IMUTIME SYSTEMS
School of Engineering Science
Simon Fraser University
Burnaby BC, V5a 1S6
ITS-ensc440@sfu.ca

February 19, 2007

Lakshman One
Simon Fraser University
Burnaby, BC, V5A 1S6

RE: ENSC440 Design Specification for Slalom Race Organizer

Dear Mr. One,

Enclosed please find our *ENSC440Design Specification for Slalom Race Organizer*. Our objective is to list the functional specifications needed for our 440 project.

We are on our way designing and building an automated kayak race timing system which can be installed on existing kayak race courses to eliminate the need for judges at each gate. This device will also wireless transmit all of this data to the judges station, providing a wealth of information for race timing and self-improvement purposes.

In the attached design specification, we outline the design and testing that our completed Slalom Race Organizer with fulfill. Furthermore, this document will list the design specifications for the successful project completion deadline in April as we hope to have a production-quality version ready at this time.

ImuTime Systems is comprised of four innovative and driven engineering students – Ashley Penna, Chris Munshaw, Kevin Lockwood, and John So. Refer to Team Organization and Company Profile section of our proposal to learn more about the ImuTime Systems team.

If you have any questions or concerns regarding the attached document please feel free contact us by email at ITS-ensc440@sfu.ca

Sincerely,

Kevin Lockwood
President & Project Manager
ImuTime Systems

Enclosed: *Design Specification for Slalom Race Organizer*

Design Specification for

# Slalom Race Organizer

***Project Team:*** Kevin Lockwood
Chris Munshaw
John So
Ashley Penna

***Contact Person:*** Chris Munshaw
cmunshaw@sfu.ca
ITS-ensc440@sfu.ca

***Submitted to:*** Steve Whitmore – ENSC 305
Lucky One – ENSC 440
School of Engineering Science
Simon Fraser University

***Issued Date:*** March 5, 2007
***Revision:*** 1.1

## *Executive Summary*

Whitewater kayak slalom racing began shortly before World War II as a branch of traditional whitewater canoe racing. This Olympic sport involves racers paddling down a natural or man-made river through hanging pairs of gates in a kayak, usually made of fiberglass or plastic. The racer must proceed through green gates in the down-river direction (see Figure 1), and red gates in the up-river direction. The overall time of the racer from start to finish is recorded, and time penalties are added for gates touched by the racer as well as missed gates. A judge watches each set of gates from the shoreline, recording time penalties for each racer that goes by. Human judging is an inefficient method that introduces a huge potential for human error.

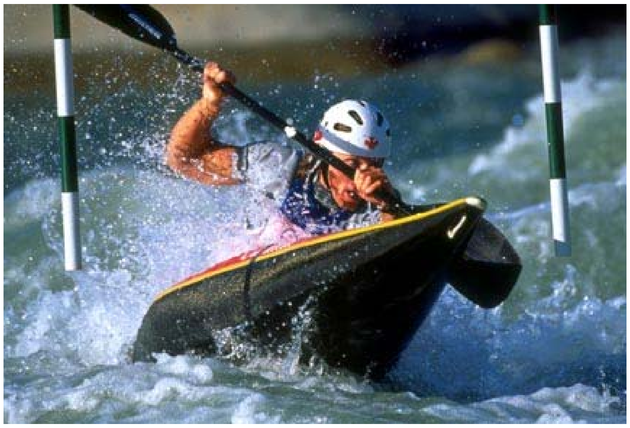


**Figure 1 - Slalom Kayak Racer – Source: IOC**

Currently, each of the gates is individually manned. A judge determines if the kayaker passes through the each set of gates, as well as determining the time at the beginning and the end of the race. Using humans for this task is cumbersome, potentially inaccurate and expensive.

This device will have the following features:

1. Sensors mounted on each of the two sets of poles can tell if the kayaker has passed through them.
2. Sensors at the beginning and end of the race can tell when the race starts and finishes.
3. Accelerometer will be able to tell if kayaker has bumped either of the poles.
4. All data will be transmitted to the judge's station for analysis.

This will be completed by April of 2007, and will be a production-ready prototype.

# *Table of Contents*

Executive Summary ... 1
1 Introduction ... 5
  1.1 Scope ... 5
  1.2 Acronyms ... 5
  1.3 Referenced Documents ... 5
  1.4 Intended Audience ... 6
2 System Overview ... 6
3 System Hardware ... 7
  3.1 Sensors ... 10
    3.1.1 IR Sensors ... 10
    3.1.2 Ultrasonic Sensors (backup configuration) ... 10
    3.1.3 Accelerometers ... 11
    3.1.4 Signal Conditioning ... 11
  3.2 Microcontroller ... 11
    3.2.1 Development Board ... 12
    3.2.2 Microcontroller ... 14
4 Gate to Computer Bridge ... 15
5 Firmware Design ... 16
  5.1 MPLab ... 16
  5.2 Core Processor Support ... 17
  5.3 Board Support ... 17
    5.3.1 Available Space ... 17
    5.3.2 Instruction Set ... 17
  5.4 Communication ... 18
6 Application Software Design ... 18
7 Test Plan ... 20
  7.1 Hardware Test Plan ... 20
    7.1.1 Sensor Test Plan: Environmental Conditions ... 20
    7.1.2 Sensor Test Plan: Performance Analysis ... 20
    7.1.3 Power Supply Test Plan ... 21
  7.2 Wireless Test Plan ... 21
  7.3 Software Test Plan ... 21
  7.4 Firmware Test Plan ... 21
8 Conclusion ... 21
Appendix A – Schematics ... 22

## *List of Figures*

Figure 1 - Slalom Kayak Racer – Source: IOC ..... 1

Figure 2 - Slalom Race Organizer System Overview ..... 6

Figure 3 – Sensor Layout 1 ..... 7

Figure 4 – Sensor Layout 2 ..... 8

Figure 5 - Network Host Block Diagram ..... 8

Figure 6 - Start/End Module Block Diagram ..... 9

Figure 7 - Gate Module Block Diagram ..... 9

Figure 8 - Helmet Module (Sensor Layout 2 only) ..... 10

Figure 9 - Button Schematic ..... 12

Figure 10 - Communication Schematic ..... 13

Figure 11 - Power Schematic ..... 13

Figure 12 - Block Diagram of Timer ..... 14

Figure 13 - MPLab Interface ..... 17

Figure 14 - Instruction Set ..... 18

Figure 15 – The Timing Computer GUI ..... 19

Figure 16 – IR Emitter and Receiver Configuration ..... 22

Figure 17 – Accelerometer Configuration ..... 22

Figure 18 - Development Board Schematic ..... 23

# Revision History

| Revision | Date | Description | Name |
|---|---|---|---|
| 0.1 | March 2, 2007 | Initial Version | Ashley Penna |
| 0.2 | March 2, 1007 | Added Executive Summary, Introduction, Scope, System Overview and Intended Audience. | Ashley Penna |
| 0.3 | March 3, 2007 | Added Wireless Overview and Testing | Kevin Lockwood |
| 0.4 | March 3, 2007 | Added to Hardware, GUI sections | Chris Munshaw, John So |
| 0.7 | March 4, 2007 | Added Microcontroller Design Specs and Firmware Testging | Ashley Penna |
| 0.8 | March 5, 2007 | Added Firmare Design | Ashley Penna |
| 1.1 | March 5, 2007 | Added schematics | Chris Munshaw |

**Table 1 Revision History**

# 1 *Introduction*

The Slalom Race Organizer is a system that will track information about a racer's progression through the course, and send this to the supervising official. This will be accomplished by mounting a module on each of the poles on a kayak course that contains a microprocessor, actuator, sensor and necessary circuitry. This data will be wireless transmitted to a central computer at the judge's station that will graphically display all relevant information. The Slalom Race Organizer will increase the accuracy and credibility of the race results, while reducing the human error currently introduced. This product will be developed to a production-ready level by April, 2007.

