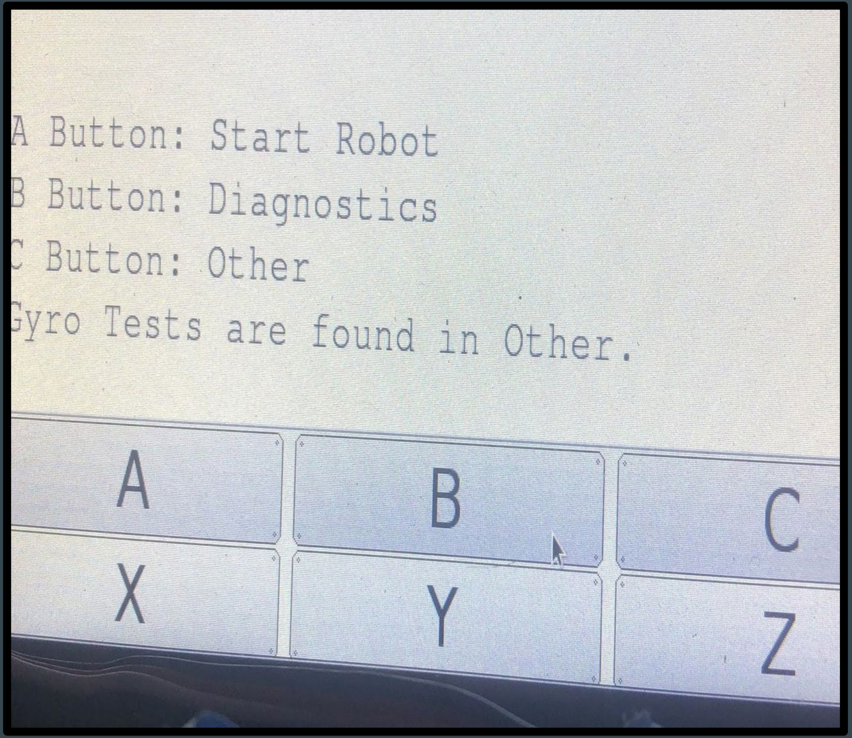
Rethinking Your Tournament UI

Carleigh Wilcox
Noble High School


```
A Button: Start Robot
B Button: Diagnostics
C Button: Other
Gyro Tests are found in Other.
```

Introduction

When executing a program on the Wallaby or Wombat, the buttons provided on screen can be turned into a user interface.

KIPR's library provides functions that allow users to interact with the buttons in code.

Tasks of the User Interface

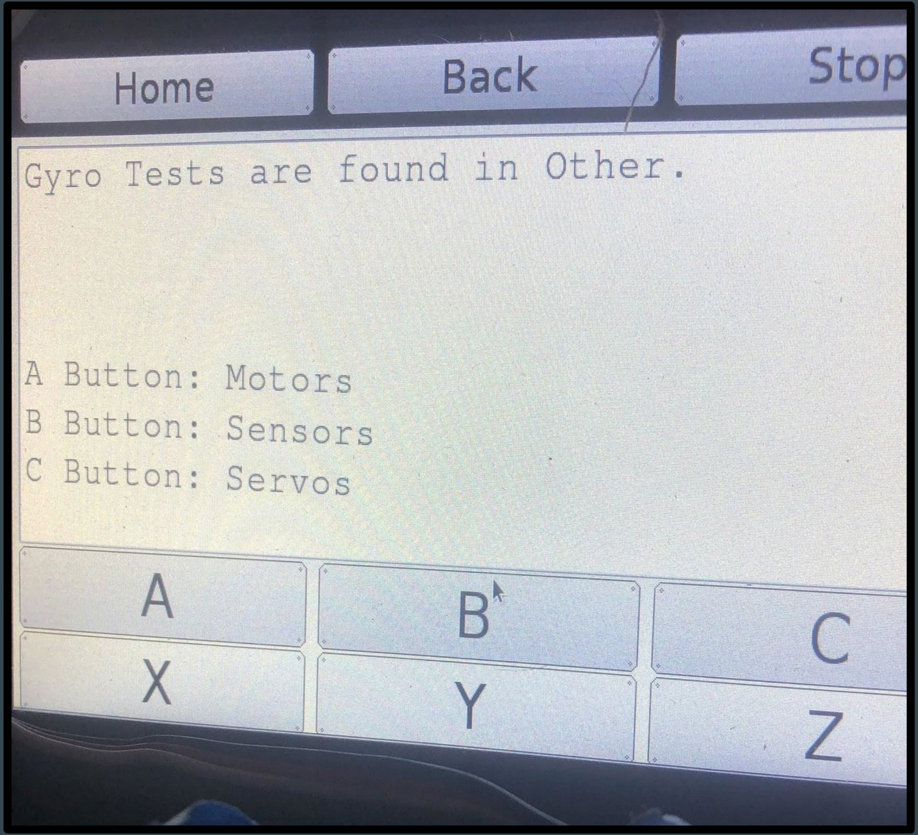
1. Change the outcome of the run
   a. Have the user click a button, each with a different value, at the beginning of the run. The value will be used in code to determine which run function to execute using a switch statement. .
2. Start the robot.
   a. One button may start the robot immediately, and another will start waiting until it detects light to start the robot.
3. Check diagnostics.
   a. Sensor readings, servo movements, and other diagnostics can be checked with a button.

