (0) value on the digital pin when light reaches a certain value.

Below is the circuit diagram of the light sensor.



On one side of the comparator is a voltage divider consisting of the LDR and 10K resistor R1 which also connects to Ain off the connector for analogue values. on the other side of the divider another potential divider is made from the potentiometer.

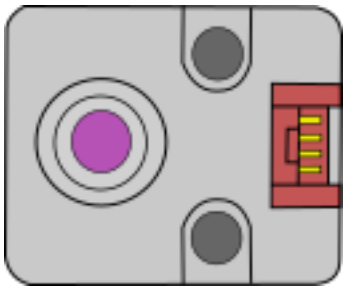
3.3.24

Makey Makey Unit

I2C Address = 0x51
Connect to Port A (**Red Port**).

3.3.25

N.C.I.R (Non Contact Infrared)



I2C Address = 0x5A
Connect to Port A (**Red Port**).

The NCIR unit is a non-contact infrared temperature sensor built around the MLX90614 sensor device and can be used for both contact and air temperature measurement.

3.3.26

NeoFlash.

◉

3.3.27

NeoHex.

3.3.28

Neopixel Strip.

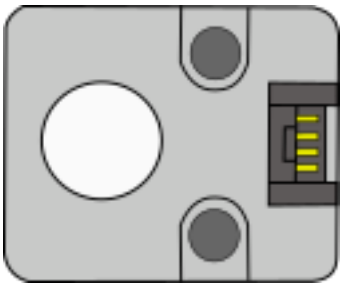
For the data sheet on the AS312 module you can visit.
https://forum.mysensors.org/assets/uploads/files/1494013712469-pir-as312.pdf

3.3.29

Neopixel Strip.

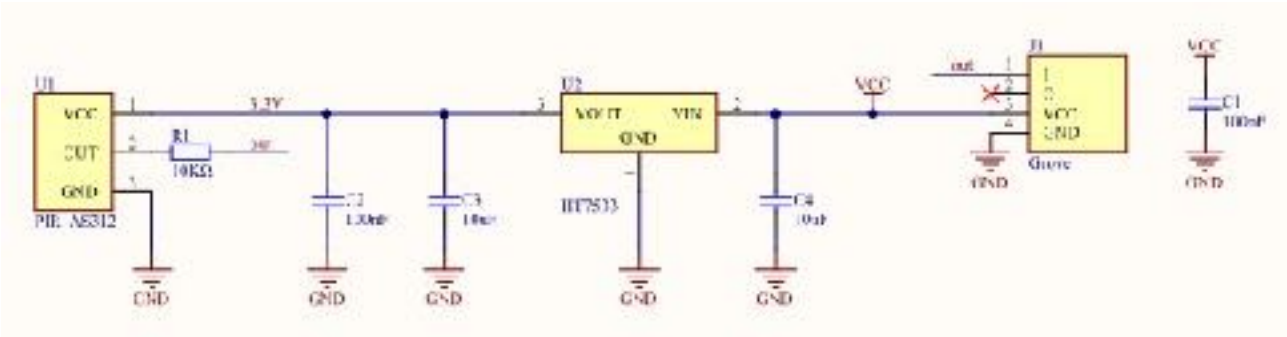
3.3.30

P.I.R. (Passive InferRed Sensor).



Connect to Port B (**Black Port)**

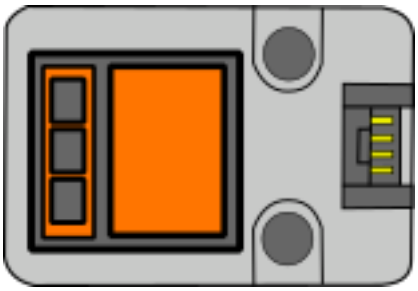
Similar to devices found in security devices, the P.I.R sensor is use to detect the presence of inferred radiation in an area. Often used to detect the presence of IR radiation given off by humans, it is only one of the many applications of inferred detection devices.
The Schematic diagram below shows how it is connected to the M5Stack.
On the left is the AS312 Pyroelectric Infrared Radial sensor which communicates with the M5Stack by setting pin 1 (A0) high when it detects inferred radiation.



In chapter 7.3 you can see an example of how I use it to trigged the Vex Robotics Ambush Striker.

3.3.31

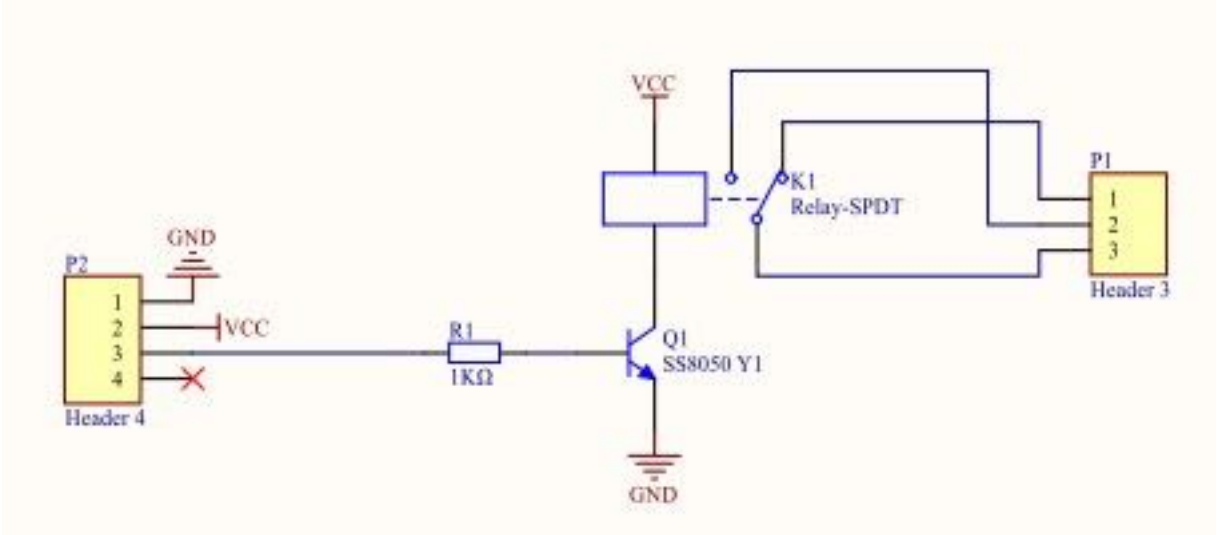
Relay Module.



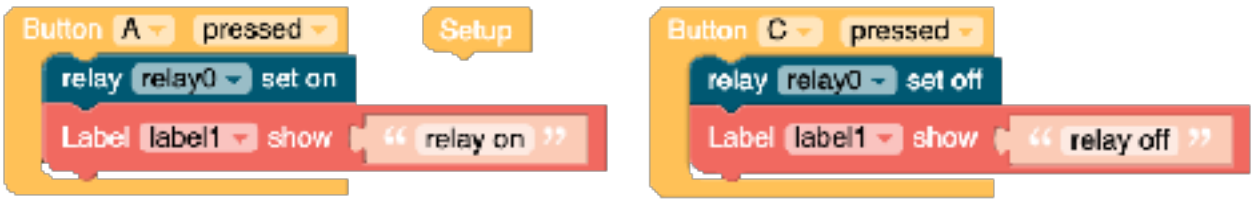
Connect to Port B (**Black Port**)

The relay module contain a Single Pole double throw relays that has a Normally Open **(NO)**, Normally Closed **(NC)**, and Common **(C)** contact connected to a 3 pin VH3.96 screw terminal.

When unpowered, the Common and Normally Closed contacts are connected together but when powered Normally Open and Common are connected together.
The schematic diagram for the relay module is below.



To operate the relay we can use the code blocks. The Code uses button A to switch the relay on and button B to switch the relay off.



3.3.32

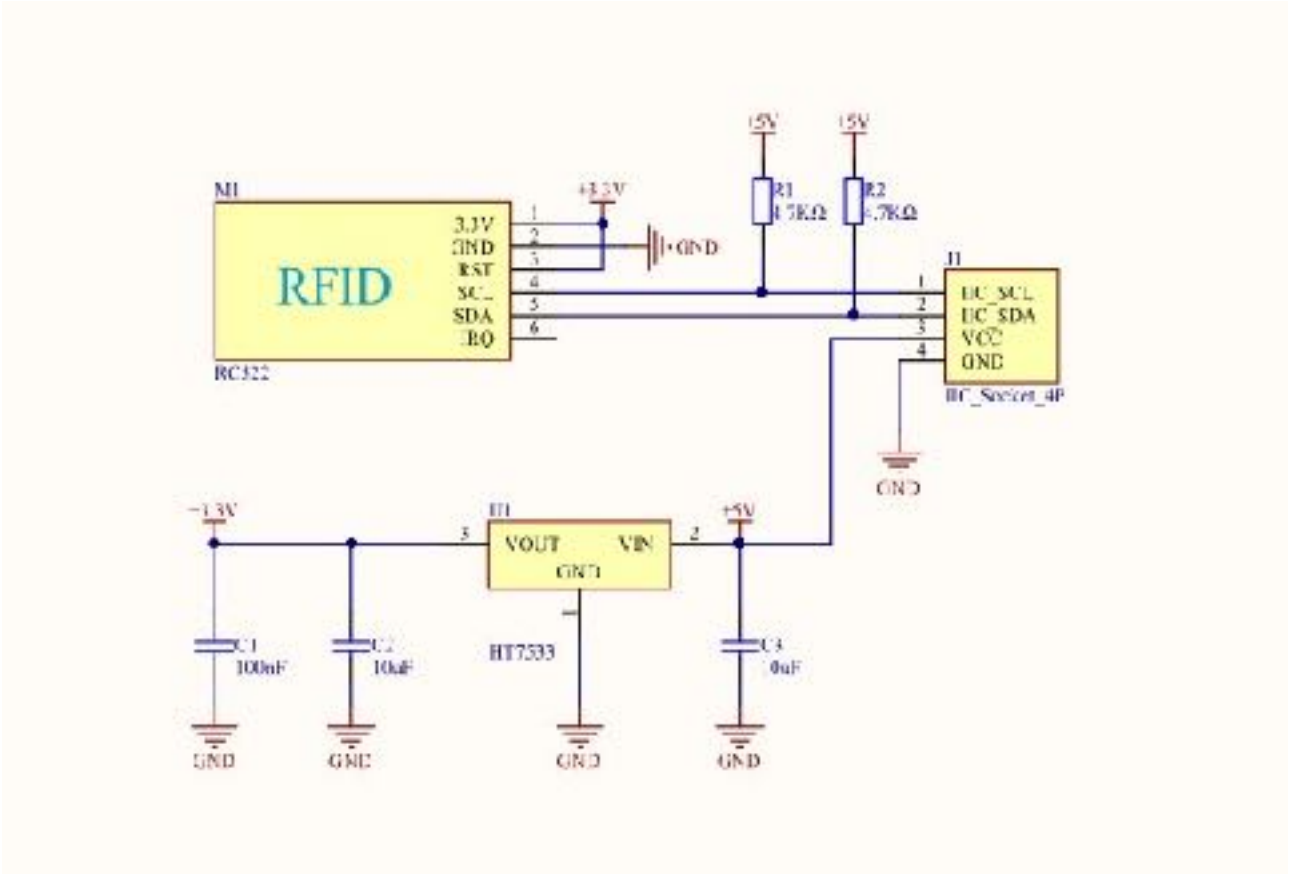
RFID Reader.

I2C Address = 0x28
Connect to Port A (**Red Port**).

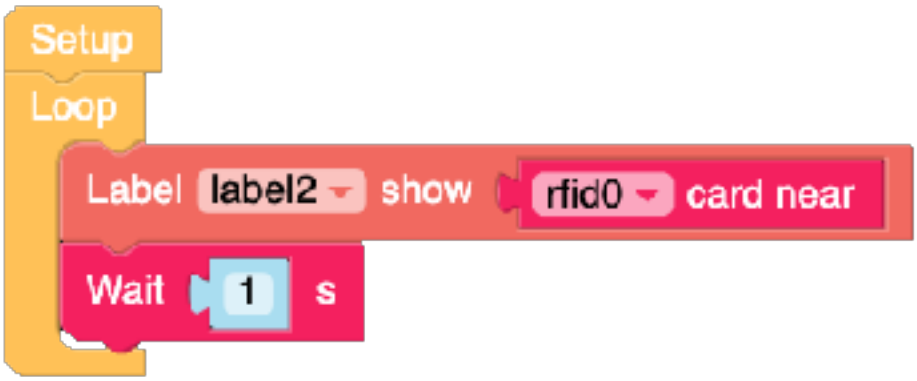
The R.F.I.D unit has been designed around the **MFRC522** chip and reads **13.56Mhz** tags that are compatible with **ISO14443A**, **MIFARE** and **NTAG** protocols.

