September 17 2012

Fly-by-wire Go-Kart

Engineering 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

Based on a 2011 3rd professional year project, the hardware and software control systems for a drive-by-wire go-kart were developed and completed. The system provides a high-level software control API for the University of Canterbury electrical engineering department's electrical go-karts. The development of a control API is of interest as a platform for autonomous vehicle research. In 2011, a group of four students initiated work on a project to produce a fully autonomous go-kart but they did not achieve this goal; the hardware components required for the project were purchased and constructed but the software control for the system was left largely incomplete and untested. This year the software for the project was completed in order to provide full control of the go-kart via a software API. The 2011 system was researched, investigated, and tested, replacement hardware was purchased, peripheral control software was written for the four boards, inter-board communication was obtained, laptop based control was developed, a high level control API was written, and the project was extensively documented and tested. Testing was completed on a workbench but the system has not been tested on the go-karts themselves. Results of the on-bench testing indicate that the control system is fully functioning and capable of full control of the go-kart via the Python API.

1. INTRODUCTION

The aim of this project was to complete the development of a software API (Application Programming Interface) for a drive-by-wire system for the University of Canterbury Electrical Engineering department's electric go-karts [1]. The principle aim of the project was to produce an API that is useful for autonomous vehicle projects. A 2011 team initiated work on the autonomous control project for an electric go-kart by a 2011 team [2] [3], but the project was not completed. The 2011 team designed and built the control hardware, and purchased peripherals to control the go-kart's steering, control its brake, and to measure its speed. In 2011 open loop control of the steering motor and brake actuator were demonstrated, as was basic communication between the 5 control boards for the kart, but most other aspects of the software for the project were non-functional.

An autonomous vehicle is a driverless car that can perform tasks in similar respects to a car with a human driver [4]. An autonomous vehicle must take input from a variety of sensors and use them to develop a model of the surrounding world and then react accordingly to navigate towards a goal [5]. Autonomous vehicles have the potential to reduce costs by removing the need for a human driver, reduce fuel usage, and reduce traffic accidents. Autonomous vehicles also enable fleet wide optimisation that can in turn reduce traffic congestion, further increase safety, and decrease overall fuel usage [6]. The study and development of autonomous vehicles at the University of Canterbury is of interest because of these potential benefits.

As the development of an entirely autonomous vehicle within the timeframe of a 3rd professional year project was deemed to be unfeasible, it was decided to focus on developing a robust control API for the electrical engineering go-karts. In previous years, teams have attempted to develop both the necessary vehicle control electronics and the computer vision. The electronics development stage has often taken much longer than expected, meaning that there was not time for in-depth computer vision research. For example, in the 2011 project, the team initially hoped to develop a working autonomous go-kart, but problems during construction forced them to present an incomplete project. Developing a solid and dependable go-kart control system will allow future students to concentrate on studying computer vision technologies without having to focus on the vehicle control electronics.

1.1 Design Overview

The system consists of a brake, motor control, speed sensor, steering, and communications board, each of which are controlled by Atmel AT91SAM7XC256 [7] microcontrollers. The boards communicate with each other over a CANBUS (Controller Area Network Bus) [8] network. The communications board communicates with a controlling laptop via a USB (Universal Serial Bus) connection and handles inter-board communication issues such as start-up and error handling. The four other boards control their respective peripherals. The laptop, which connects to the communications board, controls the go-kart via high-level API commands. In future projects, computer vision based control will be implemented on the laptop in order to make decisions about the control of the go-kart.

1.2 Design Requirements

One major downside to current mainstream autonomous vehicle technology is the cost of the data sensors that are used to survey the operating environment. A commercial LIDAR (Light Detection and Ranging) system useful for high-speed autonomous vehicles can cost around US$75,000[9]. For mainstream autonomous vehicle usage, the cost must be much less than even a single LIDAR sensor, so this project will focus on developing a low cost autonomous vehicle. This project involves only the development of a drive-by-wire system so one other aim was that the system be extensible enough that more sensor expensive systems could also be incorporated if desired. The system needed to be designed to be extensible to new sensors, as it needed to be extensible to the possibly unforeseen needs of future users.