The Design Basics

The user interface is comprised of chains of different menus:

1. Boot menu
2. Main menu
3. Sub-menus

Text printed on the screen tells the user which button does what.



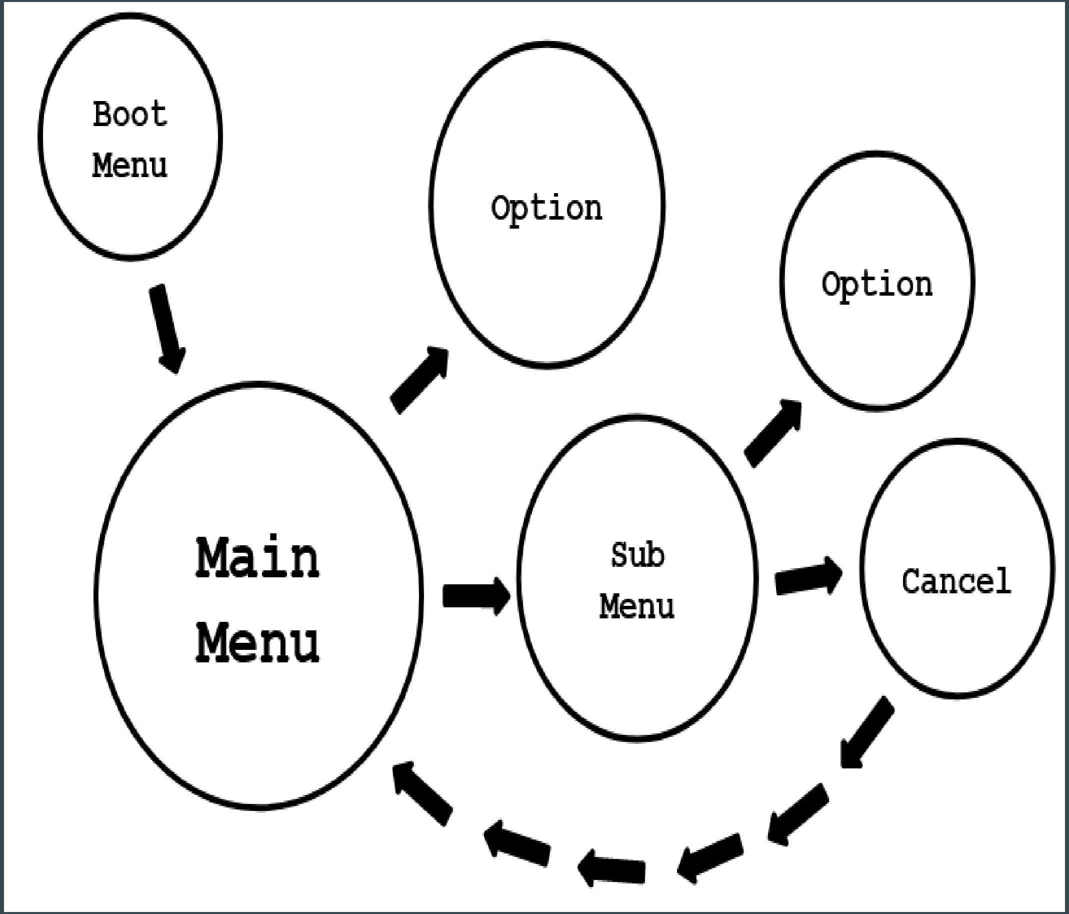
```
Gyro Tests are found in Other.



A Button: Motors
B Button: Sensors
C Button: Servos
```