Please Note that the R.F.I.D unit does not come with any sample tags.

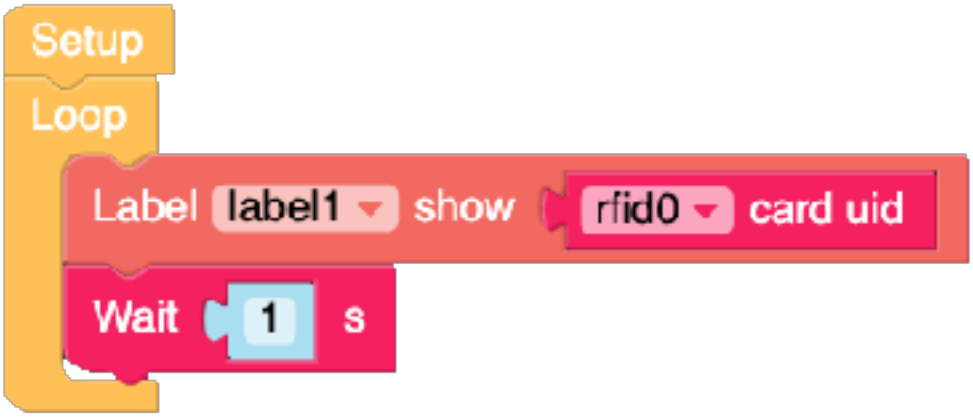
The following schematic diagrams shows how the MRFC522 connects to the M5Stack with just two pulp resistors on the SDA and SCL lines and a HT7533 power adapter to reduce to 5V down to the 3.3v that it needs.



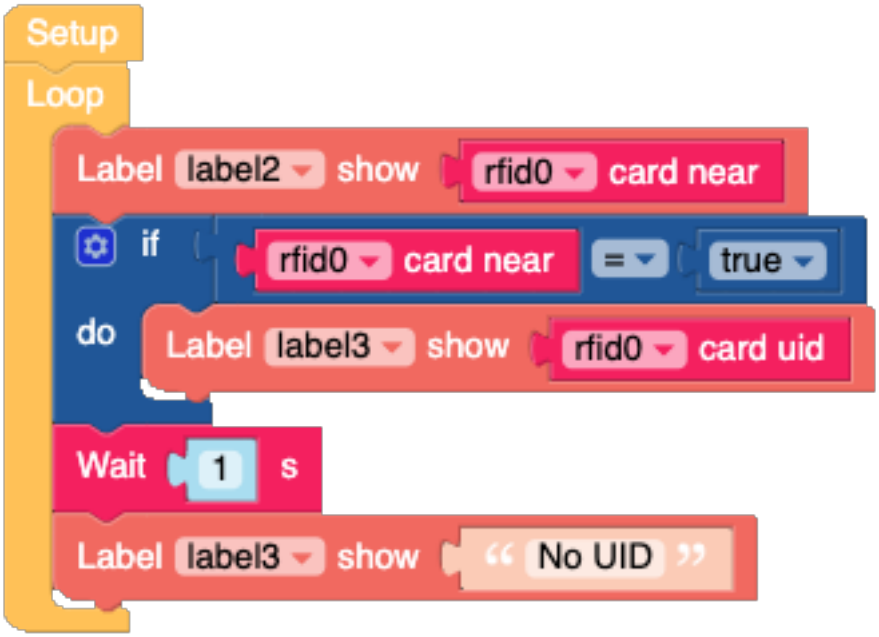
To detect the presence of an R.F.I.D tag we can use the following code.



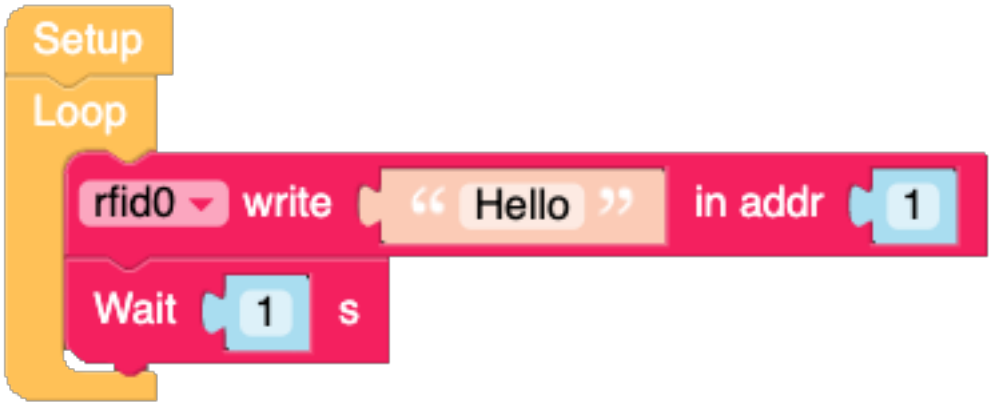
All this code does is check to see if an R.F.I.D tag is in range and print True on the screen, if it can't detect a tag then it prints false.
To read a tags data we can use the following code in UIFlow.



All the code does is to read the RFID's tag and change the Label to show the tags UID or Unique Identification code.
In the following example I have added a if statement that reads the tags data and displays it on screen if it detects the presence of a tag.



The MFRC522 can also read and write information to and from tag that support information storage like security cards.
To write information you can use code like the following.

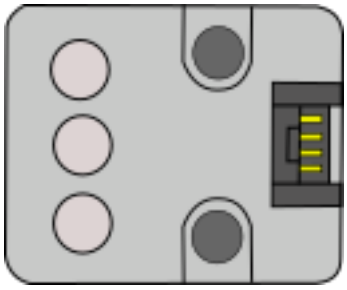


The code just blindly writes hello wether there is a card present or not.

And this code reads the information stored on the card and displays it on the screen.

3.3.33

RGB Module.



Connect to Port B (**Black Port**)

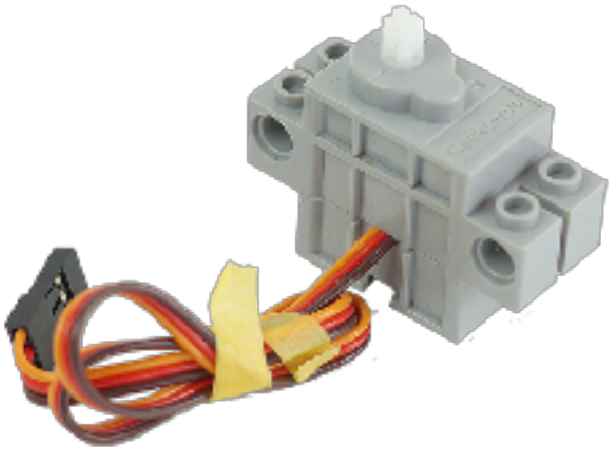
The RGB unit contains three Neopixel L.E.Ds and can be chained together because it has both and input grove connector and an output grove connector.
because it is set up to use Port B, it has its own set of control blocks.

3.3.34

RS485 adapter.

3.3.35

Servo.



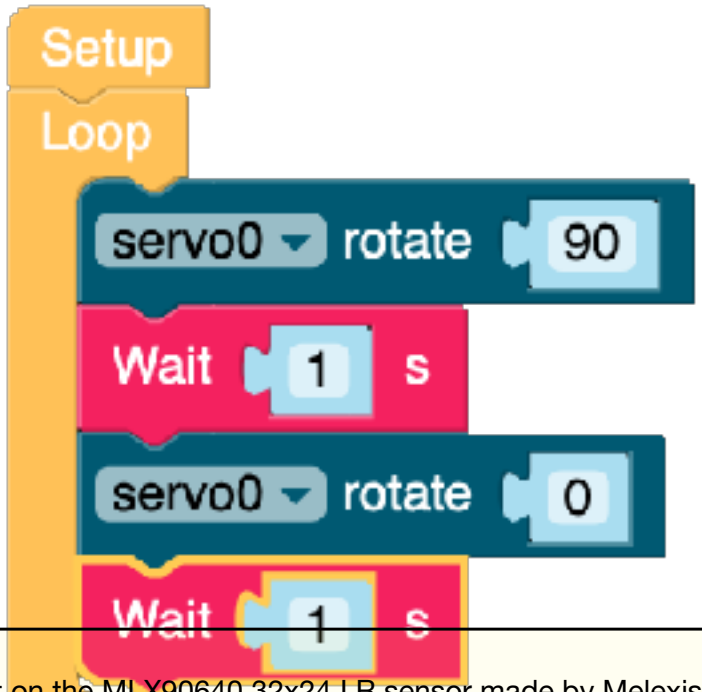
Connect to Port A (**Red Port**).
The Servo is different to other units intuit it doesn't have a dedicated M5Stack module. The Servo unit in the Units menu is a place holder for the blocks that allow us to control generic three pin servos. The image above is of a *LEGO® Technic™* compatible servo produce for the Kittenbot™ by Shenzhen Kitten Tech Co,Ltd

This setup is compatible with all servos that have the following connection pattern - Signal, VCC, GND.
Signal and GND are always on the outside so that if a non-polarised servo connection is connected the wrong way around, it wont damage the servo or the servo controller.
To control a servo we need to use a Grove cable with Dupont connections on the other end to connect the servo to the M5Stack.

Seeedstudio Grove to Male Dupont lead.

Connect the Grove lead male pins as shown below (note that the white wire is not needed and left loose).

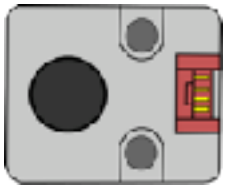
On these servos you can see that they have used brown instead of black for the GND wire and orange for the Signal wire.
Once connected we open UIFlow and add a Servo unit set to port A.
The following code set the servo to 90 degrees and then returns it back to 0 degrees.



For the data sheet on the MLX90640 32x24 I.R sensor made by Melexis you can visit.
https://www.melexis.com/-/media/files/documents/datasheets/mlx90640-datasheet-melexis.pdf

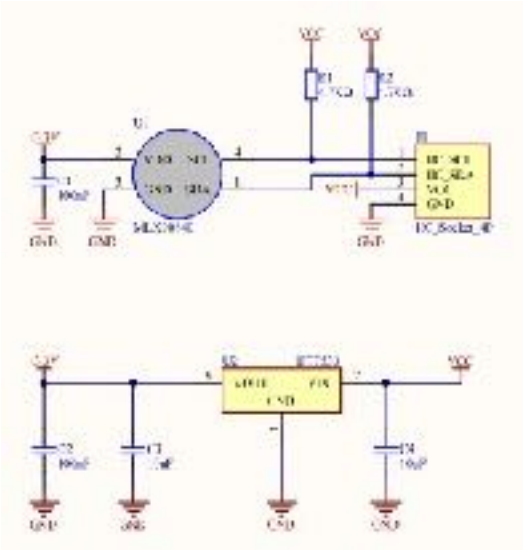
3.3.36

Thermal Sensor.



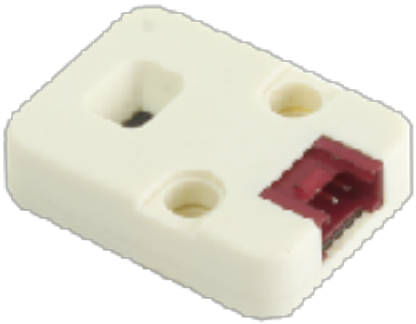
Connect to Port A (**Red Port**).

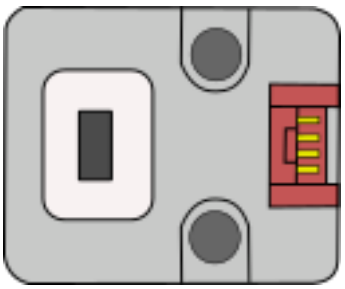
The thermal sensor has be built around the MLX90640 32x24 I.R sensor made by Melexis.
The following schematic diagram shows how it connects to the M5Stack.



