May 7 2012

Fly-by-wire Go-Kart

Specification Report

University of Canterbury Department of Electrical and Computer Engineering

Matthew Wigley
mww35@uclive.ac.nz
Student Number: 56388198

Supervisor:

Dr. Andrew Bainbridge-Smith
andrew.bainbridge-smith@canterbury.ac.nz

Abstract

This report details the specifications for a fly-by-wire control system which will be developed for an electric go-kart. This system will be designed to provide a simple and easy to use control platform is available for future computer vision students. In 2011 a 3rd professional year group project was initiated to work on this project. The hardware control boards were completed and partially tested but the software and documentation for these boards was not. First the hardware will have to be reassembled, tested and fixed, and then the software will have to be completed for each control board, after which the software governing communication between the boards and the controlling laptop will have to be completed, and then a high level API will be developed for future users. So far the hardware has been reassembled and partially tested; the software toolchain has also been compiled and some new test software has been successfully loaded onto the boards. Some success has been achieved in fixing the software bugs – the actuator PWM control and brake actuator position measurement code has been rewritten to work correctly.

Table of Contents

Project Overview

The aim of this project is to finish converting an electrically powered go-kart into a fly-by-wire vehicle so that it can be easily used as a platform for future autonomous vehicle work.

Autonomous vehicles are driverless cars which can fulfill the roles of a human driver; they take sensory input from a wide array of sensors and use it to build up a picture of the world surrounding the vehicle, then react accordingly to move towards a navigational goal. Autonomous vehicles have many possible advantages over conventional vehicles; they can save costs by removing the need for a driver, reduce fuel usage, reduce traffic accidents, and provide the possibility of fleet-wide optimization to reduce congestion and further increase safety and decrease fuel usage[1]. Because of these possible benefits research into autonomous vehicles is of interest to the Electrical Engineering department at the University of Canterbury. The vast majority of recent autonomous vehicles use expensive LIDAR sensors to scan the environment. It would be advantageous to be able to provide less expensive autonomous control, so this project will focus principally on this aspect.

In previous years various projects involving autonomous vehicles have been undertaken at the University of Canterbury. The participants usually spend the vast majority of their time working on the control hardware, leaving very little time for computer vision concepts. This project aims to provide a robustly controlled development vehicle so that in the future students will have a platform to work on without needing to consider hardware problems.

Students may want to implement a variety of computer vision methods the system must be extensible so that new sensors can easily be added. They should also not have to worry about low level details, so a high level software control API for the go-kart is also needed. The API needs to provide functionality for high level movements such as accelerating to a set speed, turning a corner sharply, or braking safely to avoid hitting an obstacle. The project will also need to be documented extensively so that subsequent work on it does not require reverse engineering previous work.

Last year work on the go-kart conversation was undertaken by a team of third professional year students[2]. Control hardware was designed and built, and peripherals to control the go-kart steering and brake were purchased. The team chose to produce a separate SAM7[3] based control board for each subsystem on the go-kart – braking, steering, communications, motor control, and speed sensing. The boards were designed to be linked by a CANBUS[4] network over Ethernet to make the system simple to extend. High level Control was designed be implemented from a laptop linked to the communications board via USB. Some software was written for the boards and the control hardware was carefully debugged. The software to control the SAM7's was not completed – there are problems with most of the systems – for example there are issues with actuator control, the CANBUS network, and USB communications.

Requirements

To be successful a fully working fly-by-wire go-kart must be produced. An easy to use high level control API should also be provided.

The requirements for this project can be roughly divided into four different categories: low level hardware, inter-board communications, USB communications, and high level software control. The low level hardware requirements include getting each of the SAM7 boards interfacing with and controlling the hardware peripherals robustly. The inter-board communications requirements involve getting CANBUS communication working correctly between the boards so that commands and information can be passed between them. The USB communications requirements involve the sending and receiving of low level commands from a laptop to the linked SAM7 boards to control the go-kart and receive sensor information. Because of the nature of this project, some aspects will have to be left for future students – for example, in the future the CANBUS protocol may have to be improved to make it more robust and the high level API may need to be extended.