Another requirement was that the system be robust and handle errors gracefully. The go-kart the system was designed for has a top speed of 40 kilometres per hour [1], and a net weight of 150 kilograms. If a control failure caused the kart to accelerate wildly out of control, it could cause serious harm to people and property. Because the system was designed for retrofitting to the electrical engineering go-karts, a requirement was that the system be simple to dismantle so that the go-karts could be used for other projects.

The main aim of the project was to develop a control API for the go-kart so it needed be fully functioned, easy to use, and well documented. If it were not possible to fully control the go-kart by the software API then the system would not be usable for its intended purpose.

1.3 Previous Work

In 2011, work was initiated on the autonomous go-kart project. The 2011 team chose appropriate peripherals to control the brake, steering, and speed of the go-kart. The overall system design was chosen. The five control boards were designed and manufactured. Two motor driver boards to drive the servos were also constructed. The hardware was tested extensively and most hardware issues were corrected. Software development was initiated but not completed – some basic open loop control of the steering motor and brake actuator was demonstrated; as was speed measurement via an inductive speed sensor but communications between boards was not fully completed. Proof of concept sending of a single byte of data from the communications board to the controlling laptop was also demonstrated. In general, the developed hardware was robust (excluding the motor drivers, which had to be replaced) but owing to time pressures, almost all software functionality was incomplete and undocumented.

2. DESIGN PROCESS

The first step involved in designing the system was determining exactly what the project needed to accomplish. The requirements were determined during the initial research stages of the project and then codified in a project specification report [10]. The next step involved understanding exactly what work had been completed by the 2011 team, what design decisions they had made, what worked correctly, and what

couldn't be changed due to time or monetary constraints even if it functioned suboptimally. Contact was made with three of the team members from the 2011 project in order to receive a first hand explanation of the progress of the project, and the documentation that was provided with the project was investigated [2]. Finally, the project was thoroughly tested to determine the functionality of each component of the project. Testing each component also helped to gain a fuller understanding of the overall design of the project.

In conjunction with the research step, purchase requirements were also evaluated. It was found that the motor drivers were non-functioning, so new motor drivers were purchased [11]. The new motor drivers also had the benefit that they were more robust, more tolerant to invalid input, and failed gracefully when overheating. Cat5e, JTAG (Joint Test Action Group) connectors, and parallel port programming cables were also constructed. The next step involved to getting arbitrary code compiling and working on the five control boards. The Linux Mint [12] operating system was installed and configured, and a GCC (Gnu C Compiler) based ARM7 (Advanced RISC Machine 7) development toolchain was compiled. The configuration steps were documented to make this process simpler for future users.

The system specifications were used to break the project down into a set of small and independent objectives and then an iterative design-develop-and-test method was used to develop the software for the project. First C code was written for open loop control of each of the peripherals, then closed loop feedback control was developed. Next, the boards were integrated via CANBUS, and system wide error handling code was written. Once the boards were controllable through the communications board, the USB communications to the laptop were developed. Finally, the high-level laptop-based control API was developed and tested.

Once the main software design process was complete, the project was tested extensively and any issues that were found were fixed. Python based functional tests were used where possible, but manual testing was required in many places due to the low level nature of the project; for example, testing error state transitions due to hardware faults required initiating a manual hardware fault so could not be automated.

Finally, extensive documentation was developed for the project. The majority of the software code was documented during development, so a large part of this stage involved tidying up the previous documentation and fixing any irregularities that were found. A nine page user document was also written to help in setting up the project and diagnosing common problems.