Menu Design

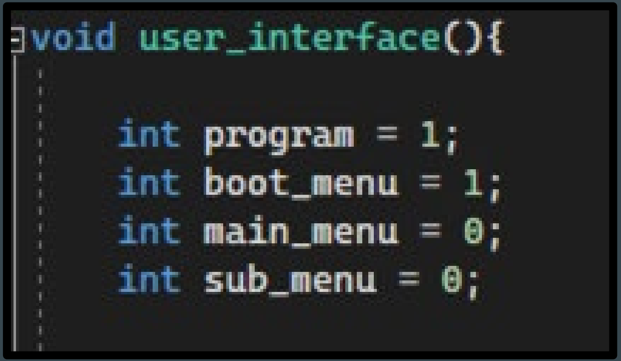
1. T**he boot menu**is optional and is shown on startup.
2. The **main menu**is the “hub,” it usually contains other menus.
3. **Options** are**actions**executed after a button push. They may or may not return to a menu.
4. **Sub-menus**are inside of other menus. All of them include a cancel button to return to the main menu.



Technical Design - True or False System

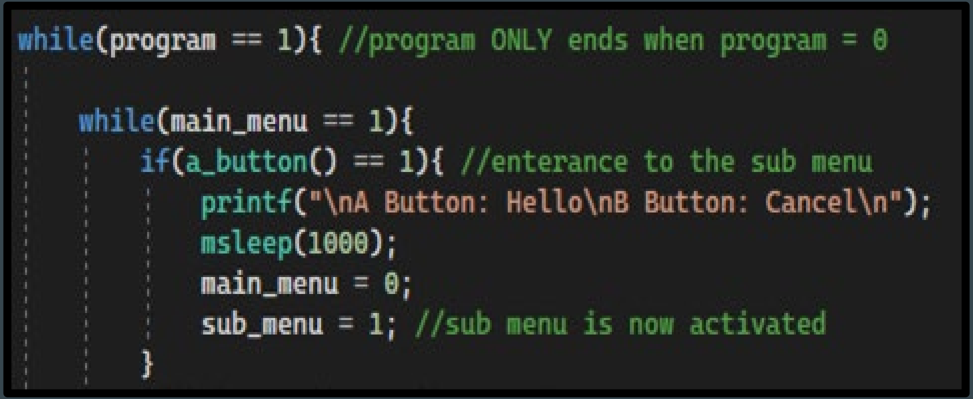
A variable is assigned to every menu at the beginning of the program. That variable equals “1” if the menu is active, and “0” if it is not. This variable is used as a condition in its respective while-loop. Also, when switching menus, the current menu’s variable is set to “0,” and the next menu’s variable is set to “1.”

Outside of the main and sub-menus, there is a while-loop containing a variable that keeps the program continuously running.



```
void user_interface(){

    int program = 1;
    int boot_menu = 1;
    int main_menu = 0;
    int sub_menu = 0;
```

