iModBot controllers

A fun and easy way to control iModBot via Bluetooth with Dualshock PS3 and PS4 controllers

Introduction

The iModBot project although it is a very sophisticated robot with a very extensive library with several features and modes, it has a great lack of control ability over it. Even though it has a dedicated area for Bluetooth control via smartphone, it is very limited, as we can only send one command at a time.

This addition in the project aims to explore the benefits of Bluetooth communication using controllers from popular consoles, such as the PS3 and PS4 (both properties of SONY). The programming to create a button layout is very easy to edit and will be referred to in this paper.

This project consists of 4 Arduino codes in the examples folder inside “iModBot” library, 2 of which use a ramp generator and the others do not, and 1 example in each Dualshock library. The libraries were created to simplify the editing of variables in the code and also to provide a better implementation in the block programming environment.

A ramp generator was used to prevent high current spikes in the motors, something that would eventually shut down the robot, because it has a current limiter in the DC-DC boost step-up and in the battery charger. This is no longer a problem, as the Step-up was removed from the board and a battery was added in series, but it was preferred to continue with a ramp generator to smooth out the current spikes and have a longer battery life.

For the example codes, the same controller codes have similarities, differing only in the ramp generator, while the functions of the buttons and analogues are the same. There are also blocks for the controllers in this project for use in Ardublock, with an area to insert the MAC Address, similar to the programming in the Arduino IDE.

Objectives

In this tutorial, the goal is to introduce and teach anyone how to set up the MAC address of controllers and insert them in the code using Arduino IDE software or ArduBlock block programming tool.
The two libraries required for this code to work will be discussed further down in this paper. The following themes will be discussed:

- MAC Address
- Libraries
- Configuration

(This tutorial already presupposes that the reader already has Arduino IDE software or Visual Studio Code installed and possesses minimal knowledge.)

MAC Address

A media access control address (**MAC address**) is a unique identifier assigned to a network interface controller (**NIC**) for use as a network address in communications within a network segment. This use is common in most IEEE 802 networking technologies, including Ethernet, *Wi-Fi*, and *Bluetooth*. Within the Open Systems Interconnection (**OSI**) network model, MAC addresses are used in the medium access control protocol sublayer of the data link layer.
As typically represented, MAC addresses are recognizable as six groups of two hexadecimal digits, separated by hyphens, colons, or without a separator.

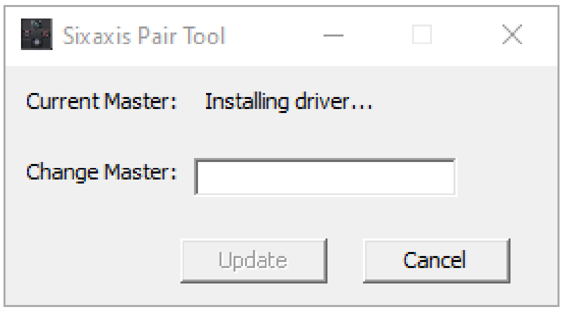
Controllers MAC Addresses

There are multiple ways access the *dualshock* controllers MAC addresses, either by using programs, accessing the device manager in windows or entering controllers settings in the console. These addresses can either be edited or read only, although editing the MAC address would imply in resetting the controller to have it connect to a console if it was previously paired with one, since the console saves the controllers MAC address.

Read/Update MAC Address

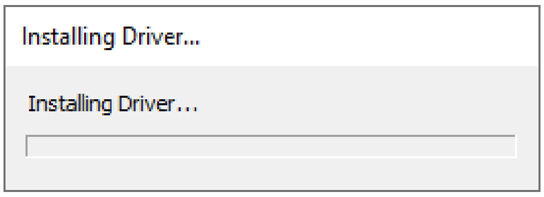
For this, you will need to download and install **SixAxis Pair Tool.** This will allow us to read or update the MAC address

In order the read or update the address, simply connect the controller to a computer. It should start installing the required drivers. Once it finishes installing, open **SixAxis Pair Tool**.
You will be presented with the following window.


Sixaxis Pair Tool

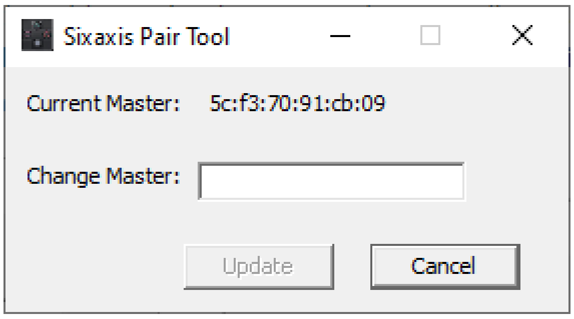
Current Master: Installing driver...

