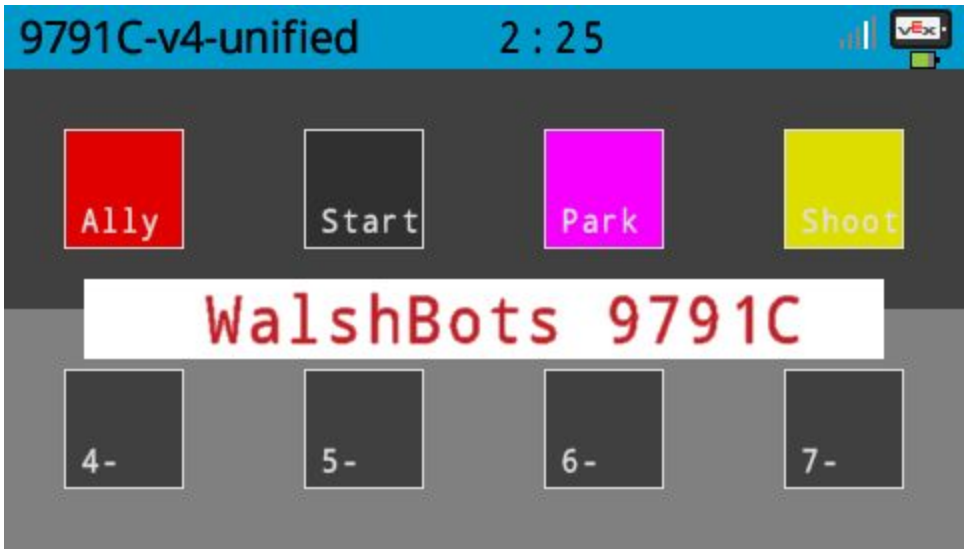
January 5, 2019 - Pascal Chesnais - Walsh Middle School

WalshBots
Autonomous Feature Selector

Introduction

With V5 Robot Brain you can load up to eight programs, which is nice as you can load different programs for different scenarios, such as which alliance are you on and which starting tiles. While this is good for new teams getting started, you have to maintain up to eight different programs each time something about the robot changes. James Pearman, VP of Engineering at RobotMatters, posted an example program on vexforum.com for using the V5 Robot Brain to select different autonomous routines by pressing square buttons on the V5 Robot Brain. In his example, he used 8 buttons to select which routine would be followed. The advantage is that you can maintain all 8 routines in one single program. The downside is you are limited to only one routine at a time. At Walsh, we modified Mr. Pearman's code such that you could control robot features/capabilities, for example, parking, shooting preload, as well as, control which position you are on the field by selecting the color alliance and starting tile (near or far from flags). This opens the possibility to have have many more than eight routines (in fact there are 256 possibilities with 8 on or off buttons). From the driver perspective, it is easy to understand, you are setting what capabilities you wish to enable depending on your alliance strategy.

Screen interface



There are eight positions on the screen. In the 9791 competition template we have four defined:

- Ally - set which alliance you are on Red or Blue.
- Start - defines the starting tile Black is near the net/flags and white is the far tile.

- Park - determine if the robot will be parking or not.
- Shoot - determines if the preload will be shot or not.

Using button results in the program

Once the buttons are selected, they set variables in the program to true or false. Then you can test those variables in your autonomous routine to adapt to your match needs.

You first declare and initialize the variables at the beginning of the autonomous routine:

```
void autonomous( void ) {

    /* initialize capabilities from buttons */
    bool allianceBlue = buttons[0].state;
    bool startTileFar = buttons[1].state;
    bool doPark = buttons[2].state;
    bool shootPreload = buttons[3].state;




   /* lower flag bumping code - only if near flag start tile */
    if(!startTileFar){            // Starting tile nearest to the
flags/net
        // STEP 1
        // if preload needs to be shot in autonomous, do it now
        if(shootPreload){
            shootPuncher();
            Brain.Screen.printAt( 60,  125, "Shooting Preload
" );
            if(allianceBlue){
                driveTurnRightDegrees(7); // need to have robot
facing more to the left
            }
        }
        // STEP 2
        // drive into first low flag
        driveDistance(50.0);
…..
```

Setting up buttons in your program

Button support definition

Global definition for button structure, this will need to be at the top of your program.

```
typedef struct _button {
    int    xpos;
    int    ypos;
    int    width;
    int    height;
    bool   state;
    vex::color offColor;
    vex::color onColor;
    const char *label;
} button;
```

Specific button definition for your needs

```
// Button array definitions for each software button. The purpose of
each button data structure
// is defined above.  The array size can be extended, so you can have
as many buttons as you
// wish as long as it fits.
button buttons[] = {
    {   30,  30, 60, 60,  false, 0xE00000, 0x0000E0, "Ally" },
    {  150,  30, 60, 60,  false, 0x303030, 0xD0D0D0, "Start" },
    {  270,  30, 60, 60,  false, 0x303030, 0xF700FF, "Park" },
    {  390,  30, 60, 60,  false, 0x303030, 0xDDDD00, "Shoot" },
    {   30, 150, 60, 60,  false, 0x404040, 0xC0C0C0, "4-" },
    {  150, 150, 60, 60,  false, 0x404040, 0xC0C0C0, "5-" },
    {  270, 150, 60, 60,  false, 0x404040, 0xC0C0C0, "6-" },
    {  390, 150, 60, 60,  false, 0x404040, 0xC0C0C0, "7-" }
};
```

As you can see we set up our Ally button to have the color RED when false and BLUE when true. All buttons have an off and on color to allow the driver to have visual confirmation of the state of the selection.

Final thoughts

From the driver's perspective this is a neat way to think about autonomous selection. There is the drawback that V5 Robot Brain screen may be difficult to reach on the field.

Many thanks to James Pearman for getting the button example on the forum. He tirelessly contributes useful know how to the community. We are glad to give back.

The example code 9791-button-template.vex also has some drive abstractions which helped our teams program their autonomous. Probably some bugs - let us know if there are improvements to be made.