1. Major goals

Low Level Hardware

- Cat5e and UART connection cables must be manufactured
- The hardware needs to be reassembled, and powered up
- The build toolchain needs to be compiled – GCC, openOCD, and GDB all need to be compiled
- Code needs to be loaded onto the SAM7 boards, and character display and debugging capability needs to be obtained.
- New motor driver boards need to chosen and be purchased

Each of the boards also need to be worked on individually:
Brake Board

- The actuator position potentiometer needs to be tested and fixed; ADC needs to be performed on it to get the brake position.
- Dual direction PWM control of the motor driver board must be obtained
- Actuator PID control must be implemented based on the potentiometer value so that the brake position can be arbitrarily.

Steering Board

- PWM control of the steering needs to be obtained
- The quadrature decoder needs to be used to determine the steering position
- An interrupt needs to be set up to get the steering limit switch data
- Code needs to be written to ensure that the steering cannot be driven past a limit switch and damage the steering motor
- Absolute position control of the steering needs to be obtained by using feedback from the quadrature decoder

Motor Control Board

- Serial data needs to be passed onto the student board

- Acceleration and deceleration of the go-kart needs to be controlled
- The brake value (obtained by the brake board) needs to be passed onto the student board

Speed Sensor Board

- The inductance sensor peripheral needs to be polled and its value recorded
- For testing purposes the speed of the vehicle should be calculated and displayed on the board

Inter-Board Communications

Rudimentary CANBUS functionality is needed. Advanced CANBUS functionality is outside the scope of this project, and could be worked on by another group in future years. For the purposes of this project the CANBUS protocol needs to reliably send and receive data between the boards. Specifically:

- Sensor movement needs to be able to be sent from the communications board to another board
- It needs to be possible to send a request for data to a board, and for the board to transmit the required data back – e.g. the communications board must be able to get the brake actuator position
- Do startup calibration correctly – the boards must switch to a running state only once all other boards have finished calibrating
- Error state transitions – if an unrecoverable error occurs on any board then all boards should transition to a failsafe state

USB Communications

It needs to be possible to send and receive information about each of the boards over USB. This can be fairly low level. It needs to be possible to:

- Send a request – e.g. to set the brake actuator to a position
- Get data – e.g. get the current brake position

High Level Software Control

A high level API needs to be written in python which provides functionality for commands such as turn left, right, slow down, speed up, and reverse. An emphasis will be made on ensuring that the API is well documented, well designed, and easy to use.

2. Measurable Outcomes

The main aim of this project is to implement fly-by-wire control for the go-kart so that it is fully controllable via laptop. In particular it must be possible to turn, brake, accelerate, decelerate, and gain a quantitative measure of the current speed of the go-kart; it should be possible to do any of these things via a single high level command.
It is possible that it will not be feasible to achieve USB control within the timeframe of the project; in this case partial success of the project can be demonstrated by some outcomes:

- Control of individual peripherals

- Peripheral state measurement and display
- Calibration of individual peripherals
- Demonstration of peripheral control from the communications board
- Demonstration of board state transitions through calibration to the run state, and error state transmissions to failsafe positions
- Demonstration of peripheral control via USB
- Measurement of peripheral state via USB

Approach and Preliminary Design

1. Approach

This project has a lot of work which can be completed simultaneously, and few critical tasks (for example only two of the SAM7 boards need to function before work can begin on the CANBUS protocol) a flexible approach could be taken in terms of the project management. Taking such a flexible approach would mean that although little work would have to go into project management but it would be very hard to measure the progress of the project and easy to get stuck on a single part of it. Instead it has been decided to specify deadlines for segments of the project, and if, for example, not all of the SAM7 boards function correctly at the deadline then work will have to progress to the next stage anyway. This will mean that even if the project is not fully completed a proof of concept should be demonstrated for each of the stages and some basic high level control should be demonstrable. Because the timeline will be highly structured it will be hard to lose focus and easy to tell if deadlines are being met.

