

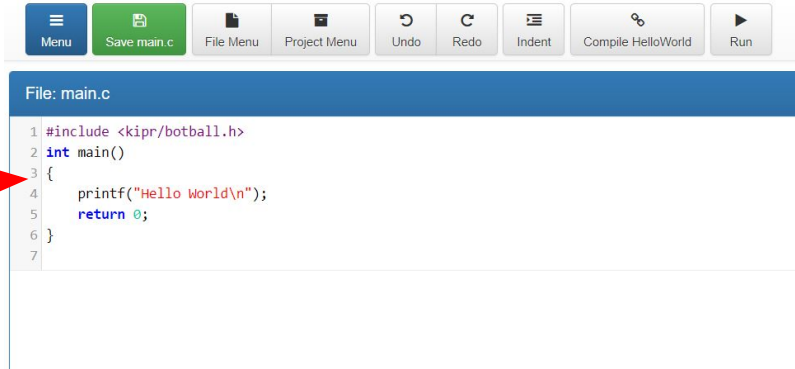
# What is a library?

A library is a collection of precompiled functions that can store your frequently used routines.

The KIPR library is used in each program that you use. You don't have access to alter the KIPR library, but you can create your own library to really help you organize your code.

Ever notice the first line of any new “Hello World” program?

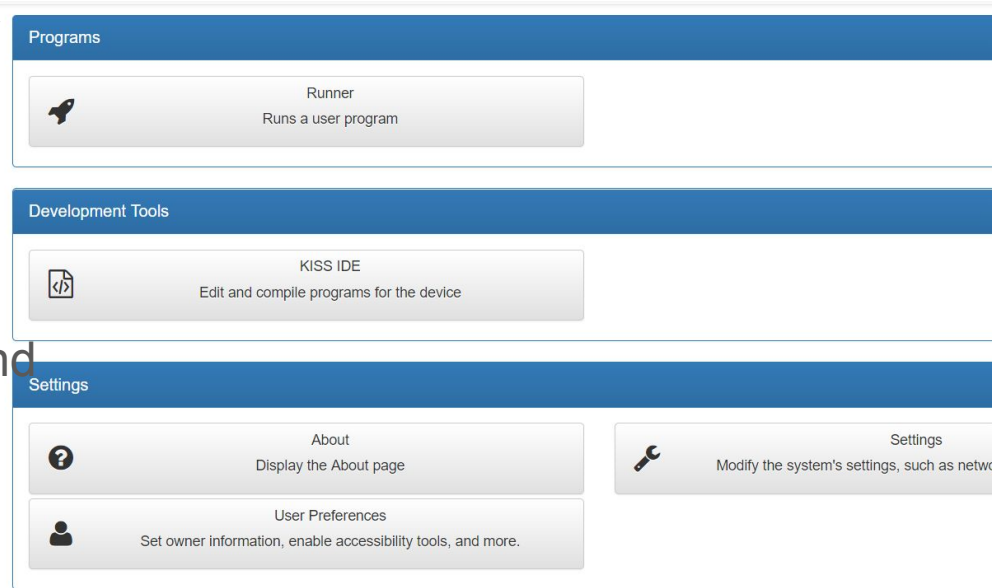
This is how we “include libraries.



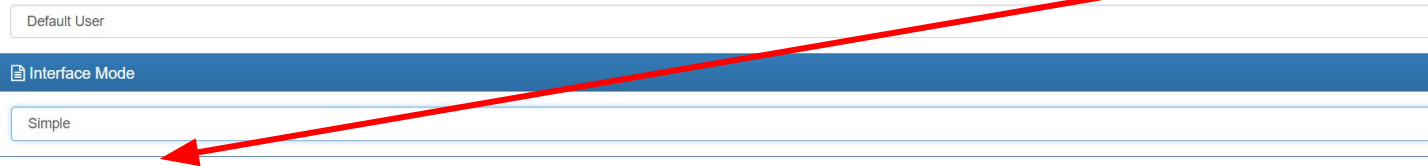
```
File: main.c
1 #include <kipr/botball.h>
2 int main()
3 {
4     printf("Hello World\n");
5     return 0;
6 }
7
```

# Creating Your Own Libraries Within the KIPR Suite

In order to create your own library, you will need to access the KIPR advanced user interface. Do this by going to this screen and clicking the “User Preferences” button.