## 1.1 Scope

This design specifications document describes the functional requirements that must be met to produce a working system that functions as mentioned in the introduction above. A full set of functional requirements in supplied for this prototype device. By the completion of this project we hope to have a working a system that produces accurate solutions up to industry standards.

The requirements listed here are determined by the design of the Slalom Race Organizer. The requirements are traceable in the design documents

## 1.2 Acronyms

| | |
|---|---|
| CPC | Central Personal Computer |
| CTR | Coordinator of Transmitting/Receiving |
| HALT | Highly Accelerated Lifetime Testing |
| IR | Infrared |
| ITS | ImuTime Systems |
| LED | Light-emitting Diode |
| MSSP | Master Synchronous Serial Port |
| RoHS | Restriction of Hazardous Substances |

## 1.3 Referenced Documents

[1] Proposal for the Slalom Race Organizer. ImuTime Systems.
[2] Canoe/Kayak Slalom Racing Rules. http://www.whitewaterslalom.org
[3] Spectrum Management and Telecommunications. Industry Canada.
[4] Maxstream Inc. www.maxstream.net
[5] Spark Fun Electronics. www.sparkfun.com
[6] Digikey. www.digikey.com
[7] Functional Specifications for the Slalom Race Organizer. ImuTime Systems.
[8] Microchip Website. www.microchip.com

## 1.4 Intended Audience

The design engineers on our team will use this document as a guideline when developing the Slalom Race Organizer.

The project manager will use this document to determine the tasks, and to ensure that the group meets performance and development landmarks. The manager will also use this document to ensure that all design requirements have been met.

Marketing personnel will use this document for promotional material, as well as initial presentations and publications of this solution.

# 2 System Overview

The kayak race organizer has four main components: the timing computer, network host, gate modules, and start/end modules. Figure 2 shows a simplified model of the interactions that occur for a typical slalom race setup. The timing computer is not included in our system but is any laptop or PC with a USB interface. The software for the computer is included, and will provide functions such as 'add/remove gates from the network', as well as providing a real time display to show the time intervals and monitoring modules attributes. These attributes include 'gate has been cleared', 'gate has been touched', etc. The network host is an external component that contains the RF transceiver used to communicate with the other modules in the network. It also includes a USB connector. The start module will be treated differently in that it will signal the start of the race time, and will not include the 'gate touch' attribute. Similarly, the end module will signal the end of the race time.

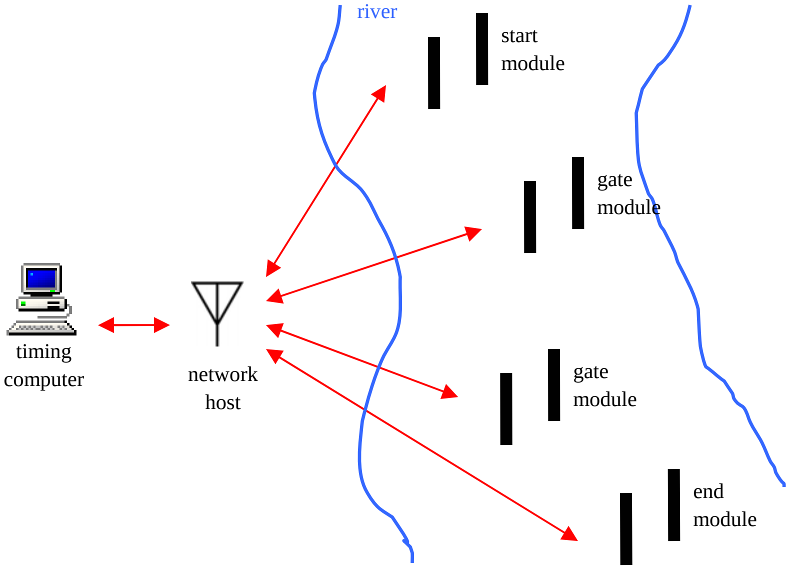


**Figure 2 - Slalom Race Organizer System Overview**

## 3 System Hardware

We have developed two possible sensor layouts for the gate and start/end modules, thus increasing the possible applications and operating environments this project can cover. The main advantages to sensor layout 1 (Figure 3) are that we achieve a higher success rate of racer detection. The main advantage of sensor layout 2 (Figure 4) is that racers are unable to trick the system into detecting a racer clearing the gate when it has not actually happened (i.e. using their paddle to break the line of sight as they pass near the gate).

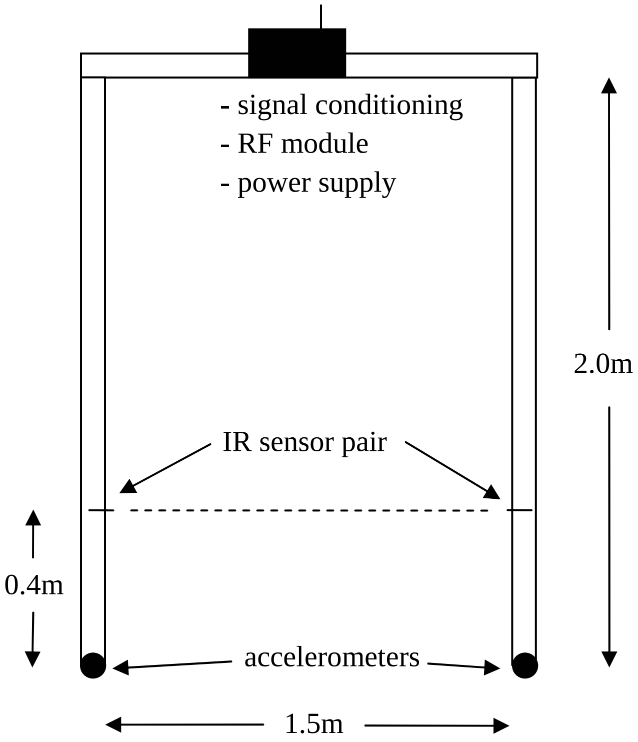


**Figure 3 – Sensor Layout 1**

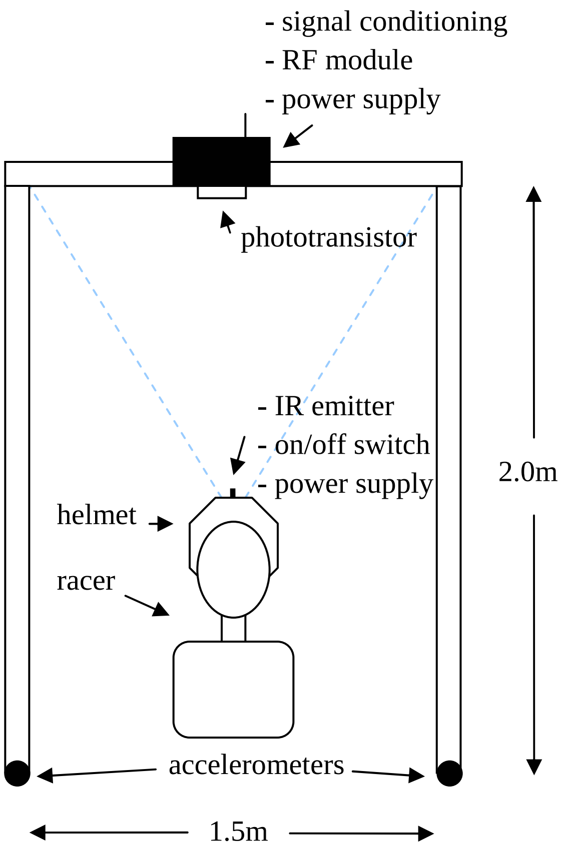


**Figure 4 – Sensor Layout 2**

Figure 5 shows the block diagram of the network host. This module connects directly to the judge's computer and provides access to the wireless network of gates.