3.3.37

TOF (Time of Flight).



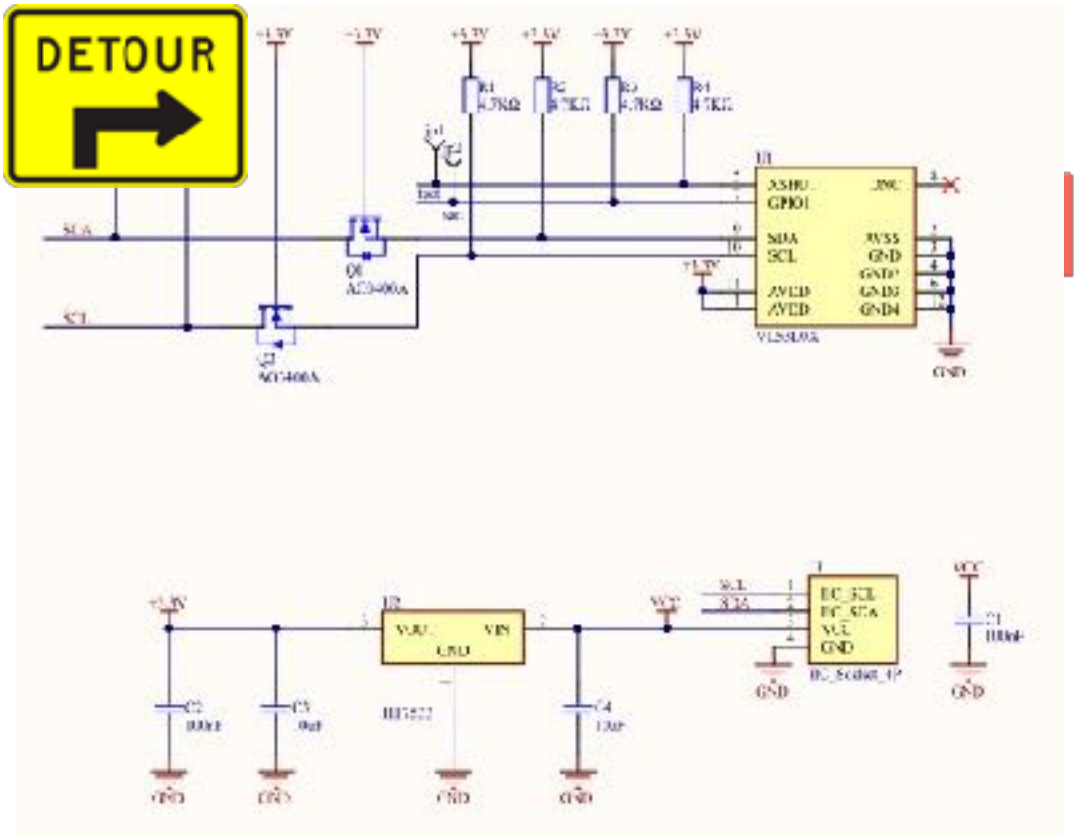


For the data sheet on the VL53L0X device you can visit.
https://pdf1.alldatasheet.com/datasheet-pdf/view/948120/STMICROELECTRONICS/VL53L0X.html

Connect to Port A (**Red Port**).
The I2C Address = 0x29

The T.O.F sensor or time of flight sensor is a high precision sensor that uses 950nm lasers to measure distance. It uses the VL53L0X sensor to communicate over I2C on port A
The default I2C Address is set 0x29. According to the datasheet, the device can be reprogrammed to take different addresses in the range of 0x08 and 0x77 however, the device defaults back to 0x29 after a power cycle to the device. UIFlow is supposed to be able to handle more than one sensor at a time but, does not provide the ability to reprogram the sensors on the fly.
Below is the schematic diagram of the TOF Unit.
To read the information from the sensor we only need one block. In the following code example we read the data from the VL53L0X device and change the label to reflect the data from the sensor.

Please note: the range of the readings is from 34 up to 8190 units however, the absolute distance is only two meters.



3.3.38

Trace Unit.

Connect to Port A (**Red Port**).
The I2C Address = 0x5A

The Trace unit is an addon for the M5Bala robot or the Lidar robot.
The module is built around the ATMEGA328 and has four I.R. LED's that give the robots the ability to follow lines.

3.3.39

Weight Sensor.

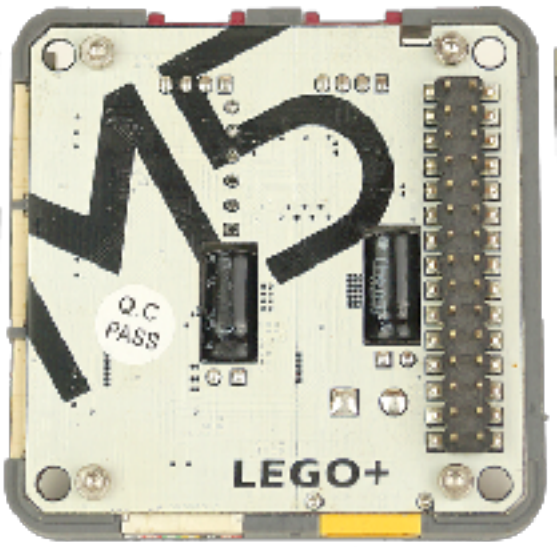
3.4

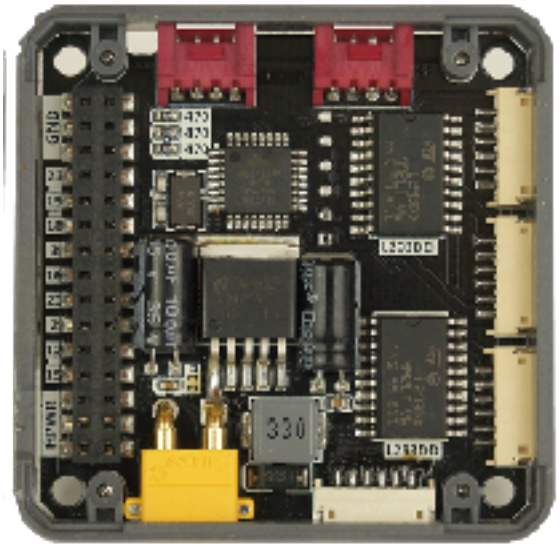
Stackable Modules

The designers behind M5Stack have taken the time to create several stackable modules that connect together using the internal bus connections.
These stackable module are as follows.

3.4.1

Lego+

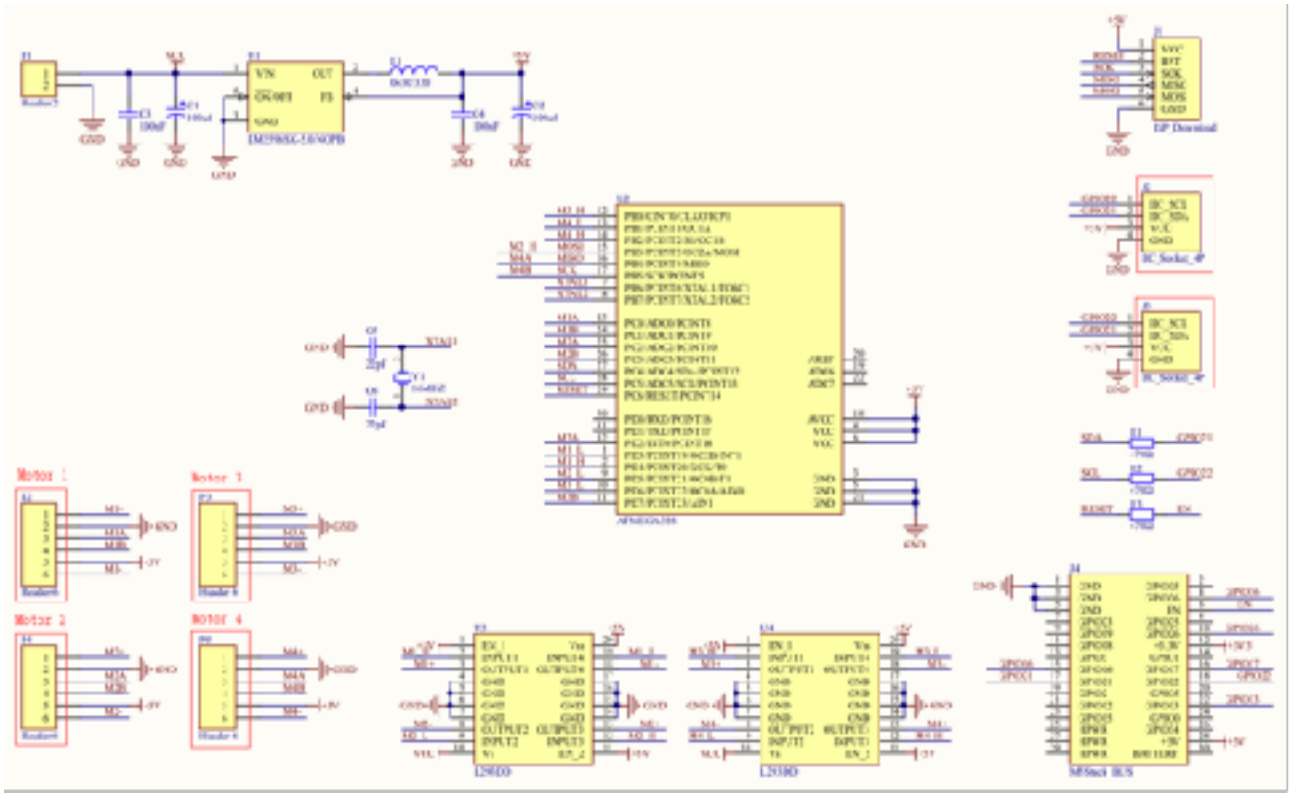




Connect to Port A (**Red Port Internally**).
The I2C Address = 0x56
The Lego+ module connects to the M5Stack by connecting between the core and the baseplate and communicates using the internal bus connector. Around the edge of the module we have four ports for connecting LEGO® Technic™ NXT motors, two I2C ports for connecting I2C devices with different addresses and an XT30 connection for providing power to the module.

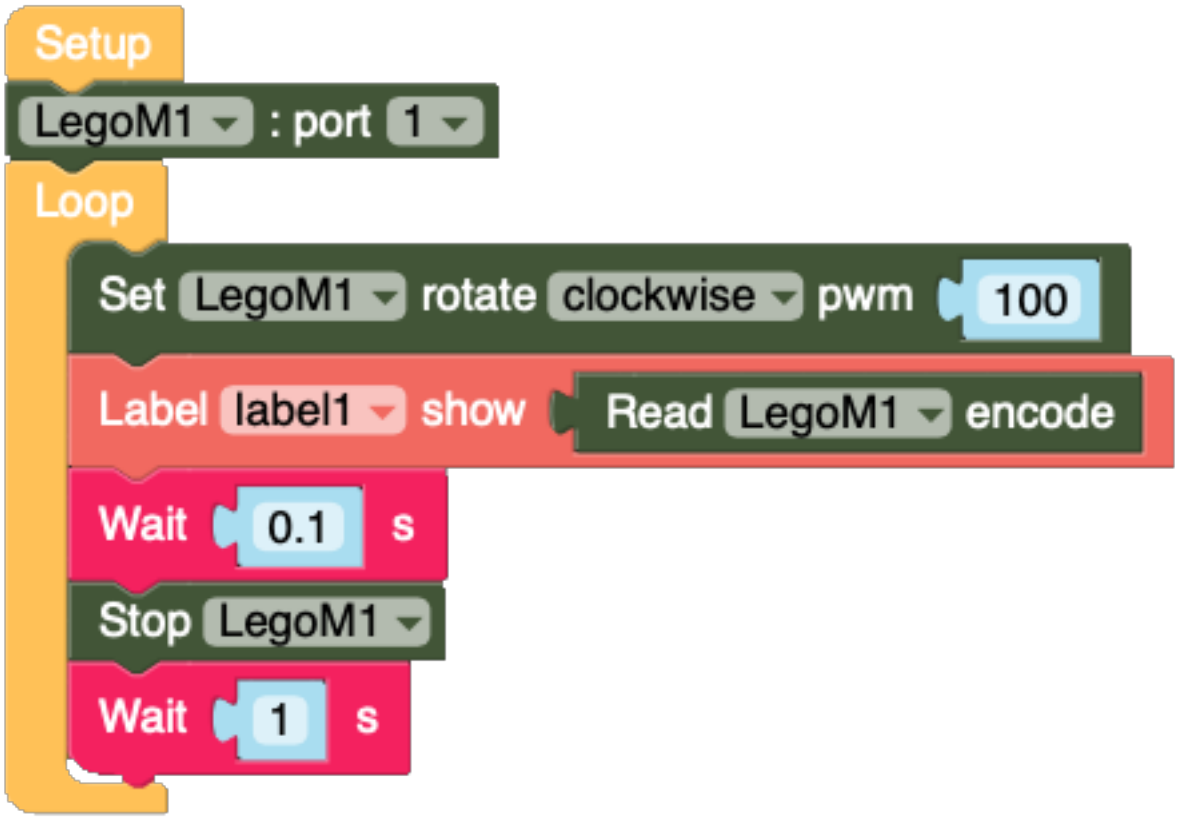
The Module can accept a voltage from 6V to 12V through the XT30 connector which is then run through an LM2596 regulator to produce the 5V needed by the logic and the motors.