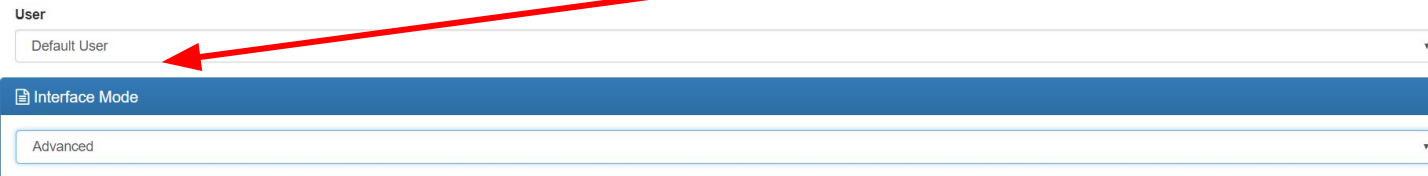


# Then you have to click on the “Simple” dropdown.



A screenshot of a web interface. At the top, there is a dropdown menu labeled 'Default User' with a downward arrow. Below it is a blue horizontal bar with a document icon and the text 'Interface Mode'. Underneath the bar is a dropdown menu currently showing 'Simple' with a downward arrow. A red arrow points from the top right towards the 'Simple' dropdown menu.

Change It to “Advanced”. Please note, the “Default User” which can be altered to your individual users at this time through the dropdown.

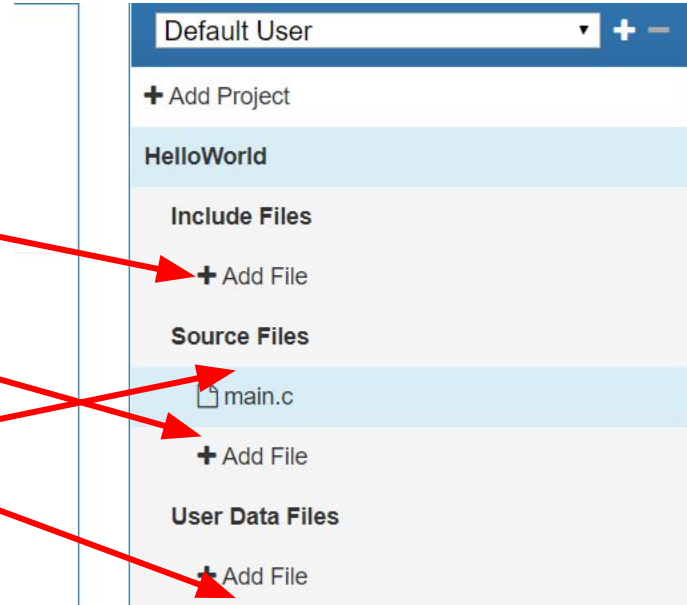


A screenshot of a web interface. At the top, there is a dropdown menu labeled 'User' with 'Default User' selected and a downward arrow. Below it is a blue horizontal bar with a document icon and the text 'Interface Mode'. Underneath the bar is a dropdown menu currently showing 'Advanced' with a downward arrow. A red arrow points from the top right towards the 'Default User' dropdown menu.

# Now go back into a Hello World program

Look over on the right side, now you will see “Include Files” and “Source Files”.

The “Add File” tabs will allow you to add Source files and Header files.



# Let's Create a Header File

After you click the “Add File” under the “Include Files”, you will be able to name your new (.h) file.

We have called this one “Drive”, but you can call it anything you want.

Create New Include File ×

---

Create a New File:

or

Upload an Existing File:

No file chosen

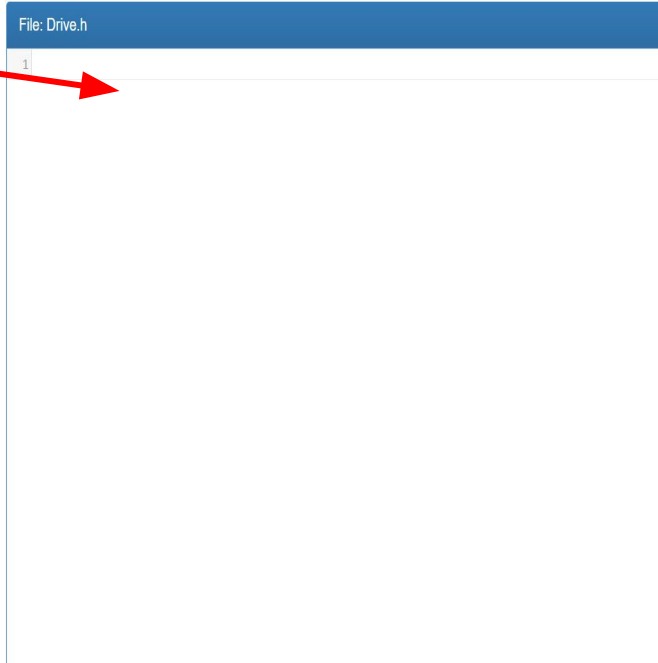
# A Library needs books...

Your brand new header file is blank.

What is a header file? A header file is a place to store function declarations, variables, and constants.

What can we put inside? We can store any variables that we want to define or the names of functions that we plan on defining in our source files.