The initial stages of the project involve mostly testing and then completing the functionality of the previous years. To do this it was decided to use an iterative testing cycle. First the smallest isolatable feature would be tested (such as receiving data from a peripheral), and if necessary it would be fixed. Then code would be written to test the next smallest possible feature (possibly relying on the now fixed feature) and any problems with it would also be corrected. This cycle will be repeated until the SAM7 hardware has had its functionality fully tested and fixed.

2. Preliminary Work

The first two months of the project were largely spent researching the project from last year, researching computer vision, and investigating the hardware and software produced during the previous project. Initially the project was thought more complete than it was later found to be, so some time was spent researching various aspects of computer vision before it was decided to narrow the scope of the project. An extensive wiki was provided as documentation by the previous group[2]. The contents of this wiki were thoroughly read, along with any information it linked to; in doing this it was found that some aspects of the project were improperly documented. Time was also spent researching the CANBUS and USB protocols, as knowledge of them will be important for the project.

After this significant time was spent studying the software which had been written by the previous year's students. This was so that a good understanding of the system architecture could be obtained, reducing the possibility of future complications. It was found that parts of the software were incomplete, and parts appeared that they would not function as expected. The PCB schematics for each of the boards were also thoroughly investigated to understand how to connect the peripherals to each of the boards.

Next work was started on testing the hardware. Cat5e Ethernet cables were manufactured[5], and the SAM7 boards were powered up. Before modification the system simply transitioned into an error state. Because of this non-functionality it was decided that an incremental testing paradigm would be the only way to properly test the software. Some time was also spent learning how the control hardware fits into the electric go-karts. Before it was possible to write any test code the compiler and debugger toolchain had to be compiled; this required downloading the sources for GCC, openOCD, GDB, and their dependencies, and then compiling them. GCC and GDB had to be compiled specifically so that they could produce code that would run on the SAM7 boards. After compiling the toolchain a simple test program to display characters on the numerical display on the SAM7 boards was written. Flashing the program onto the SAM7 required considerable effort – the instructions provided on the wiki did not work for the current version of openOCD or GDB, and workarounds had to be found before the chip could be flashed via a JTAG interface[6].

It was decided to test the motor drivers for the actuators first as any problems with these could be time consuming to fix. It was found that the provided PWM control code did not function, so it had to be rewritten. After the PWM code was rewritten it was also found that the motor driver boards would not function - the integrated circuits on the motor driver circuits had failed. It is uncertain what caused this failure, but the previous years students also had problems with the motor drivers failing. In previous years at the University of Canterbury autonomous control projects have often been blocked by problems with motor driving circuits (as happened with this project last year) so it was decided that two commercial motor driver PCBs would be purchased to replace the broken ones so that it will be possible to meet the project deadline. A reasonably priced motor driver board which met the specifications was found and will be purchased before the 11th of May.

Other parts of the system were also tested. It was found that the CANBUS protocol does not appear to currently work properly, the USB functionality is incomplete and does not function, and potentiometer data acquisition for the brake actuator does not work. The quadrature decoder and limit switches for steering are untested; as are the inductive speed sensor and the acceleration control. In doing this testing a good basic understanding of how the system works was obtained, this will make future debugging of the project faster and easier.

Budget and Timeline Summary

1. Budget

To reduce both the scope of the project and the budget all possible attempts will be made to use the hardware which was constructed last year. Some new cables had to be made as they were not supplied with the rest of the hardware, and two new motor drivers will have to be purchased because the ones designed and built in the previous year have malfunctioned.

Description	**Qty**	**Cost Per Unit ($)**	**Total Cost ($)**
RJ45 connecter	10	0.35	3.50
Cat5e Ethernet cable	5m	1.00	5.00
0.1mm^2 stranded wire	2m	0.30	0.60
2mm 10 Pin connector - female	2	2.00	4.00
2A Dual L298 H-Bridge Motor Driver	2	57.20	114.40
		Total	$127.50

Table 1. Budget.

The motor drivers are the only parts which must be purchased from an external supplier. The motor drivers will be purchased during the week ending the 11th of May, and will arrive on approximately the 21st of May.

2. Timeline Summary

Although the project is very non-linear by nature it has been broken into a set of linear steps for project management purposes; this makes it easy to measure progress and to ensure that deadlines are being met. For some parts of the project parts of it being incomplete does not block the next task; this means that if insurmountable problems are encountered in parts of the project those parts can be left off whilst still allowing partial completion of the project.

