THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING

Reliable Data Communications for Sunswift IV

Irving Tjiptowarsono

z3234744

A THESIS SUBMITTED FOR THE DEGREE OF BACHELOR OF ENGINEERING

SUBMITTED ON JUNE 2011

SUPERVISORS: GERNOT HEISER, MAHIUDDIN CHOWDURY

Certificate of Originality

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgment is made in the thesis. Any contribution made to the research by colleagues, with whom I have worked at UNSW or elsewhere, during my candidature, is fully acknowledged.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Irving Tjiptowarsono

Abstract

Sunswift IV is UNSW Solar Racing Team's entry to the World Solar Challenge, a biennial solar car race from Darwin to Adelaide through the Stuart Highway. Performing well in this race requires on-road monitoring of various parameters of the car through a telemetry network, which includes a wireless system. The system implemented in *Sunswift IV* suffered from various reliability issues. This thesis outlines the development of a new system to perform wireless communication required as part of on-road monitoring. Initial tests shows the new system has resolved various issues from the old wireless system, though its reliability has yet to be extensively tested.

Acknowledgments

The work done within this thesis would not have been possible without various contributions from a number of people:

Thanks to my supervisors, Gernot Heiser, for providing the resources to complete the project and demanding high quality work, and Mahiuddin Chowdury, who handled the administrative load within the school.

Thanks to my colleagues at NICTA, especially Etienne Le Sueur who provided a massive amount of guidance on embedded electronics, operating systems, and proofreading many, many drafts; Bernard Blackham and Aaron Carroll for answering my numerous questions.

Thanks to the members of the UNSW Solar Racing Team, both past and present; Carson Au for late night discussion of design ideas, Alexandra Boulgakov and Glen Summers for organising events and keeping the electrical team running, Campbell Mclaren for proofreading at the last minute, and everyone who put their effort into the project.

Last but not least, thanks to my family for supporting me in various ways throughout the project.

Contents

List of Figures

List of Tables

Nomenclature

CAN Abbreviation of *Controller Area Network*. a communication protocol for devices inside a vehicle, developed by Bosch.

Control car The car that follows a solar car when it is being driven in public roads.

SION Abbreviation of *Sunswift IV Observer Node*, the hardware transceiver that was designed as part of this thesis.

Solar car A car that runs solely on solar power, almost universally by using solar panels. They are most commonly made to participate in races such as the World Solar Challenge.

Sunswift IV The fourth generation solar car made by the UNSW Solar Racing Team that won the silicon class of World Solar Challenge 2009 and is the current land speed record holder for a solar powered vehicle.

The team The University of New South Wales solar racing team, the oldest student engineering project in Australia.

Chapter 1

Introduction

The *World Solar Challenge* (WSC) is the world's most prestigious solar car race, first held in 1987 and still being held every two years. The race is held through the Stuart Highway, between the Australian cities of Darwin and Adelaide. *The University of New South Wales Solar Racing Team* (UNSW SRT) has participated in each WSC event since 1996. Their best performance so far was in 2009, when their car *Sunswift IV* became the first silicon-powered solar car to cross the finish line in Victoria Square, Adelaide, winning the silicon class of the 2009 World Solar Challenge

A solar car's performance in this race is determined by strategy simulations based on the car's performance data. While *Sunswift IV* has an excellent sensor and telemetry system, she does not have a reliable on-board data transmission or recording device, which affects the team's ability to continuously monitor the state of *Sunswift IV*. This thesis analyses the issues and presents an improved system to deliver data from the solar car for monitoring.

Chapter 2

Background

2.1 World Solar Challenge overview

The World Solar Challenge is run on a set course—the Stuart Highway—which connects the cities of Darwin and Adelaide through central Australia [World Solar Challenge, 2011]. The race starts in Darwin and finishes in Adelaide, as can be seen in Figure 2.1. It is run over several days, as it is impossible to cover the entire distance in a single day. Over the course of the race, competing teams will gradually spread out along the Stuart highway, since each team will have a different cruising speed.

The race has a set of regulations governing the conduct of the event and the technical specifications of the cars. For example, solar cars are only allowed to race in a limited time slot between 0800 hours to 1700 hours every day. Furthermore, they must stop at designated stops throughout the race, called control stops. Some are 30 minute stops, while others only require 10 minutes. To ensure that all teams comply with the race regulations, each team is assigned an official observer which will travel with the team.

Each solar car must have at least two vehicles escorting it in a fleet; one at the front and another at the rear. The observer, race manager and race strategist normally ride in the car at the rear, as it gives them the best visibility to monitor and control the solar car. This can be seen in Figure 2.2. For this reason, the rear escort vehicle is officially designated as the *primary escort vehicle*. However, it is called the *control car* within the team. The latter term will be used throughout this document.

Figure 2.1: The WSC route for the upcoming 2011 race.

Each solar car may carry a battery pack on the race, up to a specified weight limit. This is intended to limit the energy storage capacity of each car. The battery allows the solar car to store energy from the solar panels for later use, such as when extra energy is required to climb hills or when running under cloud cover. Teams are allowed to start the race with a full battery pack, but are not allowed to recharge it during the race using any external power other than the solar panels. However, they are allowed to charge it while the car is stationary, usually during control stops or at the beginning and end of each day outside the allowed race time.

2.2 Strategy

Since the race is run on a set course, with a strict set of regulations, it is possible to run a computer simulation of the race. The simulation allows the race strategist to determine the fastest way to cross the entire course. This is important as solar cars run under a very

Figure 2.2: *Sunswift IV* on the Stuart Highway, followed by the control car.

limited amount of power input from the solar panels, often under 1 kW. With the low amount of energy available, the race needs to be carefully planned in order to cross the entire race in the most efficient manner possible. The strategy simulation tells the strategist the optimal speed for the solar car throughout the entire course.

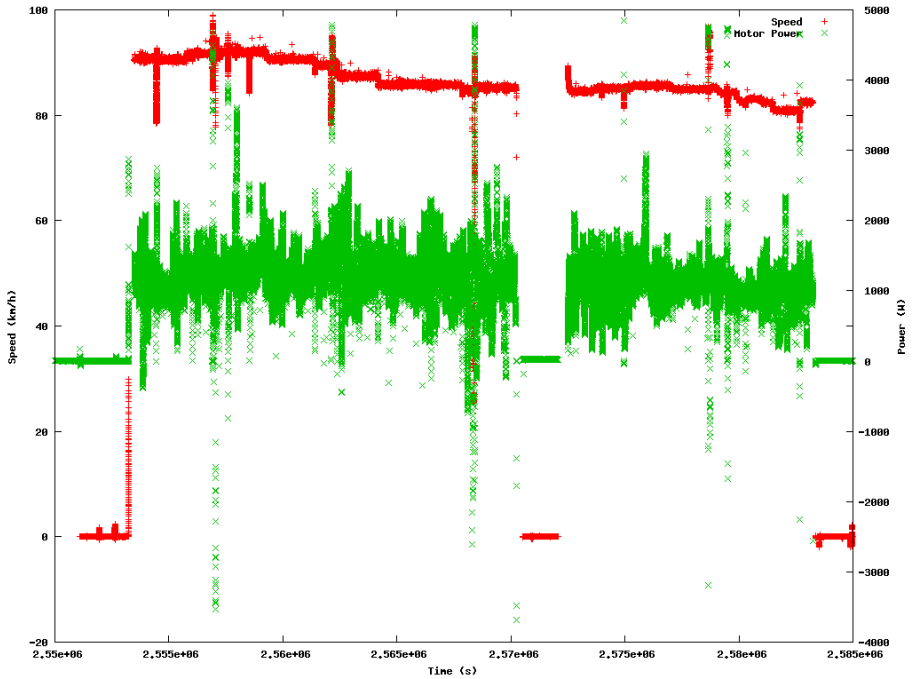


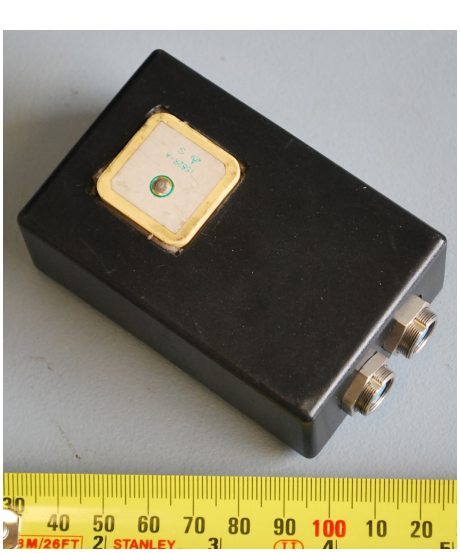
Figure 2.3: An example of telemetry data used in simulation: car speed (red) and motor power (green). This graph shows the data from Day 4 of the 2009 WSC.

2.3 Telemetry

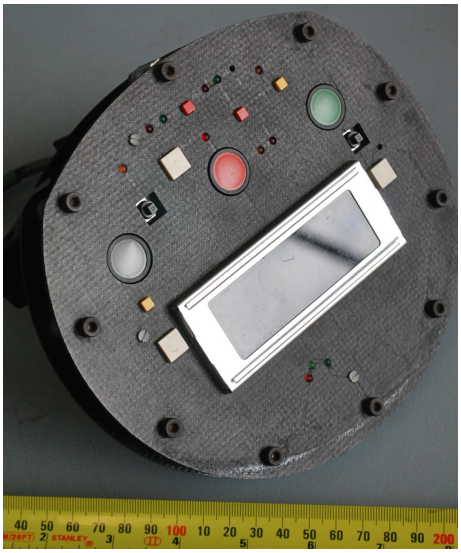
Forming a strategy simulation requires a mathematical model of the car. The simulation input has to be updated constantly with real life data while the race is running. This requires constant feedback from multiple sensors within the solar car to monitor the state of the solar car. Gathering all necessary information for the simulation is the job of the telemetry system. *Sunswift IV*'s telemetry system was designed in 2002 by then undergraduate student David Snowdon, with the biggest change being the introduction of a *controller area network* (CAN) bus with a custom protocol as the backbone of the telemetry system [Snowdon, 2002].

CAN was developed by Robert Bosch GmbH as a reliable communication network for automotive purposes. It is a half-duplex protocol using differential signalling with non-destructive collision detection [Pazul, 2005]. *Sunswift IV*'s telemetry system consists of a number of nodes which are connected by a single CAN bus. When a CAN node sends information through the bus, all other nodes will receive the message. It is up to the receiving node to filter the messages intended for itself from messages intended for other nodes. This is done using a custom protocol called SCANDAL, which is implemented on top of the CAN protocol and performs various functions such as message addressing while providing a simple abstraction to the application developer.

The CAN bus used in the car is currently configured to run at 50 kb/s, far below the maximum possible speed of 1 Mb/s. It also uses a 29 bit extended header as opposed to the 11 bit standard header length. Most of the nodes in the car are sensor nodes, which are designed to measure important parameters in the car (such as the battery voltage, motor power, and car speed). Other nodes perform different tasks, such as receiving driver's input through the steering wheel and acting as the dashboard [Wrigley, 2009]. One of the most important nodes in the car facilitates the transmission of data between the solar car and the control car. Currently, this is the responsibility of the PLEB node, as explained below.



(a) GPS



(b) Steering wheel

Figure 2.4: Two examples of telemetry nodes within *Sunswift IV*.

2.4 PLEB

The *Pocket Linux Embedded Box* (PLEB) is an embedded Linux platform developed by Adam Wiggins at the UNSW School of Computer Science and Engineering in 1998 [Wiggins, 2001]. The project aimed to build a pocket sized computer capable of running the Linux kernel. It is based on the StrongARM SA-1100 processor from Digital Equipment Corporation and runs Linux kernel version 2.4. The PLEB was designed to support additional hardware by using stackable expansion boards, similar to the modern Arduino platform. There are two expansion boards used in *Sunswift IV*. The first one provided an interface to the CAN bus using a SJA1000 CAN controller from NXP Semiconductor (formerly Philips). The other board provided Ethernet connectivity using an SMSC Ethernet controller.

The PLEB was used in the solar car to convert the data carried by the CAN protocol into the Ethernet protocol. This allows connectivity to the control car by transmitting Ethernet frames through a Netgear WG102 Wi-Fi *wireless access point* (WAP) connected to a directional Wi-Fi antenna facing the control car. An omni-directional antenna is not required as the control car is always positioned behind the solar car on the Stuart Highway, which mostly consists of long straight stretches of road. Thus, the wireless link will

always come from the same direction. In this case, a directional antenna would provide higher signal gain compared to an omni-directional antenna. It is also more resistant to the effect of electrical noise generated by the electric motor and motor controller. The antenna is positioned behind the driver's helmet, facing backwards toward the control car. It is connected to the WAP using a coaxial cable. This allows streaming of telemetry data to the strategist through another similar access point and fed back into the strategy software.

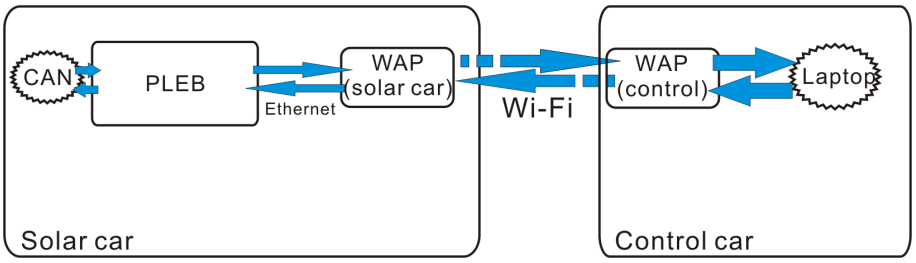


Figure 2.5: The wireless transmission scheme between the CAN bus and strategist's laptop using the PLEB and two wireless access points.

The streaming process is handled by a program running on the PLEB which uses the Berkeley sockets API to interface with the Ethernet connection. The CAN-Ethernet interface is handled by a custom protocol library called CANBRIDGE2, which was developed by Luke Macpherson in preparation for the 2003 World Solar Challenge. This system allows bidirectional data transfer, though the vast majority of the data consists of telemetry packets from the solar car to the control car. However, the strategist may occasionally send messages to nodes in the car, such as to change parameters on one of the nodes.

2.5 Issues with the current design

UNSW SRT has one PLEB device which was added into the telemetry network in 2003. As of the start of this thesis, it has been used in each race and has performed its duties without any major failure. However, there were several issues that became apparent in this system which are described below:

1. The telemetry data occasionally stopped streaming. This issue is likely caused by a bug somewhere within the software stack on the PLEB, as the wireless connection was usually found to be still working in this case. Using `ssh` to restart the

program responsible for sending telemetry data almost always fixed this problem. Furthermore, at least one part of the system was affected by electrical noise, with the data stream cutting out in the event of high current in the motor during heavy acceleration.

2. Any downtime in the data stream from the PLEB resulted in loss of telemetry data, as there was no data logging device in the car. This was made worse by the fact that the control car needs to refuel, which involves leaving the solar car at a control stop and travelling to a service station, which is usually beyond the maximum range of Wi-Fi and almost always not within line of sight.

3. The PLEB is no longer under active development. There is no documentation available and the developers have gone on their separate ways, so no changes to the software or hardware is possible without significant effort.

4. Some parts of the PLEB are no longer available for purchase, such as the StrongARM SA-1100. This means that no additional PLEB devices can be made without some major redesign.

Due to these issues, a replacement for the PLEB was needed. The blackouts in telemetry cannot be fixed in the current device without a significant effort. This loss of data greatly reduces the accuracy of the strategy simulation. Furthermore, the lack of a backup device means that if the current PLEB fails, the team would be left with no feedback on which to base the strategy simulations.

Unfortunately, this exact scenario happened during the course of this thesis. The PLEB suddenly became unresponsive on one driver training session, with no telemetry data received on the laptop. It also did not respond to ping requests or an `ssh` connection attempt. Connecting the PLEB to a serial connection shows that the PLEB was encountering filesystem errors and was unable to open a console. As none of the original developers and documentation are available, there was no method available for us to access the filesystem within the PLEB. For this reason, we were unable to perform any recovery attempt on

the PLEB. Thus, a replacement for the PLEB was urgently required as the team would not have any telemetry feedback otherwise.

2.6 Solar car telemetry systems

All solar cars in the World Solar Challenge are subject to the same race conditions and regulations. Hence, every team will face the same challenges in getting data from their solar cars. Different teams have tried different approaches for transmitting their telemetry data, some of which are outlined below.

2.6.1 University of Michigan, 2007 (Wireless mesh network)

The University of Michigan Solar car team was sponsored by Motorola, who provided them with a variety of communications hardware. The team took advantage of this and designed their telemetry system around Motorola's wireless mesh products [Motorola, 2008]. The mesh network performs data communications among all vehicles in their fleet, as opposed to just the solar car and control car. There is no mention of a data logging device in use in the solar car, though it is likely that one was used considering the size, competency and experience of the Michigan team.

2.6.2 Honda Dream, 1993 (Radio modem)

Honda, two-time WSC winner, uses a custom built telemetry system on their 1993 car [Shimizu et al., 1998]. Their system uses two devices that handles two different data streams duplexed over a single antenna. This setup was intended to reduce weight, as opposed to using two antennas. Telemetry data is transmitted directly to the control car, where it is logged. There is no data logging unit located inside the car, presumably to save weight. It is worth noting that when this race took place, hard drives had not yet reached 1 GB in capacity and small, lightweight flash memory was not available.

2.6.3 Nanyang Technological University, 2009 (Radio modem)

Nanyang Solar Team has a telemetry system based on the National Instruments CompactRIO hardware [Ren, 2010]. The team uses a pair of radio modems to transfer data to the support car. The system also has on-board data logging with 2 GB capacity.

2.6.4 Summary

Most solar car teams use off-the-shelf wireless devices to transmit telemetry data from the solar car to the control car following it. Michigan's mesh network is slightly different as it allows communication between all cars in their fleet. This is not necessary for the UNSW SRT, as the team only needs communication between two cars. It is unlikely that the team will use a wireless mesh network for this reason.

The radio modems used by Honda and Nanyang provide enough functionality for them to run their simulations. This is especially true in Honda's case as they managed to win the 1993 race with the system described above. While neither team uses a CAN bus for their on board telemetry, Nanyang's usage of on-board data logging would be a good solution to the data-loss during control stops.

While there are more solar car teams in the world than the three outlined above, most of them do not publish anything describing their telemetry system. This is understandable, as most teams consider their technical systems to provide a competitive advantage in the race. In addition, solar car teams usually perform their work under time pressure, and thus would have little motivation to publish technical reports as opposed to putting the effort to further optimise their respective cars.