3. FINAL DESIGN

3.1 Hardware Design

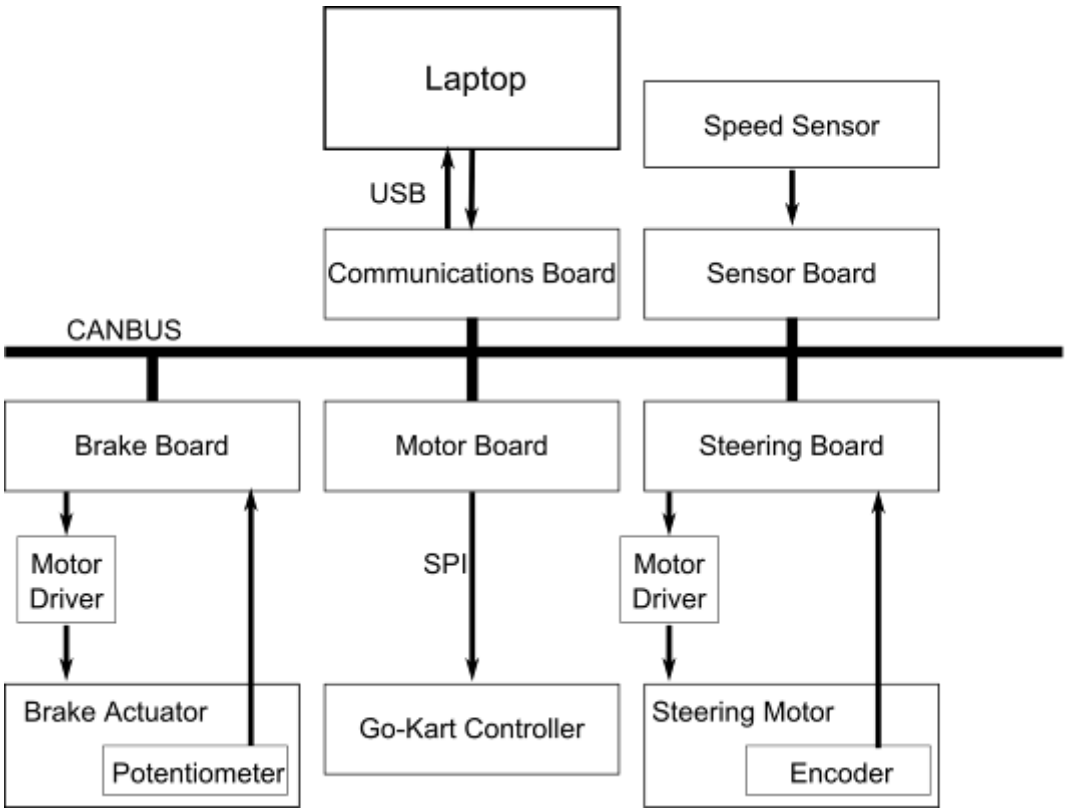

Figure 1 Diagram depicting the hardware layout of the drive-by-wire control system.

The project comprises five Atmel AT91SAM7XC256 [7] microcontroller boards connected via CANBUS over Ethernet. As shown in Figure 1, four of the boards control peripherals on the Go-Kart, while the fifth handles communication with a controlling laptop via USB. The communications board also controls synchronization between the boards – it controls board start-up, calibration, and error state transitions.