Change Master:


Installing Driver...

Installing Driver...

The program in this phase is installing the required drivers to fetch the Bluetooth ID address. This procedure works with both PS3 and PS4 controllers and should take about 5 minutes, depending on the computers speed. After the driver finished installing, 12 hexadecimal characters should appear at “Current Master:” like this:


Sixaxis Pair Tool

Current Master: 5c:f3:70:91:cb:09

Change Master:

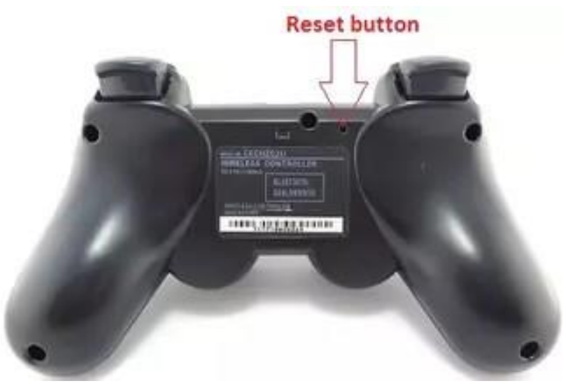
The ID presented in the area above is the Bluetooth ID Address of the controller which will be added to Arduino IDE or Ardublock. The user has the option now to either leave the MAC address untouched or update it with a new instance, for example: *01:01:01:01:01:01.*

It is generally a good idea to backup the original ID address.

Note: Editing the MAC Address, the controller will no longer pair with a console by Bluetooth. If the user wants to change the address to the original, he can do so if he has the original address with him. If this is not possible, there is a guide below on how to recover it.

PS3 controller reset

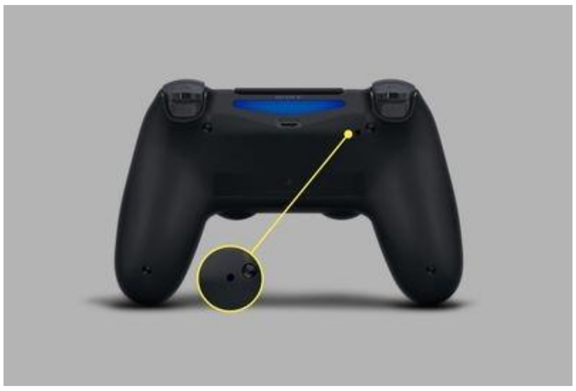
In case of a PS3 controller, to return to its original state, first press the reset button in the back of the controller for 5 seconds.



Note: Resetting the controller WILL NOT alter the MAC Address. This will have to be connected by USB to a PS3 and hold the “PS” button to pair.

PS4 controller reset

In case of a PS4 controller, to return to its original state, first press the reset button in the back of the controller for 5 seconds. This will set the MAC to the original address.



Libraries

The dualshock libraries required for this project are very similar, both operating with callback events. The code used to upload to ESP32 is inside the iModBot library. To successfully set up these libraries, follow the installation guide and download both libraries in the download tab below.

Downloads

PS3 Bluetooth Library

```
https://github.com/jvpernis/esp32-ps3
```

PS4 Bluetooth Library

```
https://github.com/aed3/PS4-esp32
```

iModBot Library

```
https://github.com/ipleiria-robotics/iModBot
```

Installation guide

Move the downloaded/cloned folders “PS4-esp32”, “esp32-ps3” and “iModBot” to your Arduino IDE libraries folder.

Note: If the user is unaware where the libraries folder is, open your Arduino IDE, go to File > Preferences, and your library folder is on the top of the window.

Troubleshooting

Note: Follow this troubleshooting guide if you are having problems with PS4 controller library.

▼ Error compiling for ESP32 board ('ESP_BT_CONNECTABLE' undeclared)

This error occurs when the ESP32 is an old version, typically V1 or V2. This can be fixed by swapping older ESP32 with newer ones, like the ESP32-DevKitC-V4. This can also be fixed via code. To do that, the reader will need to edit the “**src/ps4_spp.c”** and “**src/PS4Controller.cpp**” files with a code editor, like VSCode.

