**EE 478 Lab 2**

**Designing a High Reliability Microprocessor Based Remote Surgery System**

University of Washington – Department of Electrical Engineering

Ryan McDaniel Alyanna Castillo Patrick Ma

TABLE OF CONTENTS

[**I.** **ABSTRACT** 6](#_Toc387019242)

[**II.** **INTRODUCTION** 6](#_Toc387019243)

[**III.** **SYSTEM REQUIREMENTS** 6](#_Toc387019244)

[**a.** **Use Cases** 6](#_Toc387019245)

[**b.** **Requirements** 6](#_Toc387019246)

[Specification 6](#_Toc387019247)

[System Description 6](#_Toc387019248)

[Specification of External Environment 6](#_Toc387019249)

[System Input and Output Specification 6](#_Toc387019250)

[User Interface 7](#_Toc387019251)

[System Functional Specification 7](#_Toc387019252)

[Operating Specifications 7](#_Toc387019253)

[Reliability and Safety Specification 8](#_Toc387019254)

[**IV.** **DESIGN SPECIFICATION** 8](#_Toc387019255)

[Specification 8](#_Toc387019256)

[System Description 8](#_Toc387019257)

[Specification of External Environment 8](#_Toc387019258)

[System Input and Output Specification 8](#_Toc387019259)

[User Interface 9](#_Toc387019260)

[System Functional Specification 9](#_Toc387019261)

[Operating Specifications 10](#_Toc387019262)

[Reliability and Safety Specification 10](#_Toc387019263)

[**V.** **DESIGN PROCEDURE** 11](#_Toc387019264)

[**VI.** **SYSTEM DESCRIPTION** 11](#_Toc387019265)

[**VII.** **HARDWARE IMPLEMENTATION** 11](#_Toc387019266)

[a. Local Node 11](#_Toc387019267)

[**SRAM Interface** 11](#_Toc387019268)

[**External Clock** 12](#_Toc387019269)

[**RS232 Communications** 12](#_Toc387019270)

[**I2C Communication** 12](#_Toc387019271)

[b. Remote Node 12](#_Toc387019272)

[**SRAM Interface** 13](#_Toc387019273)

[**ADC Measurements** 13](#_Toc387019274)

[**PWM Output** 13](#_Toc387019275)

[**I2C Communication** 13](#_Toc387019276)

[**VIII.** **SOFTWARE IMPLEMENTATION** 13](#_Toc387019277)

[a. Local Node 13](#_Toc387019278)

[SRAM Control 13](#_Toc387019279)

[RS-232 Communication 13](#_Toc387019280)

[I2C Communication 14](#_Toc387019281)

[The hardware configuration created two separate I2C communication channels, greatly simplifying the software implementation. Just like the hardware setup, the software was divided into an outgoing and incoming channel. 14](#_Toc387019282)

[*Outgoing –Control Channel* 14](#_Toc387019283)

[The outgoing channel sent the speed defined by the user as an 8-bit word. The first Master Synchronous Serial Port (MSSP1) is configured as a master with a baud rate of 400kHz and no slewing. All control of the MSSP1 is done using a simple polling scheme with no interrupts. After waiting for an idle connection, the local node begins communication according to the I2C protocol. The slave address (set to 0x00 for convenience) and a write instruction are sent followed by the speed. The local node then closes the connection with a STOP signal. 14](#_Toc387019284)

[*Incoming – Monitor Channel* 14](#_Toc387019285)

[The incoming channel utilizes an interrupt-based scheme because the arrival of incoming data is arbitrary. Incoming data is received through Master Synchronous Serial Port 2 (MSSP2), which is configured as a slave device. When an address is sent on the monitor channel, the MSSP2 module compares it to the slave address (0x00 chosen again for convenience). If the address matches, an interrupt flag is asserted and the interrupt handler runs. The interrupt handler takes one of three actions depending on the state of the communication. If the data is an address, the interrupt handler clears the address from the MSSP2 data buffer, deasserts the interrupt flag and exits. If the data is actual information, the interrupt handler checks which bye of the sequence it is expecting. If it is the first byte, the MSSP2 data buffer contents correspond to the motor speed; otherwise the ISR assumes the data is an error state. For each option, the data is saved to the appropriate global variable, the byte counter adjusted appropriately, the interrupt flag is deasserted, and the display flag is set. 14](#_Toc387019286)

[b. Remote Node 14](#_Toc387019287)

[**SRAM Control** 14](#_Toc387019288)

[**ADC voltage measurement** 14](#_Toc387019289)