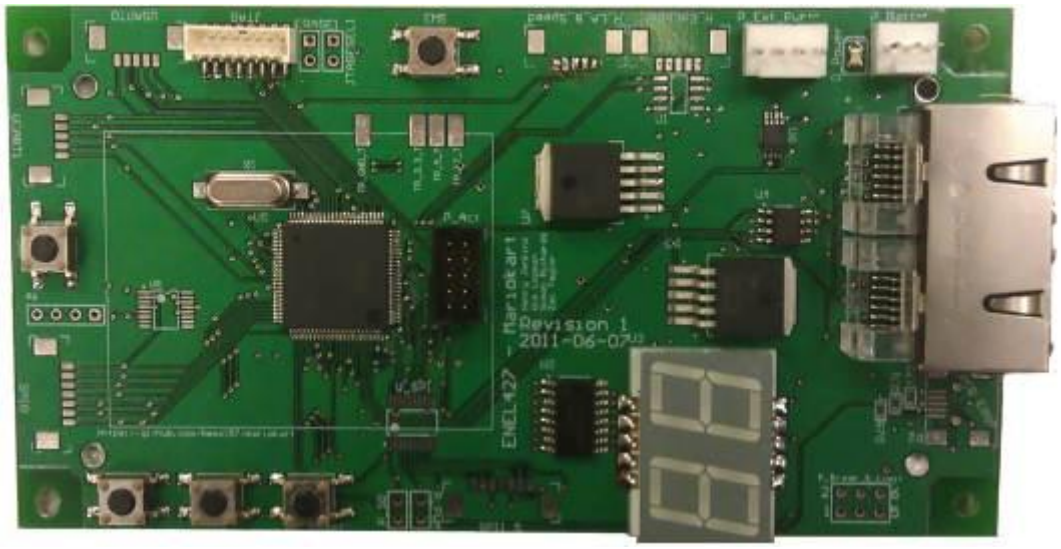

Figure 2 The completed brake board.

One of the completed control boards is shown in Figure 2. The Five control boards shared the same PCB design. A unified PCB design was used to simplify the design process and reduce manufacturing costs. ICs (Integrated Circuits) which were not required on a board were simply not included. The decision to use a single PCB was made by the 2011 group to simplify the design process.

The brake board controls the linear actuator [13] that is used to depress the brake on the vehicle. The position of the brake actuator is determined by polling a potentiometer built into the linear actuator. The motor board communicates with the accelerator control board on the go-kart via SPI to set the acceleration of the go-kart. The sensor board is an extensible board that contains sensors for the go-kart. The sensor board interfaces with an inductive proximity sensor [14]. The inductive sensor is used to measure the rotation of the wheel spokes and use this data determine the speed of the kart. The communications board handles the software side of the CANBUS communication between the boards and communicates with the controlling laptop via USB.

The steering board controls a steering motor [15] that attaches in place of the go-kart's steering wheel to provide control of the angle of the front wheels. The angle of the steering wheel is determined by a quadrature decoder that provides relative position feedback for the steering wheel. Mechanical damage could occur to the vehicle if the steering wheel were turned past its physical limits so two limit switches were placed on the go-kart to allow the steering motor to cut off. The limit switches also facilitated the determination of the centre wheel position allowing absolute position control of the steering wheel.

3.2 Software Design

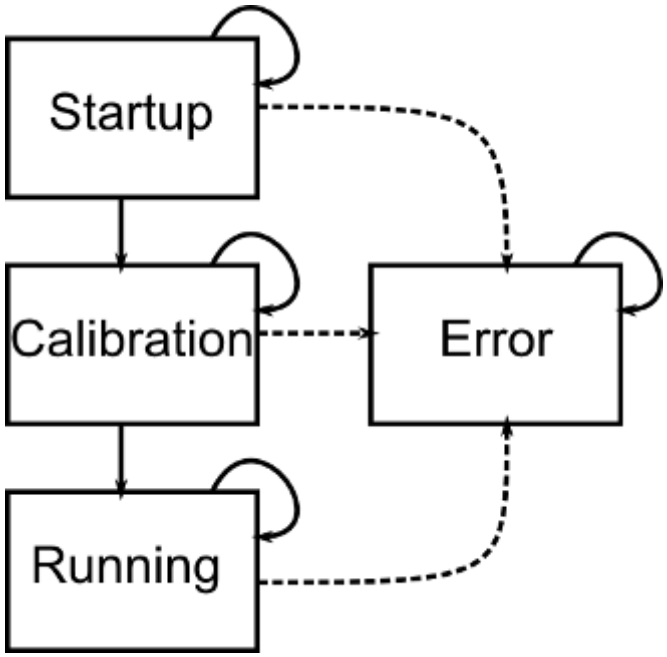

Figure 3 The system state diagram for each control board.