Chapter 3

System Design

As a result of David Snowdon's thesis [Snowdon, 2002], the team has a software and hardware framework for their on-board telemetry system. However, this framework does not cover the transmission of data outside the solar car. As previously mentioned, this role has been performed by a combination of the PLEB and WAP without any major changes for 6 years.

3.1 Defining requirements

The main goal of this thesis is to provide a better system to deliver data between the solar car and the control car on the road. The emphasis will be on the transmission of telemetry data from the solar car to the control car, though the system must be able to deliver occasional data packets in the other direction. In order to accomplish this, we shall first analyse the requirements for such a system. These requirements comes from the rules and conditions of the race, as well as the existing system already in place within the team.

1. The replacement must interface with the existing telemetry system which already exists in *Sunswift IV*. This covers the on-board telemetry network that has been in use since 2002, which has the following characteristics:

    - The replacement must be able to capture all data that is sent through the CAN bus, which runs at 50 kb/s.

- The replacement must work with SCANDAL, a custom CAN higher layer protocol used by the team.
- The replacement must be powered though the telemetry system power, which is in the form of a 12V DC bus.
- The replacement must be designed so it is turned on or off by providing or removing power. This is because the telemetry system is switched on or off by supplying or removing power instead of more complex methods such as digital signalling. Furthermore, this also means that the device must be able to handle sudden power loss, as there will be no indication of when the telemetry system will be turned off.

2. The replacement must be able to deliver data to the support crew who monitor the car. This should be done through some form of wireless communication, as the car needs to be monitored when it is moving. The support crew monitors the solar car through a regular laptop, which has an Ethernet and several USB ports. Currently there is no special purpose-built hardware used in the control car that is required for monitoring besides a Wi-Fi access point. Thus, the wireless system does not need to be backward compatible as long as it can deliver data to the support crew.

3. The replacement must be highly reliable, performing well in a mechanically harsh and electrically noisy environment. *Sunswift IV* has very little suspension travel and uses high pressure tyres, making vibration a possible issue. Furthermore, the electric motor and its controller emit significant electrical noise, as shown in Section 6.1.

4. The replacement must be available in multiple units. While only one unit is needed in the solar car, having multiple units allows the team to quickly replace the unit inside the solar car in case of failure.

5. The replacement should handle broken wireless links gracefully. This is necessary because the control car will need to refuel periodically, usually during control stops. As mentioned before, refuelling will break the wireless link between the solar car

and the control car. Thus, the replacement should at least ensure that data collected during these periods is not lost.

6. The replacement should be easy to use and require minimal maintenance. This is an important trait for the UNSW SRT because team members change regularly as old team members graduate from the university, being replaced with new team members which will require training.

7. The replacement should use a minimum amount of power. The PLEB itself consumes about 100 mA during operation, while the WAP consumes an additional 300 mA, both at 12V. Any replacement for the PLEB is expected to have a similar or lower power usage.

8. The replacement should not be excessively expensive. Spending about $1000 for a solution is reasonable, but spending $5000 should not be done unless there is a very good reason for it.

Comparing the requirements to the PLEB shows that the PLEB is capable of interfacing with the CAN network and the control car (through a separate access point). It is also reliable and low maintenance since it has been used since 2003 without any major modifications. However, it does not handle broken links gracefully, and at the time of writing no PLEB unit can be used for telemetry purposes as the last PLEB unit malfunctioned during the thesis period.

3.2 Potential replacements

As the PLEB is only a single component inside the telemetry system, any other device that can perform its duties can be used as a replacement. This opens up the possibility of taking existing hardware and implementing it as a replacement for the PLEB.

3.2.1 PLEB2

The PLEB2 [Snowdon, 2004] was the successor to the original PLEB. It is based on the Intel PXA255 processor and was designed for power management research purposes. The team has attempted twice to adapt the PLEB2 as a backup to the original PLEB, however neither project was completed due to time constraints. Its design was very similar to the original PLEB, with a separate AVR microcontroller used for power management and monitoring purposes. While there are several PLEB2 units available to the team, the project itself is no longer under active development and some critical components such as the PXA255 have reached end-of-life. Had it still been in active development, it would have been a good candidate to replace the aging original PLEB due to its similarity.

3.2.2 RMCAN Canlink WLAN

RMCAN is a German company making communication products based on the CAN bus. The team purchased a pair of their CANlink WLAN interfaces, which are supposed to connect a CAN bus to a Wi-Fi interface [RM Michaelides, 2010]. However, the team was unable to get the device to communicate with *Sunswift IV*'s telemetry system as the software inside the device was found to contain numerous bugs and issues. These issues were not resolved even when members of the team visited the company's headquarters in Germany during an unrelated trip. Thus, this device is not suitable as a replacement since it cannot fulfil the first requirement.

(a) PLEB2 (b) CANLink WLAN (c) Kvaser Blackbird

Figure 3.1: The three possible off-the-shelf alternatives to the PLEB.

3.2.3 Kvaser Blackbird

Kvaser is another company making CAN communication products [Kvaser, 2010]. Their Blackbird product connects a CAN bus with a Wi-Fi interface, not unlike the CANLink WLAN. The device does not have any data logging functionalities. It is theoretically possible to use this product alongside a separate CAN data logger, however this requires purchasing two devices or designing a separate data logger device to be used simultaneously. This would cost the team a considerable amount of money.

Furthermore, the device also lacks an external antenna connector. This means that the device cannot be used with the existing directional Wi-Fi antenna, which has been used with the WAP. This will reduce the available signal strength, and may present interference issues with electrical noise. It is not possible to determine the magnitude of this issue without purchasing a unit and performing on-road tests.

3.2.4 Custom design

There are a number of advantages to replacing the PLEB with a custom designed solution:

- The team will have complete access and control over the technical design;
- The hardware and software design can be customised to fulfil the necessary requirements;
- Development and maintenance will be much easier since all design documents, schematics and source code will be available;
- The device can be made cheaper than buying a commercial solution, due to the abscence of various costs associated with running a commercial entity.

However, there are also some disadvantages of designing a custom solution:

- The hardware and software solution might not be as well-tested compared to a commercial solution.
- There is more work involved in making a custom device as opposed to purchasing an off-the-shelf solution.

3.2.5 Summary

The PLEB2 and CANlink WLAN are not suitable as a replacement to the original PLEB. Although the Blackbird could be used, it would require purchasing multiple products, which would be very expensive, costing several thousand dollars. Furthermore, the Blackbird is more susceptible to interference issues as it cannot use an external antenna. Due to these reasons the Blackbird was also deemed unsuitable as a PLEB replacement.

As the commercially available hardware would not adequately solve the problems affecting the current system, the best course of action was to design a custom device. Having complete control over the design process also allows us to create a solution that can definitely fulfil all of the requirements, as opposed to using an off-the-shelf solution that may fulfil some, but not all of the requirements. Furthermore, a custom solution is much easier to maintain as all design documents are available. These advantages justify the amount of work that needs to be done for a custom design. The reliability of this solution could be improved by subjecting it to a large number of tests before the race.

3.3 Proposed system

The main goal of this thesis is to deliver data between two moving cars. This used to be performed by using the PLEB, two WAPs, and the CANBRIDGE2 protocol. As the PLEB acted only as a bridge between the CAN bus and Ethernet, we can design a new hardware device that can perform the same role. This device can then replace the PLEB directly without any further modification to the system.

However, technology has improved since the PLEB was first used within the team. For example, wireless communications are now commonly found within embedded systems, making it possible to avoid using a separate wireless access point. To take advantage of these improvements, we did not limit ourselves into designing a simple CAN to Ethernet bridge. Instead, we looked into designing a *system* to deliver data between the two cars. We can then design any necessary hardware as a part of that system.

The system should provide the capabilities that can fulfil the requirements outlined in

Section 3.1. Most importantly, it needs to deliver data from the CAN bus to the strategist's computer while the car is on the road. This requires the use of wireless technology. Thus, we will require two wireless transceivers in the two cars to bridge the air gap.

These transceivers are unlikely to stay connected when the control car is leaving the solar car on a refueling trip, as the petrol station might be located several kilometres away from the control stop without any line of sight. Keeping a connection during this period would require a high powered transceiver. Alternatively, two wireless transceivers might be used, a low power transceiver for normal operation and a high power transceiver for out-of-sight operation. This approach is undesirable as it introduces the additional complexities of using and switching between two different technologies. Thus, it is easier to accept that the connection will be broken during refueling as opposed to trying to create an unbreakable wireless connection.

The proposed method to circumvent a broken wireless connection is to keep a copy of the telemetry data on-board the solar car. Then, when the connection is broken, a copy of the data will still exist on the solar car. The data can then be retrieved through the wireless connection when the control car comes back within wireless transmission range, or alternatively at the end of the day when physical access to the solar car becomes possible. On-board memory for the data should be a form of nonvolatile memory, so data will not be lost if the car is turned off for any reason.

The transceiver device in the solar car has to be designed from scratch. This custom device was named *Sunswift IV Observer Node* (SION). SION needs to perform various tasks in order to deliver data. To aid the process of designing the hardware, we can split the requirements for such a device into several subsystems as follows:

- A CAN interface is needed to access the CAN bus.
- A wireless interface is needed to send data remotely.
- A processing unit is required to manage the flow of data between the two interfaces. This can be in the form of a microcontroller or an embedded computer.
- A persistent memory is required to store data in case of telemetry breakup.

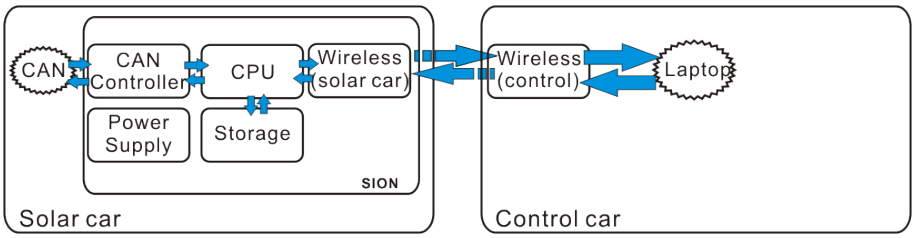


Figure 3.2: The overall system block diagram, showing the subsystems as separate blocks. The blue arrows represent the flow of data.

- A power supply circuit is required to power the other subsystems.

The transceiver in the control car can utilise off-the-shelf hardware, provided it can interface with the laptop and the wireless transceiver of choice in the solar car transmitter. In this scheme, the CAN interface allows the device to receive data from the CAN bus. The data will be recorded in memory and sent through the wireless transceiver to the laptop. The nonvolatile memory on board the solar car will ensure that a record of the data will still exist when the wireless link is broken. SION will act as a replacement of both the PLEB and wireless access point in this system.

Chapter 4

Hardware Development

As previously mentioned, SION is a custom hardware design which is made from several subsystems. Each subsystem requires an electronic hardware device which can perform its role. In turn, these hardware devices provide a platform to run software that performs the actual manipulation of telemetry data. This chapter describes the hardware devices in each subsystem, including the design decisions that were made during development.

4.1 CAN interface

The CAN subsystem's role is to interface with the CAN bus inside the solar car. This role requires the use of two hardware devices; a CAN controller and a CAN transceiver. The CAN controller handles all the intricacies of the CAN protocol such as packet structure, checksumming, and collision detection. The CAN transceiver handles the electrical signalling of the CAN bus such as converting between serial signalling from the CAN controller to differential signalling used on the actual CAN bus. The controller and the transceiver are independent of each other. Thus, it is possible to change the CAN controller without changing the CAN transceiver or vice versa, as long as both of then conforms to the CAN specification.

Sunswift IV currently uses the SN65HVD232 CAN transceiver *integrated circuit* (IC) from Texas Instruments. This transceiver was chosen back in 2002 as it was one of the first CAN transceivers available on the market that works with a 3.3 V voltage level, which

was the operating voltage of the microcontroller used in the nodes in the telemetry system. The team has never encountered any problems with this particular transceiver. Thus it was kept as the CAN transceiver for SION.

The MCP2515 from Microchip Technology is currently used as the CAN controller for all microcontroller-based nodes in the solar car. It is one of only two standalone CAN controller ICs available on the market, the other being the SJA1000 that was used in the PLEB. Both controllers have internal transmit and receive buffers so multiple messages can be queued during transmission / reception. The MCP2515 has a *serial peripheral interface* (SPI) bus which is easy to interface to a microcontroller as opposed to the memory bus used in the SJA1000. They also have different amounts of buffer, with the MCP2515 able to hold 2 CAN messages in its receive buffer, while the SJA1000 has space for about 5 messages depending on their size.

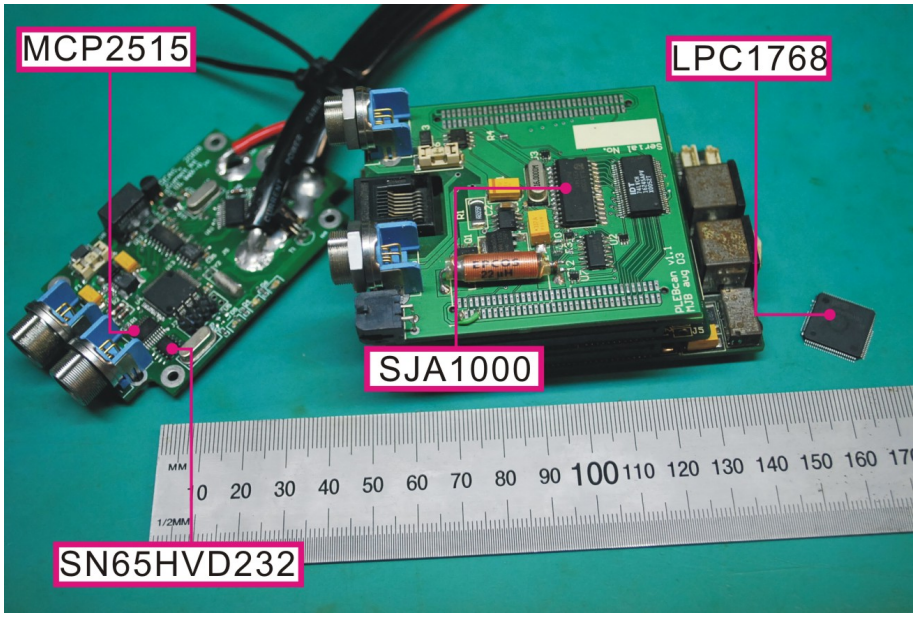


Figure 4.1: The three potential CAN interface hardware for SION, showing the MCP2515 on a current sensing node, the SJA1000 on the PLEB, and the LPC1768 microcontroller. The SN65HVD232 CAN transceiver can also be seen on the current sensing node.

An alternative to using standalone CAN controllers is to use microcontrollers with an integrated CAN controller. These microcontrollers were not available when the telemetry system was designed, which is the reason why the MCP2515 and SJA1000 were required. One microcontroller suitable to use as a CAN interface is the LPC1768 from NXP [NXP Semiconductors, 2010]. It is based on the ARM Cortex-M3 architecture running at speeds

up to 100 MHz and has 64 kB of RAM, 512 kB of flash memory, and a large number of peripherals. As the LPC1768 is based on the ARM Cortex-M3, it is compatible with the *Cortex Microcontroller Software Interface Standard* (CMSIS), which is a set of C libraries which provide hardware abstraction layers and peripheral drivers [ARM Ltd., 2010a]. This greatly reduces the amount of coding work as the hardware interface is standardised and subroutines to interface with the hardware are already available. Furthermore, the LPC1768 is widely used as the microcontroller of choice in the *mbed* prototyping platform, which can be used as a working reference to the hardware implementation as the schematics of the *mbed* are publicly available [ARM Ltd., 2010b].

By comparing the three alternatives, the LPC1768 was chosen to perform the role of the CAN controller. This choice is made primarily because using the microcontroller allows a much larger buffer space to be used. The LPC1768 has enough RAM to store more then 1000 CAN packets while still having enough space for numerous variables used in its firmware (which is described in Section 5.2).

Having a large amount of buffer reduces the frequency of interrupts on the processing unit. Processing units with long processor pipelines have a large overhead when handling interrupts, as the contents and state of the pipeline have to be stored and restored for each interrupt. On the other hand, microcontrollers such as the LPC1768 very short pipelines, which allow them to respond to interrupts very quickly. Dedicating a microcontroller as a CAN interface provided less restrictions on the choice of the processing unit subsystem. Furthermore, the CAN interface registers in the LPC1768 can be accessed directly by user code, as opposed to accessing registers through an SPI or memory bus. This simplifies software development for the CAN bus interface. The hardware design is based on schematics of the *mbed* platform, which has been shown to work in public.

4.2 Wireless interface

The wireless subsystem is required to transmit data to the solar car. To perform this role, the subsystem needs to provide a high enough data rate and be strong enough to transmit data across the distance between the solar car and the control car, which can be as far

as 90 metres. These two requirements rule out several low power and short range radio transmitters such as Bluetooth.

In choosing the suitable wireless technology to use, priority is given to the more popular technologies over the more specialised ones. Popular technologies have a much larger number of implementations and better support. This is important since designing hardware with wireless functionality is a very complex process, caused by both technical design issues and government regulation of the radio spectrum.

One of the most popular wireless technologies in the world today is Wi-Fi, commonly used for wireless networking and internet access. Wi-Fi transceivers have sufficient range to transmit data between the solar car and control car, as shown with the PLEB utilising a pair of wireless access points. They also have a high data rate, with the oldest specification (IEEE 802.11b) having a theoretical maximum speed limit of 11 Mb/s. This is far beyond the 50 kb/s maximum data rate on the current CAN network. Furthermore, using Wi-Fi allows us to reuse networking hardware and software that was used in the previous system, such as the directional panel antenna, cabling, and source code. Due to these characteristics, Wi-Fi was chosen as the wireless protocol used in the system.

While there are countless Wi-Fi implementations on the market, not every device is suitable for use in a solar car. A lot of Wi-Fi devices are sold as standalone units with a bulky size and weight, such as the access point used alongside the PLEB. Other implementations come in the form of USB-based dongles, which are normally used alongside a computer. However, these dongles required custom hardware drivers to operate, which may not be available for an embedded system as these dongles are meant to be used on a normal desktop or laptop computer.

A better solution is to use embedded Wi-Fi modules. These are small pieces of hardware which are designed to provide Wi-Fi connectivity to custom embedded devices. Some examples of Wi-Fi modules include the WiFly GSX from Roving Networks, and the Nano WiReach from ConnectOne. Both feature a *universal asynchronous receiver / transmitter* (UART) interface, an external antenna connector, and an internal *internet protocol* (IP) stack. This allows devices without an IP stack to interface with Wi-Fi by sending

data through the UART connection.The WiFly GSX has an ability to wake up, connect to a wireless network, send data and return to sleep mode in less than 100 milliseconds. On the other hand, the Nano Wireach supports *serial peripheral interface* (SPI) which allows higher data rates than UART. Both of them are suitable for the role of the wireless interface on SION, while the old wireless access point was kept as the wireless interface on the control car.