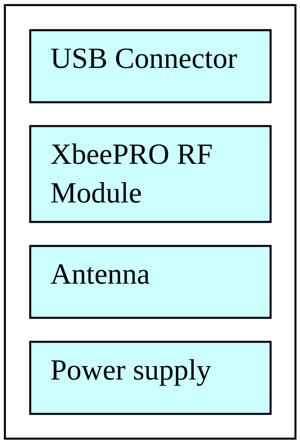


**Figure 5 - Network Host Block Diagram**

Figure 6 shows the block diagram for the start and end modules. These modules are mounted on gates similar to the gate modules, and signal the start and end of the run.

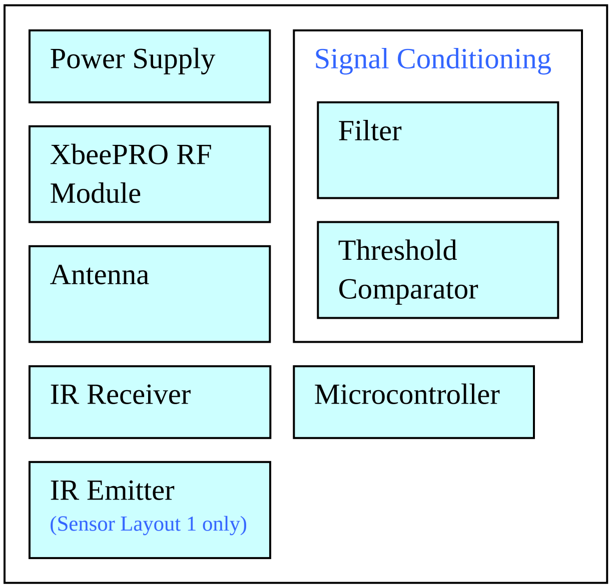


**Figure 6 - Start/End Module Block Diagram**

Figure 7 shows the block diagram for the gate modules. Each gate module is connected to a set of gates and transmits two pieces of information: whether the gate has been cleared, and whether the gate has been touched.

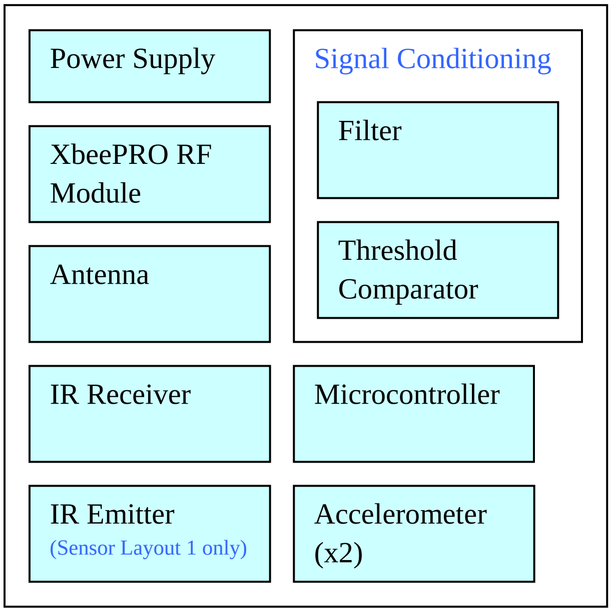


**Figure 7 - Gate Module Block Diagram**

Figure 8 shows the block diagram for the helmet module, should sensor layout 2 prove more appropriate for our applications.

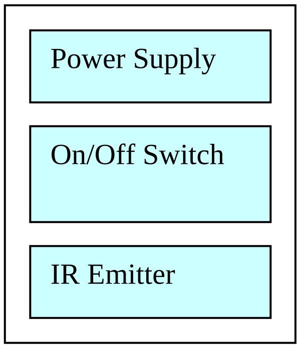


**Figure 8 - Helmet Module (Sensor Layout 2 only)**

NOTE: The IR sensor pair is still undergoing environmental tests. Should the results for low and high power IR LEDs show negative performance characteristics in outdoor conditions, the IR sensor pair will be replaced by ultrasonic sensors. Signal conditioning will not change if this change be made.

NOTE: the power supply will be a Lithium Ion battery, with power characteristics to be determined once the final circuits are designed and their overall power consumption is calculated.

## 3.1 Sensors

To guarantee that our system can operate normally in all possible environmental conditions that whitewater races are held in, we have created two backup sensor configurations. Both backup configurations involve 25kHz ultrasonic transducers. The first backup configuration involves replacing the IR emitter/receiver with an ultrasonic emitter/receiver. The signal conditioning in this setup remains the same as it would for IR sensors (see Appendix A). The second backup configuration involves two ultrasonic sensors configured as rangefinders. The rangefinders are aimed at each other in the same layout as Sensor Layout 1 (Figure 3) and a change in the reported range signals the presence of a racer between the gates.

### 3.1.1 IR Sensors

The IR sensors currently being tested are Lite-On 940 nm IR LED (of varying power capacities) and Lite-On NPN phototransistors. Acceptable signal quality is achieved at distances up to 2m using the mid range power rated LED (250mW). All IR sensors have a wide enough viewing angle that slight or brief misalignments do not interfere with the overall signal quality. Appendix A contains relevant schematics of emitter and receiver setups including signal conditioning.

### 3.1.2 Ultrasonic Sensors (backup configuration)

The ultrasonic sensors that would be implemented should the IR sensors prove inadequate under expected environmental conditions are 25kHz ultrasonic transducers. The use of these sensors requires a 25kHz signal from the microcontroller at the emitter side and a tone decoder at the receiver side. Appendix A contains relevant schematics of emitter and receiver setups including signal conditioning.

### 3.1.3 Accelerometers

The accelerometers used are the Analog Devices ±5.0g Accelerometer mounted on the Spark Fun Electronics breakout board. This accelerometer offers measurements for a range easily large enough to include all normal gate contact during slalom races. It is also very durable, being able to withstand acceleration up to 10,000g. Although the accelerometer offers measurements in three axes, we will only use two for the time being. There is no significant price difference between 2 and 3 axis accelerometers, while the 2 axis models are not available with a breakout board. We decided to use the breakout board due to limited availability of non-surface mount packages for accelerometers. The breakout board also offers bypass capacitors and easy installation in our gate modules. Appendix A contains relevant schematics of the accelerometer setup including signal conditioning.