Task Name	Duration	Start	Finish
Project Research	35 days	Feb 27	Apr 13
Deadline: Specifications Report	0 days	May 7	May 7
Low Level Hardware	**50 days**	**Apr 16**	**Jun 22**
Preparation	10 days	Apr 16	Apr 27
Motor Control Board	10 days	Apr 30	May 11
Speed Sensor Board	10 days	May 14	May 25
Brake Board	10 days	May 28	Jun 8
Steering Board	10 days	Jun 11	Jun 22
Inter-board Communications	**14 days**	**Jun 25**	**Jul 12**
Send data	5 days	Jun 25	Jun 29
Receive data	5 days	Jul 2	Jul 6
Calibration	2 days	Jul 9	Jul 10
Error State Transitions	2 days	Jul 11	Jul 12
USB Communications	**10 days**	**Jul 13**	**Jul 26**
Commands	5 days	Jul 13	Jul 19
Data requests	5 days	Jul 20	Jul 26
High Level API	**10 days**	**Jul 27**	**Aug 9**
Functionality	5 days	Jul 27	Aug 2
Documentation	5 days	Aug 3	Aug 9
Testing and Bug Fixes	5 days	Aug 10	Aug 16
Deadline: Project Inspection	0 days	Aug 17	Aug 17
Engineering Report	22 days	Aug 17	Sep 17
Oral Presentation Preparation	8 days	Sep 17	Sep 26

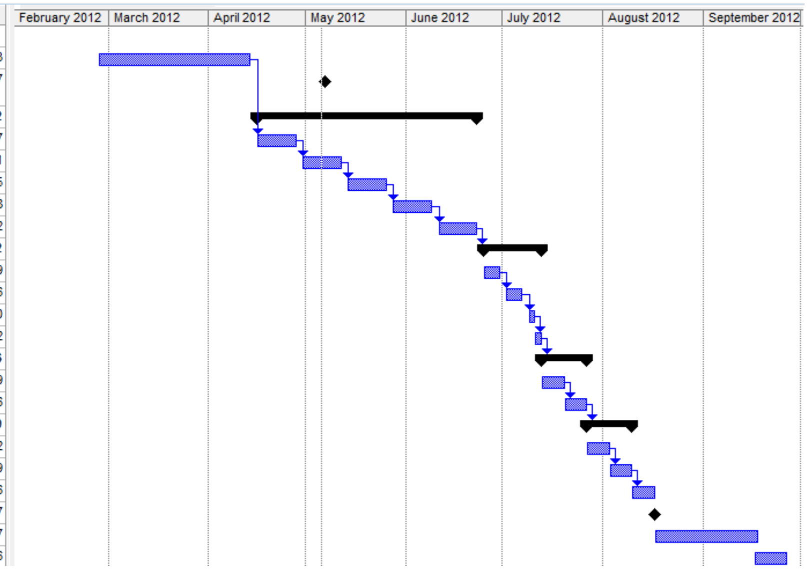

Figure 1. Gantt chart for thre project.

One blocking problem is the arrival of new motor driver boards for the actuators; progress on the brake and steering board cannot be completed until they arrive on the 21st of May. To get around this it was decided to work on the motor control board and speed sensor board before the brake and steering boards.

The important milestones for the project are June 22nd for the completion of work on the low level hardware, July 12th for the completion of work on inter-board communications, July 26th for USB communications, and August 9th for the high level API. All non-report work on the project must be completed by August 17th.

Project Risks and Conclusions

Some problems have been encountered during this project – the provided software was less complete than initially expected, and the motor drivers malfunctioned. Because most aspects of the project have now been investigated there should be no more large unforeseen problems, so a reasonable appraisable of the feasibility of the project should be possible. The need to purchase motor driver boards is one uncertainty which could negatively affect the project – if they arrive too late, or if they are inappropriate for the task this could cause the project to fail. This risk has been mitigated by carefully checking the specifications of the motor drivers which will be purchases and ensuring that they are suitable, and also by talking to the supplier to ensure that the parts will not arrive more than a week late.