4.3 Processing unit

The processing unit is required to handle the flow of data between the CAN interface, the wireless interface, and the nonvolatile memory. The choice of processing unit required analysis of features necessary for the operation of the device. One of the most important decision in this analysis is the choice between using a bare-metal approach or an operating system. The two approaches will require different hardware for the processing unit.

4.3.1 The bare-metal approach

In bare-metal systems, the program code interfaces directly with the hardware. This gives it a large amount of control over the behavior of the hardware, and by extension the entire device. This is useful to control single purpose hardware (such as the nodes inside *Sunswift IV*) as opposed to general purpose hardware (such as a desktop computer). In our case, SION can be considered a single purpose hardware, as it has a specific set of requirements that are unlikely to change over the life of the device. As the hardware is known, the program code can be optimised to deal only with the specific hardware design. This has several advantages, such as allowing the system to boot in a relatively short time and could make it easier to trace bugs, as there is less code in the device.

Bare-metal approach is commonly used on microcontrollers. In Section 4.1, we have already chosen a microcontroller to be used as the CAN interface. This microcontroller, the LPC1768, could perform the role of both the CAN interface and the processing unit, as it already has a CAN controller and is powerful enough to handle a constant stream of

data. The wireless interface could then be connected to the microcontroller using UART. The LPC1768 can also take advantage of libraries designed for embedded systems such as the Newlib C library, which can provide a subset of C library functions without the need for an operating system [Johnston, 2010]. However, it lacks enough RAM and flash memory to run large operating systems such as uClinux.

4.3.2 Use of an operating system

The direct hardware interface used in the bare metal approach can be considered a double-edged sword. The ability to interface with all registers that control the hardware also means that the developer *has* to control all necessary registers that are required for the operation of the hardware. This is a time-consuming and error-prone process, especially since modern processors are complex devices. To reduce the complexity faced by developers, special software known as an *Operating System* (OS) is available. OSes run directly on hardware and provide a simple abstraction of the hardware to the programmer. It also manages hardware resources and allocates them as necessary.

In embedded applications, OSes are used on devices that need to handle complex tasks (such as artificial intelligence), provide a general purpose platform (such as smartphones), or simply taking advantage of a feature already built into an OS without reinventing the wheel (such as a graphical user interface on an oscilloscope). SION is not a general purpose platform and does not need to handle highly complex tasks. However, it can take advantage of the abstraction layers provided by an OS, making software development much easier compared to the bare-metal approach.

OSes are commonly run on professionally designed / off-the-shelf hardware, due to the complexity of designing with relatively high-end components (which have many more issues to take into account such as signal integrity, board layout and manufacturing with leadless IC packages). A suitable hardware candidate for an OS-based approach is the Gumstix Overo Air [Gumstix, 2010]. It is a single board computer based on the OMAP3503 processor with 256 MB of RAM and 256 MB of flash memory. Gumstix also provides support for Linux as the recommended OS on the Overo Air. The Overo

Air is deemed more suitable than numerous single board computers in the market since it has a wireless chipset which provides WiFi and Bluetooth connectivity. This completely eliminates the requirement for a separate wireless module, which reduces cost and complexity. It also provides a microSD card slot, which can act as persistent memory for the telemetry data in addition to the onboard flash memory. This eliminates the need for separate hardware to provide the functionality of the persistent memory subsystem. Since the Overo Air is designed to be embedded into custom hardware design, it has a very small physical size no bigger than a stick of gum (hence the name Gumstix). This is useful as it is very lightweight, which will result in a lighter and faster solar car, as can be seen on Figure 4.2.

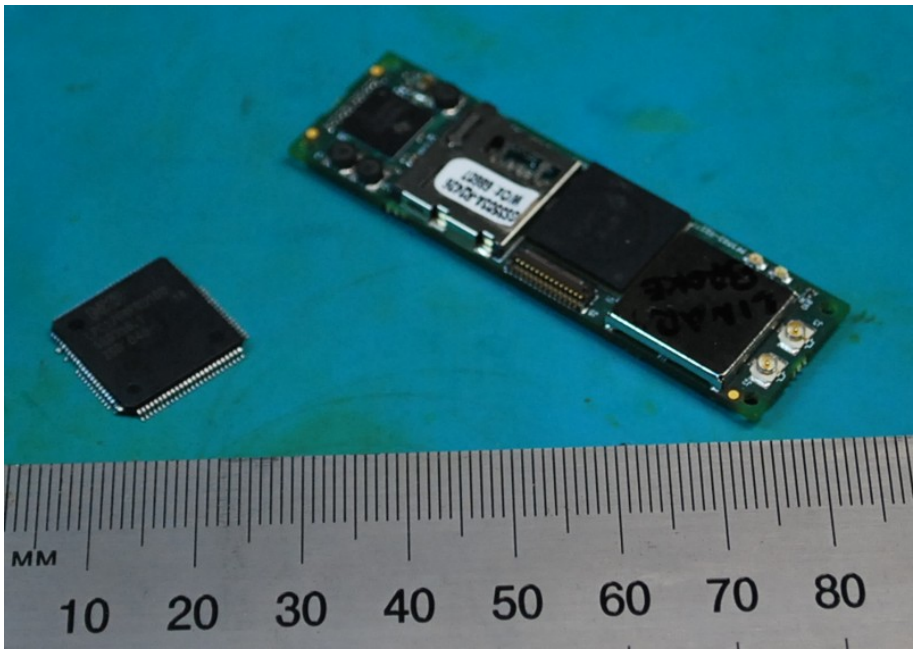


Figure 4.2: The two alternatives for the processing unit; the LPC1768 microcontroller and Overo Air single board computer.

4.3.3 Summary

Both approaches outlined above are capable of performing the tasks required to replace the PLEB. However, using an OS on the Overo Air as the processing unit has more advantages over the bare-metal approach using the LPC1768. Using an OS greatly reduces the amount of software development required as it gives a clean abstraction layer and allows code reuse from the old software, including the CANBRIDGE2 protocol. The Overo Air also has

integrated Wi-fi and microSD interface, reducing the amount of hardware design needed. Due to its advantages, the Overo Air was chosen as the processing unit subsystem, with the LPC1768 used only as a CAN interface. The two hardware devices are connected through a UART bus.

In this hardware configuration, both the LPC1768 and the Overo Air have extra unused UART ports. These were used as a serial console interface, which is essential during development and debugging. This is facilitated by using the FT2232 IC from FTDI, which can bridge two UART interfaces into a single USB interface [FTDI Ltd., 2010]. When the USB interface is connected to a host PC, the two console interfaces will appear on the host PC as two virtual serial ports.

4.4 Power supply subsystem

The CAN bus has a 12V power rail which distributes power to the low voltage electronics in the car. In each node, power supply circuitry steps down the voltage to 3.3 Volts, which is the operational voltage of the on board components. Most nodes in the car use either a buck converter (on most nodes) or linear regulator (on nodes that require low noise power, such as a current sensing node). However, SION has some extra requirements of the power supply, which will be outlined below. A new power supply design is required to fulfill these requirements.

4.4.1 Backup power

Sunswift IV has a DC-DC converter node which converts high voltage DC (about 125 V) into 12 V for powering low voltage devices in the car. This node is powered directly from the battery through a single circuit breaker which is used as a switch and for over-current protection. Toggling the circuit breaker will immediately cut power to the 12V power line and all nodes attached to it. All nodes on the car are turned on or off by providing and removing power to them. This simplifies the car wiring and design since no complex procedures are required to turn the car on or off. The abrupt power cut is not a problem

since most of the electronic devices in the car are microcontroller-based nodes which run a fixed program and do not store data internally.

However, this may not the case with SION. It would not be able to handle sudden losses in power due to its data logging operation, which perform continuous write operations to a SD card. A power outage during a write operation can result in corruption and loss of data. While there are software techniques to prevent this, such as the usage of a journaling file system or parity files, those software techniques are normally used to recover from occasional data corruption, not regular power loss. SION is expected to be switched off regularly, so a more robust protection scheme is required. The chosen method is to provide a backup power source. Using this method, SION will sense the loss of power and stop receiving data, as all other nodes on the car will be powered off. It can then flush any unwritten data to the SD card and finish any necessary write operations before losing power and turning off.

4.4.2 Electrical isolation

Electronics inside a solar car can are subjected to a variety of harsh conditions. Besides heat and vibrations, electronics in *Sunswift IV* are located in close proximity to high voltage electricity lines (potentially up to 160V). Normally, high and low voltage electronics are put on separate wiring, with any connection between the two sides facilitated by galvanic isolation devices. However, it is possible that the high voltage circuitry could come in contact with the low voltage electronics in the event of an accident.

One such case happened in the first day of the 2009 World Solar Challenge. The steering wheel electronics suddenly failed, costing the team 40 minutes as the steering wheel was changed on the side of the road. It was later found that some insulation on the high voltage wiring on the solar array had been stripped due to rubbing against the conductive carbon fibre chassis. The chassis was also accidentally shorted to the CAN network through the connector of the wireless antenna cable, which was electrically connected to ground. Thankfully, no permanent damage was done to the electronics. Had the PLEB or other critical telemetry systems been damaged, *Sunswift IV* would have been

less likely to achieve victory in 2009.

To prevent failure due to accidental high voltage exposure, SION is designed to use galvanic isolation on its CAN interface, which is provided by using the IL700 series magnetic isolation ICs from NVE [NVE Corporation, 2010]. This is used as a precaution to protect the data stored on the SD card. Since the rest of the electronics are protected behind the isolation IC, write operations will not cease abruptly due to voltage spikes. This prevents data corruption on the filesystem.

The only weak point in this scheme is the connector for the antenna. This is a coaxial cable which connects the Wi-Fi interface on the Overo Air to the panel antenna on the solar car through two metal connectors. The outer sheath of the coaxial cable is connected to the same ground as the Overo Air. As it is impossible to isolate analog signals, the metal connectors were wrapped in heatshrink tubing, which covered their metal surface and prevented them to make electrical contact with any other object.

4.4.3 Power supply design

The finished design consists of 3 stages. The first stage provides initial voltage conversion from 12V to 5V and galvanic isolation. It is based on a TDK-Lambda CC10-1205SF-E isolated DC-DC converter module [TDK-Lambda, 2010]. This module was chosen due to its high power rating (10W), which alows it to charge the backup supercapacitors (described below) within 3 seconds.

The second stage provides backup power in the form of two supercapacitors connected in series. An LTC4425 supercapacitor charger chip is used to control the charging current and perform voltage balancing on the supercapacitors [Linear Technology, 2010]. Connecting the supercapacitors in series will double their voltage from the initial 2.5 V to 5V, at the cost of reducing their capacity by half.

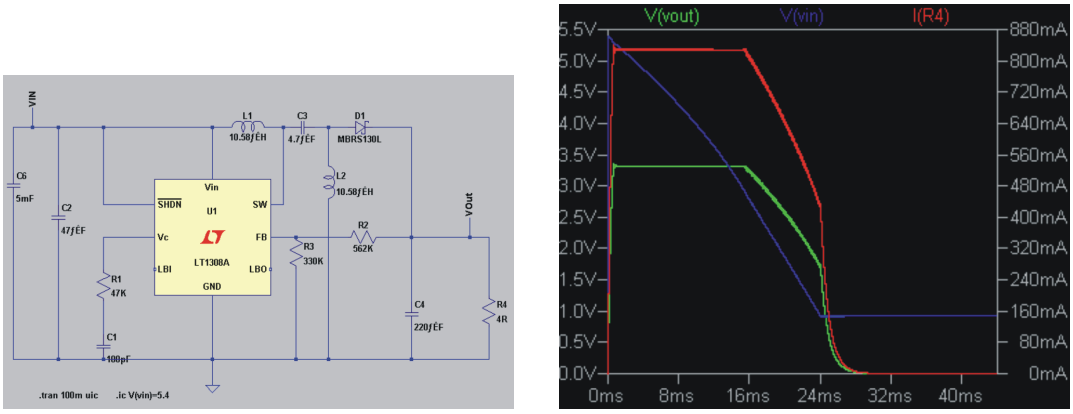
Supercapacitors are used in order to prevent a clash with the race regulations, which states that the capacity of all batteries that are present in the solar car must be taken into account when designing the main battery-pack [World Solar Challenge, 2010]. Using a battery for backup purposes could make the design of the main battery-pack more difficult,

and since a new battery pack is designed for every race, the issue will affect future battery designs. Temporary energy storage devices such as supercapacitors do not fall under this restriction. The supercapacitor sizes are calculated to provide 2 seconds of backup power. This provides more than enough time for the Overo Air to finish any necessary write operations, as the amount of telemetry data cache in the software never exceeds 1 kB. Even on a relatively slow SD card, such as ones with speed class 2 (2 Mbit/s), writing this amount of data will not take more than 5 ms [SD Association, 2010].

The third stage is based on a boost controller used in a *single-ended primary-inductor converter* (SEPIC) configuration [Analog Devices, 2010]. The SEPIC configuration allows this stage to produce a 3.3 V output from a range of input voltages, both higher and lower than the output voltage. This is necessary to extract as much energy from the supercapacitors as possible since the voltage in any capacitor will drop rapidly as its energy is drained. This is also the reason why the supercapacitors are connected in series, as having higher voltages makes it possible to extract more charge before the SEPIC regulator shuts off.

Initially, the SEPIC converter was based around the LT1308A boost converter IC [Linear Technologies, 2010]. This IC was chosen in combination with the LTC4425 as both of them were made by Linear Technology, which provides a circuit simulation program called LTSPICE. This program was used to test whether the combination of supercapacitor and SEPIC converter is feasible as a backup power scheme. While the result was favorable, the LT1308A turns out to be unable to provide enough current to the electronic subsystems contrary to the specification in the data sheet.

To remedy this issue, a replacement SEPIC regulator hardware was designed and manufactured based around the ADP1621 SEPIC controller [Analog Devices, 2010]. The hardware forms a breakout board that was inserted on top of the previous SEPIC circuit, which was disabled by removing the IC. Since the ADP1621 itself only works at a minimum of 2.9 V, the IC was pit in a bootstrap configuration. This term comes from the ADP1621 data sheet, which describes a configuration where both the input and output lines are connected to the IC power input pin through two separate diodes. This configura-



(a) Circuit (b) Simulation trace

Figure 4.3: An example of circuit simulation in LTSPICE. The simulation timespan is deliberately shortened to save memory.

tion allows the ADP1621 to continue working even when the input voltage drops below 2.9 V as the IC will be powered from the 3.3V output line. The lower input voltage limit is tested to be around 2.5V, which allows it to extract more power than the initial 2.9V limit. At lower voltages the circuit does not have enough boost ratio to output 3.3V, effectively shutting down the other electronics.

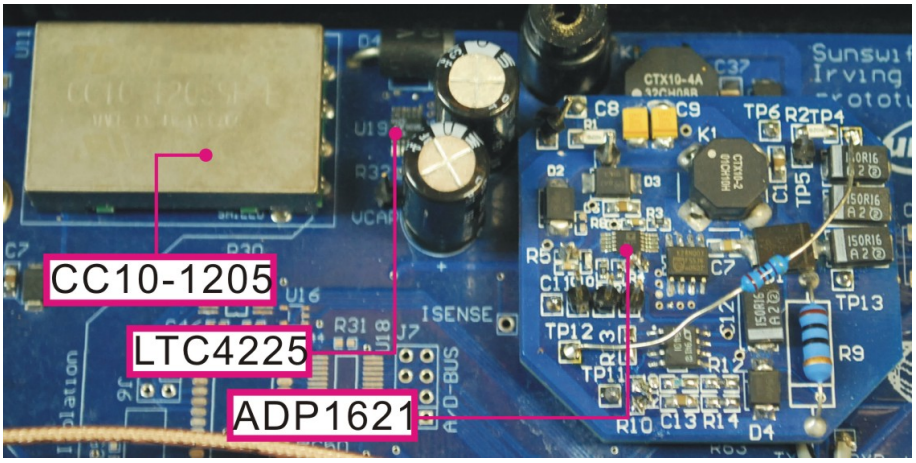


Figure 4.4: The SION power supply circuit: CC10-1205 on the left, LTC4425 on the middle, and ADP1621 on a breakout board on the right. The previous LT1308A circuit is located below the breakout board.

4.5 Summary

All hardware subsystems identified in Section 3.3 have been filled, as can be seen in Figure 4.5 below.

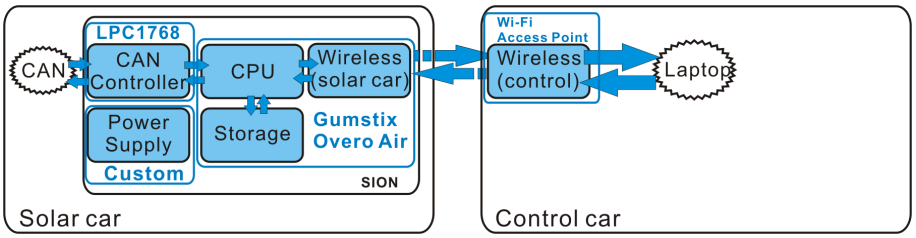


Figure 4.5: The system block diagram, showing the subsystems and the hardware fulfilling their roles.

These subsystems allow SION to fulfil the requirements outlined in Section 3.3. In particular, the NXP LPC1768 and SN65HVD232 complete the CAN subsystem, which allows SION to interface with the CAN bus, contributing towards requirement number 1. The Overo Air performs the role of the wireless, processing unit and persistent memory subsystem. This allows SION to deliver data wirelessly, contributing towards requirement number 2 and 5. The total cost of the hardware is estimated to be around $500, fulfilling requirement number 8. Some of these requirements are not yet complete, as they will require software implementation to be completed.

The schematic capture and *printed circuit board* (PCB) design was done using Altium Designer. These are available in Appendix A of this document. This contributes towards fulfilling requirement number 6, as they will allow debugging.

Chapter 5

Software Infrastructure

The hardware devices within the system hve the capability to deliver data from the control car to the solar car. This requires defining the behavior of the hardware devices in the system. Some hardware devices have a fixed function (such as the power supply), while others already have software written for them (such as the WAP in the control car). However, there are three hardware subsystems that require custom software development; they are the LPC1768 microcontroller, the Overo Air, and the laptop in the control car. This chapter describes the overall software design and specific implementations on the three subsystems.

