Gyro-Guided Skid Steer (GYGS)

Vehicle turns and goes in the commanded direction at the commanded speed

Can be used with any skid-steer vehicle with separate control of right and left wheel speeds (eg 6WD).

Driver interface is single 2-axis joystick.

Numbers in red are tuning constants. I have shown them here as numbers rather than symbols for ease of reading. In actual practice, they should be variables which can easily be adjusted for best results.

```
speed_command = max(abs(Xj),abs(Yj));
```

Xj, Yj are the outputs of a single 2-axis joystick.

```
direction_command = atan2(Xj,-Yj)*(180/pi);
```

"direction_command" is the angle of the joystick in degrees CLOCKWISE from 12 o'clock.

Warning to Excel users: atan2() is per "standard" definition http://en.wikipedia.org/wiki/Atan2#Definition

Find the shortest forward-driving rotation (either CW or CCW) required to get to the desired angle. Note: The robot should be placed facing "straight downfield" at the start of the competition so that "gyro_angle" initializes to zero at that position when powered up. Gyro_angle should be CLOCKWISE.

```
angle_error = direction_command - gyro_angle;
```

```
while(angle_error>180)angle_error-=360; [see endnote 1]
```

```
while(angle_error<-180)angle_error+=360; [see endnote 1]
```

Allow the robot to back up instead of going forward if the forward-turning angle error exceeds 100 degrees. This would be the normal desired operation. However, there will be cases where the driver wants the robot to turn through the larger angle and drive forward. When pressed, button1 overrides the "backup" feature and forces the robot to turn around and seek the forward direction:

```
if(!button1){
     if(angle_error>100)
     {angle_error-=180; speed_command = -speed_command;}

     else if(angle_error<-100)
     {angle_error+=180; speed_command = -speed_command;}
}
```

Form the rotation rate command. Full power rotation with no forward motion occurs for angle errors equal to or greater than 90 (=2*45) degrees:

```
clockwise_command = angle_error/45;
```

Add a deadband to prevent unwanted rotation commands. This deadband should be greater than any null bias in your joystick:

```
if(abs(speed_command)<0.1)clockwise_command=0;
```

Add a button to revert to non-gyro "Arcade" mode. If gyro signal becomes corrupted this allows the driver to continue to control the robot. It also allows the driver to manually rotate the robot so that the gyro can be re-zeroed during competition:

```
if(button2){speed_command=-Yj; clockwise_command=Xj;}
```

Add a button to re-zero the gyro:

```
if(button3)re_zero_gyro();
```

Calculate the left and right wheel speeds:

```
left_wheel_speed  = speed_command + clockwise_command;
right_wheel_speed = speed_command - clockwise_command;
```

Limit the wheel speed commands to +/-1:

```
if(left_wheel_speed>1) left_wheel_speed = 1;
else if(left_wheel_speed<-1) left_wheel_speed = -1;
if(right_wheel_speed>1) right_wheel_speed = 1;
else if(right_wheel_speed<-1) right_wheel_speed = -1;
```

Send the above wheel speed commands to the left and right motors.

Endnotes:

[1] Instead of the while loops, in C you can use angle_error -= 360*floor(0.5+angle_error/360);

In LabVIEW you can use the following: