EE582 Wireless Sensor Module Interim Technical Report

Team W.R.E.C.K.
Wireless Remote Electronic Car Kit

November 30, 2001

Table of Contents

Table of Figures

Group Members / Responsibilities

Erin Tilling

Electrical Engineering major specializing in Controls. Erin enjoys movies, concerts and playing guitar. In the future, Erin plans to get married in April to Mark Schnack and start her job at Proctor and Gamble in May. The user interface for the wireless sensor module was designed primarily by Erin.

Pat Stemen

Electrical Engineering major specializing in Computers. Pat enjoys movies, concerts and playing guitar badly. Pat plans to graduate and start graduate school in the fall, specializing in something. Most of the MCU research and interfacing was spearheaded by Pat.

Max Vilimpoc

Electrical Engineering major specializing in Communications/DSP. Max enjoys biking, skiing, and winning essay competitions. Max will be graduating in Spring Quarter and plans on volunteering with the Peace Corps for two years. The layout of this document reflects Max's mad skillz with Word.

Tony Pavick

Electrical Engineering major specializing in Electromagnetics and Communications. Tony enjoys engineering things for fun. In the future, Tony plans to get an advanced degree in Business or Biochemistry and Genetics. Tony worked in GPS and video research.

Gerard Garrelts

Electrical Engineering major specializing in Communications. Gerard is a member of a local band, Ordinary Peoples. In the future, Gerard plans to go on a worldwide tour with his band. Gerard created all sensor content and diagrams.

Dawn Kin

Electrical Engineering major specializing in Power and Controls. Dawn enjoys quiet walks in the park, kittens and world peace. In the future, she plans to graduate and marry rich. Dawn worked primarily with the Power Management aspects of the wireless sensor module.

Introduction

Before we discuss the technical design of our wireless sensor node platform, it is useful to discuss some of the motivations and requirements for the design. Throughout the design process, we considered the target platform to be the "Smokin' Buckeye", a Formula-1 style electric racecar designed and built by students and faculty at The Ohio State University. Although the "Smokin' Buckeye" is an extremely interesting platform to design for, the racecar has some very unique features that in some ways enhance our design, and in others constrain it.

There are several core reasons why we have chosen the electric racecar as our target platform. In comparison with a blimp-based design, the racecar allows us to concentrate less on weight constraints for the individual components of the design. For instance, with the blimp a few ounces for a single component could mean the difference between flying and not flying, whereas with the racecar, the effects of a couple extra ounces are less influential. We also feel that there are several stimulating sensor possibilities for the electric race car-based platform compared to the blimp.

What is the "Smokin' Buckeye"?

Although specific details about the "Smokin' Buckeye" design and feature set will be discussed in further sections of this design document, it is prudent to highlight a few significant features before we begin.

One of the most important features of the "Smokin' Buckeye" is its electric power source. The race car was designed from the bottom up to run off of battery power. Thirty-one lead-acid batteries power the race car, ultimately determining the length of operation. Because of its limited power supply, a large amount of resources should be dedicated to determining its remaining length and efficiency.

It is also important to remember the environment that the "Smokin' Buckeye" performs in. In a race situation, the car will be making many similar laps in repetitive fashion around a circular or ovular track. Because of the short life of the car's internal battery packs, monitoring and logging of their life and performance are extremely important features to the operators of the "Smokin' Buckeye". Throughout our design, we have tried to keep these ideas in mind. The feature set of our wireless sensor module was chosen based partially on the needs of the "Smokin' Buckeye" team, and partially on features we would like to include for other reasons.

Overall Design

In comparison to other designs for the wireless sensor module, which included multiple MCUs located throughout the Wireless Sensor Module, we decided to implement a simpler, single Microcontroller-based design.

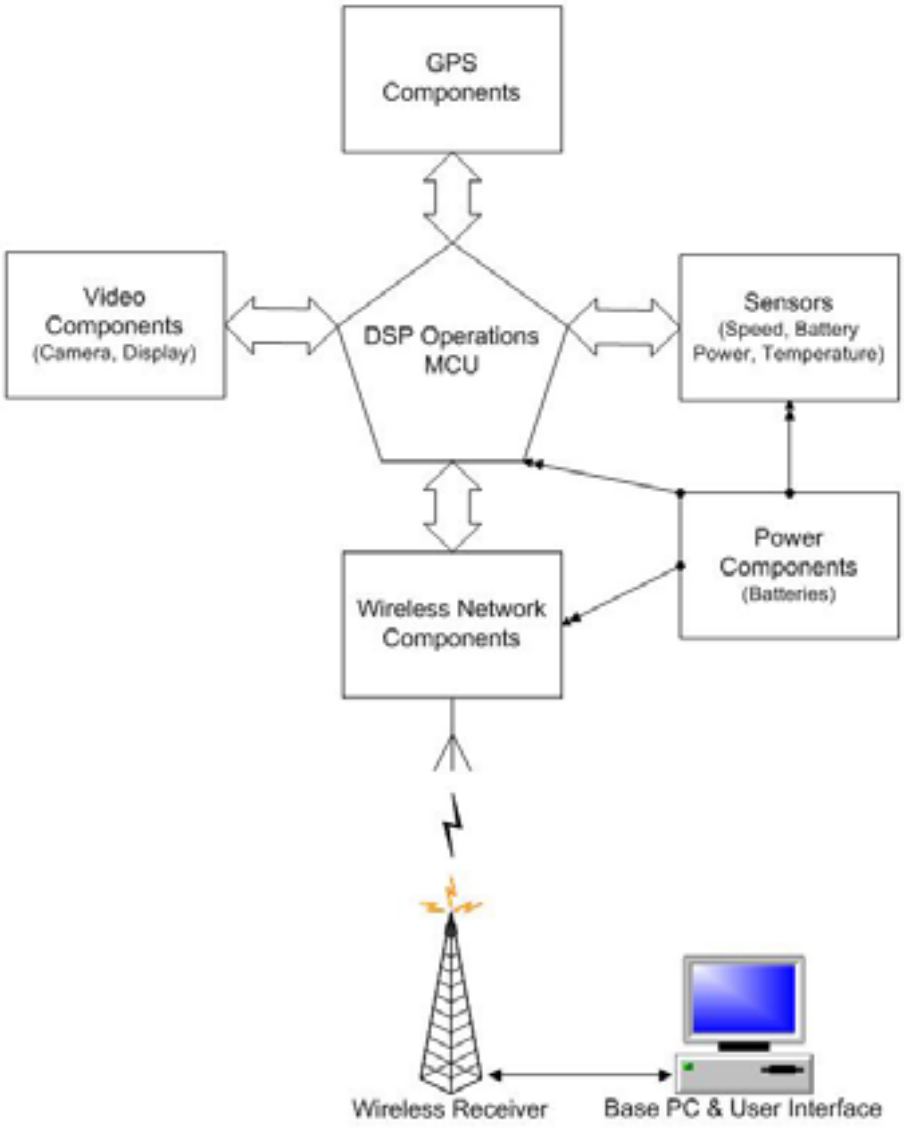


Figure 1. Wireless Sensor Module Block Diagram

Through the following pages we will discuss all facets of the wireless sensor module design. Before we begin, however, we will start with a requirements analysis and derived feature set. Then, the discussion will turn to the overall layout of the wireless sensor module.

Special care will be made at the beginning of each component section to highlight the features and requirements of the electric race car platform that drove the design decisions made. After review of the chosen products, each component section will close with comments on future changes, and room for possible improvement. In addition, interfaces between components will be discussed where applicable.

Requirements Analysis

Before beginning the design and component selection, we first looked at the core requirements for a wireless sensor node to be used on the "Smokin' Buckeye" platform.

- Separate Power Supply
- Fairly Lightweight
- Portable User Interface
- Temperature
- Acceleration
- Voltage, Current
- Digital Outputs (system status)

In addition to these requirements, several more were added by our design team.

- Rechargeable Power Supply
- 2 mile Wireless Range
- Global Positioning System (GPS) Input
- Video (Still or Motion)

These requirements led the design team to a feature set that encompassed the requirements of a possible customer, as well as their own.

High-Level Architecture

Referring to Figure 1, we can see the high-level architecture of the wireless sensor module. A single microcontroller provides both host and digital signal processing, and interfaces with sensor components. In addition, the MCU interfaces with video and global positioning system components, as well as the wireless link between the sensor module and the base station.
Other apparent features of the wireless sensor module include the power management system, which is responsible for delivering and managing power for all of the sensor module's systems. Power for the user interface and wireless module will be handled by a separate subsystem.

User Interface

The user interface for the wireless sensor module brings all aspects of the system together. It is a channel for the sensor information and allows the user to remotely control the sensor module. It also allows the user to plan for maximum power efficiency by logging the battery sensor information.

We decided use National Instruments' LabView for our user interface development. LabView provides all of the parameters that we might need in implementing the user interface and is easy to learn for a first time user. An important feature of LabView is that the user interface can be compiled into a portable (exe) file that can be used on any laptop or PC. Therefore, the laptop or PC does not need the LabView program for the user interface to operate. This offers us the flexibility of using all available laboratory resources while also allowing us to position our user interface anywhere in the field. Conveniently, LabView is already being used by the "Smokin' Buckeye" development team.

The user interface will include all of the information a user would need to track the "Smokin' Buckeye." This includes:

- Battery Power, Current and Voltage
- Temperature
- Wheel Speed
- Transmission State
- Wireless Module Battery Power
- Calculated Values
    - Efficiency
    - Acceleration
- GPS data
    - Car Position
- Video data
    - Streaming video from the cockpit of the car

User Interface Pseudocode

```
Sub SensorReset ()
      Speed.Clear
      BatteryPower.Clear
      BatteryTemp.Clear
      TransState.Clear
      RPM.Clear
      Overheat = False
      MotorTemp.Clear
      MotorOverheat.Clear
End Sub

Sub PacketWatch
      // Wait for packet
```

```
      Open packet
      Decode packet
      for (i = 0, i < #sensors, i++)
         switch(i)
            case(0)
               Speed = packet[i];
            case(1)
               BatteryPower = packet[i];
            case(2)
               BatteryTemp = packet[i];
            case(3)
               TransState = packet[i];
            case(4)
               RPM = packet[i];
            case(5)
               Overheat = packet[i];
            case(6)
               MotorTemp = packet[i];
            case(7)
               MotorOverheat = packet[i];
End Sub


Sub ModuleStandby
      MCU.Idle
      SystemStatus.IndicateStandby
End Sub


Sub ModuleShutdowm
      Speed.Shutdown
      BatteryPower.Shutdown
      BatteryTemp.Shutdown
      TransState.Shutdown
      RPM.Shutdown
      MotorTemp.Shutdown
      MotorOverheat.Shutdown
      SystemStatus.IndicateShutdown
End Sub


Sub ModuleStartup
      Speed.Powerup
      BatteryPower.Powerup
      BatteryTemp.Powerup
      TransState.Powerup
      RPM.Powerup
      MotorTemp.Powerup
      MotorOverheat.Powerup
      SystemStatus.IndicateStartup
End Sub
```