5.1 Packet manipulation

CAN occupies layer 2 and part of layer 1 of the *Open Systems Interconnection* (OSI) networking model. It handles data transmission between nodes on a local network and partially specifies the hardware / signalling implementation. The developer can utilise these features to transfer a specific amount of data on each CAN packet. The content and meaning of this data has to be provided by user software.

This functionality is provided by the SCANDAL protocol, developed by Snowdon [Snowdon, 2002]. It provides facilities available in higher layer levels of the OSI model, such as addressing. SCANDAL also acts as an abstraction layer between the developer and CAN interface. Thus, a developer who needs to send data to a particular node only need to

call a function in SCANDAL with the data and addressing information. It will then translate them into a 29-bit header and 64-bit content, which is suitable for encapsulation in a CAN packet. SCANDAL also generates a timestamp for each data packet which forms part of the 64-bit content. This means that the data packets can be delayed or arrive out of order without affecting the accuracy of the strategy software.

Another protocol used within the system is the CANBRIDGE2 protocol. It provides a way to transfer the data within a CAN packet over a Wi-Fi interface by encapsulating both the 29-bit header and 64-bit content in a frame and transmitting it over Ethernet / Wi-Fi. The header and content will have to be decoded using a separate SCANDAL implementation on both ends. Thus, SCANDAL and CANBRIDGE2 provide two distinct facilities within *Sunswift IV*'s telemetry system. They can be interpreted roughly as occupying different layers on the OSI model.

The software in the system is designed to transmit telemetry data from one car to the other. Thus, the system has to provide functionality similar to the old CANBRIDGE2 protocol. Higher layer functionality in SCANDAL is not necessary to fulfil the design requirements.

5.2 CAN / Microcontroller

The LPC1768 delivers data between the CAN bus and the Overo Air over a UART. It has an internal circular buffer that stores CAN messages temporarily. This gives the Overo Air time to boot without losing any messages. When the Overo Air is ready, it sends a signal to the LPC1768 telling it to start sending telemetry data through UART.

The LPC1768 is also used to monitor the power status of SION. In case of power loss, it will send a power loss signal to the Overo Air through the UART, telling it that the telemetry system is powered off and it is running only on backup power. The Overo Air will then flush any unwritten data to persistent memory before backup power runs out.

5.3 Gumstix Overo

There are three major design decisions involved with the software on the Overo Air. The first is the choice of operating system used. The second is the method of data storage used. The third is the wireless implementation. They are descibed in three separate subsections below.

5.3.1 Operating system

Gumstix supports the Ångström Linux distribution on the Overo Air. It uses the OpenEmbedded (OE) build system, which automates the download, patching and compilation of source code into binaries specific for such a device [OpenEmbedded, 2010]. This is similar to the more well-known Gentoo Linux distribution. Several precompiled images with default configurations are also available on the Gumstix website.

However, the OE build system is very slow. It requires several hours to download and create an OS image. Even compiling a short C program took several minutes as it has a large amount of overhead. Furthermore, OE by default creates a binary installation package instead of a simple binary image. This creates an additional overhead during development as software under development needs to be compiled and tested frequently. For these reasons, OE and Ångström distribution are not used as the OS on the Overo Air, and an alternative OS has to be used.

Running an OS on the Overo Air requires four separate components:

The kernel is the core of an OS. It contains the scheduler, virtual memory manager, filesystems, device drivers, and other various features of an OS.

Rootfs is a file system containing various necessary binaries required for operating a computer. This contains all files and directories that can be seen normally by the end user. It includes binaries such as the shell, its utilities, and OS libraries.

U-boot is a second stage bootloader which loads the kernel and the rootfs. It can be configured to perform various tasks, such as providing custom boot parameters to the kernel.

X-loader is a first stage bootloader for the OMAP3 processor. Its only purpose is to load u-boot when the OMAP3 is first started.

The kernel used is based on the *linux-omap* branch of the mainline Linux kernel [Lindgren et al., 2010]. It is configured manually to remove unnecessary modules such as various device drivers. Initially, the kernel was also expected to perform *dynamic voltage and frequency scaling* (DVFS) to reduce the power consumption on the Overo Air by reducing the clock of the OMAP3 processor. This is feasible since the load on SION is relatively low and is I/O bound. However, it turns out that DVFS support was removed from the main OMAP linux kernel repository due to unknown reasons. Thus, implementing DVFS will require a kernel from another repository. One possible repository is the Maemo project, which was the OS used in the Nokia N900 phone [Nokia, 2010]. This is feasible since Maemo is based on Debian linux and the N900 uses an OMAP3 processor, similar to the Overo Air. Using this kernel will allow the software to perform DVFS, which can reduce the power consumption of the Overo Air.

The rootfs used is based on the Debian 6.0 Linux distribution [Debian Project, 2010]. Debian is chosen for this purpose as it is familiar to us and has a long history of working well, with a reputation for stability. The rootfs is created by using the Debian network installer, which downloads all necessary files and binaries from Debian repositories. This is much faster than compiling an entire rootfs from source code, which is the method used by OE. The network installer also provides a very minimal system, which minimises overhead as SION does not require a large number of features such as video or audio output. Additional packages can be installed with relative ease by using the `dpkg` package manager.

The u-boot and x-loader binaries do not require any modification. Thus, the binaries available on the Gumstix website were used as the bootloaders.

5.3.2 On-board storage

SQlite is used to record the telemetry data on both SION and the laptop. [SQLite, 2010] It is an embedded database engine written in C which is designed to be compiled into

applications. This allows the application developer to access an SQLite database through *structured query language* (SQL) statements passed through simple function calls. The database itself exists as a single file which is compatible cross-platform, independent of the architecture of the processor it is run on. SQLite also does not require any installation or configuration.

Using SQLite allows the application to very quickly load a specific subset of telemetry data. For example, a single SQL query can be used to fetch all data packet that corresponds to the battery pack voltage. This is the main advantage of using SQLite as opposed to a plain text log. However, since SQLite does not have any server process, it has to rely on the OS and filesystem features to manage access to the database. As a consequence, a single database file can only be written by one thread at a time, though it can be read simultaneously by multiple threads. Thus, the software was designed to accommodate this restriction.

The database structure is identical on both SION and the laptop. This simplifies software design and allows the database on SION to be copied directly to the laptop if necessary. While there are multiple types of SCANDAL CAN messages, all of them have identical field size and order. The only difference between the SCANDAL packet types is the meaning of data contained within those fields. This allows us to use one database structure to store all data in the CAN messages in their separate fields. This structure allows us to use SQL SELECT statement to query individual subsets of data.

One database file is used to store CAN packets generated on a single day. Each file contains only one table named `canlog` which contains the telemetry data. This table contains eight columns, corresponding to six fields used in SCANDAL, one column for packet numbering and one column for additional timestamp. Each message is stored as a row within the table.

Each packet is assigned a sequential packet number for synchronising data between the two cars, which will be described later in Section 5.4.1. The additional timestamp column is added as a backup in case of problems with the timestamps used in the scandal protocol. One such scenario happened after the 2009 World Solar Challenge, where we discovered

Length (bits)	Datatype	Scandal channel message	SQLite column
32	unsigned integer	*None*	packet_number
3	unsigned integer	priority	priority
8	unsigned integer	message type	message_type
8	unsigned integer	node address	source_address
10	unsigned integer	channel number	specifics
32	integer	value	value
32	unsigned integer	timestamp	scandal_timestamp
64	unsigned integer	*None*	ciel_timestamp

Table 5.1: The structure of the database is based on the structure of SCANDAL messages, such as the channel message displayed here.

a bug in the code within the Global Positioning System node. This bug resulted in broken timestamp values coming from the GPS node, which means that the solar car GPS data was unusable. Fortunately, we were running our strategy simulation based on GPS data of the control car and the laptop (due to an unrelated technical issue). A detailed description of the SCANDAL protocol is available in Section 5.3 of David Snowdon's thesis and will not be repeated here [Snowdon, 2002].

The SD card will slowly fill up to its capacity as it records more telemetry data. The amount of data stored per day can be estimated by assuming a worst case scenario, where the CAN bus is fully utilised at 50kb/s whenever it is turned on. Thus, by assuming that the solar car is turned on for 15 hours on a single race day, the amount of data stored is:

$$Data \approx \frac{50000bit/s \times 3600s/hour \times 15hour/day}{8bit/byte} \approx 337500000Bytes \approx 321.86MB$$

This amount of data is enough to fill a 8 GB SD card in 25 days. While this is far longer than the length of the race (about five days), the team plans to hold multiple training runs before the race. SION is also expected to be a part of the solar car telemetry system for a long time, potentially spanning multiple years as the PLEB did. This will cause the telemetry data to eventually fill up the SD card, creating a necessity to delete old telemetry data.

Since SION is required to be as maintenance-free as possible, it has to delete old data automatically to create space for new data. This is done by creating a new database file

on the beginning of a new day. When the Overo Air is booted, a script will automatically check the amount of free space available. It will then delete some of the oldest database files to free space until a certain threshold.

Separating the database over several files allows old data to be deleted very quickly as opposed to deleting old data in single large database file. This is because SQLite does not recover storage space when its entries are deleted, preferring to marking them ”usable” in case of future writes unless an explicit command is given [SQLite, 2011]. This complicates the calculation of actual available space and causes data fragmentation in the long term. While recovering the unused space manually is possible, it requires a large amount of memory access, which may impact performance. In comparison, deleting old database files only requires a single system call to the OS, which is much more efficient.

5.3.3 Wireless

The Wi-Fi network is used in infrastructure mode, with the access point in the control car broadcasting a service set identifier (SSID) and the OS within SION connecting to it as a client. Infrastructure mode is chosen since it is the most commonly used Wi-Fi implementation as opposed to ad-hoc mode, which makes it easier to use and support in the long term.

There are two commonly used protocols for Wi-Fi; the *transmission control protocol* (TCP) and the *user datagram protocol* (UDP). TCP attempts to provide a reliable connection between two devices. It uses a system of packet numbering, acknowledgement packets and rate control to ensure that no packets are lost in transmission and all packets arrive in the same order. However, these features make TCP undesirable for our purposes. Since TCP ensures that packets arrive in order they are sent, it will hold back the transmission of newer data until the old data is sent. However, monitoring the solar car requires new data to be prioritised over old data, as we want to see the latest state of the solar car. TCP cannot be modified to support this behavior.

On the other hand, UDP only provides basic addressing and checksumming. UDP packets may arrive out of order or even missing completely. This is not a problem since

each packet has its own timestamp and we have already assumed that packet loss will occur. Due to these factors UDP is chosen as the protocol for the wireless connection.

The wireless interface is accessed using the Berkeley Sockets API. It provides a simple interface to the UDP protocol used for the wireless connection.

5.4 Laptop / Strategy

The software on the laptop provides a UDP socket that outputs the telemetry data as it comes in from the solar car. This is intended to provide the monitoring software on the laptop with an interface to the latest data coming from the solar car. Alternatively, as this software uses SQLite just like the on-board storage, any software may query the SQLite database on the laptop to get the recorded telemetry data. This is possible even when the solar car is running on the road, as SQLite support multiple read access on a single database. The SQL interface is intended to provide access to all telemetry data, which is useful to the strategy simulation.

The software on the laptop also adds a separate timestamp to the telemetry data. This timestamp uses timing information provided by the real time clock of the laptop, which is updated from *network time protocol* (NTP) servers thorough the internet. However, all telemetry nodes in the solar car, including SION, do not have access to the internet. They obtain their timing information from GPS time obtained through the GPS node in the solar car. This information is used as the base for the timestamps on the scandal protocol. Thus, the software within SION does not add extra timestamps as it will simply give an identical timestamp as the one already in the SCANDAL packet.

5.4.1 Data synchronisation after telemetry breakup

As previously mentioned, the wireless connection between the two cars is expected to break multiple times during the race. During these periods, a copy of telemetry data is stored on the solar car. When the connection is reestablished, the latest telemetry data will be sent normally. However, this will leave a gap in the database in the control car.

The missing data can have a negative effect on the strategy simulation of the car. Thus, a synchronisation method has been implemented to automatically resend telemetry data that was only present on the solar car database.

The synchronisation method is based on the fact that every packet is given a sequential number by SION. Thus, when the control car software senses a gap in the numbering, it will interpret the gap as missing packets. It will then send a request to SION, which in turn will resend the packets. A packet that has been successfully retransmitted will fill in the gap in the numbering and prevent the software from sending another request.

This system can be considered a negative acknowledgment system as opposed to the positive acknowledgment system used in TCP. Negative acknowledgment requires less traffic than positive acknowledgment, as a missing ACK packet will trigger a retransmission even when the original packet was sent successfully. It also puts the duty of organising the synchronisation process on the laptop, which is a more powerful system than the Overo Air with more resources to spare.

5.5 Compilers and tools

A challenge in developing software for the three subsystems is the fact that all three are based on different processor architectures. The LPC1768 is based on the ARM Cortex M3 architecture, while the OMAP3503 processor on the Overo Air is based on the ARM Cortex A8 and the laptop uses an Intel Core 2 Duo processor, based on the x86 architecture.

C is chosen for this purpose as it has efficient bitwise operations, which are required for performing deep packet inspection on the data packets. However, the implementation of the C language may vary between different architectures. For example, the ARM Cortex M3 does not have a hardware floating point processor, making floating point operations relatively slow. Thus, features that are limited by architecture differences are used only on their corresponding platform or are avoided altogether.

Three different compilers are needed to compile C code into machine code for the three different architectures. The compilers are all based on the *gnu compiler collection* (GCC). Binaries for the laptop are compiled with the version of gcc that comes with the Ubuntu

Linux OS on the laptop. Binaries for the two ARM-based devices are cross compiled using the ARM GCC ports, which are maintained by CodeSourcery [CodeSourcery, 2010]. The LPC1768 uses the cross compiler for *embedded-application binary interface* (EABI) targets, while the Overo Air uses the cross compiler for Linux distributions. The EABI compiler does not link against standard libraries and produces binaries that run directly on a microcontroller as opposed to running on top of an OS.

The binary for the LPC1768 was programmed to its flash memory using a *joint test access group* (JTAG) debugging interface. This is connected to the development computer using an ARM debug board from the Openmoko project, which is controlled using the OpenOCD JTAG interface software [Rath, 2010]. The binary for the Overo Air was loaded through the wireless connection on SION. The build system uses makefiles which can automatically compile the required binaries and load them to their targets.

Chapter 6

Testing and results

6.1 Electrical noise

Electrical noise was suspected as one of the issues affecting the PLEB. Ideally, resolving this issue entails analysing the electrical noise generated by the car under various conditions both in the solar car workshop and on the road. However, as the team does not have the equipment nor the expertise to analyse electrical noise, only a rudimentary test was performed using an oscilloscope.

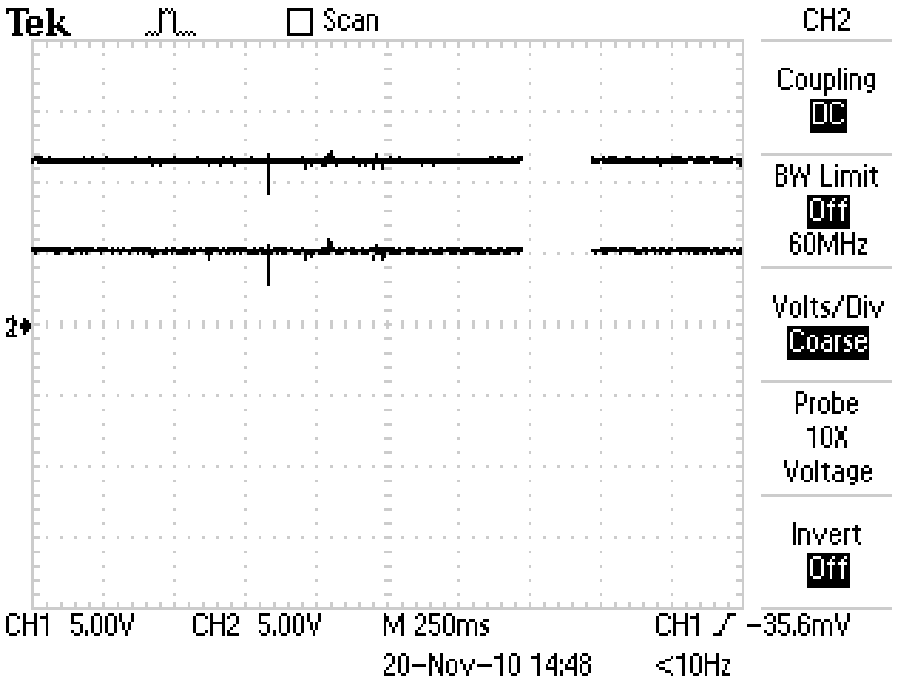
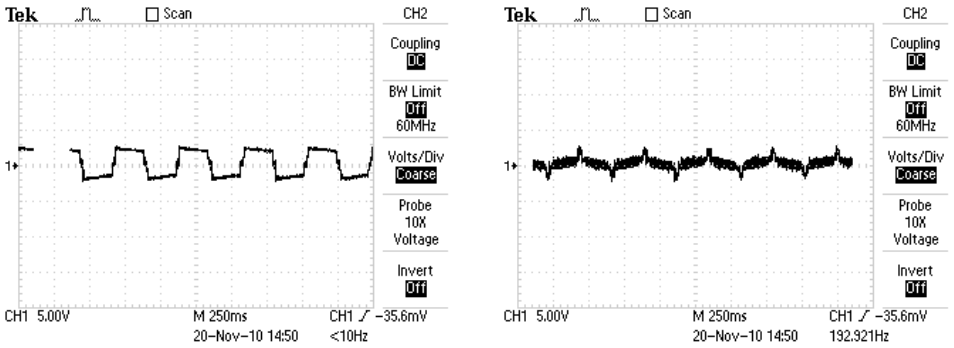


Figure 6.1: Oscilloscope trace of the 5 V and 12 V power lines.

Figure 6.1 shows the CAN power lines during motor acceleration. The dip in the trace corresponds to the time when the motor was put under maximum acceleration, which corresponds to the time when the motor is drawing a large amount of current. This offers a possible explanation to the issue where the PLEB ceased sending telemetry data during acceleration. A dip in voltage could cause the PLEB to perform a reset, requiring it to boot its OS before resuming its role.



(a) Slow speed (b) High speed

Figure 6.2: Traces on an oscilloscope due to electrical noise from the motor.

Figures 6.2a and 6.2b shows the oscilloscope trace when the ground clip is attached to the CAN ground and the probe left unattached, essentially acting as an antenna. This allows us to visualise the amount of electrical noise present around the motor. Figure 6.2a shows the trace when the motor is running at a slow speed, while Figure 6.2b shows the trace when the motor is running at a high speed. These figures show that the drivetrain in *Sunswift IV* does produce electrical noise during operation.