How do we use it in our main program? You will need to include this file as an include in your source files (like your main)



# Let's Add A Line That Will Help Always Define a Word

Let's put in a variable that will always be the same. We will “define” this. Think about this as a constant.

Notice the syntax: There is a # before the word define, then a space, the constant name, a space and a value. (no semicolon)

Anytime you use “RTmotor” in a source file, your computer will see the number 0.  
**HIT THE SAVE BUTTON!**

File: Drive.h

```
1 #define RTmotor 0
```

This will allow you to remove the number associated with a port and instead write a mav function as follows:

```
mav(RTmotor, 500);
```

If you move the motor to another port, just change the value in this header file. Save a lot of time by changing it one time here, as compared to changing it everytime it appears in your main program.



# More Books...

Now, let's add a define for the left motor and declare that we are going to make a function for driving forward. (Wouldn't that be useful)

Functions or variables can be (or should be) defined or declared in this header file.(.h)

Hint: You can have multiple .h files that can contain needed information for different purposes. For instance you could call a .h file movement, and another one servo, etc.

This is a huge organizational strategy!

File: Drive.h

```
1 #define RTmotor 0
2 #define LTmotor 2
3
4 void drive_forward();
```

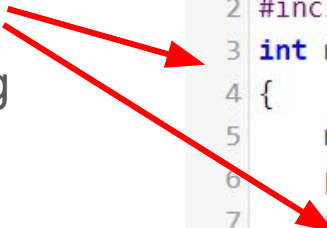
# Making the Include File Available In Your Main Program

Click back on your main.c file on the right side and add the following two lines of code: (change the word “Drive” if you named it something differently)

The include line (line 2) is how you include a library that is in this project folder.

Line 5 is using the constant that you defined in your header file.

```
File: main.c
1 #include <kipr/botball.h>
2 #include <Drive.h>
3 int main()
4 {
5     mav(RTmotor, 500);
6     printf("Hello World\n");
7     return 0;
8 }
9
```

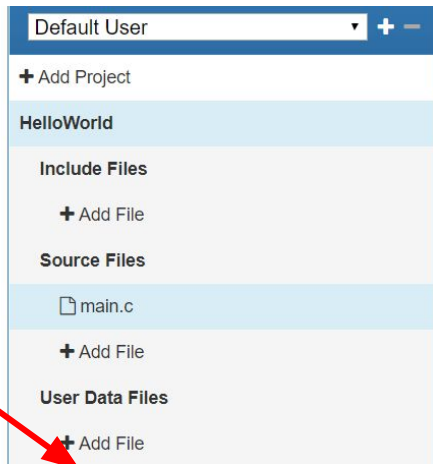


# Creating A Source File

A source file on the Wallaby can be added to hold all of your functions. (To further increase your ability to organize your code.

You can create a source file by hitting the “add file” button under “Source File” on the right hand side.

Go ahead and create a .c file to hold some of your custom functions...we will start by defining the `drive_forward()` function that was declared in the previous .h file.



# Adding Some Custom Functions

Upon creating your new .c file you should open up to a blank page. Go ahead and add the following lines of code so that this source file will have the benefits of the KIPR library as well as your custom created .h file.

File: Drive.c

```
1 #include <kipr/botball.h>
2 #include <Drive.h>
3 |
```

After you have done this, see if you can create your own function named `drive_forward()` . Give it a shot! A possible solution is on the next page.

# Making Your Source File - Maximizing Organization

So, after adding a forward driving function, your new source folder may look like this.

You can add the definition of many of your functions in these source files.

File: Drive.c

```
1 #include <kipr/botball.h>
2 #include <Drive.h>
3
4 void drive_forward()
5 {
6     mav(RTmotor, 500);
7     mav(LTmotor, 500);
8 }
9
```

# Remember and Recall...

Your source (.h) files are where your variables and functions are declared.

Your created source (.c) files are where your functions are defined.

Your `main.c` file (your main source file) is where you will use the functions, variables, and constants that you have built. It will be much better organized with proper use of header(.h) and source(.c) files.

File: main.c

```
1 #include <kipr/botball.h>
2 #include <Drive.h>
3 int main()
4 {
5     printf("Hello World\n");
6     drive_forward();
7     msleep(1000);
8     return 0;
9 }
```

# Possible Pitfalls

You can only define a variable or function 1 time. Trying to do this more than once will cause a very complex compiler error message.

You can use some of your custom functions inside of other custom functions, but the order is important in which they appear inside of your files. For instance if a function is inside of another function, the function that is inside (nested) must appear above it in the source file.

If you are trying to use a source or header file that is inside of another program folder, you will have to give the full path.(available on the file menu screen) and put quotes(“ ”) around it instead of carrots(< >). Example:

```
#include “/home/root/Documents/KISS/Default User/HelloWorld/include/Drive.h”
```