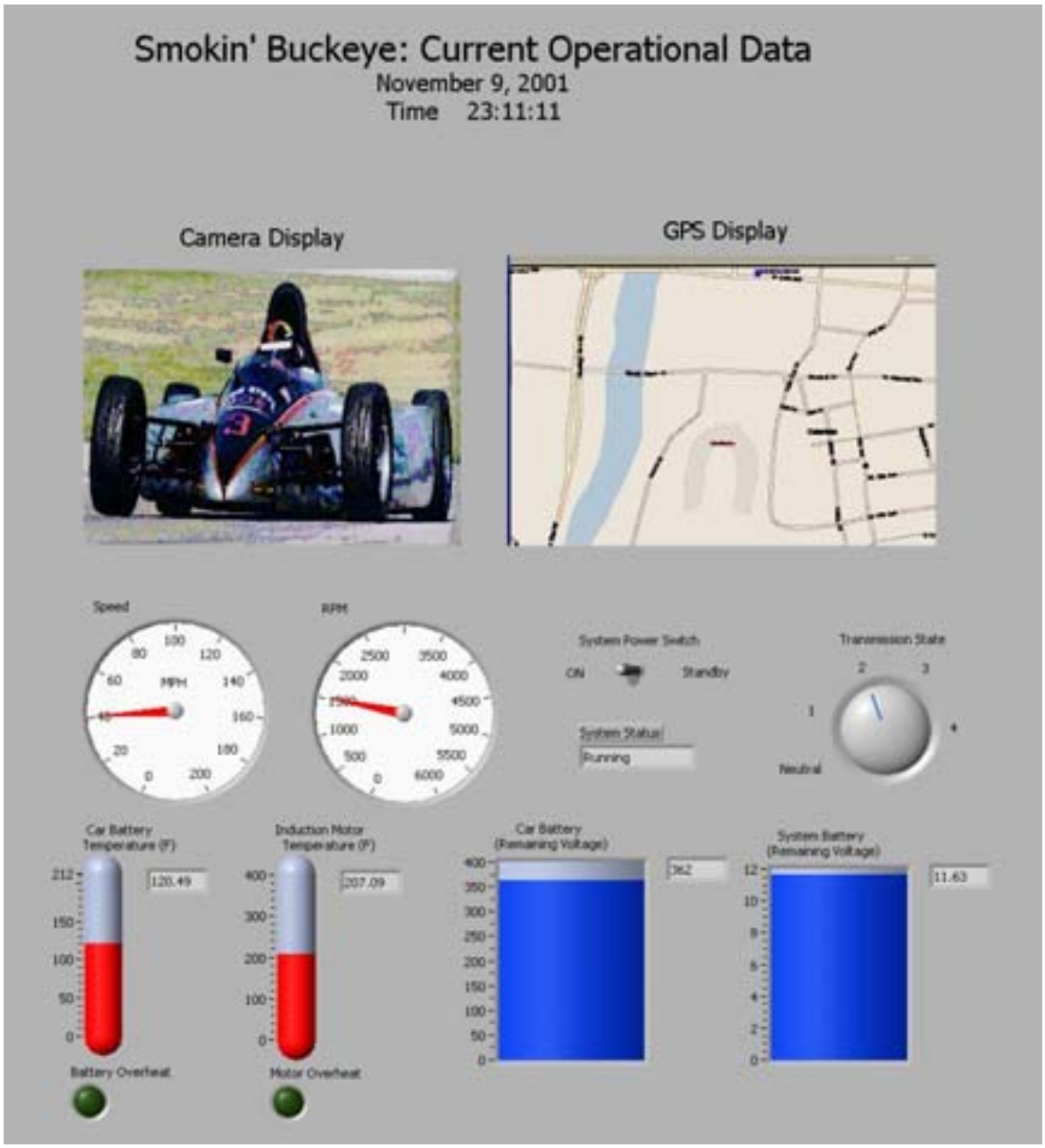


Figure 2: **User interface for the "Smokin' Buckeye" using Lab View**

Sensors

One of the most exciting elements of the Smokin' Buckeye project is the opportunity to sense many different variables. These elements have a direct impact on the car's performance and it is important that the manner in which they are sensed is chosen carefully. Team W.R.E.C.K. is exploring a number of variables that we think are crucial to the project. These variables are explored in the following paragraphs.

Motor Temperature

The temperature of the racecar's induction motor is an important variable to monitor because overheating the motor spells certain disaster. If the temperature of the motor gets too high, there is a risk of overheating the insulation on the windings of the motor and causing permanent damage to it. A common rule is that overheating an induction motor will halve the life of the motor for every 10°C past its rated temperature. Motor temperature is an important statistic to keep track of because if the motor has been overheated repeatedly, it may not be relied upon to complete a race. Just as in an internal combustion engine, temperature is a variable that needs to be closely monitored to avoid complications.

Battery Temperature

The temperature of the car's 31 lead-acid batteries should also be monitored. This would be done using the same basic techniques as used for the motor. A sensor could be mounted onto each battery and would be connected to a circuit that would trigger an alarm if the temperature of any of the batteries exceeded a certain temperature. Another way that the overall temperature could be monitored would be to average the temperature of each of the batteries and then have this value available for readout on the pit crew's laptop.

National Semiconductor LM-135[1]

The sensor that chosen to monitor the motor's temperature is the LM135 made by National Semiconductor. This IC comes in 3 different packages: Plastic, Surface Mount, and Metal Can. We hypothesized that the metal can package would be the better choice because it appears to be more durable and would conduct heat from the motor's casing the most efficiently. Some of the positive features for the LM135 are the following:

- less than 1°C error over 100°C range when calibrated at 25°C
- -55°C to +150°C Temperature Range
- Linear output
- Easy to interface to readout circuitry
- Low Power (about 1mA for normal operating conditions)
- Low Cost

[1] http://www.national.com/ds/LM/LM135.pdf

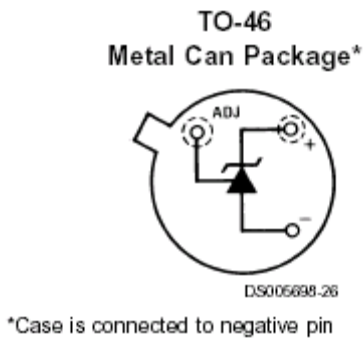


Figure 3. Schematic Diagram of the LM135

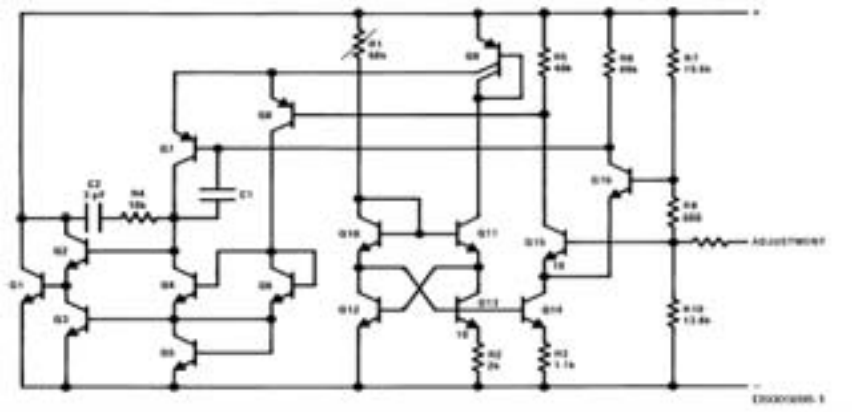


Figure 4. Internal Schematic of LM135

Accuracy:

The accuracy of the LM135 is actually much higher than necessary for this application. The sensor boast only .3 degrees of error for typical 1mA operation. A portion of the technical specifications are reproduced in Figure 5 below.

