PID Controller Tuning GUI

Wang Feixuan

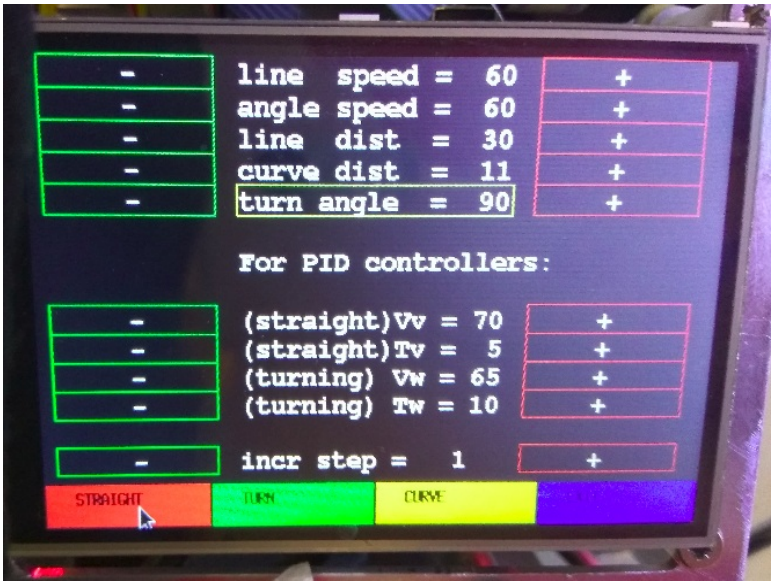
24/08/2016

1. Introduction

This program provides a graphical user interface for PI(D) controller parameter tuning on eyebot with raspberry pi3(with some modifications on the initial image). It calls RoBIOS-7 Library Functions.

2. GUI

Below is the graphical user interface for the demo(It looks very different on the remote desktop so I just took some pictures of the screen).



2.1. Parametes

```
int VWStraight(int dist, int lin_speed);        // Drive straight, dist [mm], lin. speed [mm/s]
int VWTurn(int angle, int ang_speed);           // Turn on spot, angle [rad/1000], ang. speed [(rad/100)/s]
int VWCurve(int dist, int angle, int lin_speed); // Drive Curve, dist [mm], angle (orientation change) [rad/100], lin. speed [mm/s]
```

The units of the parameters are:

parameter	unit
line speed	mm/s
angle speed	mm/s
line distance	cm
curve distance	cm

turn angle	degrees

And the calling codes are:

```
VWStraight(lin_dist * 10, lin_speed);
VWTurn((int) (turn_ang * 31.4 / 1.8), ang_speed);
VWCurve(curve_dist * 10, (int) (turn_ang * 3.14 / 1.8), lin_speed);
```

The next four parameters are for PI controllers. This V-Omega function is used:

```
int VWControl(int Vv, int Tv, int Vw, int Tw);  // Set PI params. for v and w; typical:  VWControl(9,5,6,1);
```

The final parameter **step** is the increment/decrement of every operation.

2.2. Keys

Four main keys **STRAIGHT**, **TURN**, **CURVE** and **EXIT**are created using eyebot function `int LCDMenu(char *st1, char *st2, char *st3, char *st4); // Set menu entries for soft buttons` .

The function `int KEYGet(void); // Blocking read (and wait) for key press (returns KEY1..KEY4)` is used to get key in while(true) loop, which will only be broken by exiting the whole program by when the **EXIT** button is pressed.