There are numerous shortcomings with this test. The oscilloscope can only analyse a very limited amount of the electromagnetic spectrum, thus electrical noise in other frequencies will not appear correctly. Furthermore, the oscilloscope is unlikely to have the proper analog front end to analyse electromagnetic noise. A proper analysis would require a competent operator with a spectrum analyser.

6.2 Field testing

The majority of development process was done on the bench. While this was convenient from a development point of view, bench testing does not reflect the real-world conditions that SION will be subjected to. Thus, two field tests were done during the project.

6.2.1 Land speed record attempt

The first field test happened in January 2011, during the Land Speed Record attempt in HMAS Albatross, Nowra. The event involved running a solar car without the battery in an attempt to get the highest speed possible. It was during one training session close to the event that the PLEB finally failed as described in Section 2.5. While telemetry feedback was not essential for the record attempt, it was still a major problem as we were unable to monitor the state of the solar car for any issues.

As this event happened in the middle of the development process, neither the hardware or software development was finished. However, we already had an Overo Air and its expansion board. This is already a near-complete solution, with the only missing hardware being the CAN interface. A CAN to USB dongle was then attached to it, which resulted in a complete hardware solution. The high-level PLEB software was then used to make it perform as a temporary replacement, with glue code written where necessary. The device was completed on the day before the actual event.

The newly created device worked without any major issues on the day before and during the actual event. This was somewhat unexpected, as the device had virtually no field testing. In the end, *Sunswift IV* broke the solar land speed record at 88 km/h average speed.

While this test does not involve the custom hardware and software under development, it validates the design decision to use a microcontroller (in the CAN to USB dongle) combined with an Overo Air as the hardware replacement for the PLEB.

The high level PLEB software was also found to be stable during the ordeal, with no telemetry dropout during the two days. This supported the hypothesis that the major cause of unreliability within the PLEB was its lower level software / hardware and not the high

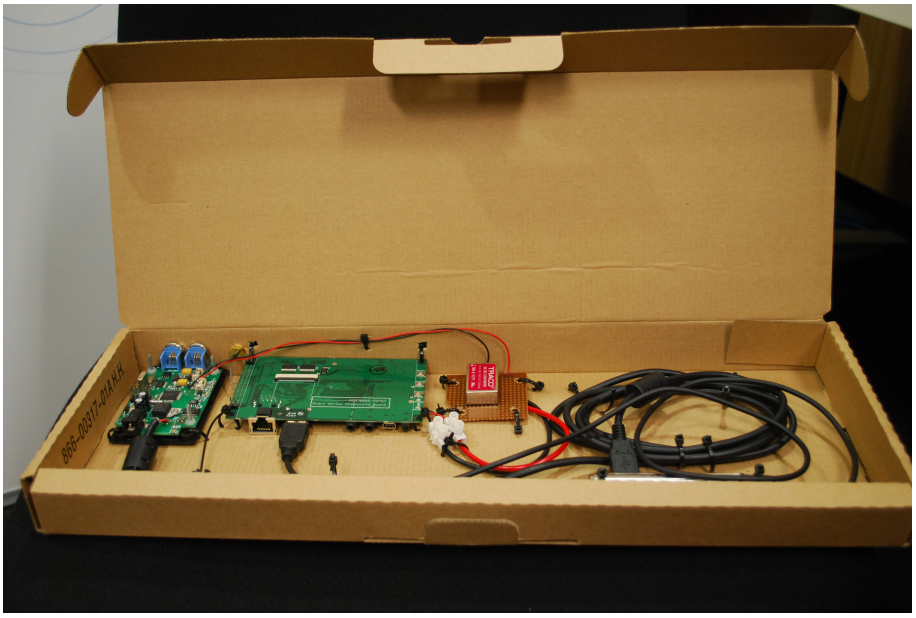


Figure 6.3: The temporary replacement for the PLEB, created for the event.

level code / wireless interface.

6.2.2 Honda Australia Rider Training track

Due to time constraints, only one field test was done with the completed system. The test was done on the *Honda Australia Rider Training* (HART) track in St. Ives, Sydney. It was part of an attempt to get telemetry data for modelling the performance of the solar car, where the team surveys various roads suitable for this purpose.

During the test, SION was able to deliver telemetry data from the solar car to the control car. It also recorded the telemetry data generated during the entire event. However, a long delay was present between events on the solar car happening and telemetry data indicating said event appearing in the laptop, exceeding 30 seconds at most. This delay was relatively consistent during the event. However, attempts to replicate the delay on the team's workshop have so far been unsuccessful.

This event shows the importance of field testing in addition to bench testing. Issues that do not show up on the bench can reveal themselves on the actual solar car. In order to create a truly reliable system, field testing should be done as much as possible in the weeks leading to the race.

6.3 Fulfilment of requirements

While the system is not completely bug-free, the majority of hardware and software design has worked consistently. To check the progress of the project, we can compare the results of the development work done so far against the original set of requirements outlined in Section 3.1:

- The completed system can deliver data between the CAN bus in the solar car and the strategist's laptop in the control car. This fulfils requirement number 1 and 2.
- The system has been found to perform data transmission consistently, which shows promise in fulfilling requirement number 3. However, it has not been extensively field-tested, which means it cannot be considered reliable yet. The only way to fulfil this is to perform more field testing in the hope of revealing any issues in the current design.
- Due to time constraints, only one prototype had been made at the time of writing. However, there is a plan to create a second device with a revised hardware design, bringing the total to two. Thus, requirement number 4 should be fulfilled before the start of the 2011 race.
- The software infrastructure provides on-board data logging and synchronisation. This ensures that few, if any telemetry data is lost at all when communication with the control car is broken, fulfilling requirement number 5.
- The system was designed to be easy to operate and require minimal maintenance. Programming interfaces was kept as simple as possible. Furthermore, all design documents and source code is available in the team's internal web site for reference. Thus, requirement number 6 can be considered fulfilled, though there might be hidden complexities that are currently hidden due to the lack of real-world testing.
- The SION hardware consumes around 272 mA at 12 V when sending telemetry data. This is lower than the combined power consumption of the PLEB and WAP, therefore fulfilling requirement number 7.

- The total hardware cost of SION is approximately $500 per device, which fulfils requirement number 8.

6.4 Comparison with the PLEB

Both the PLEB and SION perform the same basic task: delivering data between the solar car and the control car. Thus, both of them are able to interface with the CAN bus and the laptop.

The major improvement from the PLEB is SION's ability to handle broken wireless links. Previously, any telemetry data generated during this condition would be lost. SION logs all telemetry data internally, which resolves this issue. In addition, SION has more detailed documentation and design documents. It also uses less amount of power then the WAP used alongside the PLEB.

Chapter 7

Conclusions

As a result of the project, the team has a new system to obtain telemetry data from the solar car on the road. Initial testing of the system showed that it is capable of performing all of its original design criteria. The development progress should be continued to complete the full set of requirements, at which point the system will be ready to perform its role to obtain the highest possible efficiency from *Sunswift IV*.

7.1 Future work

While most of the development work has been finished, there are still several possible improvements that can be made within the system.

- The system still requires a lot of field testing to reveal any hidden issues. In particular, the potential issue with electrical noise affecting wireless transmission has not been adequately tested, as both field tests done so far did not include high power acceleration which generates a lot of electrical noise. Furthermore, the telemetry delay issue encountered in Section 6.2.2 has not been solved yet.

- More devices need to be constructed, as we will require backup devices in case of any problems on the race. A hardware revision is planned to achieve this goal and fix various minor issues on the first revision of the PCB.

- The software on the control car can be expanded to include interfaces with the

sensors in the control car, such as a wind speed sensor and insolation measurement to sense how much energy is available from the sun. It is also possible to construct a CAN-based telemetry network in the control car, using the same set of nodes as within the solar car. This allows the nodes to be interchangeable with the solar car, which allows quick replacement in case of any failures.

- The software can be improved by merging the full SCANDAL framework into the software. While SION is not a sensor node, implementing SCANDAL would allow it to send various statistics about the telemetry system, which could be useful for debugging telemetry issues. Minor modifications may also be implemented if the strategy and monitoring software requires any extra features, as they are also currently in development.

- The power consumption of SION might be improved if the *dynamic voltage and frequency scaling* can be implemented in the Linux kernel.

7.2 Final comments

Sunswift IV's performance in the 2009 World Solar Challenge was a result of continuous improvements ever since the team was first created in 1996. The development of this thesis was greatly aided by old theses, source code and design documents from various past team members. This thesis, and all documents created along the process, are also written in hopes that future members of the team can utilise it to develop further improvements to the solar car project.

Bibliography

Analog Devices. ADP1621 data sheet. http://www.analog.com/static/imported-files/data_sheets/ADP1621.pdf, 2010.

ARM Ltd. CMSIS - cortex microcontroller software interface standard. http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php, 2010a. Accessed 21 October 2010.

ARM Ltd. mbed tour. http://www.mbed.org/handbook/Tour, 2010b.

CodeSourcery. Gnu toolchain for arm processors. http://www.codesourcery.com/sgpp/lite/arm, 2010.

Debian Project. Debian home page. http://www.debian.org/, 2010.

FTDI Ltd. FT2232 data sheet. http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232H.pdf, 2010.

Gumstix. Gumstix home page. http://www.gumstix.com/, 2010.

Jeff Johnston. Newlib home page. http://sourceware.org/newlib/, 2010.

Kvaser. Kvaser home page. http://www.kvaser.com, 2010.

T Lindgren, K Hilman, and P Walmsey. Linux OMAP kernel project. http://www.omappedia.org/wiki/Linux_OMAP_Kernel_Project, 2010.

Linear Technologies. LT1308A/B data sheet. http://cds.linear.com/docs/Datasheet/1308abfa.pdf, 2010.

Linear Technology. LTC4425 data sheet. http://cds.linear.com/docs/Datasheet/4425f.pdf, 2010.

Motorola. Supporting the University of Michigan solar car team in advancing planet-friendly energy technologies. http://www.motorola.com/staticfiles/Business/Products/Wireless%20Broadband%20Networks/Intelligent%20Transportation%20Systems/Documents/_Static%20files/Mesh%20Solar%20Car_CS.pdf, 2008.

Nokia. Maemo home page. http://maemo.org/, 2010.

NVE Corporation. IL700/IL200 high speed digital signal isolators. http://www.nve.com/il700.php, 2010. Checked on 21 October 2010.

NXP Semiconductors. LPC1768 data sheet. http://ics.nxp.com/products/lpc1000/datasheet/lpc1763.lpc1764.lpc1765.lpc1766.lpc1767.lpc1768.lpc1769.pdf, 2010.

OpenEmbedded. OpenEmbedded home page. http://www.openembedded.org/index.php/Main_Page, 2010.

Keith Pazul. Controller area network (CAN) basics. Technical Report AN713, Microchip Technology Inc, September 2005.

Dominic Rath. OpenOCD home page. http://openocd.berlios.de/web/, 2010.

Neo Wei Ren. Wireless telemetry for solar car. http://www.mae.ntu.edu.sg/AboutMAE/NewsAndEvents/Documents/Event/NE2010/TelemetrySystem2010/Wireless_NEO%20Wei%20Ren_LR.PDF, 2010.

RM Michaelides. CANlink WLAN 2001. http://www.rmcan.com/uploads/tx_rmprodukte/canlink-wlan.pdf, 2010.

SD Association. SD specifications part 1: Physical layer simplified specification version 3.01. http://www.sdcard.org/developers/tech/pls/, 2010.

Y Shimizu, Y Komatsu, M Torii, and M Takamuro. Solar car cruising strategy and its supporting system. *JSAE Review*, 19:143–149, 1998.

David C. Snowdon. Hard- and software framework for the optimisation of Sunswift-II. BE thesis, School of Computer Science and Engineering, University of NSW, Sydney 2052, Australia, November 2002. Available from publications page at http://www.disy.cse.unsw.edu.au/.

David C. Snowdon. PLEB2 overview. Technical report, NICTA, Sydney, Australia, 2004. Available from http://ertos.nicta.com.au/hardware/pleb/downloads/pleb2_description.pdf.

SQLite. SQLite home page. http://www.sqlite.org/, 2010.

SQLite. SQLite query language: vacuum. http://www.sqlite.org/lang_vacuum.html, 2011.

TDK-Lambda. CC-E series data sheet. http://us.tdk-lambda.com/lp/ftp/Specs/cc-e.pdf, 2010.

Adam Wiggins. PLEB: A platform for portable and embedded systems research. Technical Report UNSW-CSE-TR-0105, School of Computer Science and Engineering, University of NSW, Sydney 2052, Australia, April 2001. Available from publications page at http://www.disy.cse.unsw.edu.au/.

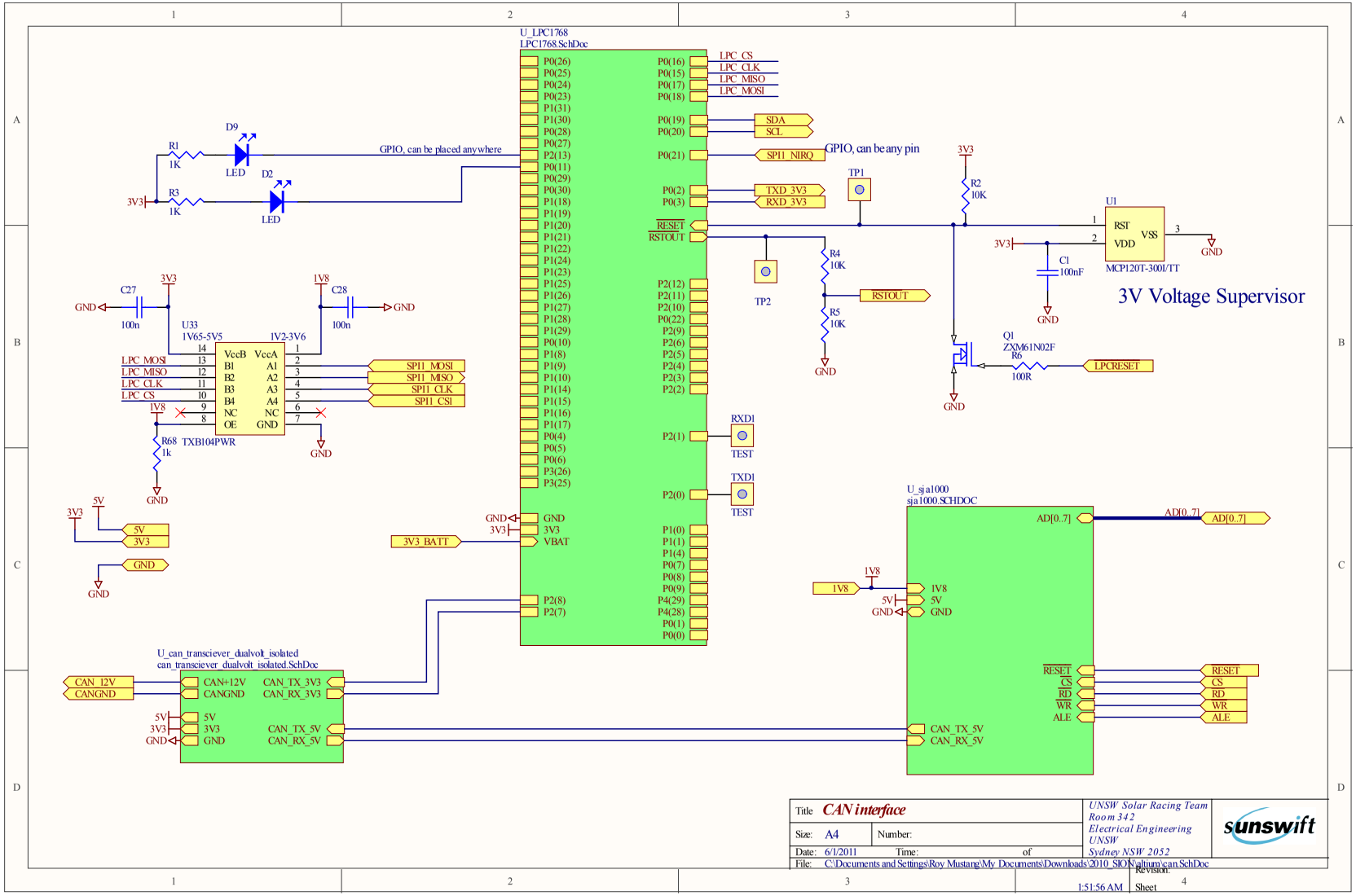
World Solar Challenge. World Solar Challenge technical regulations for the 2011 event. http://www.worldsolarchallenge.org/participants/regulations, 2010.

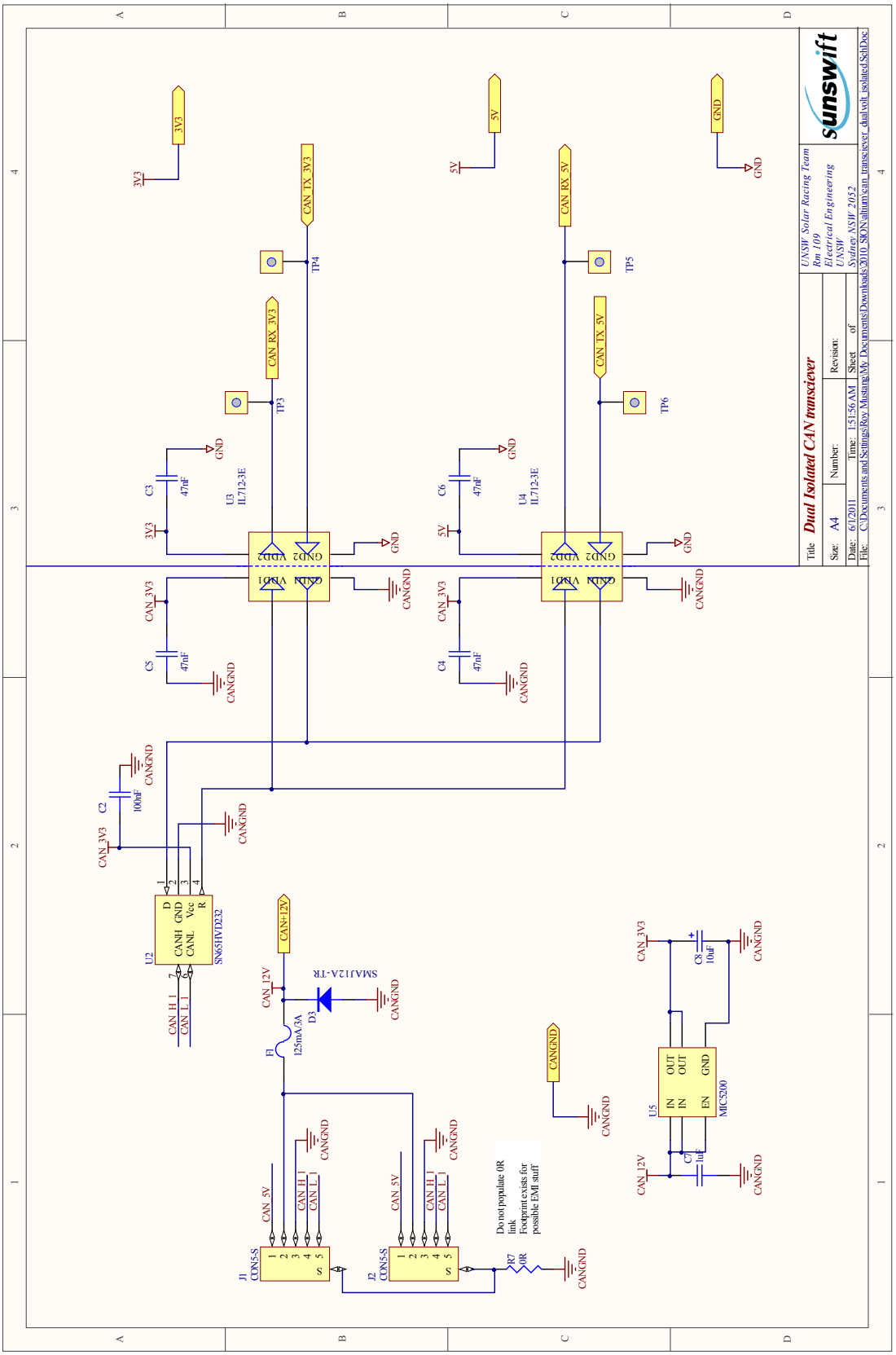
World Solar Challenge. Wsc event route 2011. http://www.worldsolarchallenge.org/home/route/, 2011.