Temperature Accuracy (Note 1)
LM135/LM235, LM135A/LM235A

Parameter	Conditions	LM135A/LM235A			LM135/LM235			Units
		Min	Typ	Max	Min	Typ	Max	
Operating Output Voltage	T_C = 25°C, I_R = 1 mA	2.97	2.98	2.99	2.95	2.98	3.01	V
Uncalibrated Temperature Error	T_C = 25°C, I_R = 1 mA		0.5	1		1	3	°C
Uncalibrated Temperature Error	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		1.3	2.7		2	5	°C
Temperature Error with 25°C Calibration	$T_{MIN} \le T_C \le T_{MAX}$, I_R = 1 mA		0.3	1		0.5	1.5	°C
Calibrated Error at Extended Temperatures	$T_C = T_{MAX}$ (Intermittent)		2			2		°C
Non-Linearity	I_R = 1 mA		0.3	0.5		0.3	1	°C

Figure 5. Temperature Accuracy of LM135

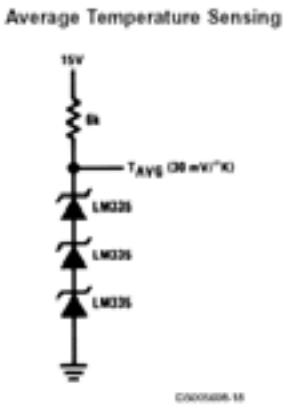


Figure 6. A temperature sensing circuit.

Wheel Speed: Philips KMI15 Rotational Speed Sensor[2]

The speed of the car is an obvious sensing requirement for both the driver, but is also helpful to for the pit crew. The crew can compare the speeds of previous laps or even previous races to give advice to the driver as to how he/she should be driving. The best choice for wheel speed sensors that W.R.E.C.K. found was the KMI 15 made by Phillips Semiconductor. This sensor consists of three major components: the magneto resistive sensor, a permanent magnet that is attached to the sensor, and an integrated signal conditioning unit. The reason Philips was chosen was due to its ability to effortlessly interface with digital systems. This is mainly due to the signal conditioning unit included with the KMI 15. If supplied 5 V, the output of the sensor will be a 5 V square wave of appropriate frequency. The KMI15/16 also has the ability to sense speed accurately at speeds below 5 mph.

Highlights of the KMI 15:

- Wide air gap allowed because of high sensitivity of sensor
- Zero Speed Detection
- Insensitive to vibration
- Wide Operating Temperature Range

[2] http://www.semiconductors.philips.com/pip/kmi15/4#datasheet

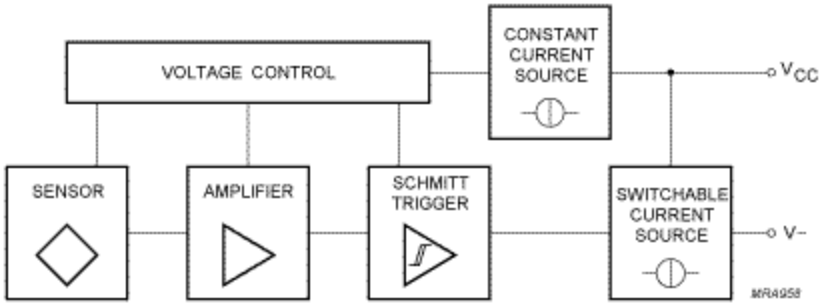


Figure 7. Block Diagram of KMI 15

Acceleration

The acceleration of the car is another important variable because it can be used to calculate both the position and velocity of the car using algorithms that are put into place either in the car, or on the laptop in the pit. Using the data sent by the accelerometer, the car's location can be accurately plotted using the GPS system. This will be described in more detail later on in the report.

One of the best accelerometers on the market is the ADXL202[3] made by Analog Devices. This small chip can be easily interfaced to many common microprocessors and the instructions on how to do so are included on Analog Devices' website. Some benefits of the ADXL202:

- 2 Axis of Acceleration Sensing on a Single IC Chip
- 5 milli-G Resolution
- Duty Cycle Output with 1ms Acquisition Time
- Low power > 0.5mA (ADXL202)
- Direct Interface to Popular Microprocessors
- BW Adjustment with a Single Capacitor
- +2.7V to +5.25V Single Supply Operation
- 1000g Shock Survival
- Low Cost

[3] http://products.analog.com/products/info.asp?product=ADXL202

GPS

The Global Positioning System (GPS) provides the Smokin' Buckeye racing team with some interesting options. Many of these options will provide the car with additional functionality, but some will simply add to the entertainment value of the car. The extent to which these options can be employed is dependent upon the accuracy and refresh rate of the GPS unit as well as the bandwidth of the wireless subsystem. While choosing a GPS system, it is necessary to weigh the benefits of the system against the cost and degree of difficulty required to interface the GPS with the MCU.

Practical Uses of a GPS

Using the GPS to determine track location is a multi-purpose operation. While operating on a closed track, the number of laps remaining can be counted to calculate distance remaining. On an open road, a GPS unit would be beneficial in calculating distance to the finish line. In either case a software program could be installed on the host computer that would calculate the remaining distance. Using this calculated value and the car's average energy consumption, a race team could establish the speed that the car must maintain in order to have enough battery power to complete the race.

When coupled with sensor readings from the cars power supply, logging GPS coordinates would enable the race team to determine energy consumption at given average velocities. Because the car will consume more energy while accelerating, acceleration readings must also be tracked and accounted for. This practice could prove especially beneficial during pre-race activities. Mapping the path of the car for several laps around a track would enable the team to determine a path of least energy. The aforementioned readings also allow the team to determine whether or not energy consumption varies with remaining voltage.

