Vehicle Interface Controller (VIC)

Lewis Collum

Contents

1 Overview

We are using an Arduino Uno to take in i2c steering and drive values to control the vehicle's steering servo and electronic speed controller (ESC). We refer to the Arduino as the Vehicle Interface Controller (VIC).

The VIC is programmed in C++ (not using Arduino's packages). This gives us control over the implementation details of utilities like strings, pwm, usart, and i2c. Note that this decision was made for our own academic curiousity.

2 Application Makefiles

The VIC uses GNU Makefiles. We have modularized the makefile process, such that when starting a new application file (i.e. a source file which contains `main`) the makefile to build the application only needs to contain

```
TARGET = <name of your application>
ROOT = ../..
include $(ROOT)/mk/all.mk
```

where `ROOT` is the directory that contains the `mk` directory.

3 Making a New Application

We have included a bash script in `app` called `createNewApp`. To make a new application, we can go to the app directory and run

```
./createNewApp <name of application>
```

This script ensures there are no existing applications and if that is true:

1) will make a directory with the same name as our application;
2) will make a `cpp` source file with the name as our application;
3) will create a Makefile, filled with everything necessary to build our application.

4 Flashing the Arduino

To flash the Arduino, we go to the directory of an application (in the `app` directory), and then we run `make flash`. The build files are exported to the `app/build` directory.

5 Serial Interface

To communicate with the Arduino, via Serial, we use `picocom`, a linux terminal utility. As part of our modular makefile system, we can open `picocom` by going to the directory of an application (in the `app` directory), and running `make com`.