Regardless of these problems it should be possible to produce a successful outcome as long as there are no new major problems. Progress has been made in understanding how the system works; a long time was spent investigating the hardware and software provided from last year's project. This means future development will have a fairly fast turnaround as the preliminary research needed has been completed. Some progress has been made in fixing bugs in the software code; and now that the software toolchain has been compiled and openOCD configured it is easy to test new changes on the SAM7 boards meaning that development can progress quickly.

This project has some critical points of failure, where if these parts of the project do not succeed then further higher level work cannot continue. If the CANBUS or USB functionality cannot be completed then it may be impossible to start work on the high level API or fly-by-wire control. Preliminary investigation indicates that it should be possible to get enough CANBUS and USB functionality that the system can function, so this should not be a problem. A single SAM7 board failing would mean that the project could not be fully completed, but partial control using the other boards should still be possible to accomplish.

In the event that the project is completed before the deadline date some basic computer vision will be implemented as a proof as concept.

Bibliography

[1] A. Forrest, M. Konca, Autonomous Cars and Society, Worchester Polytechnic Institute, MA, USA. 2007. URL: http://www.wpi.edu/Pubs/E-project/Available/E-project-043007-205701/unrestricted/IQPOVP06B1.pdf

[2] H. Jenkins, W. Looman, S. Richards, Z. Taylor, Team Ramrod / Mariokart Wiki URL: https://github.com/team-ramrod/mariokart/wiki

[3] SAM7X512/256/128 Preliminary. Atmel Corporation, San Jose, CA, USA, 6120i edition, May 2012.

[4] ISO 11898-1:2003 - Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signaling, 2003, URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33422

[5] S. Nikkel, How to wire Ethernet Cables, 2011, URL: http://www.ertyu.org/steven_nikkel/ethernetcables.html

[6] A. Dushistova, A. Rusev, Debugging with JTAG, MontaVista Software, Inc. 2008. URL: http://mvista.com/download/vision08/Debugging-with-JTAG-Vision2008.pdf

[7] H. Jenkins, Embedded Hardware Design For An Autonomous Electric Vehicle, University of Canterbury, New Zealand, 2011

[8] Z. Taylor, Development of a Marker Following System for use in the Mariokart System, University of Canterbury, New Zealand, 2011

[9] W. Looman, Software Engineering Practices in the Mariokart System, University of Canterbury, New Zealand, 2011

[10] S. Richards, Safety by Design for the Mariokart System, University of Canterbury, New Zealand, 2011

[11] J. Feder. Google Wants To Work With Automakers On Driverless Cars. May 2012. URL: http://www.thecarconnection.com/news/1075872_google-wants-to-work-with-automakers-on-self-driving-cars

[12] Z. Jia1, A. Balasuriya, S. Challa, Vision Based Target Tracking for Autonomous Land Vehicle Navigation: A Brief Survey, September 2008. URL: http://www.benthamscience.com/cseng/samples/cseng2-1/0004CSENG.pdf

[13] D. Gennery, A Stereo Vision System For An Autonomous Vehicle, Stanford University, USA.

[14] A. Kelly, A. Stentz, Stereo Vision Enhancements for Low-Cost Outdoor Autonomous Vehicles, In 1998 IEEE International Conference on Robotics and Automation, Carnegie Melon University, USA

[15] S. Se1, P. Jasiobedzki, Stereo-Vision Based 3D Modeling for Unmanned Ground Vehicles, MDA, Space Missions, Canada. 2006. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.84.3870

[16] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2009.

[17] M. Barr, Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition, O'Reilly, 2006

[18] AT91SAM ARM-based Flash MCU, Atmel, 2011, URL: www.atmel.com/Images/doc6175.pdf

[19] T. Carr, Motor Driver 2A Dual L298 H-Bridge, New Zealand. 2012, URL: http://www.mindkits.co.nz/store/movement/motor-driver-2a-dual-l298-h-bridge

[20] L298 H-bridge Dual Bidirectional Motor Driver, Canakit, Canada, 2010. URL: http://www.canakit.com/Media/Manuals/UK1122.pdf