With the growing popularity of racing, there may be an audience for internet published real-time race positions. If every car was equipped with a GPS unit, each unit's coordinates could be collected at a base station, overlaid onto a track map, and broadcast over the internet. This process would be relatively simple and if implemented would be done so by the racing commission, so cost, other than the GPS unit, is not applicable.

Potential Drawbacks of a GPS

Cost is the biggest and most obvious drawback of a GPS. Most GPS units cost between one hundred dollars and several thousand dollars and can be purchased from a wide variety of distributors. The accuracy of these units will vary from 75 m to 1 cm, but with increased

accuracy comes increased price. Many units come with an available Differential GPS input. This feature allows a stationary receiver to send correctional code to our moving unit that is on the car. This doubles the cost of system because we must buy two receivers. One possible solution is to propose the idea of internet race positions to a governing body and encourage them to purchase a DGPS receiver that can transmit stationary position data to all race participants.
An alternative, and less costly solution is to forego the DGPS setup and instead install an accelerometer in the racecar. The concept behind this setup is that given a set of coordinate readings and the acceleration between them we can accurately calculate the position of the car.

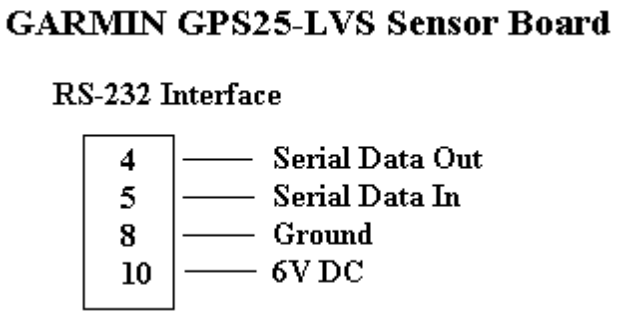
Choosing a Project-Compliant GPS

Because our wireless communications system and all of its components will be powered with a series of 6V batteries, it is beneficial to select a GPS module that can operate at 6V or 12V. The size of the module is also of important consideration because of the limited space available on the car. The Garmin GPS25-LVS fits our requirements quite well. The unit weighs just 1.3 ounces and consumes .5 W of power. GPS data is output via an RS-232 interface. An external antenna is essential because the GPS module will be safely enclosed in the car. This means that an internal antenna would get very poor reception. A patch antenna is preferred because of durability and aerodynamic factors.

Interfacing the GPS with the MCU

The GPS unit is interfaced with the rest of our system through the use of a DB9 serial port. There is a choice of three data protocols, TSIP, TAIP, NMEA. We have decided that the positional data will be sent to the host PC and combined with data from the accelerometer. This will give us accurate positioning data.

The figure below shows the RS-232 Interface required for communications with the GPS unit.



Video

Perhaps the most challenging aspect of the wireless sensor module is the video system. Although an excellent feature for the end user, video is extremely difficult to implement properly in a cost-sensitive and bandwidth-limited situation.

Still vs. Motion Pictures

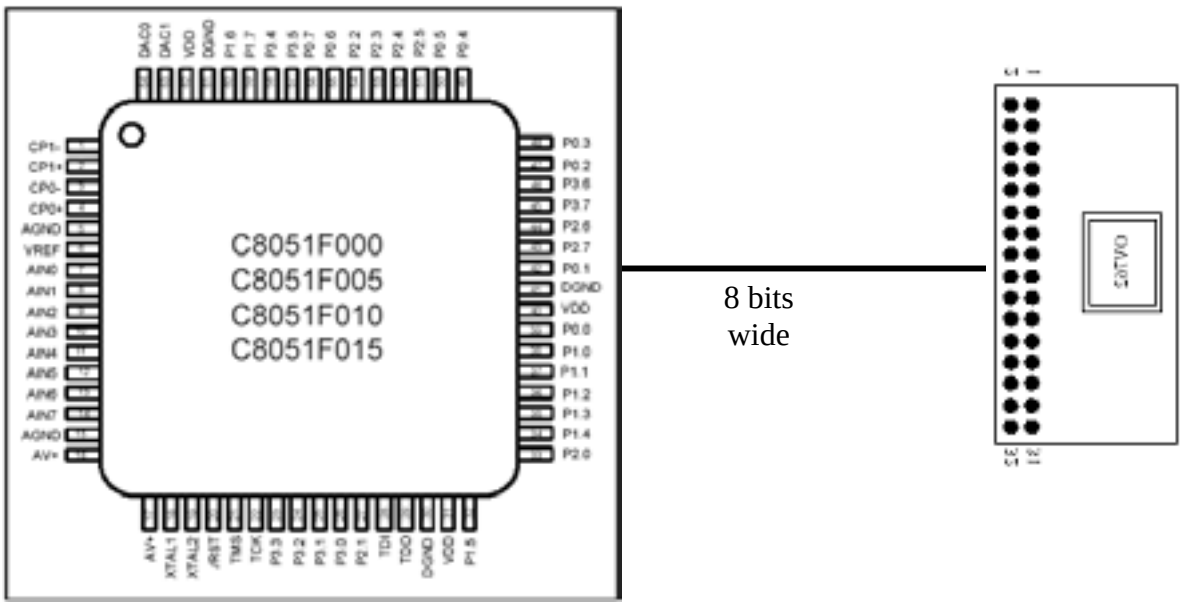
Due to the limitations on the bandwidth of our wireless subsystem, it may only be possible to periodically transmit still images back to the user interface. Ideally, we would transmit full-motion video, however we have yet to work out the implementation details.

CCD Color Camera Module Information

Due to the digital nature of the MCU, we are limited in the hardware we can use for video input. It will be very difficult to digitize analog video from a standard video source with a composite output. With these constraints in mind, we have chosen a color camera module that has a well-known interface with a serial digital output.

A kit comprised of an OmniVision CCD camera and some additional hardware is available on the market. This kit is unique because the interface is I2C, a well-known standard serial interface, conveniently part of our Cygnal MCU. Parameters such as exposure, gamma, gain, white balance, color and matrix windowing are programmable through the I2C interface.

Video data is output over an 8 bit digital I/O port, which is integrated onto the same PC board as the camera module. It will be possible to connect these 8 bits to one of the Digital I/O ports on the Cygnal MCU and transmit the video data in a raw format to the User Interface. Once at the user interface, the host PC will be required to do the majority of the video processing, including image reconstruction and playback.