### 3.1.4 Signal Conditioning

**IR Sensors:** *Emitter* – the emitter is powered by a constant 2V supply that allows for a low load resistance that will not overheat when the emitter is pushed to near its maximum forward rated current.

*Receiver* – the phototransistor operates under a constant supply voltage (connected to the collector) which puts the transistor in saturation mode when the IR signal is detected. The output is filtered using a first order active low pass filter, which allows us to adjust the cutoff frequency to account for water spray and gate flexing. The signal is then sent to a comparator with an adjustable threshold of approximately 0.5V (this threshold depends on the signal strength, which is proportional to the gate width). This will output a clean 5V signal for the microcontroller each time the gate is cleared.

**Ultrasonic Sensors:**

*Emitter* – The emitter is powered using a 25kHz square wave generated by the microcontroller

*Receiver* – The receiver signal is filtered using a first order active low pass filter and sent to a tone decoder for analysis. If the tone decoder output is then inverted using an inverting amplifier and clamped using a clamping circuit. This signal is then sent to the microcontroller.

**Accelerometers:**

The accelerometer signals (X and Y axes) are both filtered using bypass capacitors and then sent to a comparator with a threshold of approximately 1.5V (this will remain variable until an appropriate threshold for normal racer/gate contact is achieved). The comparator outputs a clean 5V signal to the microcontroller each time the accelerometer is triggered.

## 3.2 Microcontroller

The microcontroller and attached development board, PIC16F777-I/P by Microchip and 40 Pin PIC Development board by Sparkfun.com will implement many aspects of our system. These include the following

- **Button**
- **Communication when programming**
- **Power Supply**
- **Additional Hardware**
- **Timer**
- **Analog to Digital Conversion**

## 3.2.1 Development Board

### *3.2.1.1 Button*

The button on each of the development boards will be used as an automatic restart to the microcontroller. This button is a switch that passes current when pressed. The button came with the development board, however a part number is not included in the provided schematic (see Figure 9 below for button portion of schematic for the development board).

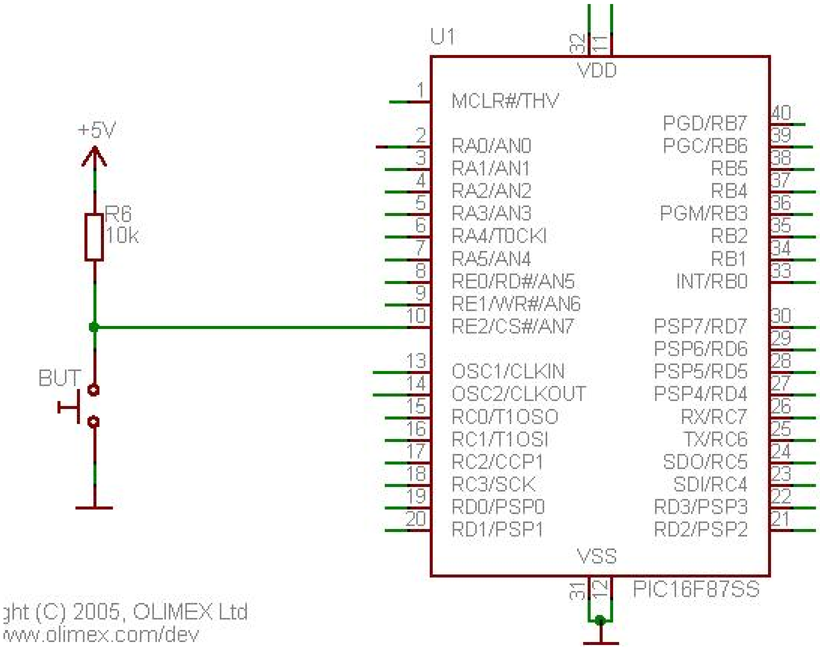


**Figure 9 - Button Schematic**

### *3.2.1.2 Communication when Programming*

The development board also contains circuitry to enable us to communicate with this development board and the microcontroller. This is done using a serial cable (RS232) and a Dual EIA-232 Drivers/Receiver (MAX232 in the communication schematic, Figure 10 below). This system allows us to communicate with the microcontroller though the development board.

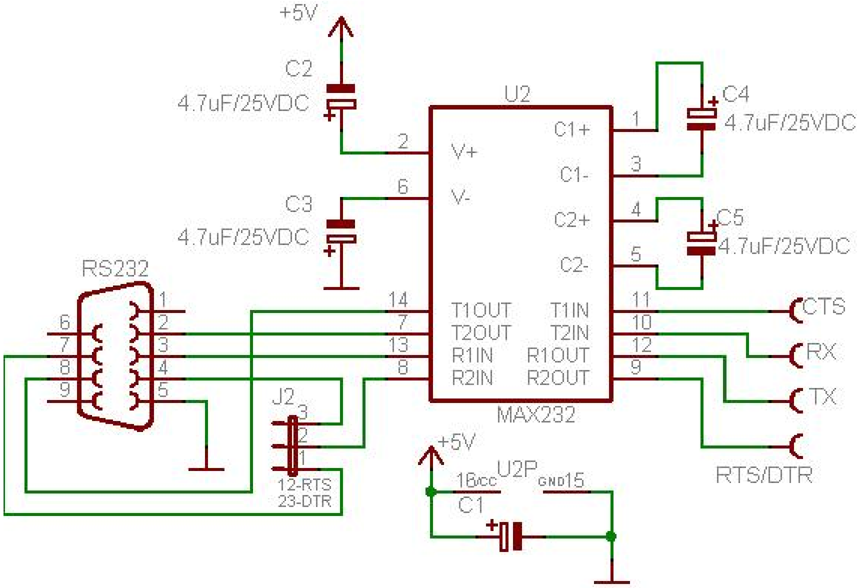


**Figure 10 - Communication Schematic**

### *3.2.1.3 Power Supply*

The PIC18777-I/P microcontroller requires a 5V power supply in order for the chip to function. This will be held constant using a linear regulator, specifically LM317. This linear regulator ensures that the output current leaving it is 1.5A at all times, which makes this output node 5V for the microcontroller. In figure 4 you can see the circuit provided that will accomplish this.