The five control boards were programmed using the C programming language. As shown in Figure 3, Each board implements a standard set of control states – startup, calibration, running, and an error state. The board initially starts in the startup state and continually monitors its CANBUS mailbox for a state transition request from the communications board. When the board receives a request it sends back an acknowledgement to the communications board. Once the communications board has received confirmation from all boards, it sends a request to the boards that they enter their calibration state. Once required peripheral calibration is complete, the boards respond to a communications 'calibration complete request' and with an affirmative. Once all boards have signalled completion of calibration the communications board sends a command for all boards to start running.

The communications board continually sends requests for board-specific data to each of the peripheral controller boards and saves these recorded values in an internal buffer. The communications board responds to data value requests by sending its buffered data value to the appropriate board or to the laptop. The communications board continually monitors the USB laptop connection for requests and commands. The communications board forwards control commands onto the appropriate board as a CAN message.

The communications board also centralizes error state changes if an irrecoverable error, such as a hardware failure, occurs. The board that caused the error moves into its system error state and periodically broadcasts to the communications board that it has entered the error state. Upon reception of an error state broadcast, the communications board orders all other boards to enter the error state. Each board implements an error state that moves any controlled peripherals to failsafe positions.

The communications board also periodically polls the boards, and if one does not respond in a timely fashion it transitions to the error state.

The brake and steering boards both perform closed loop proportional control on their actuators. PID control was implemented but proportional control was deemed sufficient. When the communications board sends position requests to the boards, they reply with a CAN packet containing the motor position. When the communications board sends a set position request, the board acknowledges the command and uses proportional control to move the actuator to the requested position. The brake board initially turns to a 'brake on' state during calibration, and the steering board calibrates itself by touching each of its limit switches thereby determining the centre wheel position.

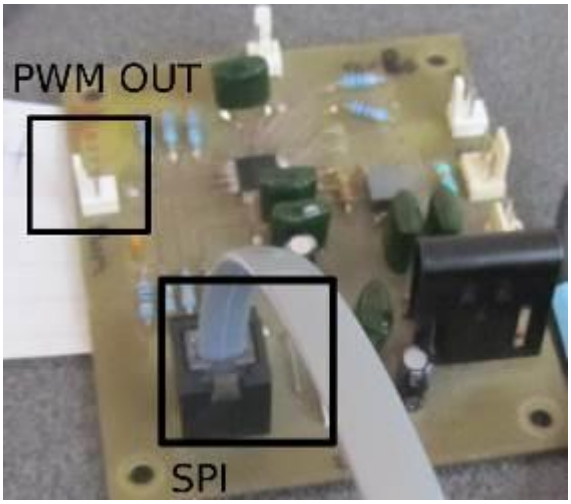

Figure 4 The Atmega8 based go-kart accelerator control board. Digital SPI accelerator input produces a signal on PWM OUT that controls the go-kart motor.

The sensor board responds to requests for a sensor's data by sending the requested data in a CAN message to the requester. Currently the sensor board only controls an inductive speed sensor. The motor board can perform either closed loop speed control or acceleration control. For acceleration control, the board sends the requested acceleration value directly to the go-kart control board via SPI. For speed control, the motor board makes asynchronous requests to the sensor board for the cart speed and uses the received values to perform closed loop speed control on the kart. The control board on the go-kart, shown in Figure 4, was originally created to take input via an analogue accelerator potentiometer. In order to facilitate digital communication with the motor board, the code on the go-kart control board was modified to take a SPI based accelerator input instead.