An ATMEGA328P is used to communicate with the M5Stack and to drive the L293DD motor controllers used to drive the four LEGO® Technic™ NXT motors.



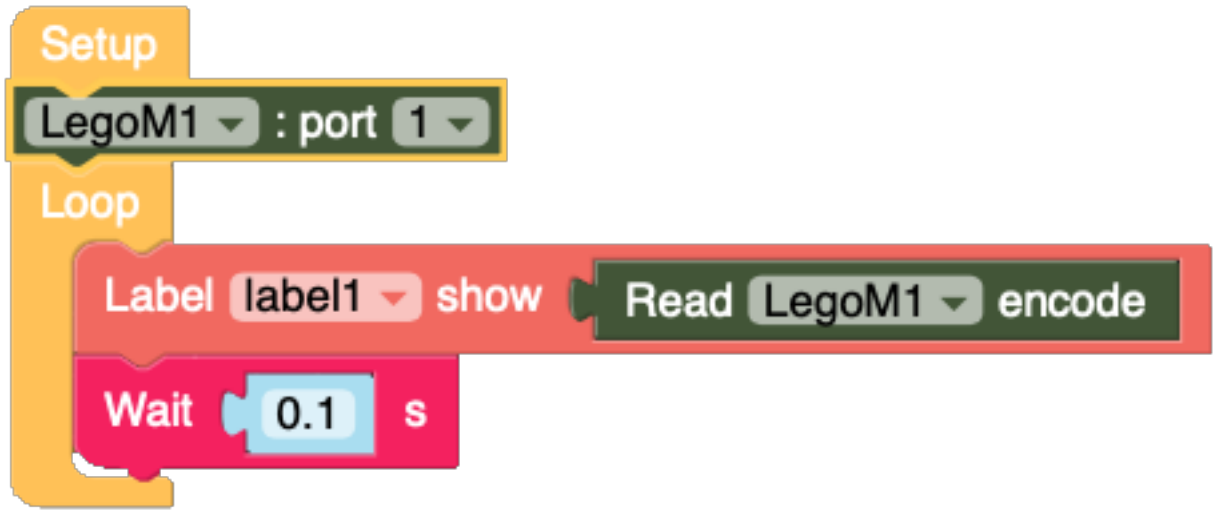
In the schematic diagram above we can see how everything connects together and how the I2C ports connect to the main I2C bus connected to GPIO pin 21 and 22 which is the same pins used by the RED PORT on the Core Unit.

In order to control LEGO® Technic™ NXT motors in UIFlow we can use the following basic code. In this example I didn't need to use an external power supply as I only had one motor connected. The "LegoM1:Port1" block tells the M5stack where to find the motor connected to the Lego+ Module and is placed outside of the loop because the code only need to know the location when it first starts. The loop set the motor running while reading the encoder for 0.1 seconds and then stops the motor before waiting 1 second before restarting the loop.
Because the encoder returns a value which the label prints on the screen, the longer the program runs, the higher the value will climb



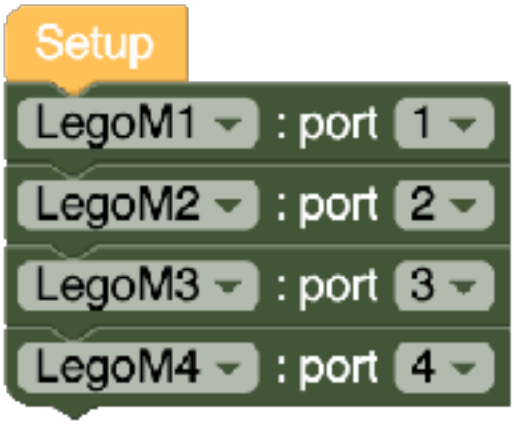
.

We can read the encoder without running the motor and use it as a form of controller using the following code.



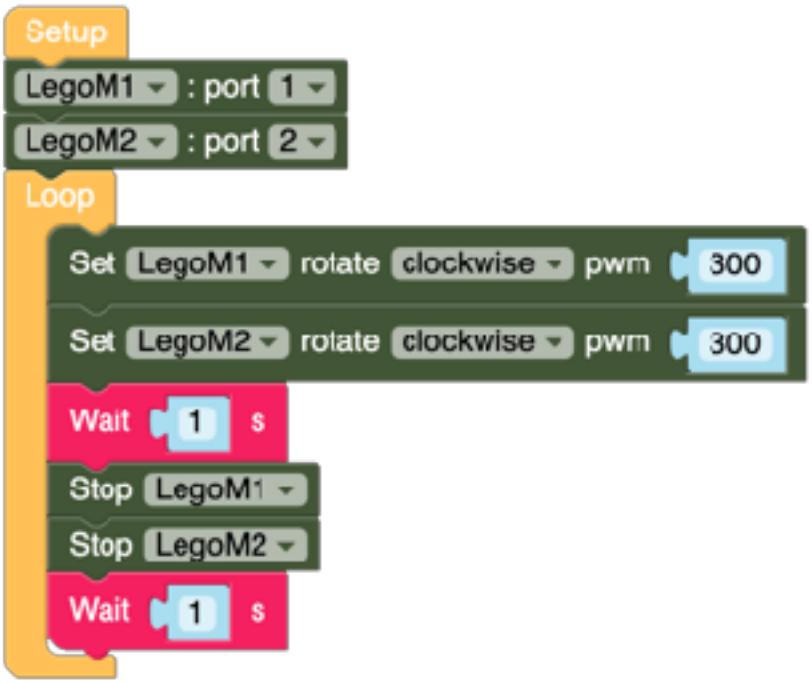
This code read the encoder and just increments the value shown on the screen.

When Using the Lego+ module in UIFlow you need to first define the motors connected to the module. This is done with the **LegoM1:port1** block, open up the block draws and drag out the block, placing it directly under the **Setup** Block. You need a separate block for each of the motors connected to the four motor ports around the side of the module.



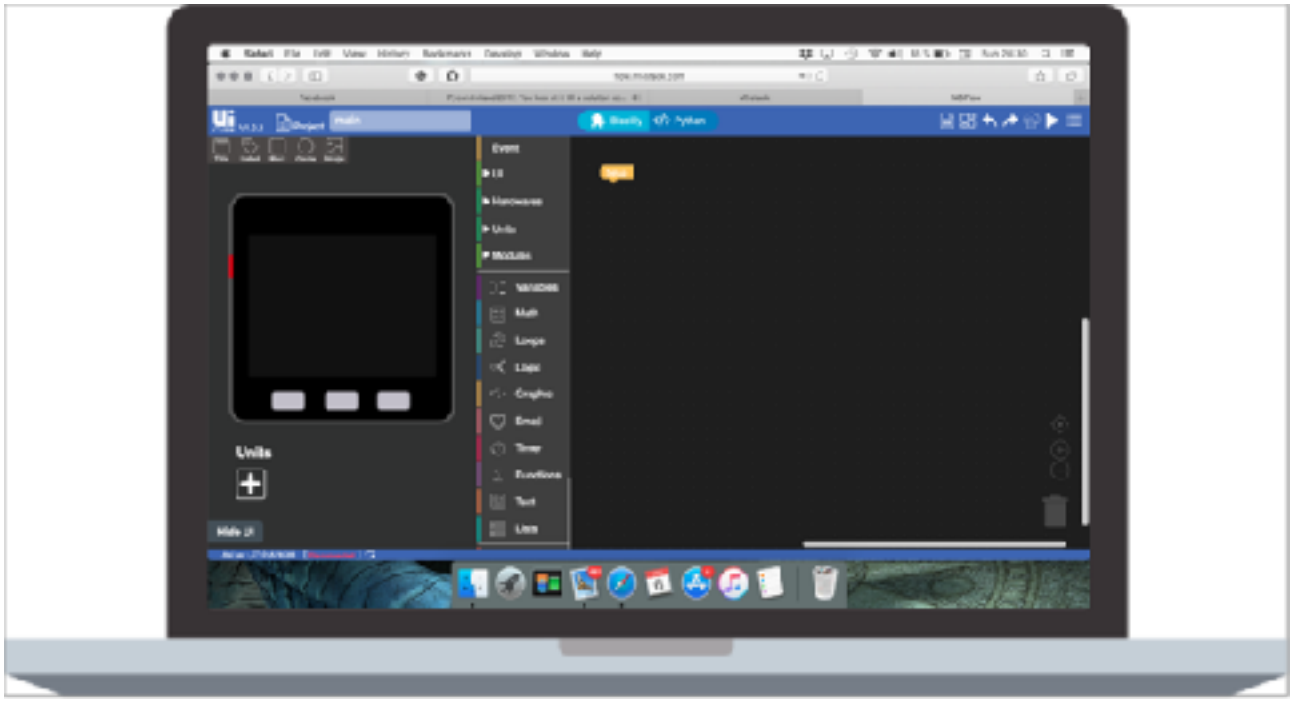
In the image above I have set the motors to each of the four port systematically however we can set the motors and port in any order to match the physical connection of motors to the module e.g. LegoM2 could be on port 3.

In the following code I am using two motors connected to ports 1 and 2 to drive forward one second with a PWM frequency of 300 which results in a very rough equivalent of 70mm of travel per second as measured on a metal ruler.



4

UIFlow



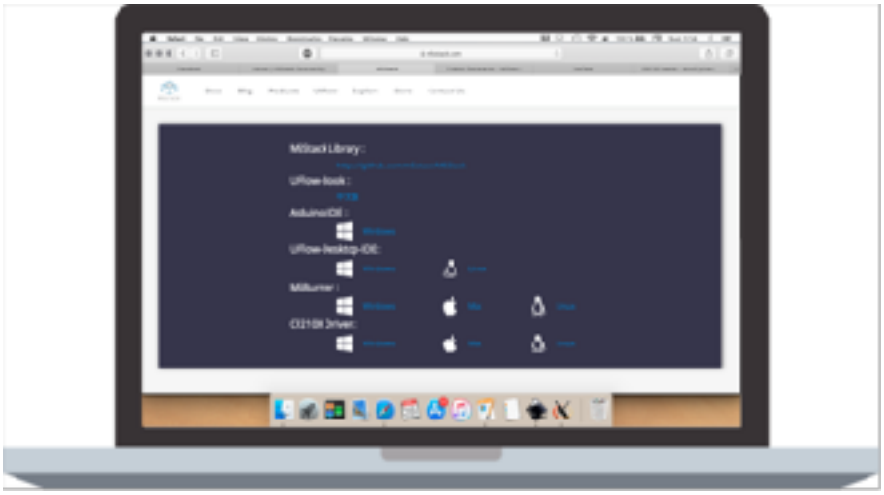
UIFlow is the **I**ntegrated **D**evelopment **E**nvironment **(IDE)** created by M5Stack and is the third incarnation of the programming environment.
In various documents references are made to M5Cloud and Moments, these are the two previous versions and are no longer available.

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.

4.1

Installing the UIFlow Firmware.