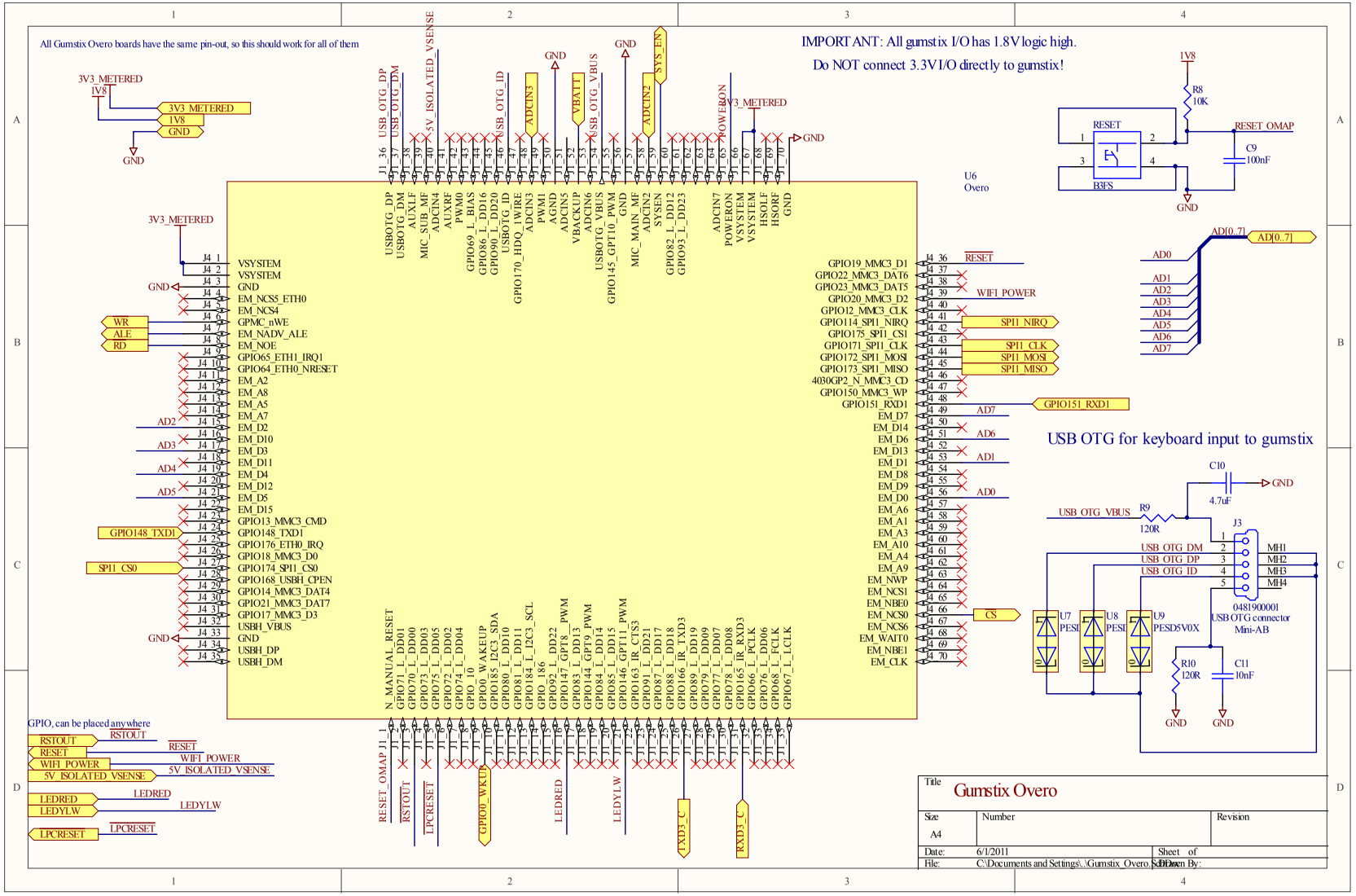
Andrew Wrigley. Steering-Integrated driver controls for Sunswift IV. BE thesis, School of Electrical Engineering, University of NSW, Sydney 2052, Australia, November 2009. Available from publications page at http://www.disy.cse.unsw.edu.au/.

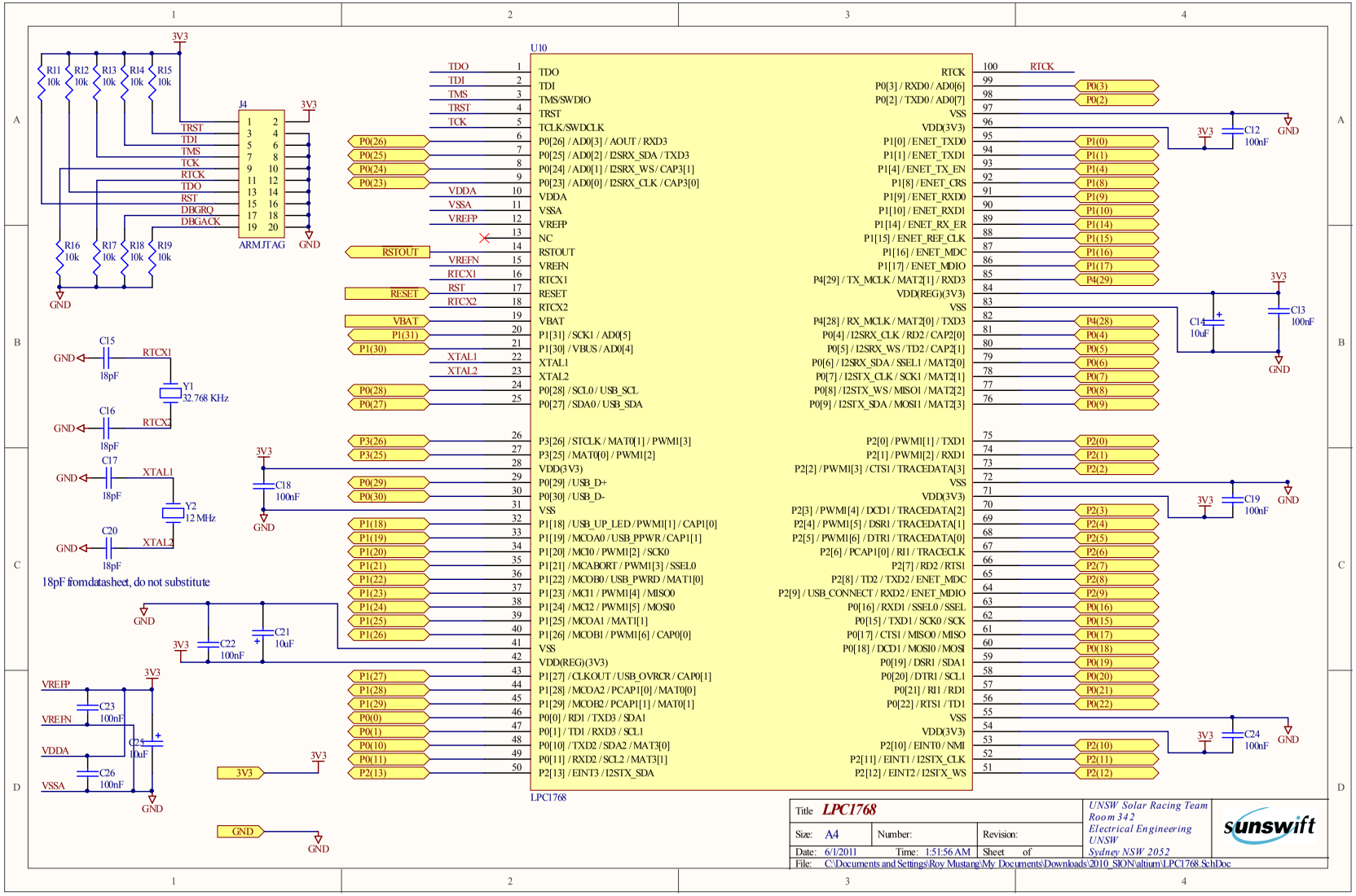
Appendix A

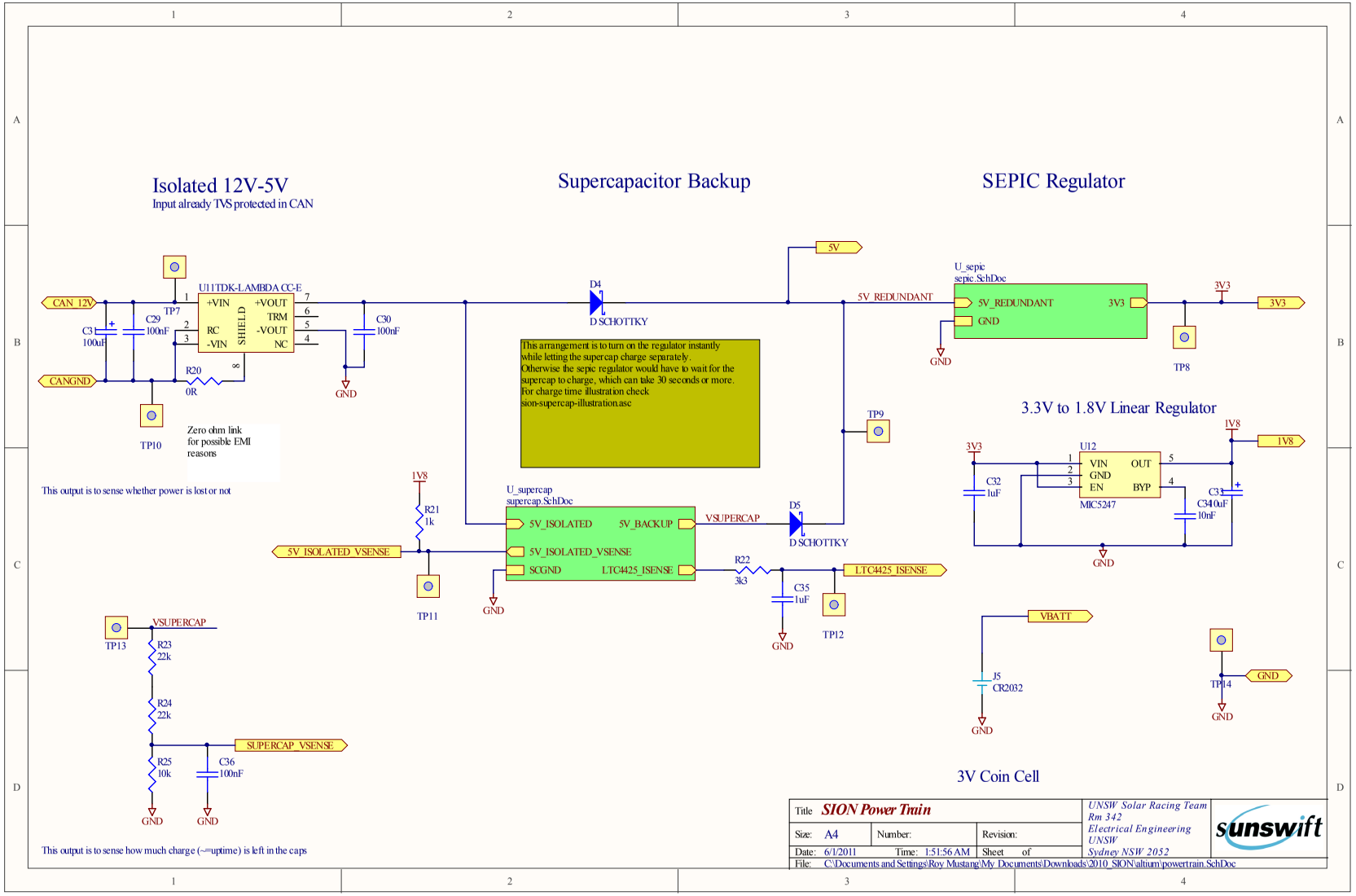
Electrical schematics and PCB layout

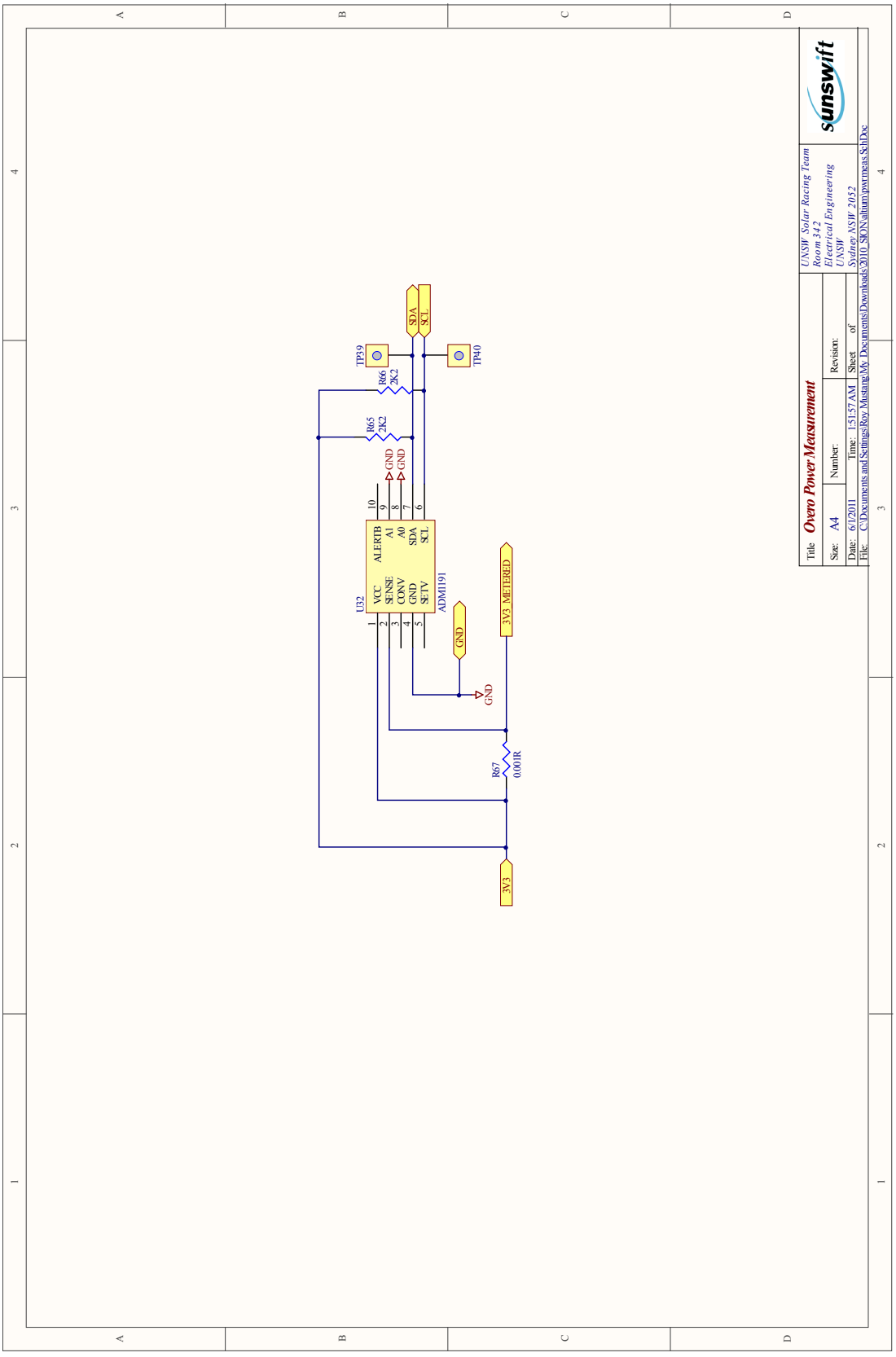

U_LPC1768
LPC1768.SchDoc
GPIO, can be placed anywhere
GPIO, can be any pin
3V Voltage Supervisor
MCP120T-300I/TT
ZXM61N02F
TXB104PWR
U_can_transciever_dualvolt_isolated
can_transciever_dualvolt_isolated.SchDoc
U_sja1000
sja1000.SCHDOC
Title CAN interface
Size: A4
Number:
Date: 6/1/2011
Time:
of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIO\Altium\can.SchDoc
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift
Revision:
1:51:56 AM
Sheet


Dual Isolated CAN transciever
UNSW Solar Racing Team
Rm 109
Electrical Engineering
UNSW
Sydney NSW 2052
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:56 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIGN\altium\can_transciever_dualvolt_isolated.SchDoc
sunswift
3V3
5V
GND
CAN_TX_3V3
CAN_RX_3V3
CAN_RX_5V
CAN_TX_5V
TP3
TP4
TP5
TP6
U3
IL712-3E
U4
IL712-3E
C3
47nF
C4
47nF
C5
47nF
C6
47nF
CANGND
CAN_3V3
CAN_5V
U2
SN65HVD232
D
GND
Vcc
R
CANH
CANL
C2
100nF
CAN_H_1
CAN_L_1
CAN_12V
CAN+12V
F1
125mA/3A
D3
SMAJ12A-TR
J1
CON5-S
J2
CON5-S
R7
0R
Do not populate 0R link.
Footprint exists for possible EMI stuff
U5
MIC5200
IN
OUT
EN
C8
10uF
C7
1uF