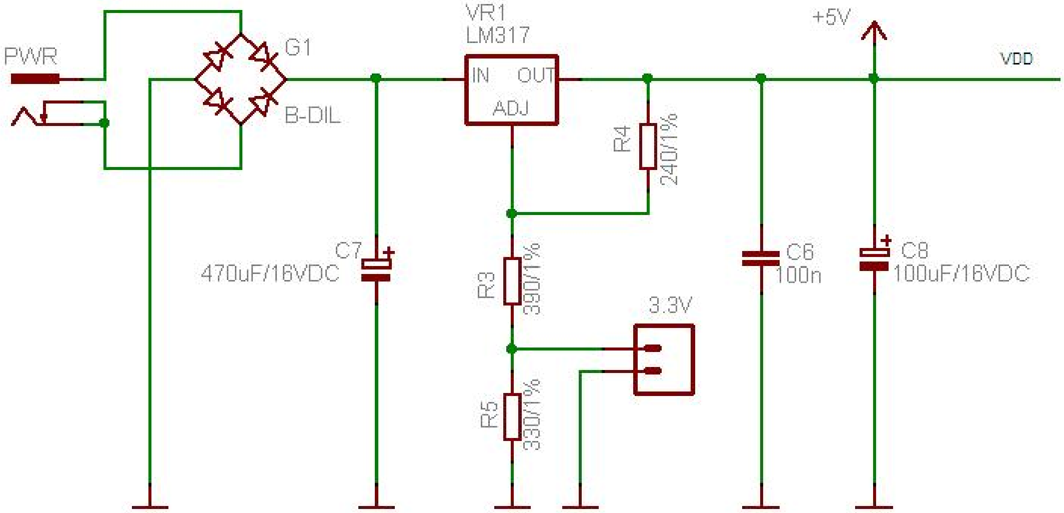


**Figure 11 - Power Schematic**

### *3.2.1.4 Additional Hardware*

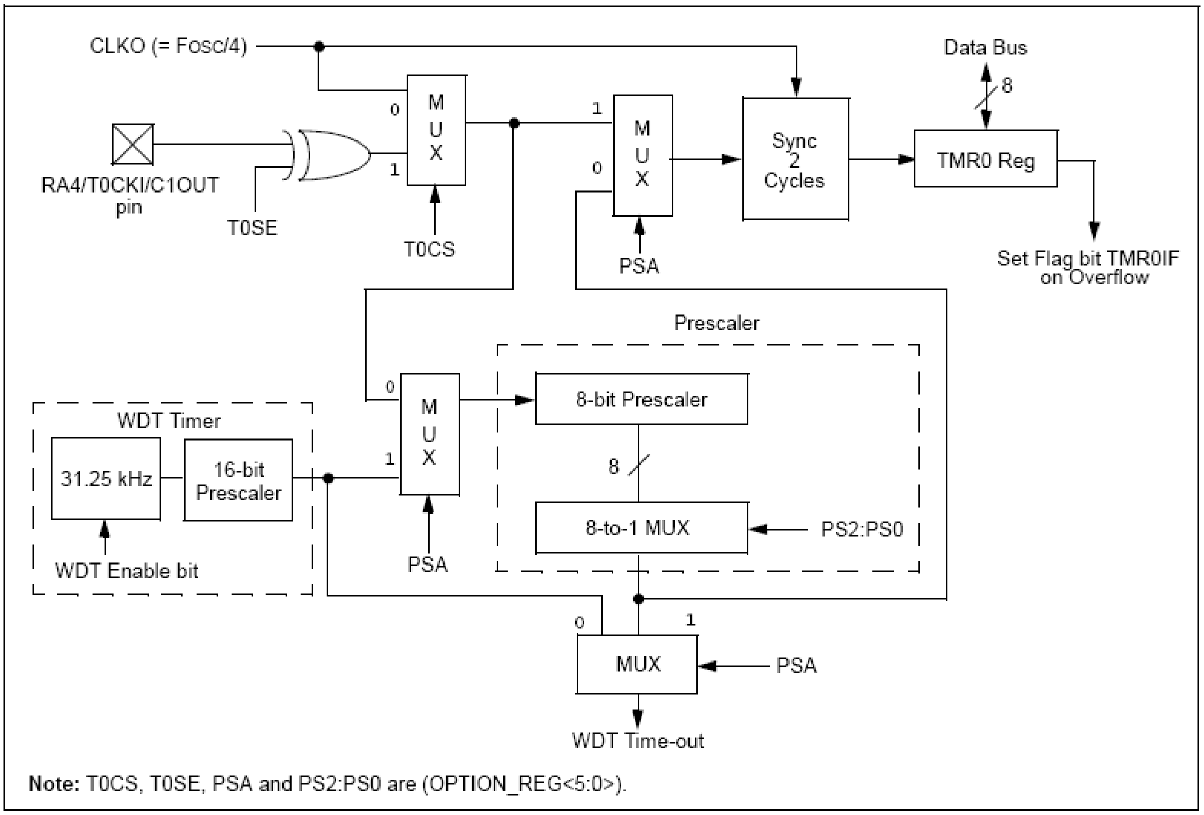
Additional hardware such as sensors, transmitters, filters, etc. will be mounted onto the development board in the additional space available, however will be discussed in their respective sections.

### 3.2.2 Microcontroller

#### *3.2.2.1 Timer*

Using the Timer0 Module available on the PIC18777-I/P we will be able to implement a timer that will reset when the kayaker goes though the first gate and will be stored to another location for transmission of the split time when the user passes through that particular gate. We will be using the internal clock set, and writing ‘0’ to the 8-bit timer register when the user passes through the first gate and reading the clock value when the user passes through that individual gate. The following figure 5 shows how this timer works.

**FIGURE 6-1: BLOCK DIAGRAM OF THE TIMER0/WDT PRESCALER**



**Note:** T0CS, T0SE, PSA and PS2:PS0 are (OPTION_REG<5:0>).

**Figure 12 - Block Diagram of Timer**

#### *3.2.2.2 Analog to Digital Conversion*

The analog to digital converter on the PIC16777 converts an analog input into a corresponding 10-bit number. There are 14 of these A/D converters available on this chip, which we will use for sampling the sensor level, sampling the accelerometer, etc. Also an additional feature available in this chip is that the channels have a programmable acquisition time. The user selects a new channel for conversion and then sets the GO/DONE bit immediately. When the GO/DONE bit is set, the selected channel is sampled for the programmed acquisition time. This reduces the firmware overhead needed to create a sampling time.

## 4 Gate to Computer Bridge

Communication and control of both the setup and configuration of the gates modules will be accomplished through wireless communication using XbeePRO RF modules. These modules operate in several different modes and have a variety of methods for network configuration and addressing.

### The Network

Due to the versatility of the XbeePRO modules, many different networks are possible. We chose our networking to comply with IEEE 802.15.4 standards. The main theme of the entire setup of this timing system is one central computer from which all the gates can be controlled and polled. With this theme in mind a Nonbeacon Network with a Coordinator is used. The Coordinator is equivalent to the theoretical CTR mentioned in the Functional Specification document, and acts as the network host. All the gates RF modules are called End Devices. They only have knowledge of the network host and automatically associate with the CTR upon power up. This allows networks to be dynamic and controlled at the CTR, which is tied to the software. Each gate current is identified by its unique 64bit serial number. This allows a maximum of $2^{64}$ gates. However working with 64bit ID tags may prove cumbersome and overkill for our needs, especially considering the maximum number of gates used in official races is 25. In future shorter 16bit ID tags can be custom created and used.

### The Power

Firstly note that the CTR modules will powered of the PC’s USB port. Since the gate RF modules will use portable power, which ideally will have a very long continuous lifecycle, as both powering and controlling the RF modules are critical. The RF modules operate most efficiently at 3.0V. When a module is transmitting data, is consumes up to 215mA* for approximately 0.5sec**. While receiving data, it consumes up to 55mA for approximately 0.5sec**. This means that the average current consumption will be around 200mA for gates which mostly act as transmitters. The RF modules will enter sleep modes when not transmitting or receiving, which consumes only 50µA. There are lower consumption modes available, but there require a longer wakeup time. Waking from sleep modes will be controlled by the gate modules microcontrollers, and triggered by the sensors detecting an approaching kayaker. Therefore over the course of an entire busy day it can be estimated that the gate RF module will need to transmit at least 50 times and receive 5 times. When not doing either of these, it RF module will be sleep mode. If an entire day is estimated as 8hours the total power consumption by a single RF module is only approximately 0.0019Amp hours. Therefore with careful monitoring these RF modules can be made to consume very little power.