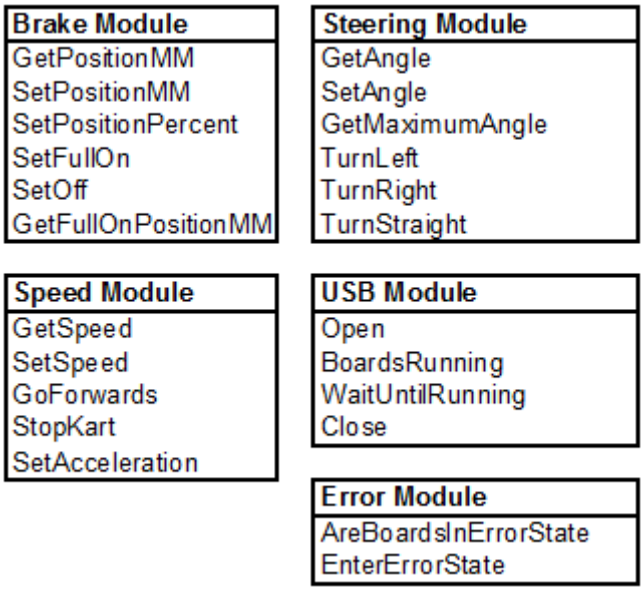

Figure 5 The public functions in the high-level software API.

The laptop-based control API was written in Python to facilitate the writing of high-level computer vision code. As shown in Figure 5, The API provides functions to connect to and interact with the control boards via a USB connection to the communications board. The system comprises of five main modules - a connection module, an error module, a brake module, a steering module, and a speed control module. The modules were designed to be simple to use while providing all control functions that might be required by users. Python based functional testes were also written.

4. DISCUSSION

The project brief was open ended – a final goal could have been chosen that ranged between full autonomy and partial peripheral control. The goal of implementing drive-by-wire control was chosen as an attainable but challenging goal. Because of the goal choice, it was possible to meet most of the design specifications while leaving enough time remaining to thoroughly bug fix and document the project. The project goal and specifications did not require any significant changes for the duration of the project; these static goals simplified coding because modules did not have to be rewritten due to specification changes.

The initial specification process also allowed the project to be separated into small, mostly independent goals. Project progress was measured on a chart similar to Table 1 in section 5.1, and provided a simple visual indicator of project completion. Individual sub-goals were progressed towards in an iterative design-implement-test cycle. Because each sub-goal was chosen to be small design-implement-test iterations could be performed rapidly, allowing errors to be found quickly. Because many of the sub-goals were independent, it was possible to take a break and work on another goal if serious problems were faced.

One issue with the design process was that it was not possible to evaluate how well some parts of the code that were developed in 2011 functioned until scaffolding code had been developed. For example, it was not possible to test USB communications until the communications board was functioning correctly. To work around this issue, the assumption was made that any non-testable components were non-functioning until proven otherwise. Assuming the code would not work ensured that time was budgeted even for worst-case scenarios. All areas of the 2011 code were found to be incomplete or have serious issues, so this budgeted time was invaluable in completing the project on schedule.

Hardware issues were identified during project development. Workarounds had to be developed to produce a working product. The two supplied motor drivers did not function. In order to facilitate completion of the project before the deadline two commercial motor drivers were purchased – these motor drivers functioned correctly and were more robust to errors. Board connector also presented issues; Molex PicoBlade [16] connectors were chosen because of their small size. The PicoBlade connectors broke repeatedly meaning that many cables had to be resoldered. Headers were soldered onto the boards and used as a temporary measure. In the future, a more robust set of headers should be selected and used to replace the PicoBlade connectors.

It is planned that the project be worked on by students in the future so documentation and clarity was of paramount importance to the project. A nine page document that detailing information about the system was developed. The document contained information on rebuilding the system, board specific information and issues, peripheral information, TODO's, and contact information. Photographs of the connected boards were also taken to show how the system connects together. The photographs were labelled to highlight important information about the project in a simple to understand fashion. Code quality and commenting was also focused on; detailed comments were provided to help maintainers understand how each sub-system worked and how to use it.

