UWA Digital Embedded Systems Lab Prep 7:
EyeBot 7 - Driving

October 2018 - Marcus Pham

Revision: Eyebot7 Usage

To use the EyeBot7 library you will have to include the header, “**eyebot.h**”, to your C source file.
To compile, you must compile using **gccarm**, followed by the standard c compiler options, eg –o <filename>, -Werror, -Wall, -pedantic.
You however **will not** have to specify the standard (eg. std=c99/std=gnu99).

Running the compiled program is as simple as typing: **./<programname>** in the folder containing the program.

Remember to collect **1x Car, 2x Batteries and 1x Charger** from the front desk before the lab. Charge one pack whilst using the other.

When turning off, Press **SYSTEM -> ADMIN -> SHUTDOWN** to shutdown then use switch on side to turn off once screen turns white.

Write all your code in **/home/pi/usr/** to be able to access them the screen.

Basic command line usage

a. **cd <directory>** – change directory
b. **ls** – list contents of directory
c. **cp <filename> <location>** - copy file to desired location
d. **mv <filename> <location>** - move/overwrite file
e. **nano <filename>** - a basic text editor for command line, to exit/save <ctrl+x>
f. **pkill <processname>** - kills a process/program
g. **<ctrl+c>** - kills the currently running process on that command line terminal
h. **mkdir <dirname>** - makes a new directory

Position Sensing Device (PSD)

The Eyebot is equipped with 3x PSDs by default.

The arrangement is:
PSD1 - Centre
PSD2 - Left
PSD3 - Right

To use the PSD sensors, you are able to call **PSDGetRaw(<PSD no.>)** to get an integer representative of the distance away from the PSD. The farther away from the device, the smaller the value.

To tune the PSD for the correct values, you can access the hdt (hardware decription table) file at **/home/pi/eyebot/bin/hdt.txt.** This file contains settings and tables for tuning the Eyebot.

To tune the PSD sensor, by using a basic look-up table, take readings from the PSD at various distances.
Each entry on the table linearly scales to the range of the PSD sensor (12bit resolution, max = 4096), and the default size of the table is 128.
Hence, to tune, you will approximately increase the value the the PSD sensor provides by 32 (4096/128) and record the real distance at that position.

The quickest method is to either write your own code to output the current PSD readings in real time, otherwise you can use the inbuilt hardware code, via HARDWARE->PSDs->PSD<no.>

You are able to make multiple tables for each PSD, however you will need to link the PSD to the corresponding table in the hdt.txt file.

Remember to save afterwards.

Once tuned, you can use the function PSDGet(<PSD no.>) to obtain the real world distance (in mm).

V Omega Driving control

To drive the car, there are already inbuilt low-level functions that implement PID control on the motors.

There are various functions available for simple driving.

Straight:

VWStraight(int dist, int lin_speed);
Drive straight, dist [mm], lin. speed [mm/s]
eg. VWStraight(100, 100); → 100mm @ 100mm/s

Turning:

VWTurn(int angle, int ang_speed);
Turn on spot, angle [degrees], ang. speed [degrees/s]
eg. VWTurn(180, 360); → 180o @ 1 rev/sec aprox.

Curve:

VWCurve(int dist, int angle, int lin_speed);
Drive in a Curve, dist [mm], angle [degrees], lin. speed [mm/s]
eg. VWCurve(100, 20, 100) → Drive 10cm of a curve of 1 rad angle @ 10cm/s

- **Hint: These functions overwrite any other previous VW function when called.**

Completion:

VWDriveDone()
Can be used as a flag in order to know if the car has completed a VW task or not.

- Hint: VWDriveDone returns 1 when VW is not used. Therefore in order to use this function properly it is better first to create a loop waiting for VWDriveDone to be 0 and then another loop waiting to be 1.

There are many other functions available to you to use on the Eyebot7, they are all available on the API at: http://robotics.ee.uwa.edu.au/eyebot7/doxygen/html/