*Notes:*

*This is current consumption is based on using the highest output setting for transmission. This can be reduced pending the results of fidelity tests in harsh environments.

**This time is merely an overestimate. Pending the results of testing, this number should be quite smaller, on the order of 10msec.

### The Modes

There are several modes in which the RF modules can operate. The modules must first be pre-programmed to be configured to a specific mode of network communication. This is done on development programming boards prior to installation of the RF modules.

**Transparent Operation** is simply a data transfer mode. Only raw data can be transfer between the CTR and End Devices. Data in UART serial format is inputted to a transmitting module from its microcontroller. This UART serial data is then received at the destination and input to the recipient module's microcontroller. Since the RF modules are pre-programmed and configured, the user has little to no control of adjusting parameters of the network in the Transparent operation. Modules can only be automatically added or subtracted from the network.

**API Operation** is a highly expanded format of communication. Under this mode of operation, data takes on a specific frame format. Using this frame format the host application (in this case the CTR) can not only transmit and receive data, but can interact with specific modules networking capabilities and low level firmware settings. Thus more advance and adaptive networking is possible and can be controlled by a user. This mode of operation is far more complex and may prove to be too complicated and give too much power to a user. Thus implementing this mode of transmission will be used in future models of the ITS Slalom Race Organizer.

# *5 Firmware Design*

Our firmware will be developed using the MPLab Integrated Development Environment which is an integrated toolset for embedded applications.

## 5.1 MPLab

MPLab runs a 32-bit application on Windows and can be used for application development and debugging. It also provides a graphical user interface for easy manipulation. MPLab is used for 8/16-bit microcontrollers. MPLab contains:

- An interface to debugging tools
- A full-featured editor with color coded context
- High-level source code debugging
- One touch assemble (or complile) and download to PIC micro emulator and simulator tools
- Degub using:
    - Source files (assembly or C)
    - Mixed assembly and C

This user interface can be seen in figure 10 below:

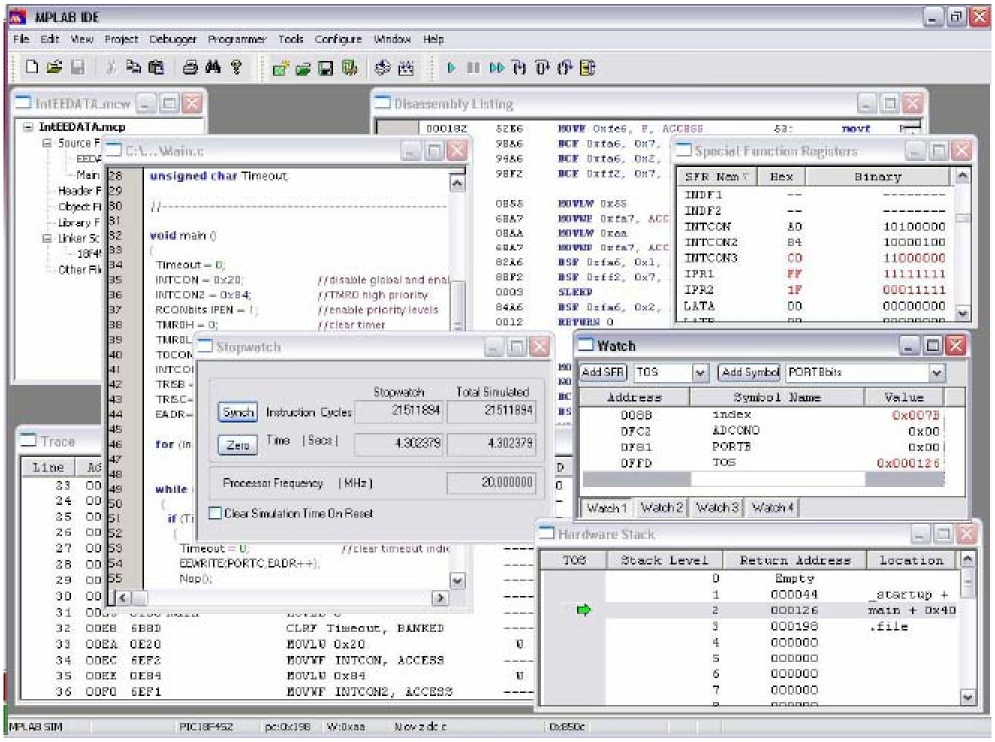


Figure 13 - MPLab Interface

## 5.2 Core Processor Support

The processor in our microcontroller that will implement out code is a CMOS

## 5.3 Board Support

### 5.3.1 Available Space

| | |
|---|---|
| Flash Program Memory (14-bit words) | 8K |
| Data Memory (bytes) | 386 |
| Interrupts | 17 |

### 5.3.2 Instruction Set

For the PIC16777 there are 35 instruction sets available for creating our firmware. These will be a sufficient amount to do all necessary tasks. There are three basic categories:

- Byte-oriented operations: The file register designator specifies which file register is to be used by the instruction set. The destination designator specifies where the result of the operation is to be placed.

- Bit-oriented operations: The ‘b’ represents a bit fild designator which selects the bit affected by the operation, while ‘f’ represents the address of the file in which the bit is located.

- Literal and control: The ‘k’ represents an eight or eleven-bit constant or literal value.

These can be graphically viewed in the following figure 11:

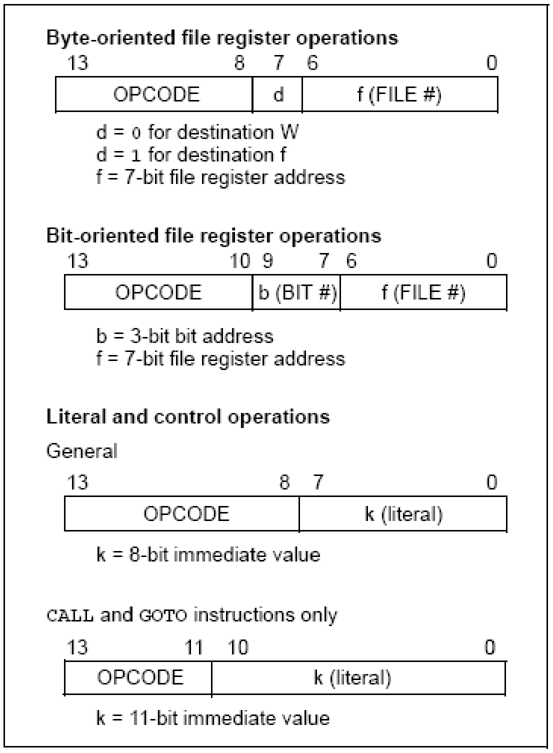


**Figure 14 - Instruction Set**

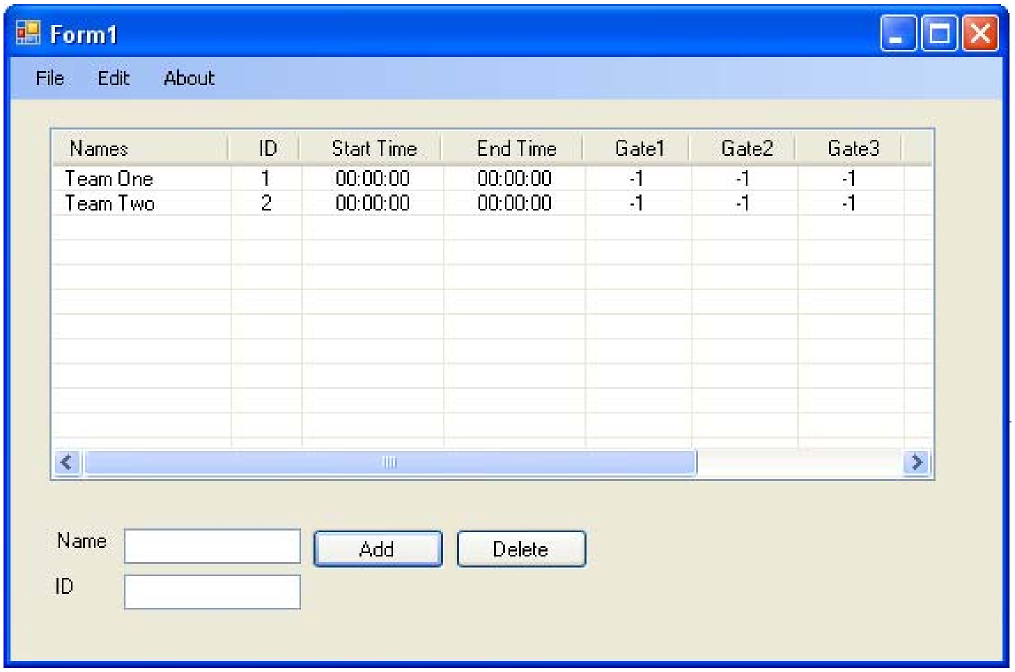
## 5.4 Communication

This microcontroller can be communicated with using the Master Synchronous Serial Port (MMSP) with 3-wire SPI and I^2C (Master and Slave modes).

**MSSP**: The MSSP module is a serial interface which is used for communicating with other peripheral or microcontroller devices. These could be EEPROMS, shift registers, display drivers, A/D converters, etc.

# *6 Application Software Design*

The software application allows the user at CPC to control the race organizer through a graphical user interface (GUI, Figure 15). The GUI aims to be as simple as possible, while retaining sufficient functions to run a race smoothly. The user can add or remove kayak teams and gates from the race easily. The GUI will keep the status of the race updated and presented to the user.



| Names | ID | Start Time | End Time | Gate1 | Gate2 | Gate3 |
|---|---|---|---|---|---|---|
| Team One | 1 | 00:00:00 | 00:00:00 | -1 | -1 | -1 |
| Team Two | 2 | 00:00:00 | 00:00:00 | -1 | -1 | -1 |

**Figure 15 – The Timing Computer GUI**

## Functions

a) Add Name/ID: The user can add a team by simply fills out the Name and ID textboxes, and presses add. The ID is an integer starting from 1. It gives each team an identity for the gate sensors to distinguish from each other.

b) Delete Name/ID: A kayak team is removed from list by selecting the team and press delete. Selection can be done by clicking anywhere on the row. The team entry will be removed from the table.

c) Timing: Start time is determined when the sensor at starting point detects the kayaker passing. Similarly, end time is determined when the sensor at finish point detects the kayaker passing. If timing is performed manually, user can directly input the times onto the table.

d) Add/Remove Gate: User can add gate to or remove gate from the system by clicking on the corresponding column of the table and press add/remove gate button. Button is not shown in KayakGUI diagram.

e) Gate Monitoring: When CTR receives a transmission from an end device, an event is raised and the software reads from the hardware register. The address of sender and the data are processed. The table automatically updates the changes to keep user informed.

f) New File: Creates a blank table. All settings are back to default. User will be prompted to save existing records before data is erased.

g) Save File: Saves the table in .txt format. The file can be easily loaded to other applications, such as Microsoft Excel.

h) Open File: Loads the data saved in a txt file to the table.

### Program Structure

The programming language used is Visual Basic due to simplicity of creating windows application. A custom class Racer is constructed to store all information of a kayaker, including name, ID, and time penalties added for each gate. The class provides functions for the GUI to set or get data.

The table displays all information related to the race. Columns are, in order from left to right, name, ID, start time, end time, gate1, gate2, and up to gate 25, and lastly the total time. Name is a string inputted by user. ID is the identity assigned to the kayaker to be recognized by sensors. Start time and end time record the time kayaker spends during the course. Gates 1 to 25 show whether the kayaker successfully passes each gate. The default values assigned to gate columns are -1(inactive). If a gate is added, its value becomes 50, the highest time penalty in seconds. If the gate is flawlessly negotiated (passed without touching gate poles), its value becomes 0. If it is negotiated with a contact, its value becomes 2. At the end of race, the total time is calculated by the formula:

end time – start time + time penalties added

Each row represents a kayaker or kayak team. A row is added and deleted by the Add Name/ID and Delete Name/ID functions as described above.

# *7 Test Plan*

Each subsystem will undergo several functionality and robustness tests. Similarly once all major integrations have completed, overall product testing will begin. An important aspect of the product is that it will operate under rigorous outdoor conditions not found in labs. A HALT has been designed to test the product under temperatures beyond what is expected in the field operation. This is done with the help of Xantrex Inc, using Qualmark 3.0VB thermal chambers. The results of the HALT will allow for good estimates of possible field failures, expected failures rates, and expected lifetimes.

## 7.1 Hardware Test Plan

### 7.1.1 Sensor Test Plan: Environmental Conditions

All sensors are to be tested in the following environmental conditions: sunny/dim outdoor light, dry/rainy weather, cold/warm temperatures. We will also test sensors over dry land, fresh water, and salt water to ensure that optical reflection does not interfere with our success rate of racer detection. Many of the performance characteristics of our sensors will depend on the weatherproofing we can provide for the various modules. For this reason, we will do some environmental tests without our prototype casing, and further tests with the prototype casing. Cases will be shock resistant plastic that offer easy access without allowing for water intrusion once sealed.

### 7.1.2 Sensor Test Plan: Performance Analysis

A performance analysis will be completed to ensure that we are achieving a near perfect rate of success in terms of the detection of racers. This test will focus on the signal quality, which may be an issue due to the 2.0m distance between the accelerometers and the signal conditioning circuitry. Noise may be induced on this long wire, and could simulate a gate touch which has not actually occurred.

#### 7.1.3 Power Supply Test Plan

Since batteries are to be used to power each module, a detailed test is required to determine the power consumption and supply requirements. This in turn effect the battery life, a goal which we have set to be 6 hours.

### 7.2 Wireless Test Plan

It has been writing in papers that wireless transceivers can be subjected to a reduction in fidelity when used over water surfaces. These tests can be complete initially over a pool. Further range tests will be completed across the Alex Fraser River. Since the RF modules will only be maximally 10ft off the ground, line-of-sight range is limited. Since RF communication is best under perfect line of sight, fidelity will be testing will obstacles obstructing line of sight, and ranges of up to 500m. Once the Wireless System has been integrated with the microcontroller and software, fidelity and range testing can begin. Testing of the automated network subroutines can be accomplished as well.

### 7.3 Software Test Plan

The functionality of Kayak Race GUI will be tested with the software debugger, provided by Visual Basic 2005. The test will ensure that all forms, buttons, menu items work as expected, and the functions perform smoothly, as described in Application Software Design section. Another test will be performed to ensure that software correctly processes the incoming data from the receiver, and issues commands to distant gates.