Inside **ps4_spp.c**, in lines 86 to 90, you should see something like this:

```
#if CONFIG_IDF_COMPATIBILITY >= IDF_COMPATIBILITY_MASTER_D9CE0BB
  esp_bt_gap_set_scan_mode(ESP_BT_CONNECTABLE, ESP_BT_NON_DISCOVERABLE);
#elif CONFIG_IDF_COMPATIBILITY >= IDF_COMPATIBILITY_MASTER_21AF1D7
  esp_bt_gap_set_scan_mode(ESP_BT_SCAN_MODE_CONNECTABLE);
#endif
```

Replace with this code:

```
//#if CONFIG_IDF_COMPATIBILITY >= IDF_COMPATIBILITY_MASTER_D9CE0BB
//   esp_bt_gap_set_scan_mode(ESP_BT_CONNECTABLE,  ESP_BT_NON_DISCOVERABLE);
//#elif  CONFIG_IDF_COMPATIBILITY  >=  IDF_COMPATIBILITY_MASTER_21AF1D7
  esp_bt_gap_set_scan_mode(ESP_BT_SCAN_MODE_CONNECTABLE);
//#endif
```

Inside **PS4Controller.cpp**, add the following code to line 44, before *PS4Controller::begin* function:

```
#define  ESP_BD_ADDR_STR  "%02hhx:%02hhx:%02hhx:%02hhx:%02hhx:%02hhx"
```

▼ PS4 controller shuts down after connecting

This problem usually happens when controller is given a new MAC address or is paired with a different device. This is not a very common problem, but the fix is simple. For this, you must have python installed in your system.
After completing the python installation, press **Windows + R**, write "cmd" and hit OK. A dark window should appear and you'll have to insert the following command:

```
pip install esptool
```

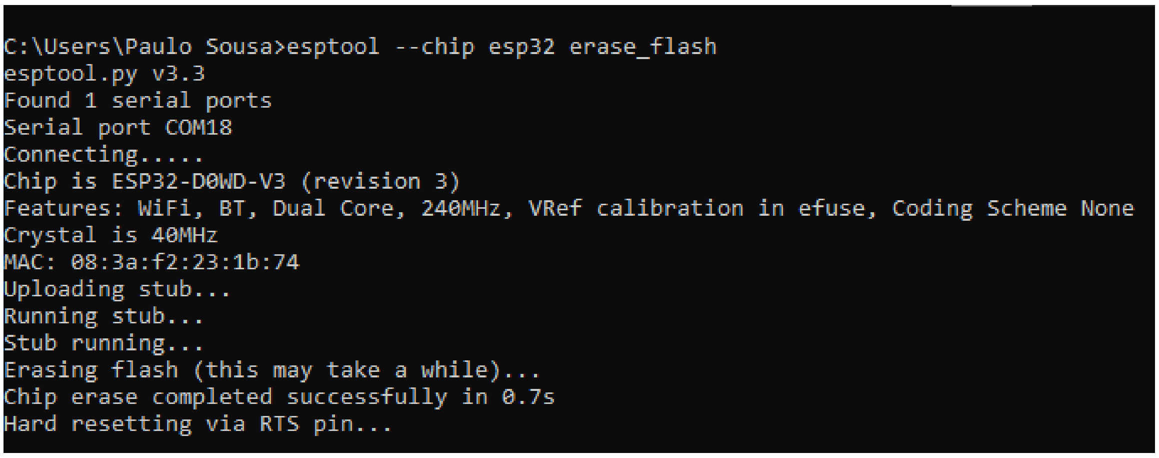
After successfully installing esptool, you will run the required code to erase the flash in your ESP32, since that is where the problem is located. Connect ESP32 to the computer and run the following command:

```
esptool.py --chip esp32 erase_flash
```

If the previous command did not work, try this:

```
esptool --chip esp32 erase_flash
```

After pressing enter, you should be greeted with the following lines:



```
C:\Users\Paulo Sousa>esptool --chip esp32 erase_flash
esptool.py v3.3
Found 1 serial ports
Serial port COM18
Connecting.....
Chip is ESP32-D0WD-V3 (revision 3)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 08:3a:f2:23:1b:74
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 0.7s
Hard resetting via RTS pin...
```

Now, you should be able to connect your controller to the ESP32 without any problems.