[The analog to digital module measures the actual voltage across the motor. The setup for the ADC involves configuring the ADC clock source, input channel, and acquisition time. These parameters are defined as follows. 14](#_Toc387019290)

[ ADC clock source is 1/64th the primary oscillator frequency 14](#_Toc387019291)

[ ADC acquisition time is 12 time periods 14](#_Toc387019292)

[ Input channel is channel 14 (Pin 10) 14](#_Toc387019293)

[ ADC interrupt is disabled 14](#_Toc387019294)

[ ADC result is right justified 14](#_Toc387019295)

[ ADC voltage references are the internal voltage references (5V and GND) 15](#_Toc387019296)

[Since no interrupts are used, the microcontroller polls the ADC status flag until a computation is complete. The result is read into a temporary register and converted to a zero to 200 scale by multiplying the result by 50, bit shifting to the right by 8 and adding 1. To reduce the unnecessary memory writes, and because the precision of the ADC is greater than the noise from the input signal, the current motor speed is only updated if the new measurement is 1% greater or less than the value for the actual speed held in memory. If this is the case, a flag is set to send this value to the local node. 15](#_Toc387019297)

[**PWM output** 15](#_Toc387019298)

[The pulse width modulation (PWM) output has only a minimal software implementation –the majority of the work is handled by the hardware module. Within software, the PWM has two primary components: setup and adjustment of the duty cycle. Setup occurs once each time the microcontroller is reset but adjustment occurs continuously while the PIC18 is running. When PWM setup occurs, the following registers are set: 15](#_Toc387019299)

[ PR4 is set for 20kHz frequency 15](#_Toc387019300)

[ T4CON is set for a 1:1 pre-scaling factor 15](#_Toc387019301)

[ CCP4CON is set to select PWM mode 15](#_Toc387019302)

[ CCP4CON and ccPR4L are set to an initial 50% duty cycle 15](#_Toc387019303)

[ The Timer4 interrupt flag is cleared 15](#_Toc387019304)

[ The GPIO pin B4 (pin 21) is configured as an output 15](#_Toc387019305)

[To adjust the PWM duty cycle, the PIC18 calls the compiler function SetDCPWM4(). To pass an appropriately scaled value to the PWM module, the global control variable is multiplied by 5, giving a range from 0 to 1023. The PWM duty cycle is adjusted each time the main program loop is executed. 15](#_Toc387019306)

[**I2C Communication** 15](#_Toc387019307)

[Much of the remote node I2C software operates using the same code as the local node. For a general description of operation refer to the local node section. Specifics for the remote node are provided below. 15](#_Toc387019308)

[*Outgoing –Monitor Channel* 15](#_Toc387019309)

[The outgoing channel sent two bytes of data to the local node. The first byte encodes the actual motor speed following the coding convention used throughout the system. The second byte contains the error state encoded as an integer. The MSSP1 module is configured just like the local node using the same polling scheme. 15](#_Toc387019310)

[*Incoming – Control Channel* 15](#_Toc387019311)

[The incoming channel utilizes an interrupt-based scheme in the same manner as the local node. However, since the remote only receives a single byte containing the set speed, the interrupt service routine merely check whether the MSSP data buffer contains an address or actual data: actual data is stored to the set speed global variable; an address is simply cleared from the buffer and discarded. 16](#_Toc387019312)

[**IX.** **TESTING** 16](#_Toc387019313)

[a. Test Plan 16](#_Toc387019314)

[b. Test Specification 16](#_Toc387019315)

[**c.** Test Cases 16](#_Toc387019316)

[**X.** **RESULTS** 16](#_Toc387019317)

[a. Presentation 16](#_Toc387019318)

[b. Discussion 16](#_Toc387019319)

[c. Analysis 16](#_Toc387019320)

[**XI.** **ANALYSIS OF ERRORS** 16](#_Toc387019321)

[**XII.** **WHY PROJECT DIDN’T WORK** 16](#_Toc387019322)

[**XIII.** **SUMMARY** 16](#_Toc387019323)

[**XIV.** **CONCLUSION** 16](#_Toc387019324)

[**XV.** **APPENDIX** 16](#_Toc387019325)

[a. Charts and Tables 16](#_Toc387019326)

[b. Graphs 16](#_Toc387019327)

[c. Code 16](#_Toc387019328)

1. **ABSTRACT**

This lab provides experience in using the MPLabX development environment for programming the PIC18F25K22 microchip

1. **INTRODUCTION**
2. **SYSTEM REQUIREMENTS**
   1. **Use Cases**
   2. **Requirements**

## Specification

### System Description

This specification is a draft for the requirements in embedded circuitry for a microprocessor based remote surgery system. The system is a prototype and proof of concept for a larger system. The circuitry must be able to communicate between a local node, connected to a computer, and a remote node, connected to a surgery robot motor. A remote node must be able to maintain motor speed, and the local node must be able to communicate with a PC to send information and receive commands. The product is meant to be developed in two phases. The first phase would be to implement the local and remote communication networks, and the user interface. The second phase will include adding support for the remote feedback channel through which warnings, data, and alarms are sent.