Microcontroller Unit (MCU)

The microcontroller until will be responsible for performing the following tasks during normal operations:

- Reading of raw sensor data
- Communicating with GPS device
- Receiving and repackaging video data
- Building Packets for transmission over wireless link
- Communications with wireless subsystem

Traditionally, each of these tasks may require a separate microcontroller with serial or parallel communications between them. However, by using a single, yet powerful, microcontroller, we can perform all of these tasks on the same device.

Component Selection

There are many options on the market for microcontrollers, all offering a wide variety of features. After comparing products from several companies such as the PIC microcontroller, and the Atmel units, we decided that the Cygnal 8051Fxxx series microcontroller was the best product for the job.

The Cygnal 8051Fxxx series is based on the Intel 8051 architecture. The 8051 is licensed free of charge by Intel to other manufacturers such as Cygnal, increasing the installed base of the architecture. The features that make the Cygnal microcontroller more appealing than other comparative products include:

Raw Processing Power

The 8051Fxxx series microcontrollers have a 20 MIPS (Million Instructions Per Second) central processing unit. This will be more than adequate to handle the processing requirements of our wireless sensor node, including video and GPS system processing. In addition, should power management become an issue, we can step back the clock frequency to extend battery life.

Digital I/O

Perhaps the best feature of the Cygnal microcontroller is the abundance of Digital Input / Output ports. The basic Fxxx-series microcontroller, the 8051F000, has 32 1-bit input / output ports that can be toggled individually. In addition to these simple I/O ports, there are several built-in timers and a standard serial communications UART buffer.

Inter-device wired communications

Another benefit of the Cygnal 8051F000 microcontroller is the abundance of on-board inter-device communications systems. The most notable of these busses are the SPI (Serial Peripheral Interface) and the SMBus (Systems Management Bus). These wired interfaces will make connection to other devices easier.

Power Management

Power management is a very important consideration in the design of the wireless sensor module. While the components of the module do need to be powered, we do not want to use the racecar batteries as a power source.

Battery Power

The racecar uses 31 Optima Spiral Wound Lead-Acid batteries, which are twelve volts apiece and packaged into battery boxes that contain two batteries. Due to controller limitations, one box only contains a single battery. Eight boxes are situated on each side of the car and weigh 720 pounds total. Since these batteries weigh so much, and the goal of any race car is to limit weight as much as possible, the power supply weight for the node must be minimal.

There are many battery options available, and each having positives and negatives. Lead-acid type batteries are inexpensive, reliable and have a fairly long life cycle. However, they have a relatively low energy density as well as manufacturing and disposal hazards. We compared numerous other types of batteries, such as Lithium, Nickel-Cadmium, and NiMH. Nickel-Cadmium stood out for its smaller size, longer lifetime, and greater energy density than Lead Acid. Unfortunately, there is the possibility for performance problems at high temperatures. Also, they are more expensive than Lead-Acid, yet are cheaper than the other alternatives. A benefit of this type of battery is the rechargeable nature and high duty cycle, therefore cost will not be as great an issue.

The battery we chose is a hobby-type 6.0V Nickel-Cadmium battery for radio-control applications. To provide the 12 volts needed to power the higher voltage components in our module, two of these batteries would be connected in series. For the lower voltage devices requiring 5 volts, a few options are available. First, a series regulator could be placed between our power supply and our 5V digital devices. There are many corporations that manufacture 12V to 5V voltage regulators. Our second option is a rail splitter, which we will have available by simply connecting in parallel to a single battery of our power supply.

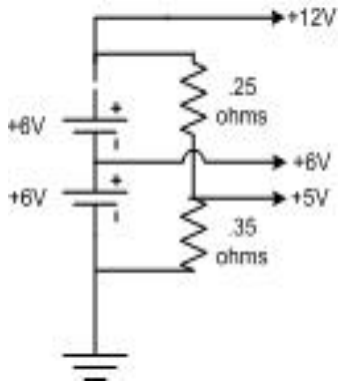


Figure 8. Rail Splitter and Series Regulator

Power Management Techniques

There are several techniques that can be considered for maximum power conservation of the wireless sensor module. First, the lowest voltage supply possible should be used. This is the reasoning behind using 12 volts as an input to the system. Also, power consumption is directly proportional to the clock frequency (SYSCLK) of a microcontroller. When designing the internal oscillator configuration, the lowest possible frequency required should be chosen to operate the device until a condition occurs that requires high frequency operation. (See Figure 9)

The Cygnal C8051 device that we have chosen also features power management modes of IDLE and STOP. During IDLE, the CPU and FLASH memory are taken offline by setting the Idle Mode Select Bit (PCON.0) to '1'. External peripherals of the CPU remain active during this mode, including internal clocks. The CPU exits IDLE mode when an enabled interrupt or reset occurs.

STOP mode completely shuts down the CPU and oscillators, which shuts down all digital perepherals, as well. This is done by reducing the SYSCLK frequency to zero, and can be used to save power when no device operation is required. The CPU can then be "woken up" when needed by using a Comparator zero reset.

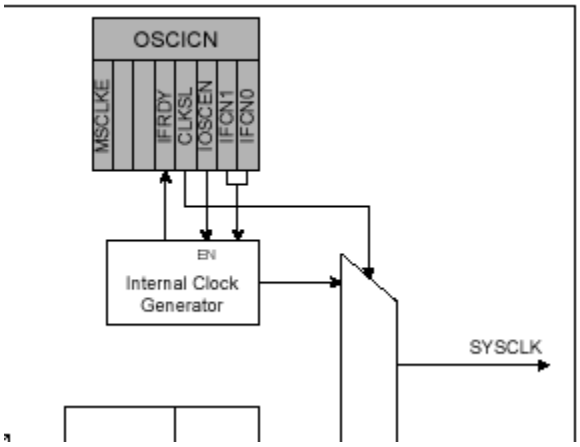


Figure 9. Cygnal Internal Oscillator Control Port

Assimilating information from the figure above and Page 99 of the 8051F000 microcontroller datasheet[4], we see that our options for internal system clock generation are the following:

Bits1-0: IFCN1-0: Internal Oscillator Frequency Control Bits

00: Internal Oscillator typical frequency is 2MHz.
01: Internal Oscillator typical frequency is 4MHz.
10: Internal Oscillator typical frequency is 8MHz.
11: Internal Oscillator typical frequency is 16MHz.