4.1 Evaluation Against Specifications

Table 1 Evaluation of Final Project Against Project Specifications

Low Level Hardware		**Steering Board**		**Speed Sensor Board**		**High Level API**		**Testing**	
Manufacture Cables	√	Obtain Steering Angle	√	Poll Inductive Speed Sensor	√	Brake Control	√	Functional Tests	√
Reassemble Hardware	√	PWM Angular Speed Control	√	Calculate and Return Speed	√	Steering Control	√	Develop Unit Tests	X
Compile Build Toolchain	√	Closed Loop Steering Control	√	**Communications**		Speed Control	√	Test on Bench-Top	√
Load Arbitrary Code	√	Limit Switch Errors	√	Laptop Connect Via USB	√	**Error Handling**		Test on Go-Kart	X
New Motor Drivers	√	**Motor Control Board**		CANBUS Communication	√	Propagate Errors	√	**Documentation**	
Brake Board		Send Acceleration Over SPI	√	Send Data Requests	√	Board Malfunction Error	√	Comment .C Files	√
Obtain Brake Position	√	Speed From Sensor Board	√	Send Commands	√	Board Removed Error	√	Comment .py Files	√
PWM Speed Control	√	Closed Loop Speed Control	√			Errors to Laptop	√	Document Setup	√
Closed Loop Control	√	Develop Kart Code	√			Robust to Bad Commands	√	Document TODO's	√

Table 1 shows the final state of the project when evaluated against the project specifications. Two major goals were not met. Low-level unit tests were not developed. Unit tests would have facilitated testing the system piece by piece and made the debugging of errors simpler. Functional tests were sufficient to ensure that the project functioned correctly but did not provide information about the source of the error. The final system was also not tested on the go-kart. Testing on the go-kart

may have produced errors that were not visible in on-bench testing, but it is expected that the on-kart testing will go smoothly due to the lack of problems during on-bench testing. Thirty-six out of thirty-eight goals were completed, and the two incomplete goals had minimal effect on the functionality of the system so it should still be fit for purpose.

4.2 Testing

During initial development ad-hoc white box testing was performed on each component – after a change was made to a board it was tested to ensure it still functioned correctly. Because of the difficulty of writing tests for embedded systems, automatic tests were not written until laptop communication was functioning correctly. Once laptop communication was possible, functional tests [17] were written in to test all of the boards. Many of the functional tests require manual supervision because they rely on observing the physical output of a servo or actuator motor and comparing it to on-screen results. The main difficulty with relying on functional tests was that it made it more difficult to determine the cause of a fault. In practice, this was not an issue because new faults generally resulted from the most recently changed module. The form of the error messages also gave clues about the source of the problem. Unit tests should be added by a future team to remedy these issues.

4.3 Future Work

Due to the open ended and nature of the project, some work on the project was left for future teams. An outline of work that could be completed in the future was been documented and is provided in the project directory. As discussed in section 5.1 of this report, the project generally meet its initial specifications so most future developments involve possible extensions to the project, although on-kart testing does still need to be completed. The project was designed with extensibility in mind so if future teams decide a necessary feature is not present it should be possible to add it in.

In order to facilitate future work, a GitHub project was been set up [18] that contains the code and documentation for the project. The documentation attempts to detail issues which future teams may encounter, and specifically deals with issues that were encountered when trying to set up the project after receiving it from the 2011 team. The project has been left in such a state that it should be possible to use it in an autonomous system without modification, but there are issues that it would be advisable to address first.

The first major step in the future should be to ensure that the control platform is solid and robust. Extensive testing on the kart needs to be carried out and unit tests should be developed. The are other issues which should be addressed – more robust connectors should be chosen and soldered onto the PCB's, a hardware error on the speed sensor board needs to be corrected, more work should be done in analyzing possible errors that could occur, and heat sinks should be added to the motor driver ICs. These issues should not affect the ability of the go-kart to perform basic autonomous tasks but fixing them will make the platform more robust and stable.