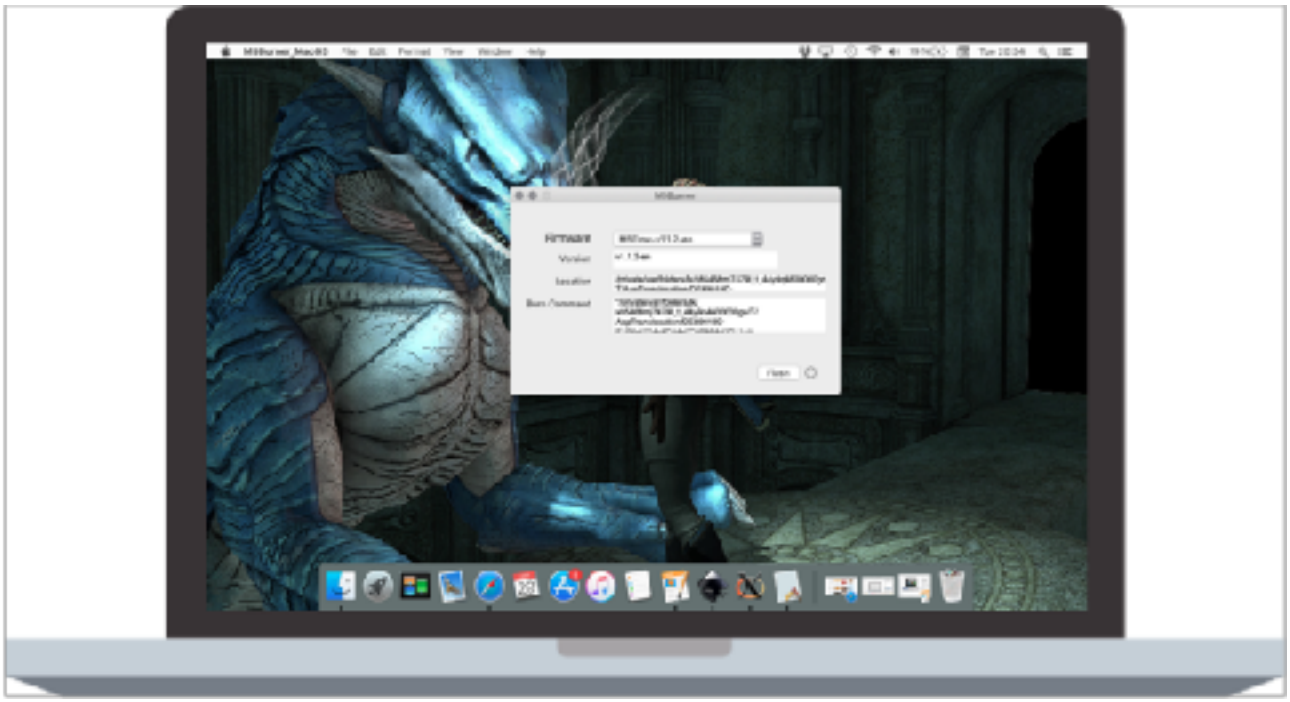
To use UIFlow on the M5Stack devices and to also update UIFlow to the latest firmware versions you need to follow the following procedure that works on all M5Stacks and M5Sticks.
First we need to go to the M5Stack homepage found at https://www.m5stack.com/
At the top of the screen you will see the title 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.
Once the driver is installed you can download the M5Burner app for your computer which is needed to install and update firmware.

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.

When M5Burner starts up you will be presented with a panel matching the screenshot on the following page.



The top box contains the current available firmwares while the following give information on the firmware including the flashing commands.
Make sure that the M5Stack or M5Stick is plugged in, select a firmware version (you can upgrade and downgrade firmware from M5Burner) and press flash. When the circle next to the flash button has filled, wait thirty seconds and then close down M5Burner and disconnect the M5Stack.
Connecting to UIFlow uses the same steps as in Chapter 2.

We write code in blocky by selecting blocks from the containers in the middle of the screen and drag them on to the right of the screen (the workspace) placing them one under the other. When the code is run it always starts at the "Setup" block and travels sequentially done the screen one block at a time.

The blocks in Blocky are separated into three groups loops, Actions and Variables.

Loops
Loop blocks repeat the actions that have been placed inside them.

Actions
Actions are the "Do Something" parts of code. These can be interfacing with hardware or preforming actions like mathematics.

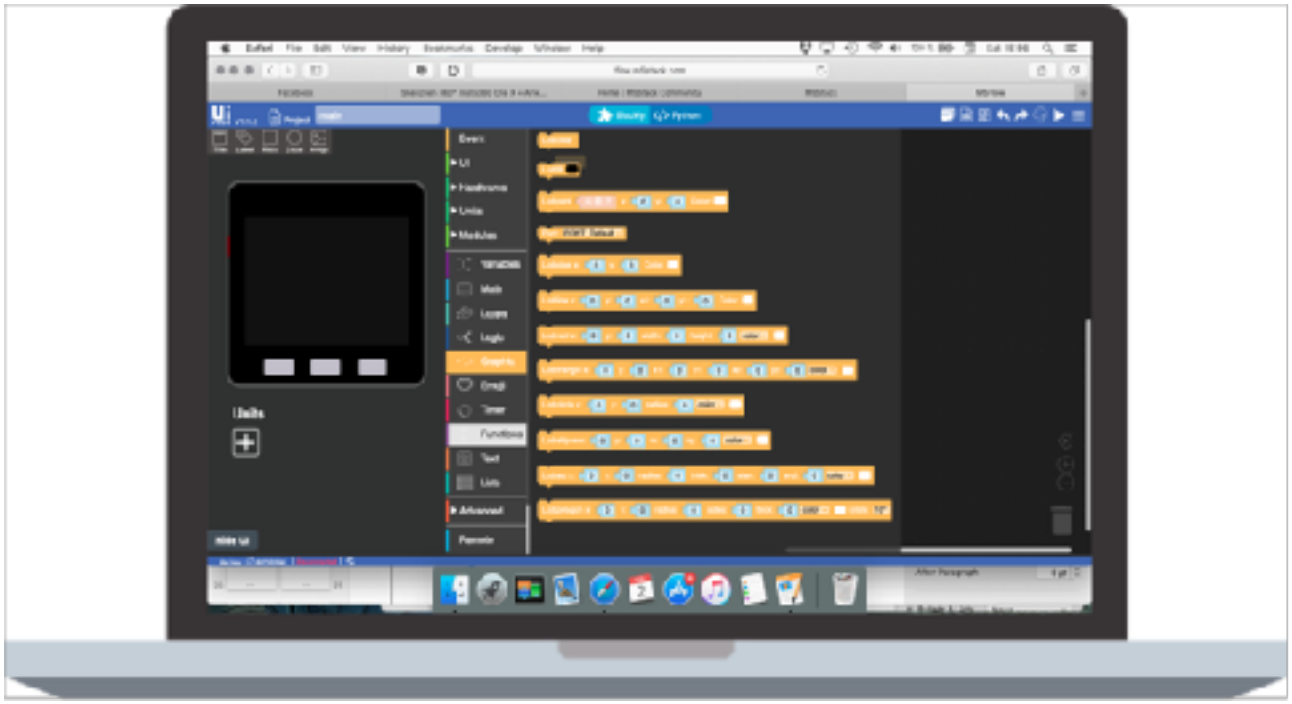
Variables
Variable are blocks which can contain values returned from sensors or values set by the programmer.

Throughout this chapter I will explore the various blocks available in UIFlow, how they interact with each other and with hardware.
I have divided chapter four into seperate sub chapters based on their function types.

- Images, Graphics and Text,
- Events,
- Internal Hardware,
- Units,
- Modules,
- Variables,
- Maths,
- Loops,
- Logic,
- Timer,
- Functions,
- Lists,
- Advanced Functions,
-

4.2

Images and Graphics.



4.1.1

Virtual User interface designer and 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. 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 Neopixels 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

Table 01 UI Block compatibility.

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 text 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.

4.1.2

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.
All images must be unto a maximum size of **320 pixels wide** by **240 pixels high**, have a maximum file size of **24Kbytes** and must be in **.JPG** or **.BMP** format.

- **.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.

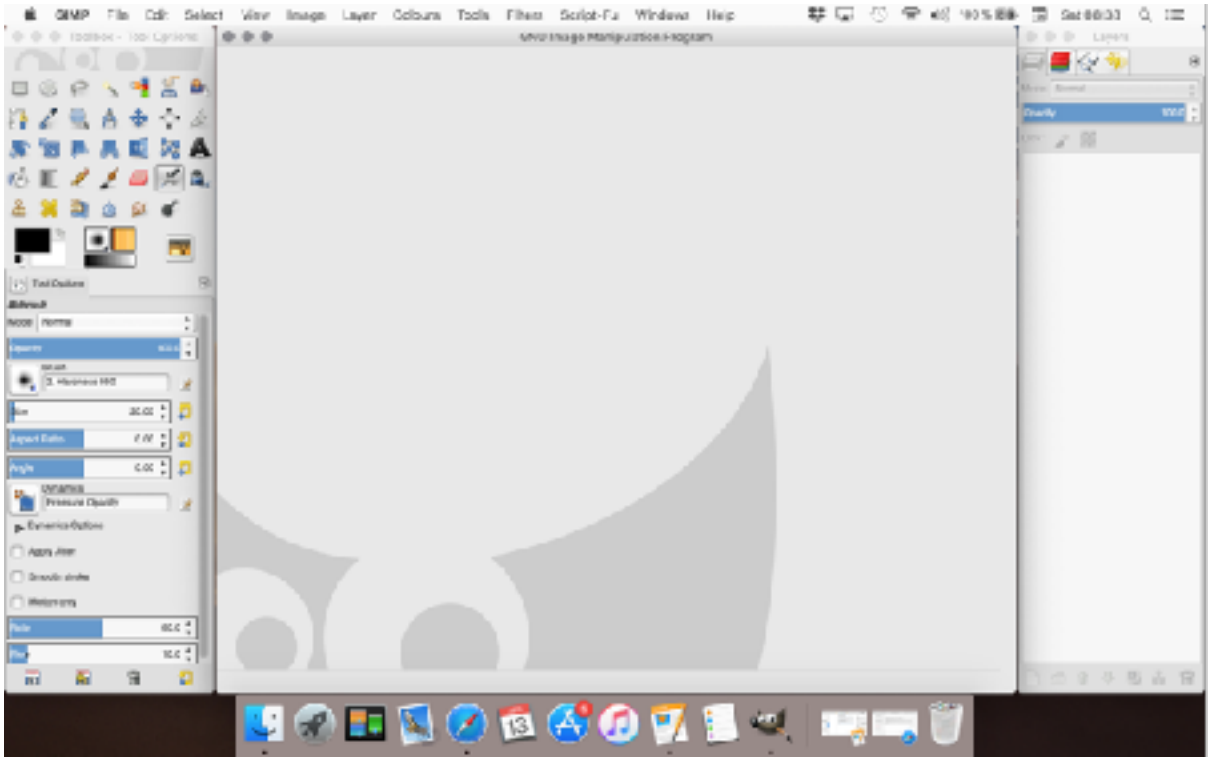
.**BMP** images are supported.

- 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).



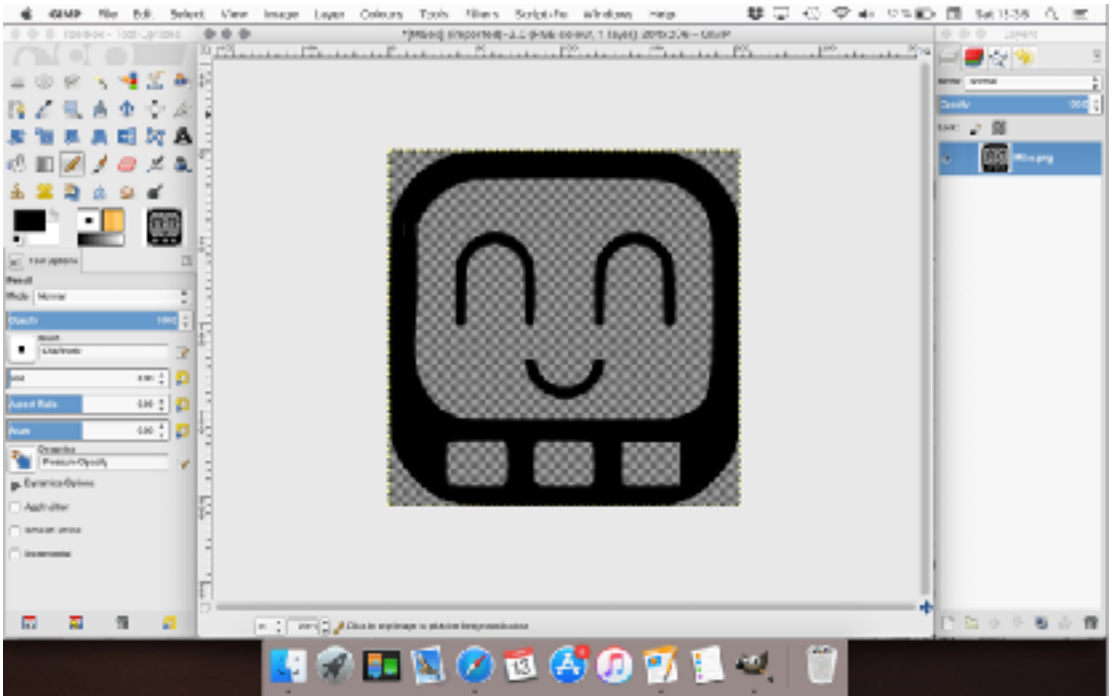
Screenshot of how G.I.M.P's windows are configured on OSX 10.14.2

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.