Power Management Summary

A number of options are available for managing and minimizing the amount of power used by the wireless sensor module. Using Nickel-Cadmium batteries offers us economy and rechargability. In addition, our chosen microcontroller provides energy efficiency through internal clock regulation.

[4] http://www.cygnal.com/datasheets/c8051fxxx.pdf

Wired and Wireless Communication

Wired Communication Standards

During the development of the race car communications package, we investigated a large number of options for both wired and wireless information transfer. Indeed, quite a number of different systems are available on the market in development kit form. Choosing the proper product required investigating a number of criteria such as performance, cost, and power consumption.

In the racecar, we decided that using a standard bus would result in the most efficient transfer of sensor information to our microcontroller units (MCUs). Rather than inventing our own bus, we noted that a number of MCUs already had bus interfaces integrated into their input-output subsystems. A variety of acronyms entered the development process here: Serial Peripheral Interface (SPI), Inter Integrated Circuit (I2C), System Management Bus (SMBus), Microwire, Controller Area Network (CAN), and others. From among these options, we decided that the Serial Peripheral Interface (SPI) bus would allow us to achieve our development targets with the most flexibility and speed.

SPI

With a top data rate of over 1Mbps, the Serial Peripheral Interface offered us more than enough performance to measure and disseminate information from sensors mounted around the car. Physically, the SPI-bus is built on a four-wire plus ground connection network. Three of the wires, known as MOSI (Master Out Slave In), MISO (Master In Slave Out), and SCLK (System Clock), can be shared among all devices attached to the bus. These three wires represent the physical medium for data exchange. The fourth wire, SSn (Slave Select n), must be attached in a one-to-one ratio with each device. The SSn line acts as an enable/disable switch for each device, and determines which one will have access to the bus at any given time.

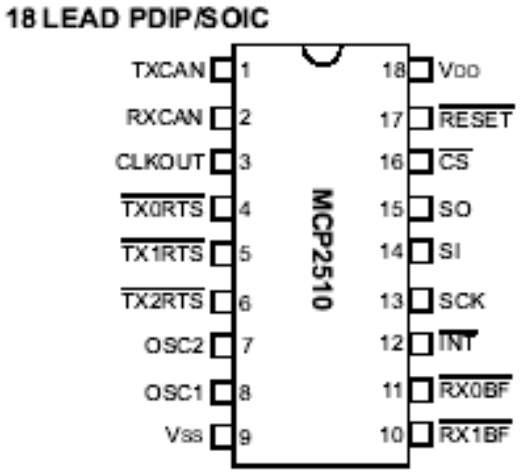
From our research, it was clear that a number of vendors offered direct conversion Analog-to-Digital sensors, which could detect physical phenomena and output measurements directly into an SPI-bus compatible data type. Having such capabilities built into the sensing devices correlates to a shortened development cycle.

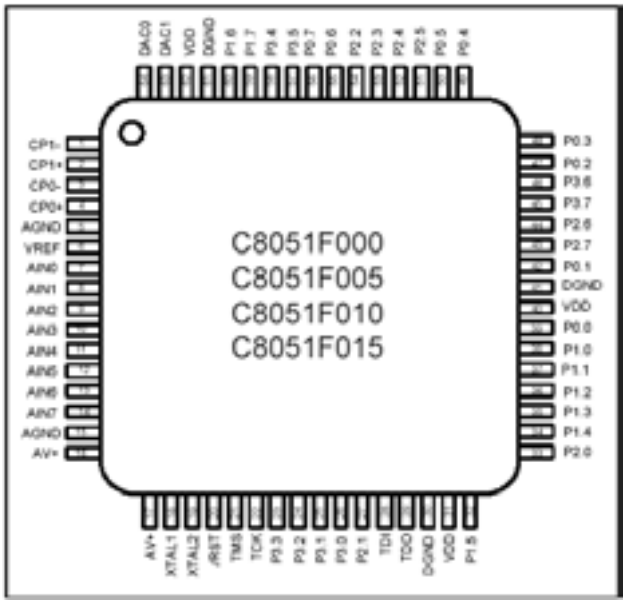
CAN

However, as it turns out, another bus type is already in place on the electric racing vehicle. Rather than use the SPI-bus, the maintainers of the vehicle chose to implement a Controller Area Network (CAN) bus, which is a growing standard for control and communication in automobiles around the world. As of 1999, the number of CAN systems in operation is well over 150 million.[5] In its physical form, the CAN bus topology can assume a number of different forms, depending on the developer's preference in implementation. Networking media include wired links, optical, and as well as wireless links. The bus can be implemented as a two-wire, terminated parallel-access form, or as a single wire link. Because of this reduction in wiring requirements, the costs are lower for designing around the CAN bus.

[5] http://www.can-cia.de/cg.htm

SPI – CAN Interface





It turns out that a plausible solution lies in the Microchip MCP2510 CAN bus interface chip. Using this chip, it is possible to convert a standard SPI interface signal into a signal compatible with the Controller Area Network. The following is a table that lists the necessary SPI connections between the Cygnal MCU and the MCP2510.