5. CONCLUSIONS

The hardware and software required to control a University of Canterbury Electrical Engineering department go-kart was developed. The work that was completed was based upon the initial work of a 2011 team who designed the bulk of the system and purchased and developed the hardware that was necessary for the project. The project involved the programming of five SAM7 based control boards that communicated with each other via a CANBUS network. Code was also written so that a communications board could communicate with a controlling laptop via USB. A high-level control API for the system was written in the Python programming language. Extensive documentation was also developed to ease development for future users of the system. By breaking the project down into small sub-components and working on them sequentially, it was possible to efficiently manage the complexity of the project and meet the project specifications. Functional tests were developed and the system was tested extensively on a bench-top, but the system has not been tested on a go-kart. Bench-top testing indicates that few, if any, issues will be found during on-kart testing. Some issues still exist within the system, and these should be addressed by a future group to ensure the system is robust; unit tests should be written, new board connectors should be obtained, and any issues found during on-kart testing should be fixed. Once these issues have been addressed, the system should provide a stable, reusable, and extensible platform for autonomous vehicle development.

6. REFERENCES

[1] "Junoir Sport JS80IIR Owners Manual", TJ Power Sports. Texas, U.S.A, 2008.

[2] H. Jenkins, W. Looman, S. Richards, Z. Taylor, "Team Ramrod / Mariokart Wiki", https://github.com/team-ramrod/mariokart/wiki, accessed 14 Sep 2012.

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

[4] "autonomous vehicle - technical definition", http://computer.yourdictionary.com/autonomous-vehicle, accessed 17 Sep 2012.

[5] R. Siegwart, R. Nourbakhsh, "Introduction to autonomous mobile robots", MIT Press, Cambridge, 2004.

[6] R. O'Toole, "Gridlock: why we're stuck in traffic and what to do about it", Cato Institute, 2009.

[7] "SAM7X512/256/128 Preliminary". Atmel Corporation, San Jose, CA, USA, May 2012.

[8] "ISO 11898-1:2003 - Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signalling", http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=33422, accessed 17 Sep 2012.

[9] A. Priddle, C. Woodyard, “Google discloses costs of its driverless car tests”, http://content.usatoday.com/communities/driveon/post/2012/06/google-discloses-costs-of-its-driverless-car-tests/1#.UFWx8YKhnEM, accessed 17 Sep 2012.

[10] M. Wigley, “Fly-by-wire Go-Kart Specification Report”, Christchurch, New Zealand, 2012.

[11] T. Carr, Motor Driver 2-5A MC33926, New Zealand. 2012, http://www.mindkits.co.nz/store/movement/motor-driver-2-5a-mc33926, accessed 12 Aug 2012.

[12] “Linux Mint”, http://linuxmint.com/, accessed 17 Sep 2012.

[13] “M-Track Features”, Warner Linear, www.warnerelectric.com/pdf/P_1581_MTrack1_pg6-9.pdf, accessed 17 Sep 2012.

[14] “Factory Automation – Inductive Sensors”, Pepperl Fuchs, Germany.

[15] “IG-52GM Datasheet”, Taiwan Hsiang Neng Dc Motor Manufacturing, http://www.superdroidrobots.com/product_info/IG52GM-04.gif, accessed 17 Sep 2012.

[16] “Molex Limited. (2011) Picoblade 1.25mm (.049) pitch connector system”, Molex, http://www.molex.com/molex/products/family?key=picoblade&channel=products&chanName=family&pageTitle=Introduction, accessed 17 Sep 2012.

[17] M. Pezze, M. Young, “Software Testing and Analysis: Process, Principles, and Techniques”, Ch.13, John Wiley & Sons.

[18] M. Wigley, “Mariokart2”, https://github.com/mataamad/Mariokart2, accessed Sep 2012.