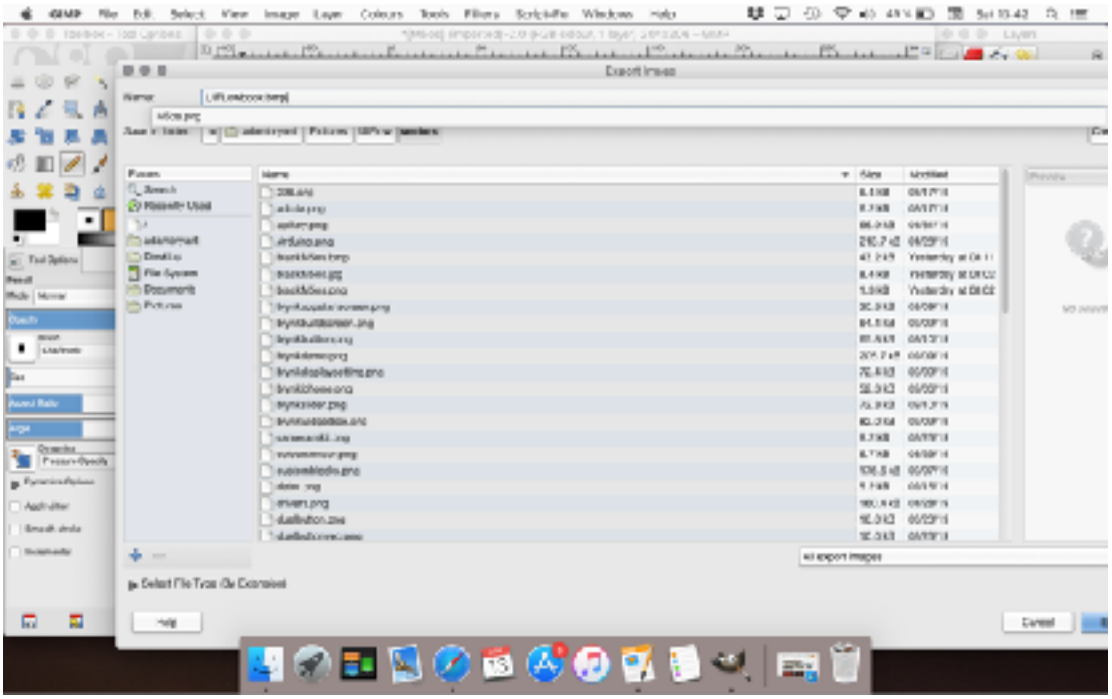
To get started, first we need an image.

The Image I am using above has been provided by Lucas from M5Stack.

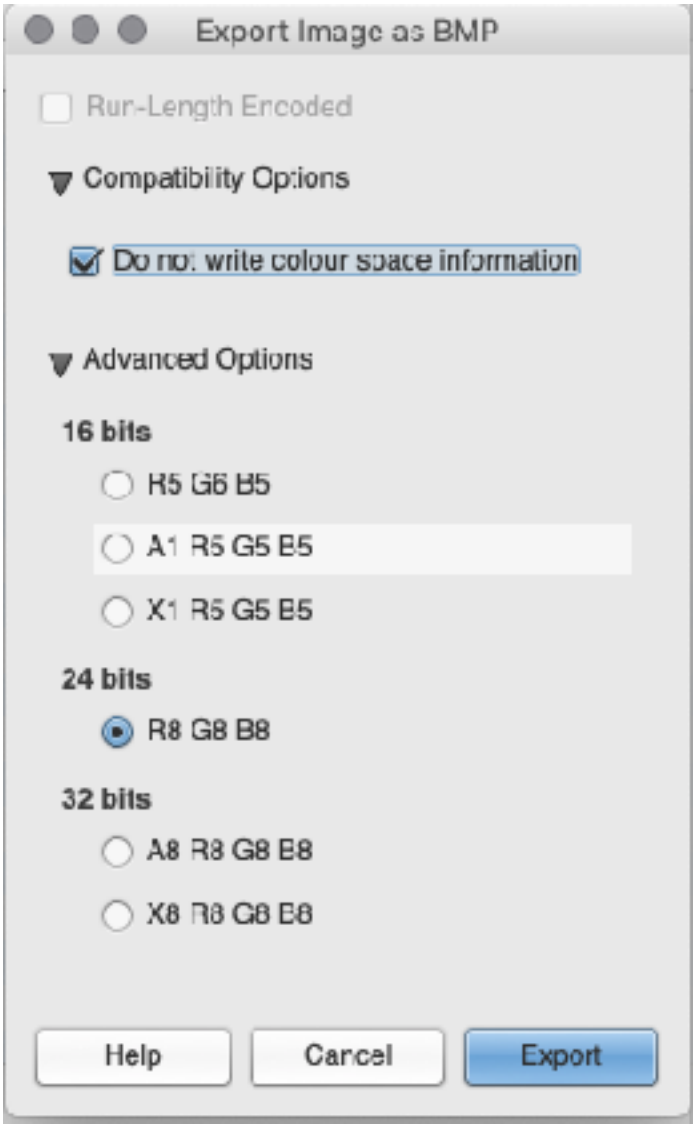


When loaded up we can see that our image has a transparent background. The available file formats that Micropython can read do not at present support transparency and when we save the image the transparent background will turn white.

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.



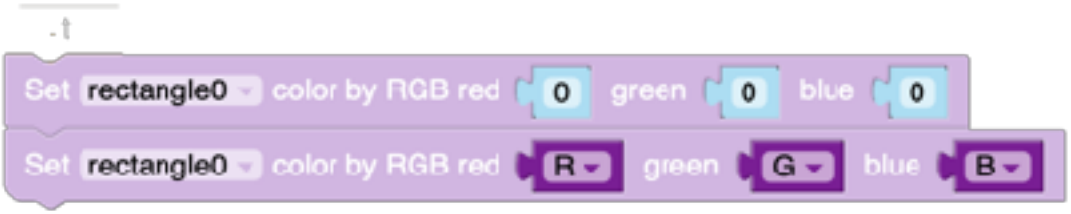
In this window click on the arrow next to **Compatibility Options** and a box will appear with a tick box next to **Do not Write colour space information.**
Click the box to make an arrow appear in it and then click the **Advanced Options** arrow and the above options will appear. Click on the circle 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.

4.1.3

Colour Setting.

The WS2812, RGB LED, and the colour screen on the M5Stack share the same colour setting functions. Colour can be set using the colour picker or by setting the individual channels R, G, and B, with a value of 0 to 255 for a total of 16 Million possible shades and colours.

We are not restricted to manually putting in the values for colours and brightness, the value boxes allow for the use of a variable block to control the values from code calculation.



The top block shows the default options for the colour value block. The light blue blocks are the ones that you add the values from 0 to 255. In the bottom block I have added the purple Variable blocks R, G, and B, which are place holders for the variable that can be controlled.

The colour picker only has a choice of seventy colours and shades which are connected to the predefined colour pallet defined in Micropython.

In addition to the colour selection, we also have a separate brightness block where we can set the value manually or via a variable.

4.1.4

Graphics Menu.

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

Lcd.fill

Lcd.fill wll 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 Menu

The Emoji menu contains blocks that allow us to draw Emojis and control the background shown behind them.

There a 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.

Set cell colour.
This block allows you to set the colour of each individual cell.

Change Background Image.
Allows you to chose one of the eight built in backgrounds that show behind the emoji.

5

Micropython

As mentioned previously UIFlow also has a window that allows us to program directly in Micropython. This is not the only was to write Micropython on the M5Stack, there is also the built in R.E.P.L environment that can be directly accessed on the M5Stack and M5Sticks.

To access R.E.P.L on the M5Stack we need to plug in the M5Stack to a USB port and open the command prompt/ terminal program. Once open then then type in the following and hit Return.

```
screen /dev/tty.SLAB_USBtoUART 115200
```

Once the computer has connected to the M5stack it will look like the terminal has frozen, Press CTRL+A and the R.E.P.L command prompt will open.

6

Examples and Experiments.

In this chapter I will show you what can be done using just the items available in the M5Stack range.

6.1

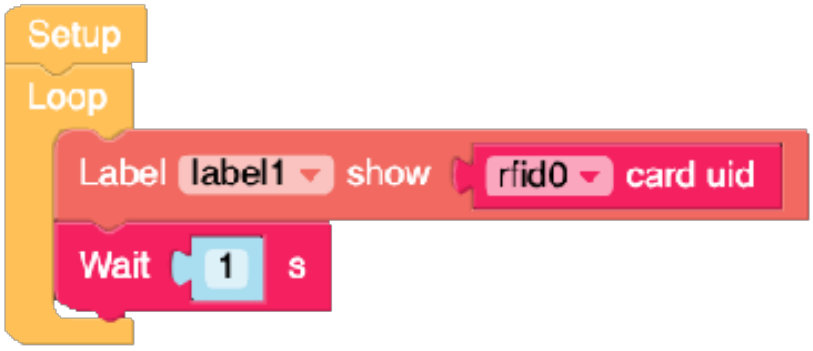
R.F.I.D Door Access Control.

In this example I will attempt to you you how to make a rudimentary R.F.I.D security tag controlled door access system. This example is only meant as a demonstration and not meant for full use as it has various security flaws.

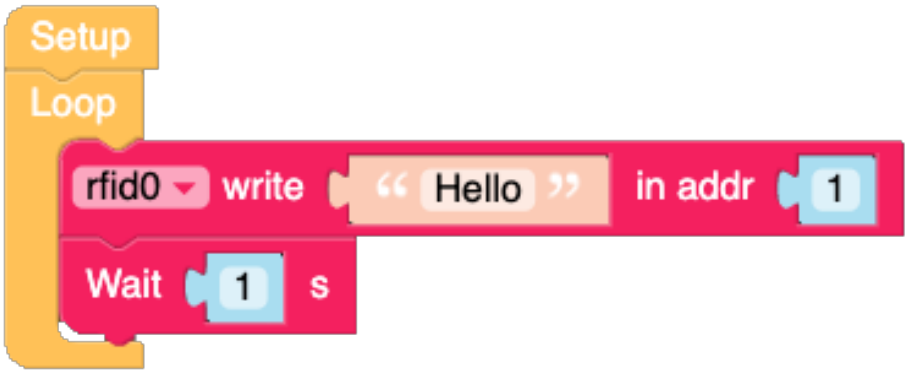
To make the R.F.I.D door access control you will need the following items.

M5Stack,
R.F.I.D Unit,
Relay Unit,
Two Connector Cables,
Some R.F.I.D computable tags or cards.

First take one of the cables and connect one end to **PORT A** and the other end to the **R.F.I.D Unit**.
Take the second cable and connect one end to PORT B and the other end to the **Relay Unit.**
Next we take the UID reader code from R.F.I.D Unit chapter and make a note of the codes stored on the cards.