2.3. Buttons

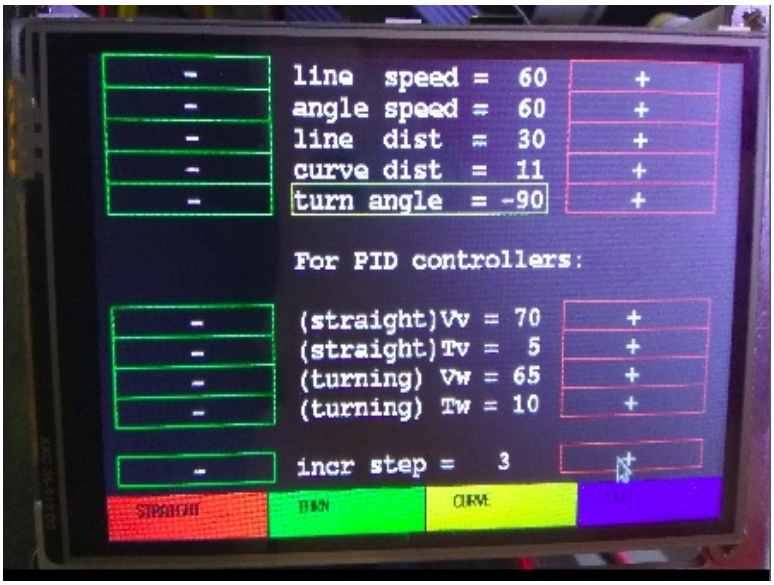
For each parameter two buttons are created to decrese/increase the value. Note that for `VWTurn()` and `VWCurve` angles can be above or below zero for anti-clockwise and clockwise. The two buttons for `turn angle` can only change its absolute value and the yellow button for itself can change its sign. Other values can only be `>= 0` .

The buttons are created with `LCDArea()` for the boxes and `LCDSetPrintf()` for the marks. As I'm not quite sure about how these functions are realized and fonts can look very different on differernt hardwares, the parameters for these functions are tested specially for this raspberry pi3.

The buttons are detected with the function

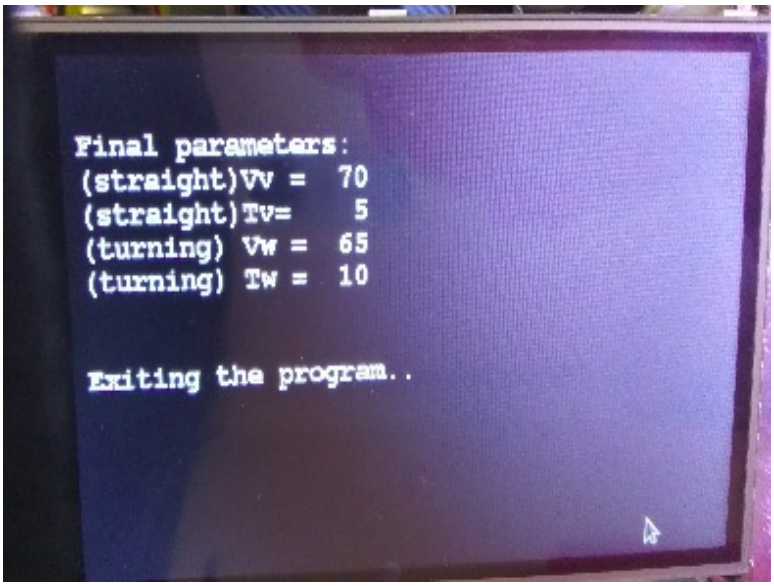
```
int KEYGetXY (int *x, int *y);  // Blocking read for touch at any position, returns coordinates
```

The function `KEY_Decode()` is used to decode the buttons according to their position(`Point(x, y)` and its row, col, and something else). A lot of values are *defined* to make this decoding simpler.



2.4. Exit

The program is (and can only be on the raspberry pi only) exited when the **EXIT** button is pressed. When this happens, the current four PID controller parameters shall be printed on the screen and will be written to `PID_tuning.parameters` for next test. All the parameters and the current time shall be recorded in `PID_controller.log`.



3. Compile and run

I installed `clang` and created a command `clangarm` just like `gccarm` as a lot of things are not supported in the low version gcc provided by raspberry pi3.

The files `control.c` and `PID_tuning.parameters` are located in /home/pi/usr/software/control and so should the `PID_tuning.log` be. Note that if `PID_tuning.parameters` is not found the default values of the parameters in the program are used. Compile

```
clangarm control.c -o ../control.o
```

and run the program in `software` on the pi.

4. TODO

More detailed descriptions and contact information are written in the code and parameters for the GUI may need changing when implemented on other hardware.