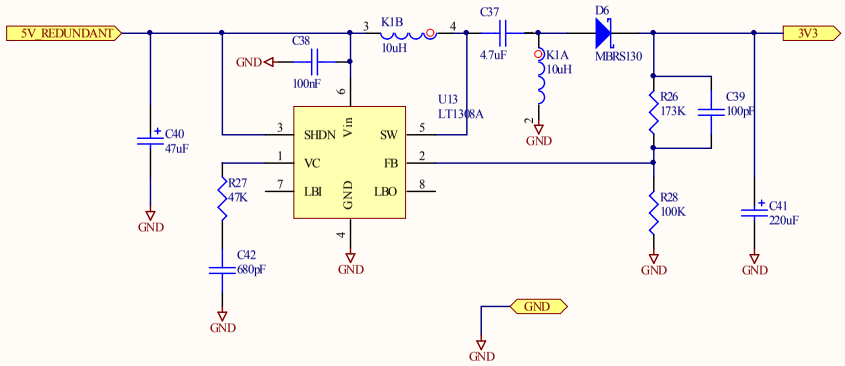

All Gumstix Overo boards have the same pin-out, so this should work for all of them
IMPORTANT: All gumstix I/O has 1.8V logic high.
Do NOT connect 3.3V I/O directly to gumstix!
U6
Overo
USB OTG for keyboard input to gumstix
GPIO, can be placed anywhere
Title
Gumstix Overo
Size
A4
Number
Revision
Date: 6/1/2011
Sheet of
File: C:\Documents and Settings\...\Gumstix_Overo.SchDoc
Drawn By:


U10
LPC1768
ARMJTAG
J4
3V3
GND
TDO
TDI
TMS
TRST
TCK
RTCK
RST
DBGRQ
DBGACK
R11 10k
R12 10k
R13 10k
R14 10k
R15 10k
R16 10k
R17 10k
R18 10k
R19 10k
C15 18pF
C16 18pF
C17 18pF
C20 18pF
Y1 32.768 KHz
Y2 12 MHz
RTCX1
RTCX2
XTAL1
XTAL2
18pF from datasheet, do not substitute
C18 100nF
C19 100nF
C22 100nF
C21 10uF
C12 100nF
C13 100nF
C14 10uF
C24 100nF
C23 100nF
C25 10uF
C26 100nF
VREFP
VREFN
VDDA
VSSA
RSTOUT
RESET
VBAT
P1(31)
P1(30)
P0(26)
P0(25)
P0(24)
P0(23)
P0(28)
P0(27)
P3(26)
P3(25)
P0(29)
P0(30)
P1(18)
P1(19)
P1(20)
P1(21)
P1(22)
P1(23)
P1(24)
P1(25)
P1(26)
P1(27)
P1(28)
P1(29)
P0(0)
P0(1)
P0(10)
P0(11)
P2(13)
P0(3)
P0(2)
P1(0)
P1(1)
P1(4)
P1(8)
P1(9)
P1(10)
P1(14)
P1(15)
P1(16)
P1(17)
P4(29)
P4(28)
P0(4)
P0(5)
P0(6)
P0(7)
P0(8)
P0(9)
P2(0)
P2(1)
P2(2)
P2(3)
P2(4)
P2(5)
P2(6)
P2(7)
P2(8)
P2(9)
P0(16)
P0(15)
P0(17)
P0(18)
P0(19)
P0(20)
P0(21)
P0(22)
P2(10)
P2(11)
P2(12)
Title LPC1768
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:56 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SION\altium\LPC1768.SchDoc
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift

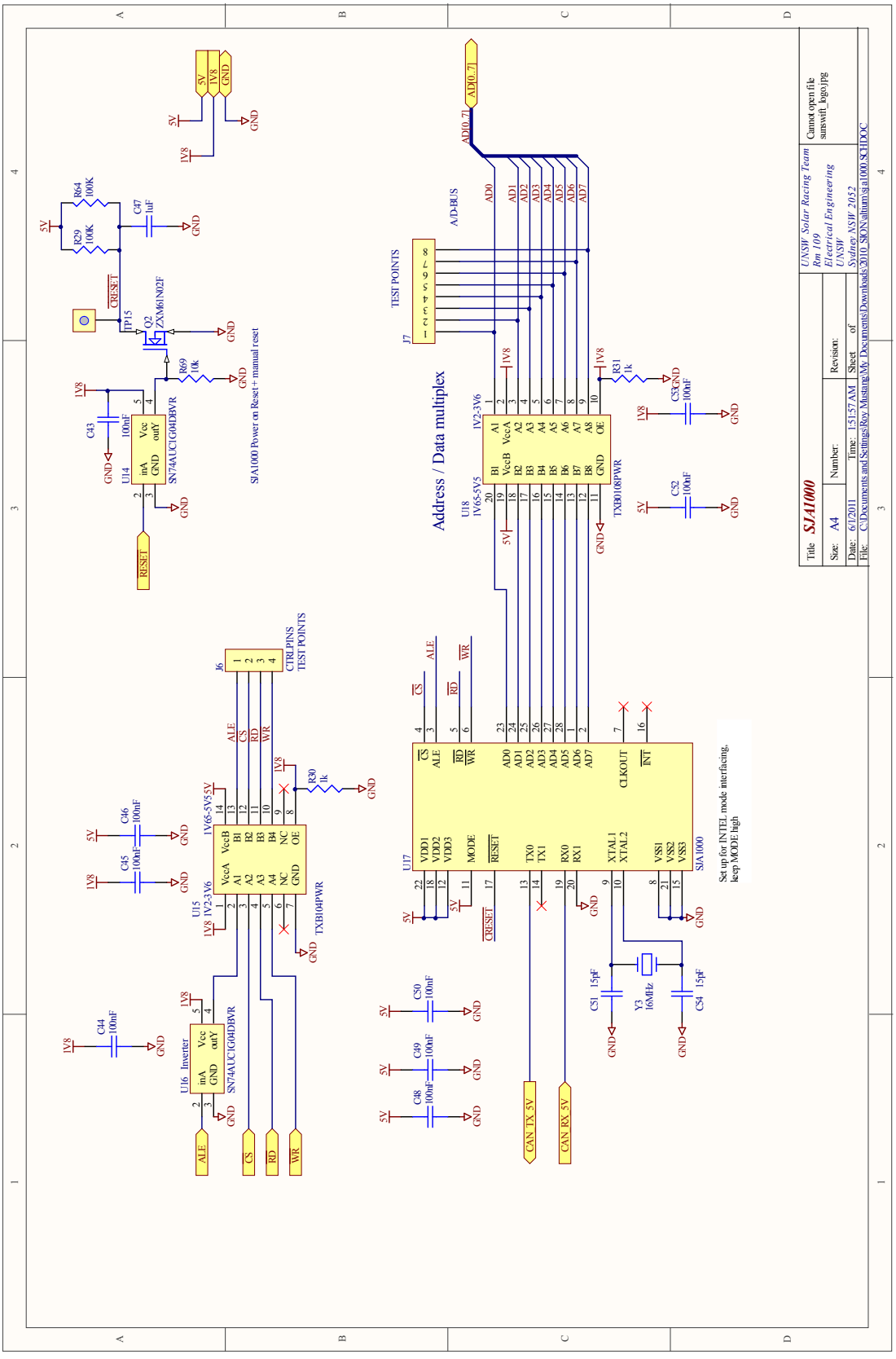

Isolated 12V-5V
Input already TVS protected in CAN
Supercapacitor Backup
SEPIC Regulator
CAN_12V
CANGND
U11TDK-LAMBDA CC-E
+VIN
RC
-VIN
SHIELD
+VOUT
TRM
-VOUT
NC
C3 100uF
C29 100nF
C30 100nF
R20 0R
TP7
TP10
GND
Zero ohm link for possible EMI reasons
This output is to sense whether power is lost or not
D4
D SCHOTTKY
5V
5V_REDUNDANT
This arrangement is to turn on the regulator instantly while letting the supercap charge separately. Otherwise the sepic regulator would have to wait for the supercap to charge, which can take 30 seconds or more. For charge time illustration check sion-supercap-illustration.asc
U_sepic
sepic.SchDoc
3V3
TP8
TP9
1V8
R21 1k
U_supercap
supercap.SchDoc
5V_ISOLATED
5V_BACKUP
5V_ISOLATED_VSENSE
SCGND
LTC4425_ISENSE
VSUPERCAP
D5
D SCHOTTKY
R22 3k3
C35 1uF
TP11
TP12
3.3V to 1.8V Linear Regulator
U12
VIN
GND
EN
OUT
BYP
MIC5247
C32 1uF
C33 C34 0uF 10nF
VBATT
J5 CR2032
TP14
TP13
R23 22k
R24 22k
R25 10k
C36 100nF
SUPERCAP_VSENSE
This output is to sense how much charge (~=uptime) is left in the caps
3V Coin Cell
Title SION Power Train
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:56 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SION\altium\powertrain.SchDoc
UNSW Solar Racing Team
Rm 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift

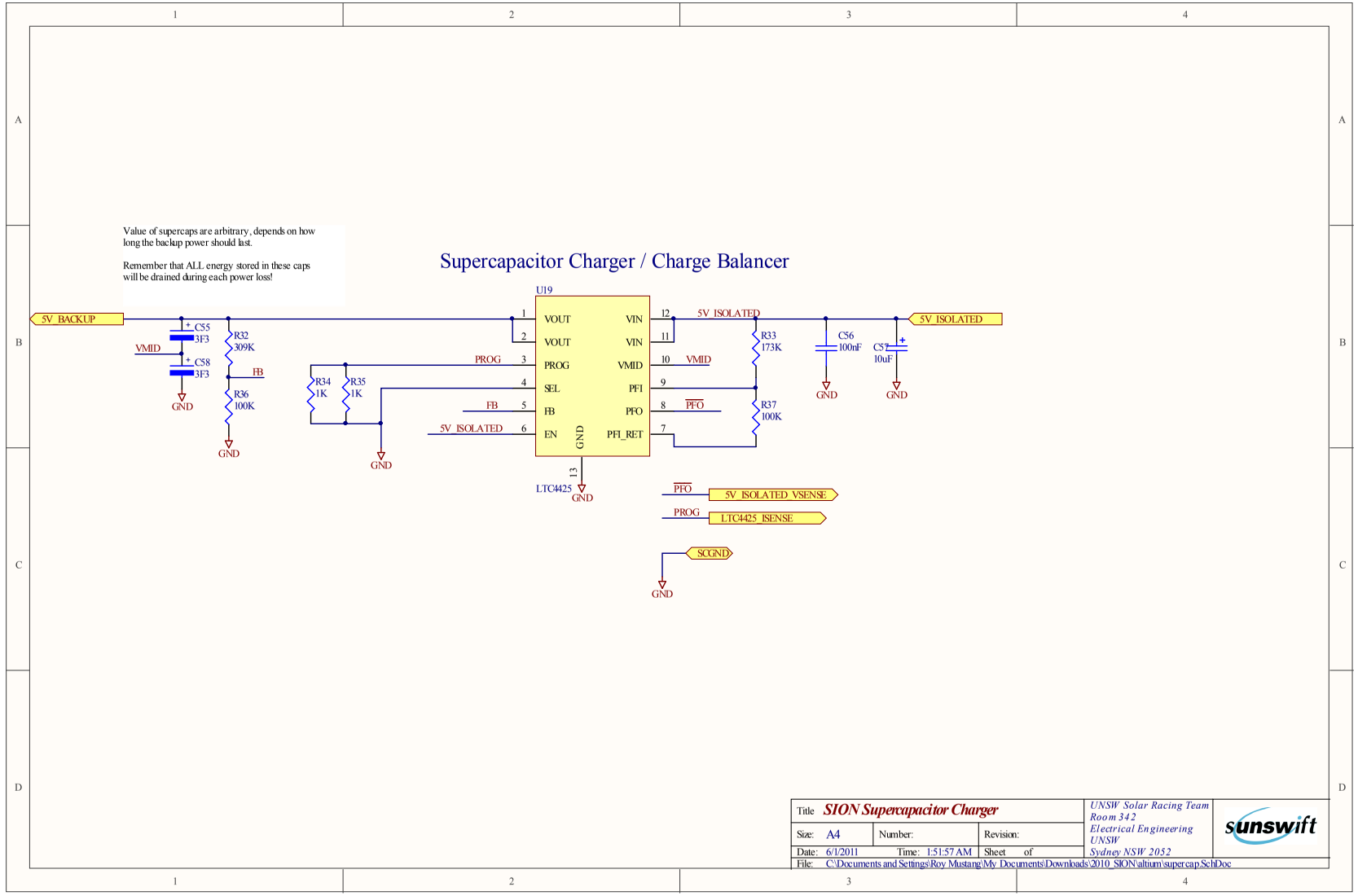

U32
ADM1191
VCC
SENSE
CONV
GND
SETV
ALERTB
A1
A0
SDA
SCL
1
2
3
4
5
10
9
8
7
6
R65
2K2
R66
2K2
R67
0.001R
TP39
TP40
SDA
SCL
GND
GND
GND
GND
3V3_METERED
3V3
Title: Overo Power Measurement
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:57 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIGN\altium\powrmeas.SchDoc
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift