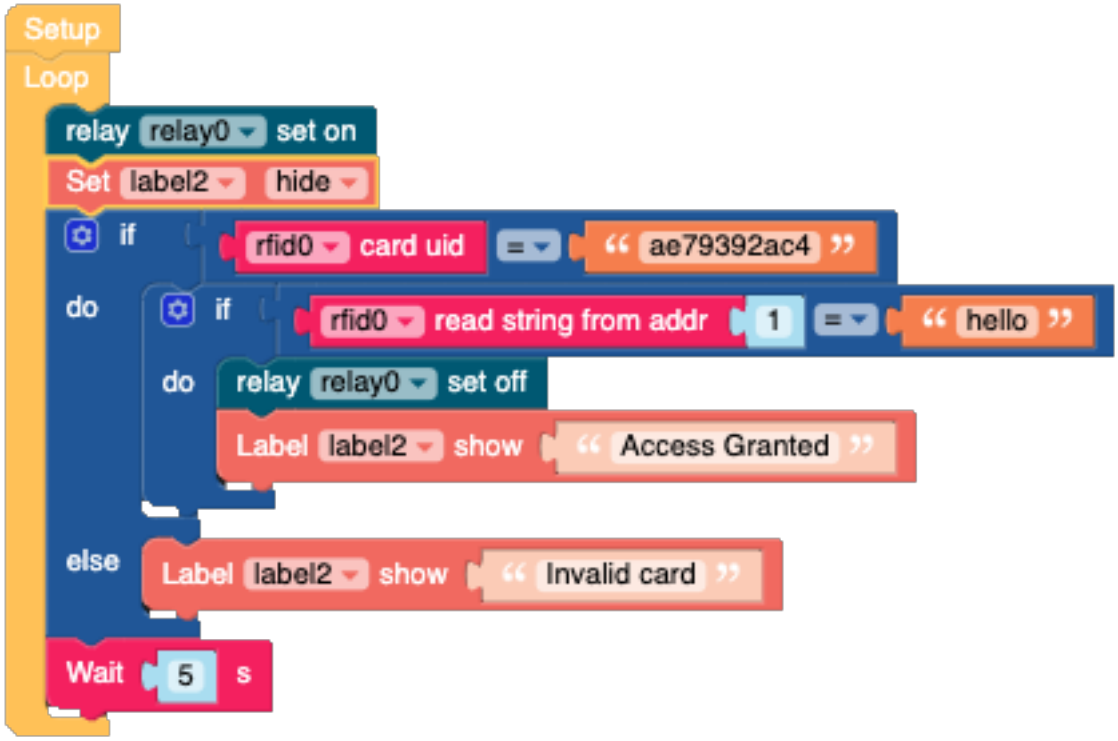


To add a little bit of additional security to the cards we will write some information to the cards limited memory. Using the following code you can write some simple data like a code word to the memory.



Here I just chose hello as I couldn't think of anything else at the time.

Place an R.F.I.D computable tags or card on the unit and run the code. We don't need to download the code as we will only do this once. after a few seconds remove the card and reset the M5Stack. Next it time for the code for the door access. Copy the following and download it onto the M5Stack.



When you you will hear the relay click as it is switched on. The code check for the presence of a card with both UID and string shown and if it detects both it will deactivate the relay. If it doesn't find both strings the relay will stay on and "Invalid card" will be printed on the screen.

I mentioned earlier that there are security reasons why this is just to be used as a demo for the code. The first is that the UID and string are clearly visible in the code and as such, anyone has access to it. The next is that you can overwrite the code via the USB and the relay will no longer activate. Next issue is the hardware, by just connecting the cable to the relay, the whole system is deactivated and likewise if the battery runs out, again the relay will deactivate again rendering the whole system useless.

7

Moving On.

In this chapter I will show you what else can be done with the M5Stack using some additional hardware and electronics.

In this chapter I will show you the following projects.

DIY Angle sensor.
(Mostly recycled) Pocket Laser Engraver.
Automated Ambush Striker.

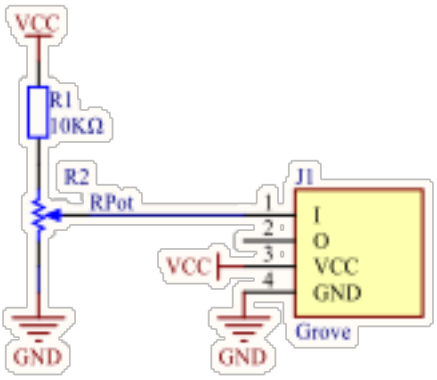
7.1

DIY Angle Sensor.

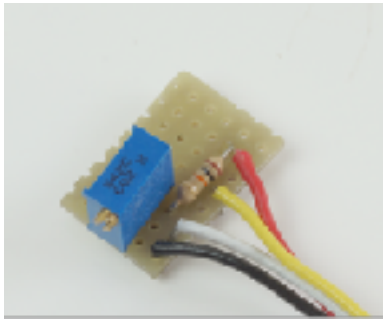
In this chapter I will show you how to build a version of the official Angle sensor including how to create a prototype P.C.B using Fritzing.
The reason I chose to copy the official Angle sensor is because M5Stack has gone to the trouble of releasing the schematic for it online and so means that we have a prebuilt version which we can compare our own to and it only uses five components if you count the pcb and the grove lead.

For more information on the official sensor you can read chapter 3.3.3.
For Fritzing, you can download it for free from http://fritzing.org/home/
All components were found online.

Lets start with the schematic circuit diagram,

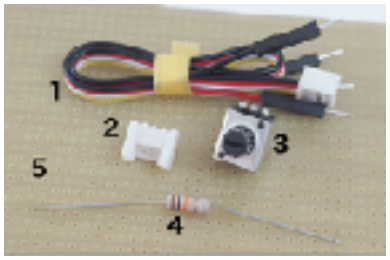


In the diagram we can see that **R1** is a 10K resistor which is connected between **VCC** and the input and a variable resistor between the input, **R1** and **GND.**



This is the early prototype I made based on the schematic diagram. While the parts are not finger friendly, it works.

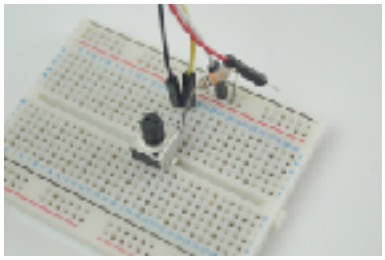
Step 1 - The Parts



Here are the parts required.

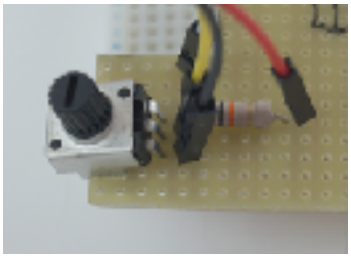
1, Grove to Dupont lead.
2, Grove PCB Connector
3, 10K Potentiometer,
4, 10K Resistor,
5, P.C.B.
6, Bread Board (Not shown here.)

Step 2 - Breadboard Layout.



Here I have played out the parts on Breadboard to test the circuit.
The GND (Black) wire is connected to one side of the potentiometer.
The Red wire connects to the resistor.

Step 3 - Proto PCB design.

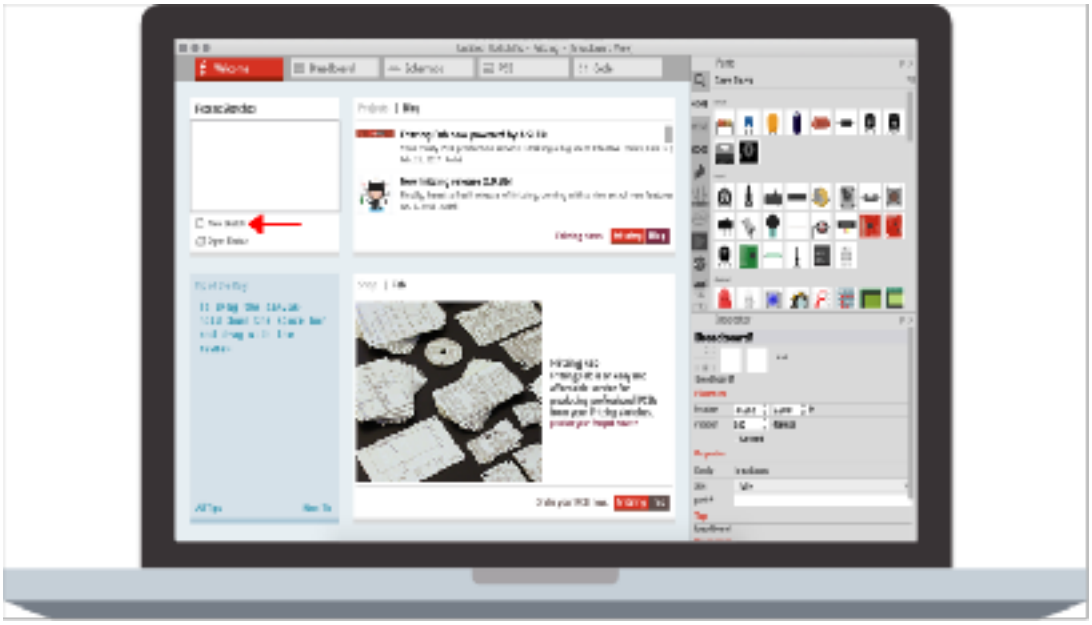


The potentiometer has a pair of legs connected to the metal shroud that stops it sitting flat on the P.C.B. Here I had to cut two slots in order to fit the potentiometer flush on the board. I haven't soldered the components in place as this is just to show PCB layout.

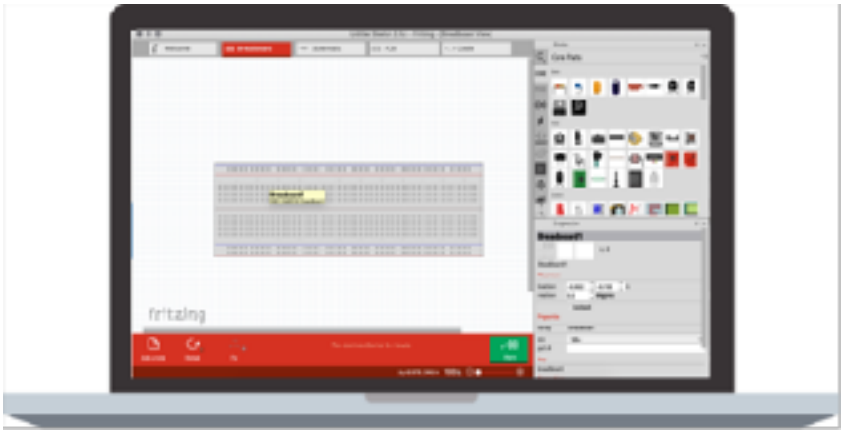
On each of these three steps I used a multimeter set to resistance mode to make sure there were no shorts between VCC and GND.

In order to make a P.C.B. we need to use some software. Because of its very simplistic nature I chose to use Fritzing for the P.C.B. design but it does have some issues.

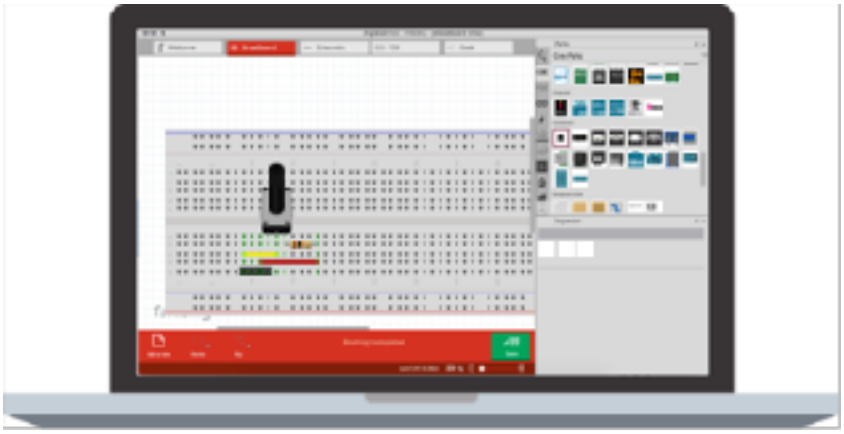
First we need to load Fritzing and start a new project.



