**Recommended reading: ‘getting started\_2012’** to understand the system modules and what each is responsible for and how they interact.

**INSTALLING LINUX MINT (DUAL BOOT)**

Loading code onto the boards is done using OpenOCD and the department's USB-JTAG adapter. This can be done under both Windows and Linux using the command line. However when it comes to debugging using usb\_cdc and DEBUG modules in the code, Linux was found to be easier.

In 2013 a Linux Mint 14 (MATE) distribution was installed alongside the existing Windows install on the development laptop. A good guide to partitioning drives and installing Linux Mint was found at <http://www.linuxbsdos.com/2012/06/06/how-to-dual-boot-linux-mint-13-cinnamonmate-and-windows-7/> .

Downloading the required software and following the above guide allows the creation of a dual-boot setup into either Linux Mint or Windows while maintaining the Windows boot loader.

It is probably easiest to setup an installation distribution of Linux Mint on a USB flash drive from which the installation onto HDD can be made. A good guide to creating a bootable USB can be found at <http://en.kioskea.net/faq/2469-unetbootin-installing-linux-using-a-pen-drive-bootable-usb> .

The Linux Mint distribution itself can be downloaded from the Linux Mint website. In 2013, Linux Mint 14 (MATE) was used. This is very similar to the Ubuntu work environment.

**Recommended reading: ‘getting started\_2012’** to understand the system modules and what each is responsible for and how they interact.

**FLASHING THE BOARDS**

Working under Linux was found to be the simplest for flashing and debugging (command line). The ECE Wiki provides good instructions on building the ARM toolchain (GCC) under Linux and was used in 2013. A guide to connecting to the boards using OpenOCD and the department's USB-JTAG connectors is also available. As the Boards run SAM7 processors there is plenty of resources and help in the department for them. Dr Michael Hayes has great experience in the area and may be able to help.

The makefile supplied with the source code builds the whole project (all modules). The make-program-module commands did not appear to work therefore the following process was used:

Navigate to ‘Software’ directory

* ocd //run openOcd from makefile and detect board presence
* make //make all modules
* arm-eabi-gdb -batch -x program.gdb build/motor.elf   
   //(Linux)  
  arm-none-eabi-gdb -batch -x program.gdb build/motor.elf   
   //(Win)  
    
   //make program didn’t work  
   //replace ‘motor’ with the name of the module you want to program
* Cycle the power  
    
    
  Note: the board you are flashing needs to be separately powered. The JTAG adapter connection does not supply power.

**OPERATION**

Once the system is powered up – operation starts automatically.   
The comms board immediately starts polling the other boards and checking that all the correct boards are connected.

Then the system moves to the calibration phase. If this successfully completes, the system moves to the running state where interaction with the laptop can take place.   
If calibration is unsuccessful, or there is an error at any point the system moves into the error state. A power cycle is needed to get out from this state – meaning a complete restart. It would be nice to have some alternative transitions from the error state during development.

Laptop interaction is done via script files through a high level python API. These can be based on the demo/calibration script files in Software/highLevelControl.

**HARDWARE CONFIGURATION**

The boards are connected together the same as described in the 2012 guide and shown in the Images folders under documentation.

The main changes in 2013 were mounting the components on the go-kart and powering the boards and actuators.

**PCBs**

The 5 PCBs and 2 motor driver boards were placed carefully around the kart as shown in the images, with the motor control board up the back near other control circuitry. This is important as the SPI cable between the motor control board and the student PWM board must be kept short.

A ground connection was added between the motor driver boards and respective PCBs – connectors are supplied for 4 connections rather than the 3 made in 2012 (they were not grounded).

It is pretty clear that the on board connectors, connections between the boards, board housings and placement on the go-kart is not robust enough for outdoor movement of the kart. Therefore this needs a complete re-work.

**Brake**

The new brake actuator bracket is secured to the chassis with two bolt-washer-nut assemblies. The brake actuator can be taken in and out of the bracket once it is on the kart by completely retracting the brake. This can be easily done without running the entire system by simply disconnecting the actuator from the motor driver board and supplying it with 12-24V from a DC bench supply. Reversing polarity reverses direction.

Ensure current is limited to around 2A when running actuators off bench supply. They can draw about 1.5A when operating loaded and together.

**Steering**

The steering actuator connects to the steering column in place of the steering wheel. Spacing washers should be added under the bolts (used for the steering wheel) so that they extend completely into the slots in the steering servo.