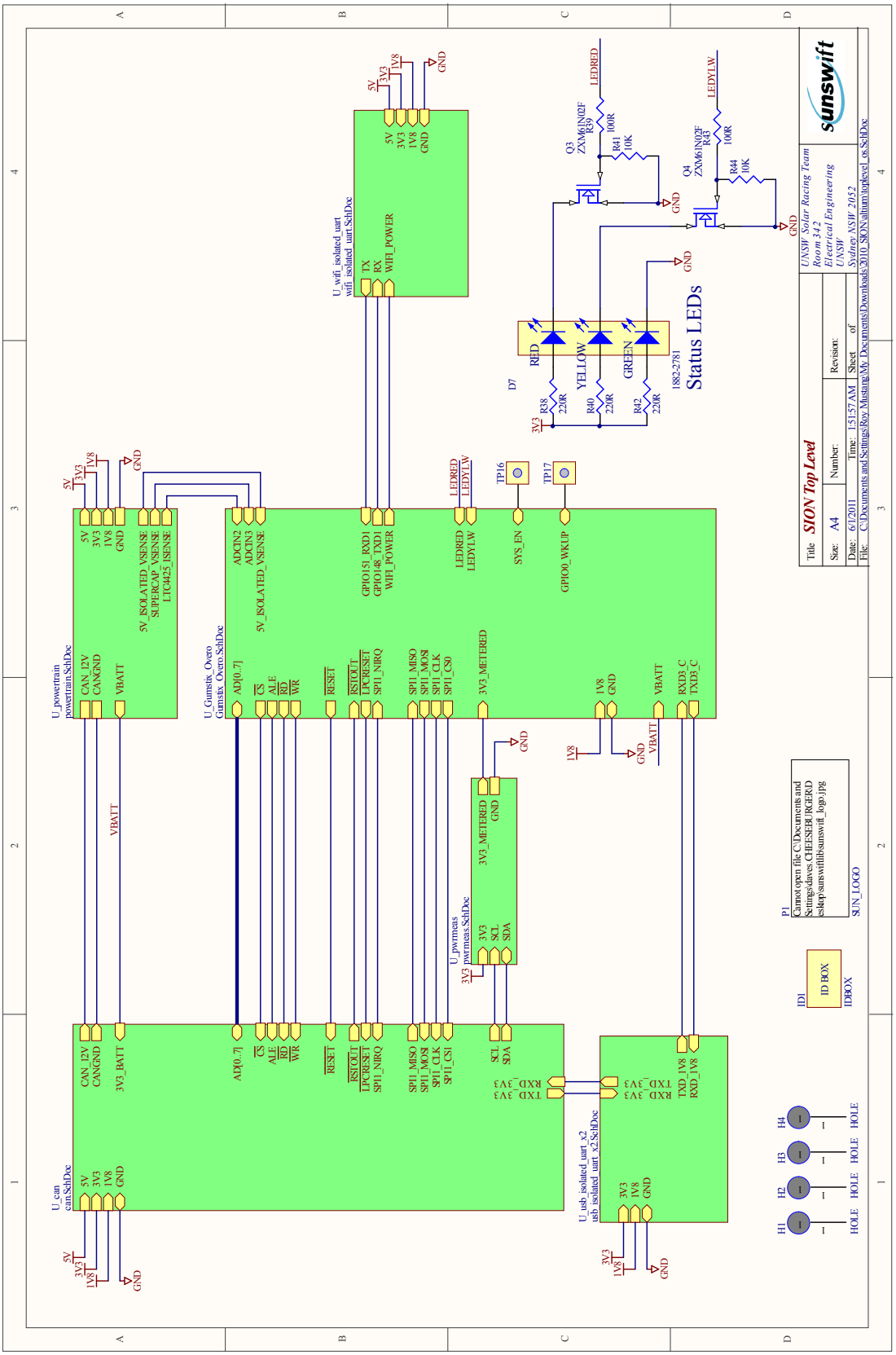
0.8 - 5V to 3V3 SEPIC Regulator

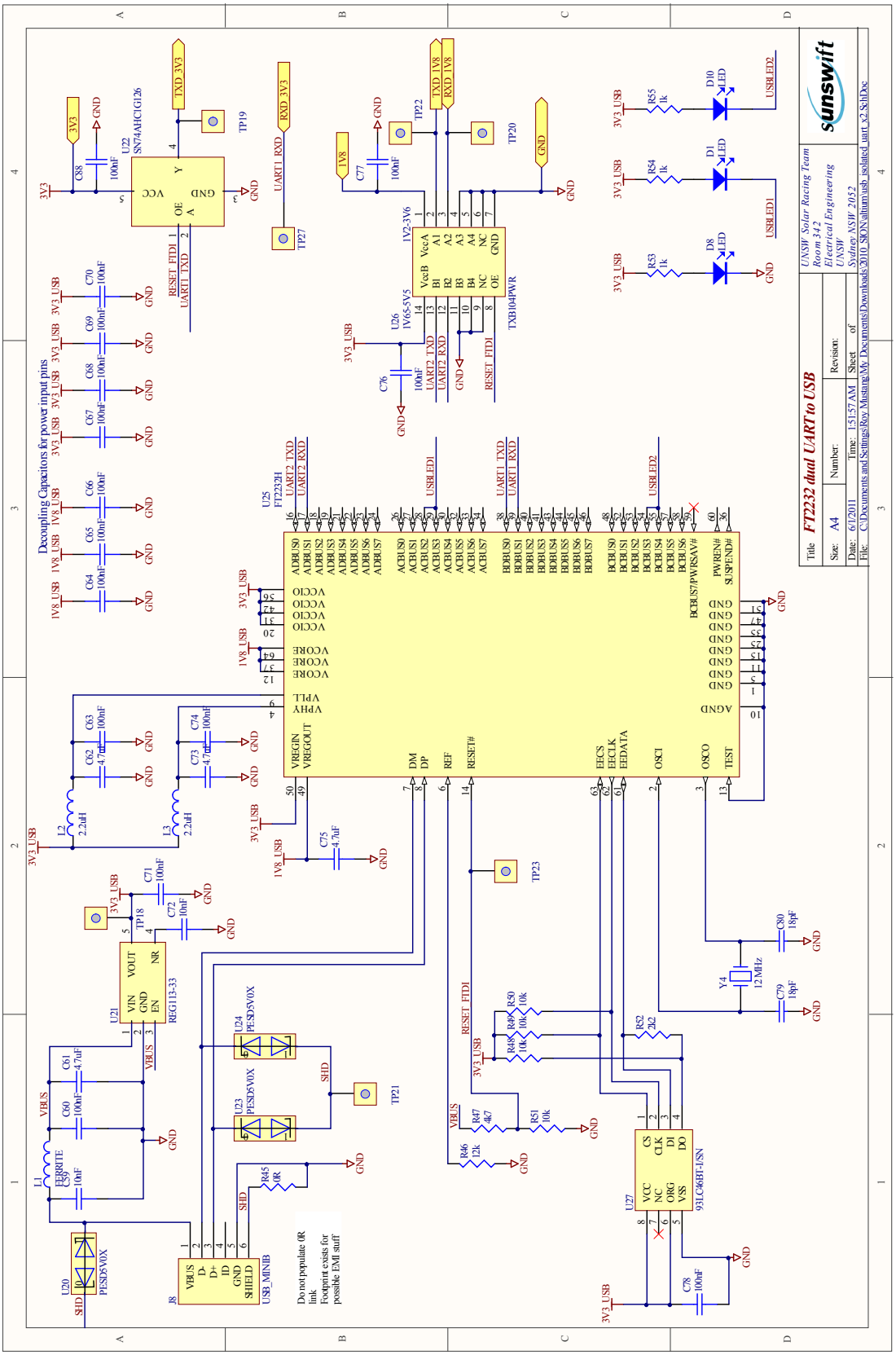


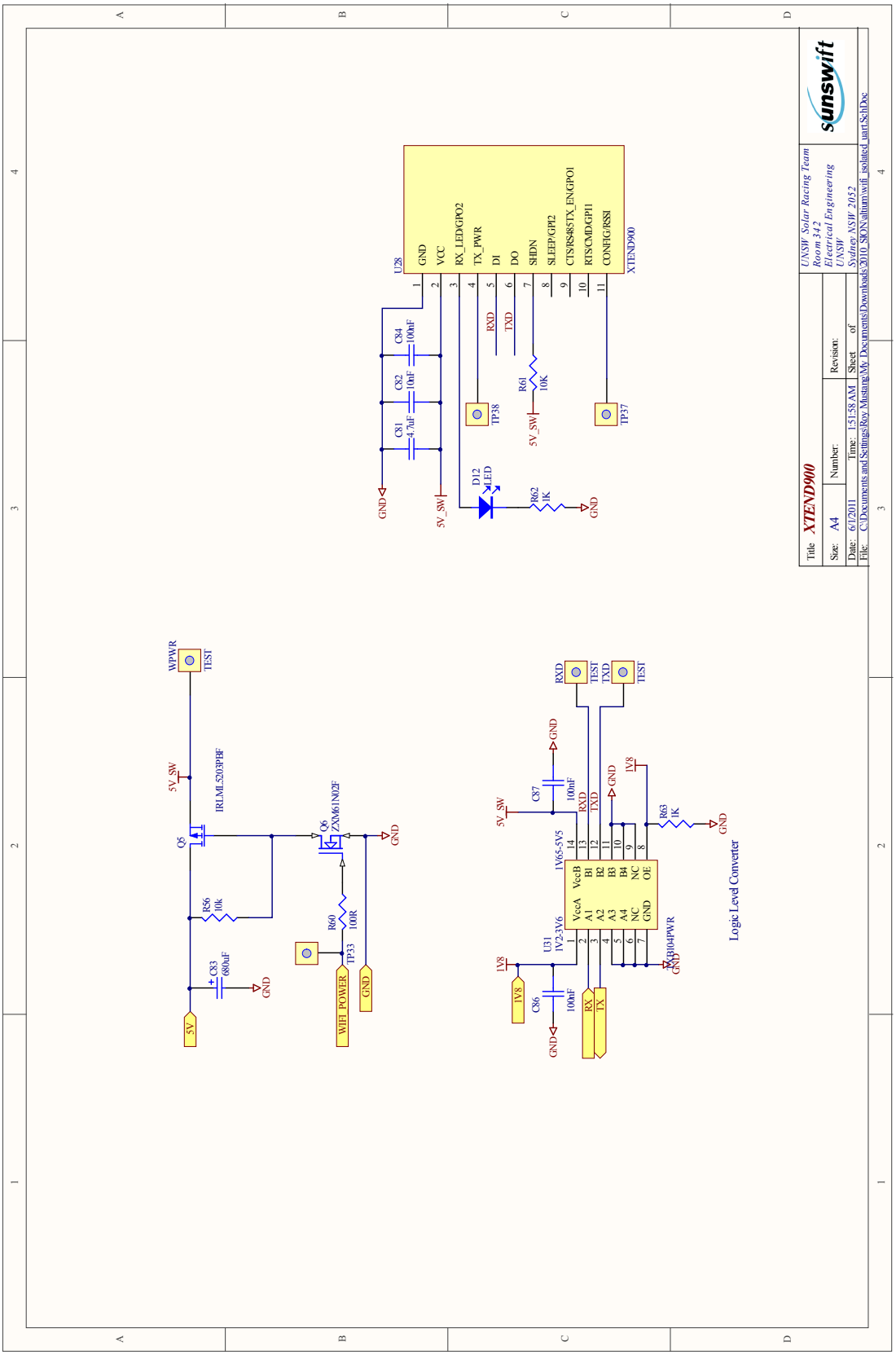
Title	*SION Sepic Regulator*		*UNSW Solar Racing Team* *Room 342* *Electrical Engineering* *UNSW* *Sydney NSW 2052*
Size: A4	Number:	Revision:	
Date: 6/1/2011	Time: 1:51:57 AM	Sheet of	
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SION\altium\sepic.SchDoc			

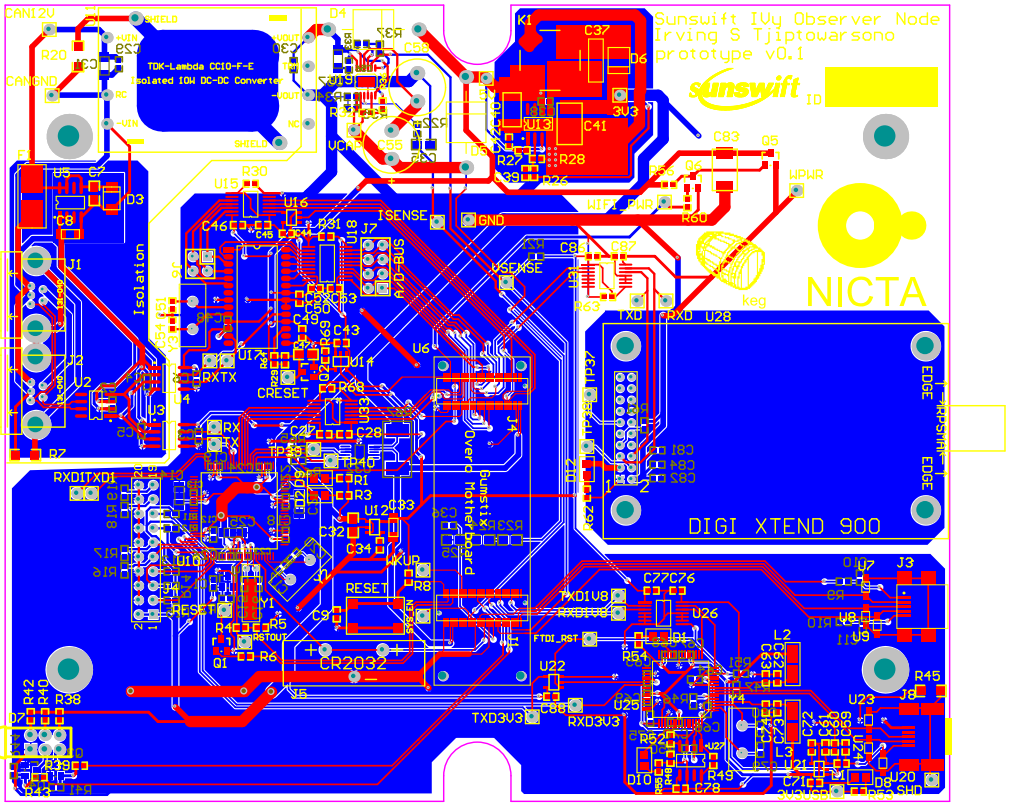

Address / Data multiplex
SJA1000 Power on Reset+ manual reset
Set up for INTEL mode interfacing,
keep MODE high
CTRL PINS
TEST POINTS
A/D-BUS
TEST POINTS
U116 Inverter
U117
SJA1000
U18
TXB0108PWR
U15
TXB104PWR
U14
SN74AUC1G04DBVR
Title SJA1000
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:57 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIGN\altium\sja1000.SCHDOC
UNSW Solar Racing Team
Rm 109
Electrical Engineering
UNSW
Sydney NSW 2052
Cannot open file
sunwift_logo.jpg


Value of supercaps are arbitrary, depends on how long the backup power should last.
Remember that ALL energy stored in these caps will be drained during each power loss!
Supercapacitor Charger / Charge Balancer
U19
5V_BACKUP
C55 3F3
C58 3F3
VMID
R32 309K
R36 100K
FB
R34 1K
R35 1K
PROG
FB
5V_ISOLATED
VOUT
VOUT
PROG
SEL
FB
EN
GND
VIN
VIN
VMID
PFI
PFO
PFI_RET
5V_ISOLATED
R33 173K
R37 100K
C56 100nF
C57 10uF
5V_ISOLATED
VMID
PFO
LTC4425
PFO
5V_ISOLATED_VSENSE
PROG
LTC4425_ISENSE
SCGND
GND
Title SION Supercapacitor Charger
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:57 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SION\altium\supercap.SchDoc
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift

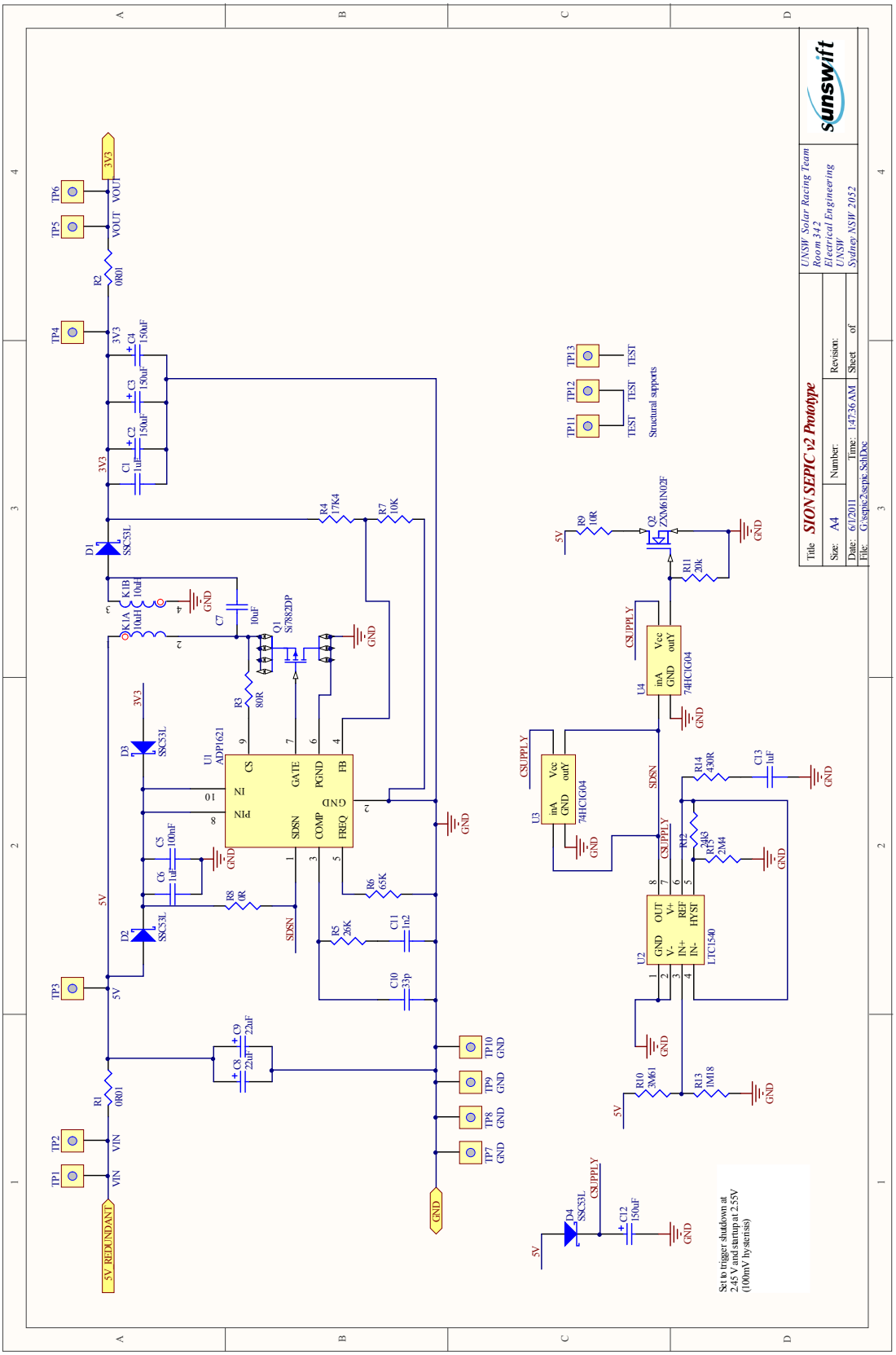

Status LEDs
SION Top Level
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift
Title
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:57 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SION\altium\toplevel_os.SchDoc
U_powertrain
powertrain.SchDoc
U_Gumstix_Overo
Gumstix_Overo.SchDoc
U_wifi_isolated_uart
wifi_isolated_uart.SchDoc
U_pwrmeas
pwrmeas.SchDoc
U_can
can.SchDoc
U_usb_isolated_uart_x2
usb_isolated_uart_x2.SchDoc
Q3
ZXM61N02F
Q4
ZXM61N02F
R38
220R
R40
220R
R42
220R
R39
100R
R41
10K
R43
100R
R44
10K
RED
YELLOW
GREEN
1882-2781
D7
TP16
TP17
SYS_EN
GPIO00_WKUP
LEDRED
LEDYLW
ID BOX
IDBOX
ID1
P1
SUN_LOGO
H1
H2
H3
H4
HOLE

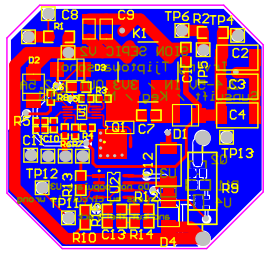

FT2232 dual UART to USB
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
sunswift
Title
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:57 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIGN\altium\usb_isolated_uart_v2.SchDoc
Decoupling Capacitors for power input pins
Do not populate 0R link.
Footprint exists for possible EMI stuff
U25
FT2232H
U26
TXB104PWR
U27
93LC46BT-I/SN
U21
REG113-33
U22
SN74AHCG126
U20
PESD5V0X
U23
PESD5V0X
U24
PESD5V0X
J8
USB_MINIB
Y4
12 MHz
TP18
TP19
TP20
TP21
TP22
TP23
TP27
D1
D8
D10
USBLED1
USBLED2
RESET_FTDI
UART1_TXD
UART1_RXD
UART2_TXD
UART2_RXD
3V3
3V3_USB
1V8
1V8_USB
VBUS
SHD
GND


U28
XTEND900
GND
VCC
RX_LED/GPO2
TX_PWR
DI
DO
SHDN
SLEEP/GPI2
CTS/RS485TX_EN/GPO1
RTS/CMD/GPI1
CONFIG/RSSI
C84 100nF
C82 10nF
C81 4.7uF
D12 LED
R62 1K
R61 10K
TP38
TP37
RXD
TXD
5V_SW
WPWR
TEST
IRLML5203PBF
Q5
Q6 ZXM61N02F
R56 10k
R60 100R
C83 680uF
TP33
WIFI_POWER
5V
U31 1V2-3V6
1V65-5V5
VccA VccB
A1 B1
A2 B2
A3 B3
A4 B4
NC NC
GND OE
C87 100nF
C86 100nF
R63 1K
TXB104PWR
RX
TX
1V8
Logic Level Converter
Title XTEND900
UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:51:58 AM
Sheet of
File: C:\Documents and Settings\Roy Mustang\My Documents\Downloads\2010_SIGN\altium\wifi_isolated_uart.SchDoc
sunswift



Sunswift Room 342 EE Building UNSW, Sydney Australia 2052	ENGINEER: Irving Tjiptowarsono PCB DESIGNER: Irving Tjiptowarsono DATE: 6/1/2011 FILE NAME: sion_pcb_v5.1.PcbDoc	TITLE: SION PART NO.: Number DWG NO: Number	REV: 1 SCALE: Scale


UNSW Solar Racing Team
Room 342
Electrical Engineering
UNSW
Sydney NSW 2052
Title: SIONSEPIC v2 Prototype
Size: A4
Number:
Revision:
Date: 6/1/2011
Time: 1:47:36 AM
Sheet of
File: C:\sepic2\sepic.SchDoc
sunswift
Structural supports
Set to trigger shutdown at 2.45 V and startup at 2.55V (100mV hysteresis)



Sunswift Room 342 EE Building UNSW, Sydney Australia 2052	ENGINEER: Irving Tjiptowarsono	TITLE: Sepic revision	
	PCB DESIGNER: Irving Tjiptowarsono		
	DATE: 6/1/2011	PART NO.: Number	REV: Rev
	FILE NAME: sepic2.PcbDoc	DWG NO: Number	SCALE: Scale