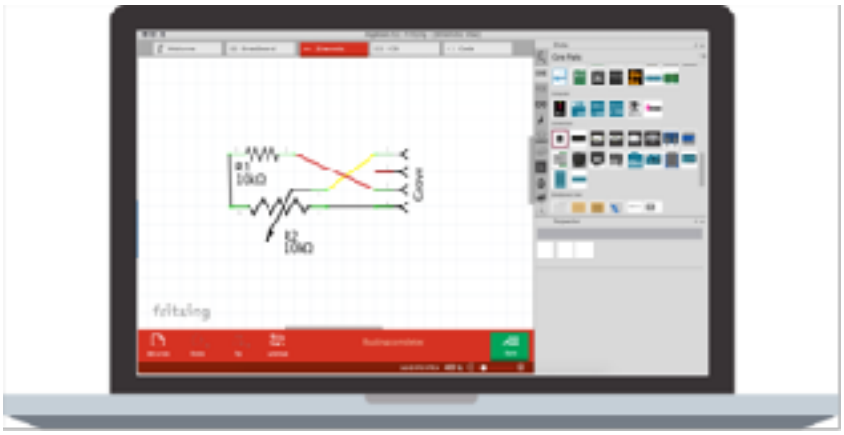
You will then be taken to the breadboard window when we place the parts for rough connection.



drag the components from the panel on the left and place them on the breadboard as shown below.

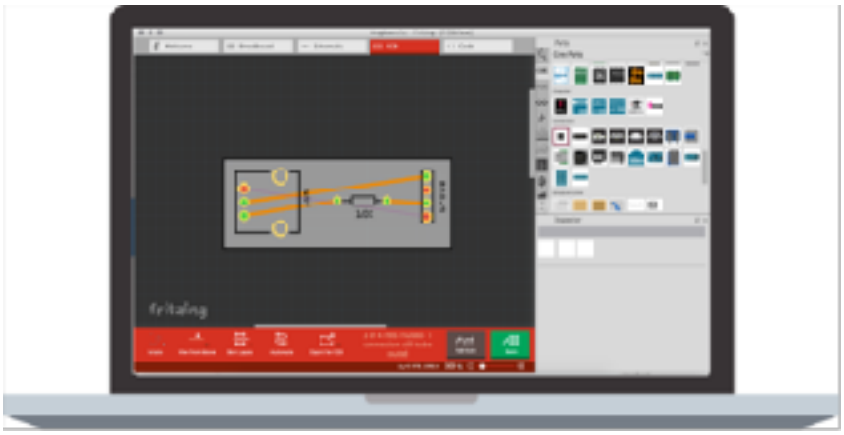


I have used a four pin header here to represent the grove connector as there isn't one built into the parts library. the red and yellow lines are the VCC wire and the A0 wire connecting the header to to the resistor and middle pin of the potentiometer. When finished we can move onto the schematic panel.



When you first switch to the panel after adding components to the breadboard they will be scattered around the screen. Move the components into position as show (right click to bring up the rotate menu) and double click the dotted lines to change them to wires.
Here I have changed the wire colours to match the ones used on the breadboard view and in the grove leads.

Once we have the schematic sorted out, save the file and move over to the PCB panel.



I have forgotten to take a screen shot here but when you switch to the PCB panel the parts will be scattered around the screen again. Move them into place as shown (do not place the header all the way to the edge!) and again double click the dotted lines to connect them together. You will noticed that I didn't connect the GND wire between the potentiometer and the grove connection here, this is for a reason. Right click the unconnected pin of the potentiometer and click "Set as ground seed" and then right click the bottom pin of the grove connector.

Now go to Routing>Ground Fill>Ground fill top and bottom. the unused area of the PCB will be filled with copper and the two pins we set as ground seeds will be connected together.
I forgot to change the grove connection here and had to select it and change some sizes.

Click on "Pin Spacing" and set to 2mm and then click on "Ring Thickness" and set to 0.3mm. if Ring thickness is any bigger the connections on the pcbs will be shorted together.
Re do the Ground fill and save the file again.

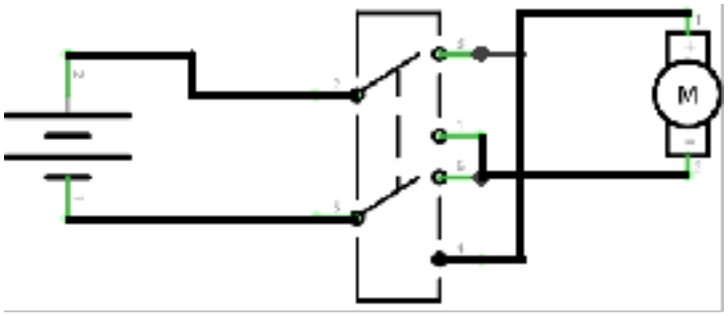
The next step is to generate the PCB files needed to create a PCB. at the bottom of the screen is a button that says "Export for PCB", click on the little down arrow next to it to bring up the export options. Chose the "Extended Gerber" option and then save to an empty folder.
Next open the folder in your computers file browser, selected all the files generated by frizzing and compress into a zip file. No you can send this file to a PCB fabrication house as it should be good for production.
Below is the finished PCB and the assembled version.

BLANK PCB IMAGE MISSING!

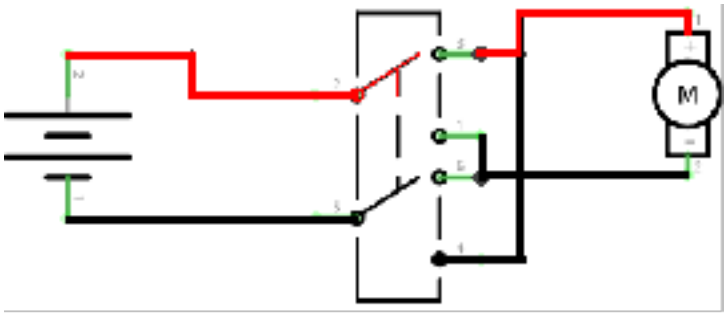
7.3

Automated Ambush Striker.

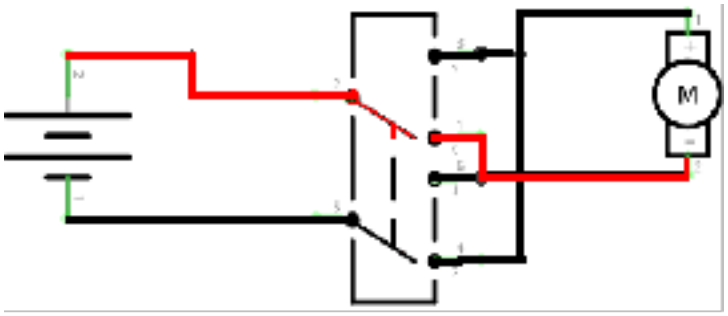
The Ambush Striker is a motorised ball catapult produced by **VEX Robotics.**
Because of the simple one motor operation of the Ambush Striker, we can easily use the **Relay** module to control it with a P.I.R. **Motion Sensor**.
Because of the simple circuit used to power the Ambush Striker, not much electronic knowledge is needed to connect it to or M5Stack.
The following circuit diagram is a simplified schematic of the strikers motor circuit.



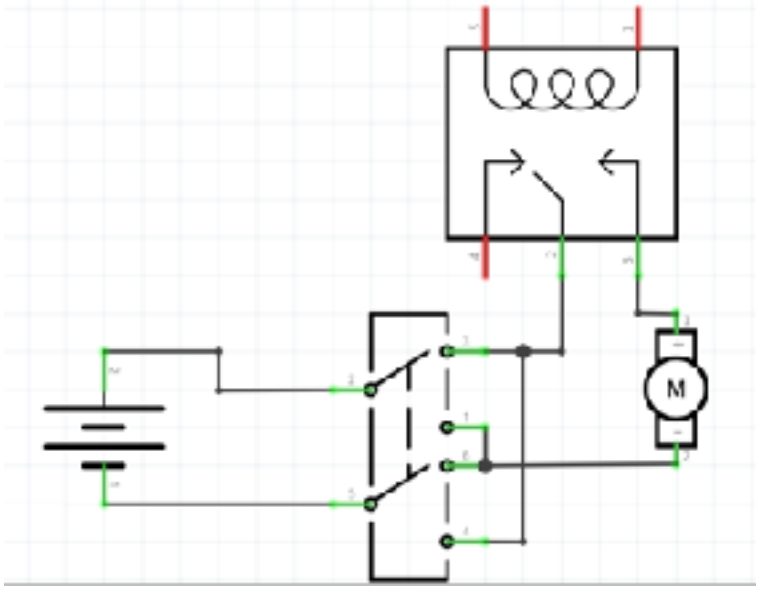
The circuit use the DPDT switch (Double Pole Double Throw) to reverse the polarity of the motor.
In normal use the power flows along the red line to the motor .



By flipping the DPDT switch which has it connections connected together, the power now flow the opposite way around.



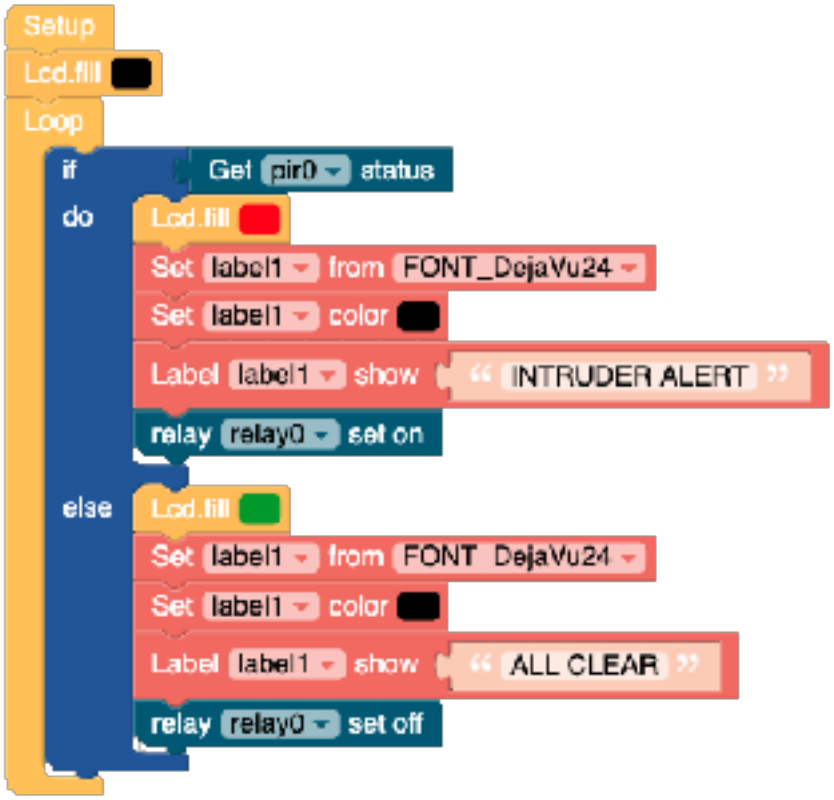
To control the motor with the M5Stack we need to use the relay module connected between the switch and the motor.



The existing switch has a latch inside it that allows us to lock the existing switch on and control the Ambush Striker with just the relay (or have a safety switch). The wire from the switch is connected to the Common connection on the relay and the wire to the motor is connected to the Normally Open connection. This is done by just cutting the red wire in the middle **(DO NOT CUT THE BLACK WIRE!)**, stripping the two red ends, putting the bare conductors into the screw terminals and tightening them up.

To sense the presence of someone in the room, we will use a P.I.R. Motion sensor however, both of devices require the use of the I/O port **(Port B)**. We could use **Port A** in Digital I/O mode or we could use the **Grove Hub** or the **Grove T**. This is possible because while both devices use the same I/O port, the P.I.R uses **pin 4** which is the input pin and the relay uses **pin 3** which is the output pin.

In the code example below I have used and "If/else" logic block to get the status from the P.I.R, change label1 to "All Clear" and fill the background green or "Intruder Alert" with a red background and trigger the relay if it detects the presence of a person.
Because I have place the "If/Else" logic block in a loop, the code automatically resets the relay when the area is clear.



This is just a simple of the implementation of some code and you could use a "Wait" block after the "Relay set on" block to keep the ambush firing for longer.

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

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 table is a list of know I2C addresses used by M5Stack devices.

Device	Address
ADC	0x48
Card KB	0x5F
Colour Sensor	0x29
DAC	0x60
ENV (DHT12)	0x5C
ENV (BME280/BMP280)	0x76
EXT I/O (PCA9554PW)	0x27
Heart	0x57
Joystick	0x52
Makey Makey	0x51
N.C.I.R.	0x5A
R.F.I.D	0x28
Thermal	0x33
Trace (See 3.3.38)	0x5A

Table 1 - Unit I2C addresses.

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
Fn	NULL	NULL	NULL	NULL	NULL

Key	Value	Sym+Key	Value	Shift+Key	Fn+key
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