The steering servo can also be operated from bench supply similar to the brake – ensure current is limited.

**Power supply**

The 5 PCB boards are powered via 24V drawn from the control circuitry at the back of the kart. This is a fused (I think 3A, need to check again) bus that powers the go-kart’s motor monitoring control circuitry and the student PWM board as well. A connector was created to draw from this 24V bus on the kart control board and split to power both the PWM board (the short end of the split) and the brake board (the long end of the split). The brake board regulates this to 5V and powers the other boards via CANBus.

Not all of the boards have working 24V to 12V regulators therefore the brake board was given 24V and 12V was taken from its 12V test pin to other boards. The speed sensor needs 12V supply.

Powering On: The CANBus system in this configuration is powered on via the control switch on the front panel. This ensures that all control modules (5 CANBus PCBs + PWM student board + go-kart’s control board) get power at the same time. For some reason if the CANBus boards and the PWM board were powered at different times, the duty cycle response would *sometimes* change.

The actuators are powered via a 3A fused line directly from the battery but after the emergency stop contactor. The power (24v) is actually given to the motor driver boards which use it to drive the actuators.

**Steering Limit switches**The steering limit switches were connected to the chassis using nylon bolts. This prevented the motor EMI picked up by the chassis interfering with the steering board.

The limit switch connections and actuator power lines should still be on the go-kart, if not Ken Smart should know where they are/ be able to advise how to re-create them.

**PWM STUDENT BOARD**

This board is an intermediate board between the CANBus motor control PCB and the go-kart motor control circuitry. It generates a PWM signal with a duty cycle corresponding to that passed over SPI. In doing this it checks the brake and current (LEM) sensor on the motor.

The code in main.c under ‘Current\_Modified\_Code\_2013’ was loaded onto the board using Atmel Studio. Atmel studio can be freely downloaded but is also available in the DSL. The board is flashed using SPI therefore a AVR ISP MKII will need to be borrowed off Phillip Hof for connecting to USB on a computer. A vero-board-SPI connection is supplied for connecting the MKII to the student board.

The current code allows successful motor control up to a duty cycle of about 50% (remember the PWM passed from the student board to the go-kart is actually inverted). It was not tested passed this. Unfortunately Edsel’s test rig was not properly calibrated for loaded conditions when testing had to be done. Therefore before testing under loaded conditions (with the wheels on the ground) the code on the PWM board should be checked on the test rig, and correct over-current shutdown should be confirmed.

**GUI**

A test GUI was created to allow easy control of the go-kart modules. This in the ‘highLevelControl’ section and can be run from the command line after the system has started and transitioned to the running state. Starting the GUI will initiate a connection with the CANBus system. You can control the kart via sliders, buttons and text inputs.

This is only a test GUI, bad inputs are not accounted for so be careful what you give it.

Some limits are set within the GUI code and can be adjusted if needed.

**COMPUTER VISION INTEGRATION**

Computer Vision code and documentation is located in Software/Computer\_Vision.

‘Computer\_vision\_project’ contains the original checkerboard tracking algorithm and test videos used for the COSC428 project. The research paper explains the functionality of the algorithm.

The Computer vision folder was used as the workspace folder for modifying the algorithm under eclipse – hence all the .project type files.

The ‘check\_cmake’ contains the modified algorithm (named ‘Checker’) compiled via cmake to a form accessible via the ‘CV\_interface’ python scripts. These python scripts use process forking to run the checkerboard tracking algorithm as one process and pipe its results (an x-y-z position vector) to a python process which interprets the data and sends commands to move the go-kart as required. Unfortunately the current method requires a copy of the go-kart API in this folder, which is not great as too copies have to be maintained identical. This is a path/permission problem and should be solvable so that only one copy of the API is maintained.

If changes to the computer vision detection part of the algorithm are made it will have to be recompiled with cmake.

The images are very helpful for setting the system up. As is the 2012 documentation. The 2011 documentation/work was not found all that helpful for the work done this year but may be of use if you are redesigning the PCBs.

If you have any problems:

Ken Smart (machines lab) – where the go-kart is kept   
 He’s the project take shape over a number of years and is very helpful.

Edsel Villa (power electronics) – knows the go-kart motor and drive requirements

Myself: Nissanka Weerekoon – I will not be around Christchurch but will try help via email if you are confused about anything done in 2013 – nis.wren@yahoo.co.nz