MCU Port.Bit	**MCU Physical Pin Number**	**Function**	**MCP2510 Pin Name**	**MCP2510 Pin Number**
Port 0.0	39	SCK	SCK	13
Port 0.1	42	MISO	SO	14
Port 0.2	47	MOSI	SI	15
Port 0.3	48	NSS	CS	16

Sensor Bus Pseudocode

```
/* Cygnal MCU SPI pseudocode */
entity CygnalMCU is
      port(SCK  : output bit);
      port(MISO : input  bitvector(0..7));
      port(MOSI : output bitvector(0..7));
      port(NSS  : output bit);
end CygnalMCU;


/* Microchip MCP2510 SPI pseudocode */
entity MCP2510 is
      port(SCK   : input bit);
      port(MISO  : output bitvector(0..7));
```

```
      port(MOSI  : input  bitvector(0..7));
      port(NSS   : input  bit);
      port(RESET : input  bit);
      port(TXCAN : output bit);
      port(RXCAN : input  bit);
      port(CLKOUT: output bit);
end MCP2510;

entity GenericSensor is
      port(ADCOUT : output bitvector(0..7));
      port(CLK    : input  bit);
end GenericSensor;

architecture dataflow of SPIBus is
      MCP2510.SCK    <= CygnalMCU.SCK;
      CygnalMCU.MISO <= MCP2510.MISO;
      MCP2510.MOSI   <= CygnalMCU.MOSI;
      MCP2510.NSS    <= CygnalMCU.NSS;
end dataflow;

architecture dataflow of MCP2510 is
      TXCAN <= MOSI;
      RXCAN <= GenericSensor.ADCOUT;
      MISO  <= RXCAN;
end dataflow;
```

Wireless Communications

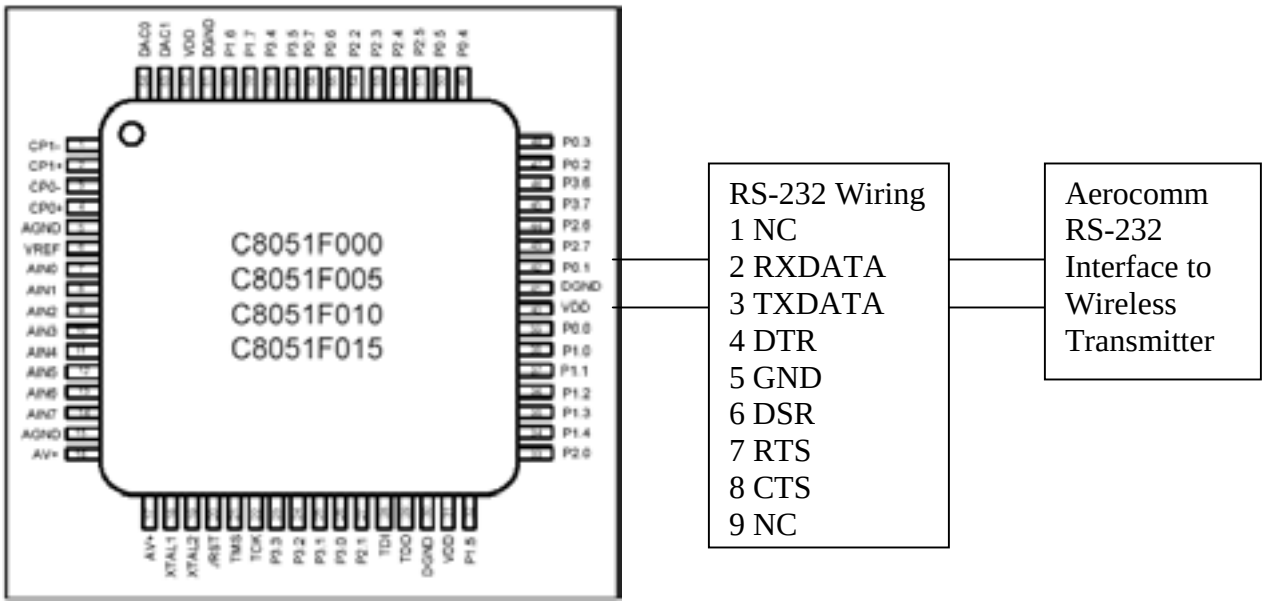
For the wireless transmission of data from the moving vehicle to our data-acquisition system, a number of potential receiver/transmitter pairs were investigated. The transmitters were generally designed around two frequency ranges, 900MHz and 2.4GHz. The 900MHz modules were generally lower-cost and lower range, usually reaching a limit of 1000' line-of-sight operation. The 2.4GHz modules were generally higher-cost and higher-range, usually capable of exceeding 2 miles of line-of-sight operation. On a positive note, most of the parts listed in the table below also had a standard serial interface, which would enable rapid integration with our MCU.

Three essential factors were used to benchmark the utility of each solution: range, data rate, and price.

Company	Product	Range (ft)	Price Component Eval Kit ($)	Data Rate (kbps)	Power Requirements (V)	Channels	Duplex
Radiotronix, Inc.	EWM-900-FDTC	1000	$69 / $299	19.2	3	56	Full
Linx Technologies	MDEV-900-HP	1000	$53 / $299	50	9	8	Half
Coyote DataCom, inc.	DR-915L	2500	$76 / $319	50	7.5-12 / 75mA	62	Full
Aerocomm	AC5124C-200	10000	No Cost	1Mbps	5V	77	Full

Table 1: Wireless System Development Kit Comparison

From the above table, it is reasonable to note that there are two technologies competing to satisfy our wireless communications requirements. From Coyote DataCom, inc., we have an inexpensive 900MHz receiver/transmitter which is capable of half-mile communication. From AeroComm, a 2.4GHz solution exists with an excellent data rate, and excellent range, but will likely be much higher in cost than the DataCom unit. Initially, Team W.R.E.C.K. chose the DataCom unit for our wireless requirements. However, given its incredible data rate performance and ~2 mile line-of-sight range, we now believe that the Aerocomm unit would provide a better solution for our wireless communication needs.



Wireless Transmission Arbitration Pseudocode

```
/* assemble a packet of data */

for (;;) {
   transmitbuf[0] = gpsbuffer;
   transmitbuf[1] = voltagebuffer;
   transmitbuf[2] = currentbuffer;
   transmitbuf[3] = rpmbuffer;
   transmitbuf[4] = tempbuffer;
   transmitbuf[5] = transmissionstatebuf;
   transmitbuf[6] = wmodbattpower;
   transmitbuf[7] = videodatalength_hi;
   transmitbuf[8] = videodatalength_lo;
   transmitbuf[9..20000] = videodata;
   send(transmitbuf);
}
```

Conclusion

Throughout this document we have shown the evolution of a wireless sensor module from a high-level architecture to an interconnected series of components. The components work together to provide the required features and functionality necessary for the product. Special care was also taken in choosing components for the design to fit the requirements defined by the “Smokin’ Buckeye” design team.