### 7.4 Firmware Test Plan

The board support testing is straightforward. MPLab Integrated Development Environment is an integrated toolset for the development of embedded applications for our specific microcontroller. The MPLab Debugger performs an initial hardware check each time when connecting to the board or writing to the microcontroller, and this will ensure that the board support portion of the firmware is functioning. This MPLab Debugger will also be used for debugging development firmware during the process of creating it.

## *8 Conclusion*

The focus of our design is to provide the most robust and reliable system possible. While we keep in mind that the overall cost of this system should not climb too high, some components have been chosen based on quality as oppose to price reasoning. Our goal is to achieve a 99% success rate in terms of the detection of racers between the gates, and racer contact with the gates. If the system misses a racer that proceeds between or touches the gates, our failure rate will increase. Additionally, should the system detect a racer or racer contact when it has not actually occurred, our failure rate will again increase. Due to the unpredictable and sometimes extreme environmental conditions in which this system will be put to use, this may be a significant problem. For this reason we have focused on ensuring we use the most reliable parts and circuits available.

# *Appendix A – Schematics*

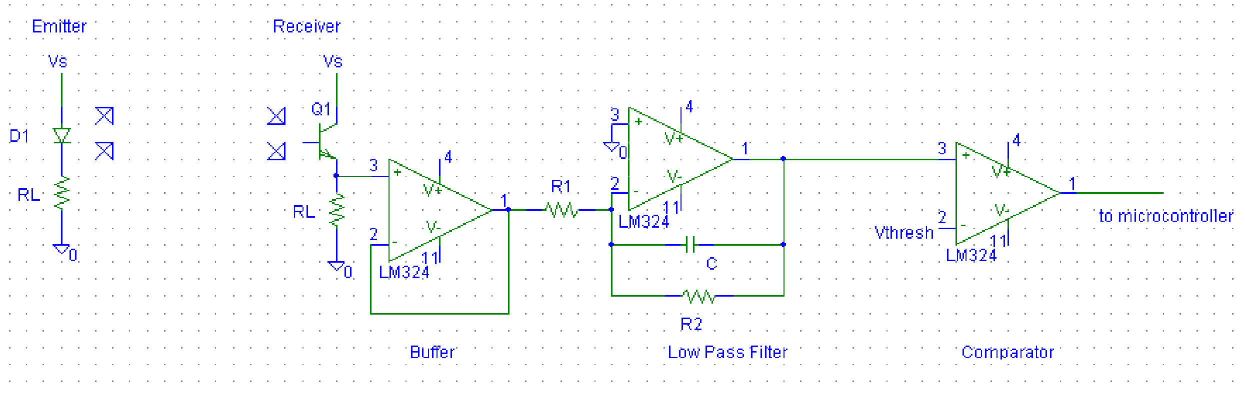


**Figure 16 – IR Emitter and Receiver Configuration**

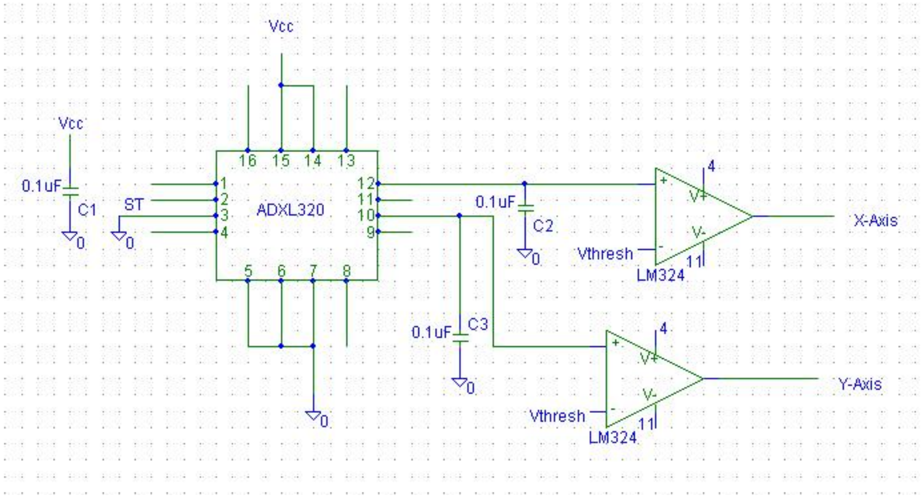


**Figure 17 – Accelerometer Configuration**

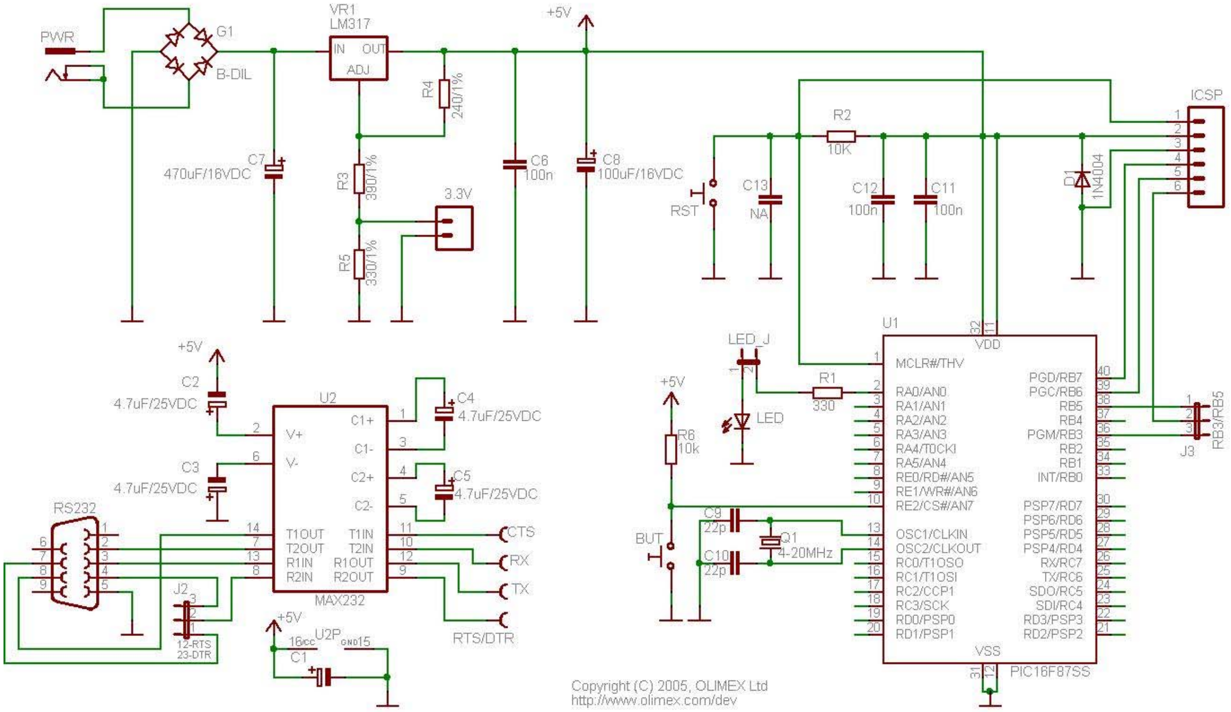


**Figure 18 - Development Board Schematic**