Configuration

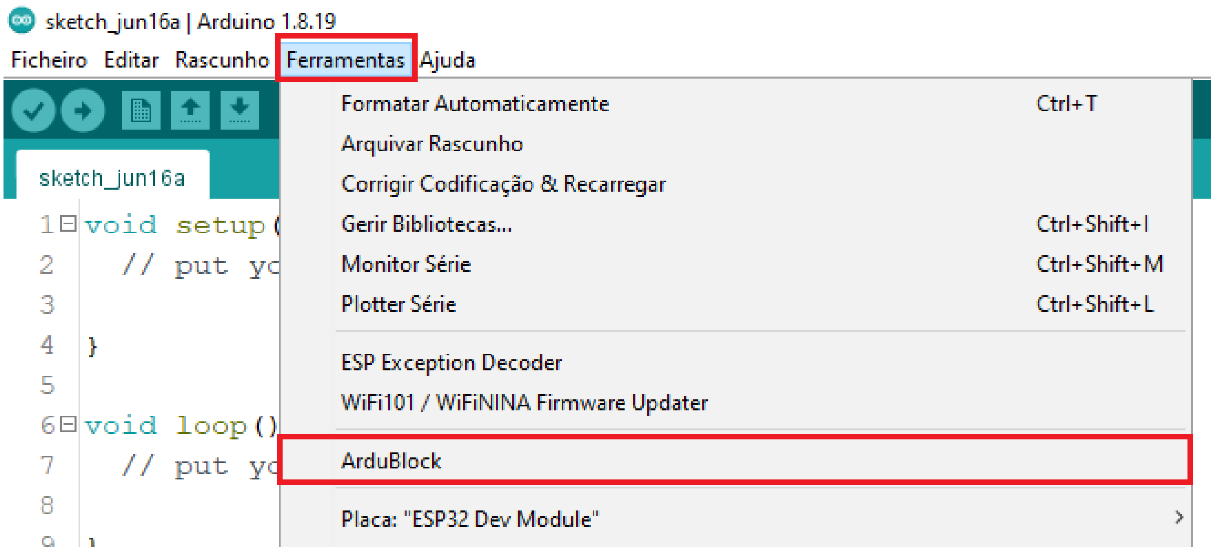
To modify the MAC address in the code, just open the Arduino IDE, go down to the "void setup()" function where there is a line with the following code. The procedure is identical to both parties:

```
Ps3.begin("01:02:03:04:05:06"); // PS3 controller example
PS4.begin("07:08:09:0A:0B:0C"); // PS4 controller example
```

The address obtained from the SixAxis Pair Tool program (Current Master) must be copied into the brackets in the PS3 or PS4 function.

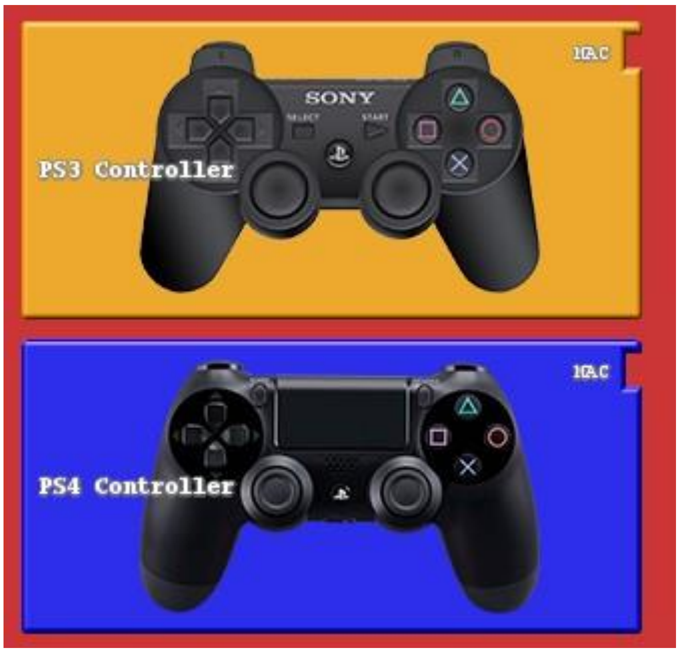
Note: PS3 controller **will not** work with PS4 code and vice-versa.