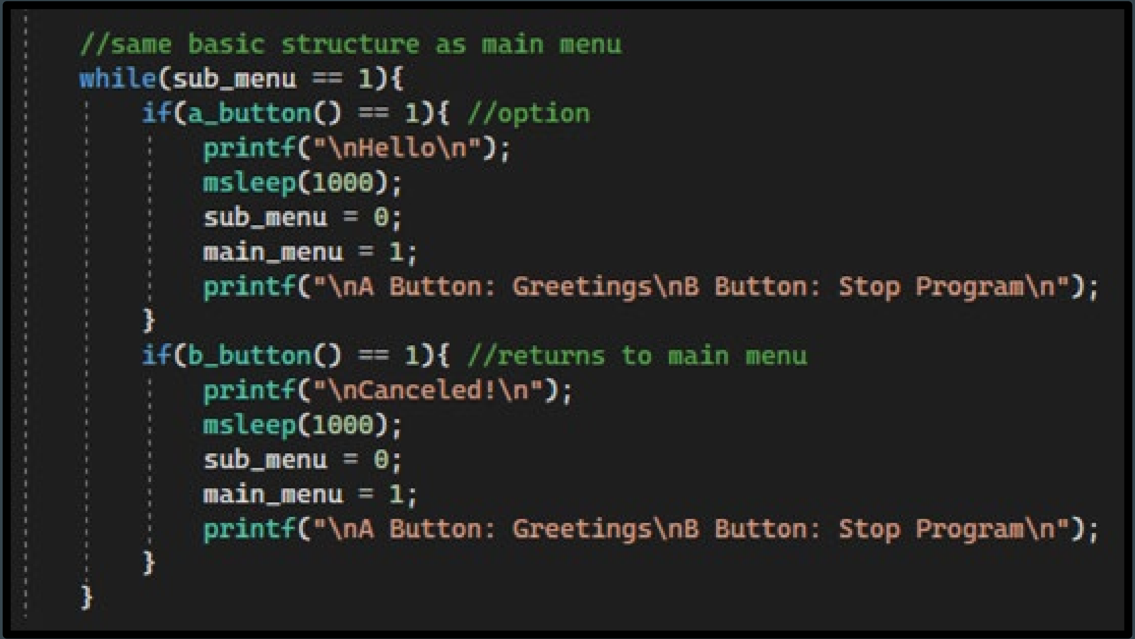


```
while(program == 1){ //program ONLY ends when program = 0

    while(main_menu == 1){
        if(a_button() == 1){ //enterance to the sub menu
            printf("\nA Button: Hello\nB Button: Cancel\n");
            msleep(1000);
            main_menu = 0;
            sub_menu = 1; //sub menu is now activated
        }
```

Technical Design - If Button is Pressed

During each menu's while loop are if-statements that check if certain buttons are pressed.



```
//same basic structure as main menu
while(sub_menu == 1){
    if(a_button() == 1){ //option
        printf("\nHello\n");
        msleep(1000);
        sub_menu = 0;
        main_menu = 1;
        printf("\nA Button: Greetings\nB Button: Stop Program\n");
    }
    if(b_button() == 1){ //returns to main menu
        printf("\nCanceled!\n");
        msleep(1000);
        sub_menu = 0;
        main_menu = 1;
        printf("\nA Button: Greetings\nB Button: Stop Program\n");
    }
}
```

Technical Design - Small Basics

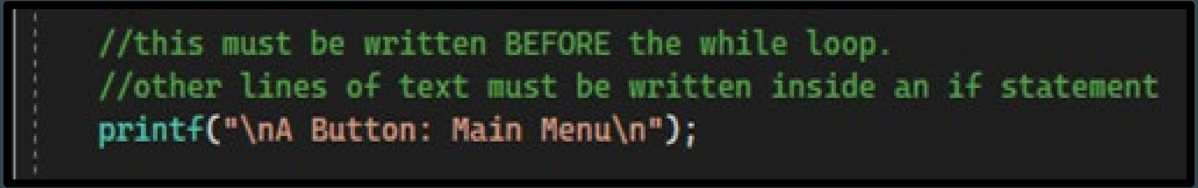
Printing

Print-statements should be written *before*the next menu is accessed.

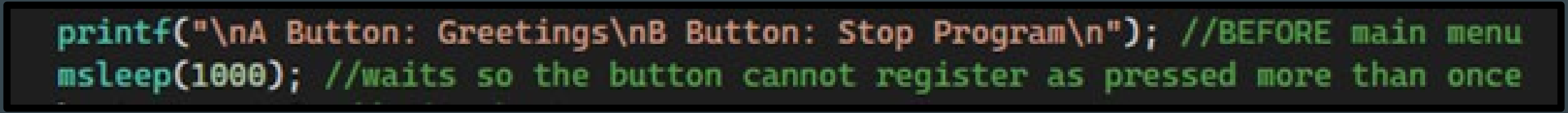
“/n” should be written before and after each line.

Waiting

After each button press there should be an “msleep” to ensure the button is not pressed more than once.



```
//this must be written BEFORE the while loop.
//other lines of text must be written inside an if statement
printf("\nA Button: Main Menu\n");
```



```
printf("\nA Button: Greetings\nB Button: Stop Program\n"); //BEFORE main menu
msleep(1000); //waits so the button cannot register as pressed more than once
```

Conclusion

This new user interface is a worthy inclusion for any tournament bot. It negates the need for a computer to communicate with the bot, and it holds an endless amount of potential for programmers to take advantage of.