To change the MAC address in Ardublock, simply open the Arduino IDE, go to Tools > ArduBlock.


sketch_jun16a | Arduino 1.8.19
Ficheiro Editar Rascunho Ferramentas Ajuda
sketch_jun16a
Formatar Automaticamente Ctrl+T
Arquivar Rascunho
Corrigir Codificação & Recarregar
Gerir Bibliotecas... Ctrl+Shift+I
Monitor Série Ctrl+Shift+M
Plotter Série Ctrl+Shift+L
ESP Exception Decoder
WiFi101 / WiFiNINA Firmware Updater
ArduBlock
Placa: "ESP32 Dev Module"

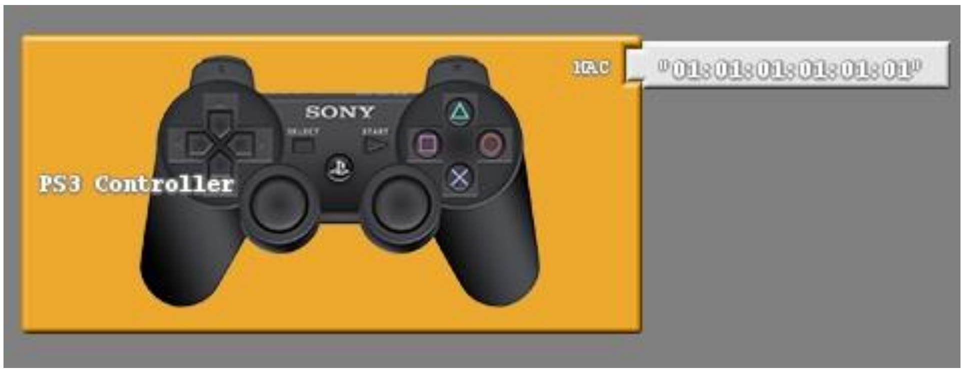
```
void setup(
  // put yo

}

void loop()
  // put yo

}
```

Next, you will be presented with the block program and at the bottom of the menu on the left side, there is a family of blocks called "Dualshock", and open that one. After opening it, you can see that there are two blocks: one for the PS3 controller and one for the PS4 controller.



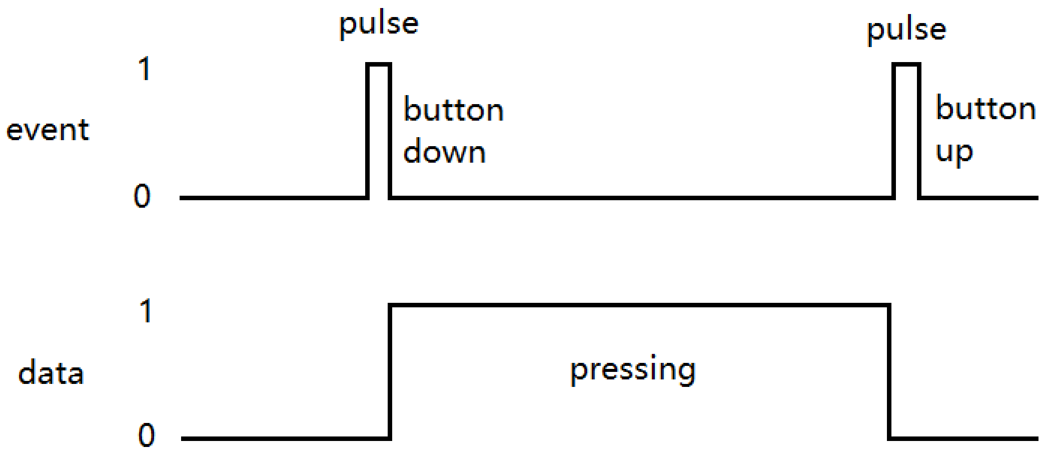
These blocks are very similar and the procedure is the same for both. With this, drag the one you want to use into the gray area of the program, and modify the MAC address so that it is identical to that of your controller. This address will need to have quotation marks at the beginning and end of the address. If it is not entered with quotation marks, the program will not recognize the address as a String and therefore will not work. It is important to know that the controller blocks do not have

the quotes by default.
The address in the block must be identical to the following example:



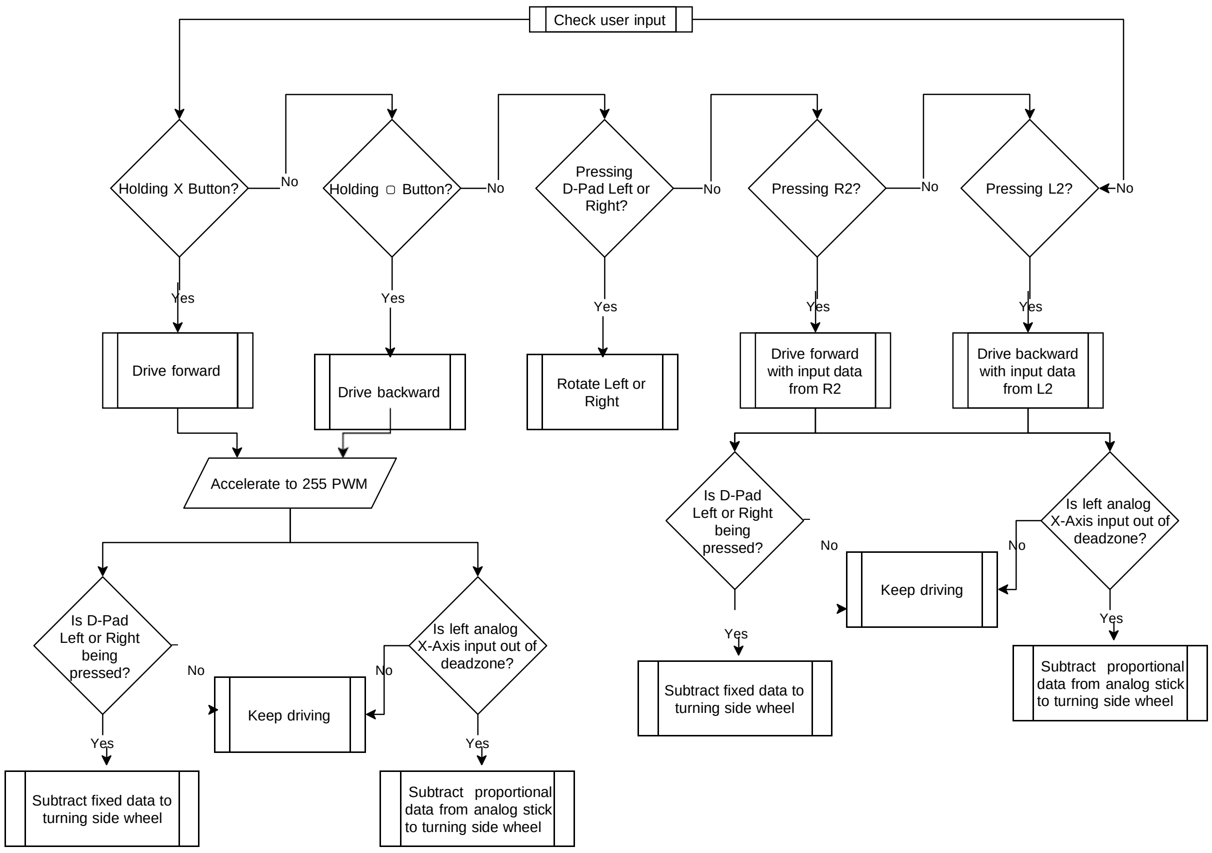
Operation

Both libraries are quite different in 1 aspect, that being the events section. When the user presses a button, this will trigger an event called `button_down` .This acts as a pulse which lasts a loop cycle. Whilst the button is pressed, this is returning a `true` value in `data` section. After the user stops pressing, a new event called `button_up` will trigger a pulse that also lasts a loop cycle. Out of both libraries, only PS3 has events, while PS4 is currently still on the works.



These events are very useful as they allow us to identify when a button is no longer pressed or when we want to change the state of a variable.

The following image is a block diagram of the microcontroller operation for the "drive()" function.


Check user input
Holding X Button?
No
Holding ▢ Button?
No
Pressing D-Pad Left or Right?
No
Pressing R2?
No
Pressing L2?
No
Yes
Yes
Yes
Yes
Yes
Drive forward
Drive backward
Rotate Left or Right
Drive forward with input data from R2
Drive backward with input data from L2
Accelerate to 255 PWM
Is D-Pad Left or Right being pressed?
No
Keep driving
No
Is left analog X-Axis input out of deadzone?
Yes
Yes
Subtract fixed data to turning side wheel
Subtract proportional data from analog stick to turning side wheel
Is D-Pad Left or Right being pressed?
No
Keep driving
No
Is left analog X-Axis input out of deadzone?
Yes
Yes
Subtract fixed data to turning side wheel
Subtract proportional data from analog stick to turning side wheel