### Specification of External Environment

The unit is meant to be used in the medical environment. Concerns will be blood and corrosion, water, gasses, and sterilization. The system may have to operate in a frequency that does interfere with other medical devices, or produce any gasses that would contaminate a sterile environment. Finally, the system should not have a negative effect on the power draw of the environment.

### System Input and Output Specification

#### System Inputs

User commands from PC terminal output.

Set motor speed

Increment motor speed

Decrement motor speed

Start motor

Stop motor

Motor voltage level

Power On/Off

#### System Outputs

Voltage control to motor

Display to PC Terminal

Warnings

Alarms

Current motor speed

### User Interface

The user interface will be a command line presented through a terminal window. The user can enter the following commands:

Set speed:

Sets the speed of the motor to a specified value

Increment speed:

Increments the current motor speed by 0.5%.

Decrement speed:

Decrements the current motor speed by 0.5%.

Start:

Start the motor.

Stop:

Stop the motor.

In addition to the command line interface, the system will have an ON/OFF button for power.

### System Functional Specification

The system takes and executes user commands from the PC terminal with the intent of remotely operating motor speed through a network. These commands are turning the motor on/off, setting the motor speed, and increasing and decreasing the motor speed.

The system has two main parts – the local and remote node. Both nodes communicate with each other and have a separate memory to store data. The user interface (PC terminal) is connected to the local node, which processes user input. The commands are then sent to the remote node to control the motor, which in turn measures the current status and sends it back through the local node to the PC.

### Operating Specifications

The system shall operate in a sterile medical environment.

Temperature Range 20-23 C

Humidity Rang is 20-60%

Power 5V

### Reliability and Safety Specification

The robot surgery system shall comply with the following safety standards

* Meet government regulations regarding medical devices as outlined by the FDA’s CFR (Code of Federal Regulations)
* Continue to safely function in the absence of power (blackout, etc.)
* All outer parts of must be sterilized before use

1. **DESIGN SPECIFICATION**

## Specification

### System Description

This specification is a draft for the embedded circuitry for a microprocessor based remote surgery system. The system is a prototype and proof of concept for a larger system. The circuitry must be able to communicate between a local node, connected to a computer, and a remote node, connected to the surgery robot motor, through an I2C connection. A remote node must be able to maintain motor speed by controlling the voltage applied to the motor. The local node must be able to communicate with a PC through an RS232 serial connection so that it can receive commands and send data to be displayed. The product is meant to be developed in two phases. The first phase would be to implement the local and remote communication networks, and the user interface. The second phase will include adding support for the remote feedback channel through which warnings, data, and alarms are sent.

### Specification of External Environment

The unit is meant to be used in the medical environment. Concerns will be blood and corrosion, water, gasses, and sterilization. The system may have to operate in a frequency that does interfere with other medical devices, or produce any gasses that would contaminate a sterile environment. Finally, the system should not have a negative effect on the power draw of the environment.

### System Input and Output Specification

#### System Inputs

User commands from PC terminal output.

Set motor speed

Increment motor speed

Decrement motor speed

Motor voltage level

Power On/Off

#### System Outputs

PWM voltage signal to motor

Display to PC Terminal

Warnings

Alarms

Current motor speed

### User Interface

The user interface will be a command line presented through a terminal window. The user can enter the following commands:

Set speed:

Sets the speed of the motor to the specified value.

Increment speed:

Increments the current motor speed by 0.5%.

Decrement speed:

Decrements the current motor speed by 0.5%.

In addition to the command line interface, the system will have an ON/OFF button for power.

### System Functional Specification

The system takes and executes user commands from the PC terminal with the intent of remotely operating motor speed through a network. These commands are turning the motor on/off, setting the motor speed, and increasing and decreasing the motor speed in 0.5% increments.

The system has two main parts – the local and remote node. Both nodes communicate with each other through an I2C connection and have a separate SRAM to store data. The user interface (PC terminal) is connected to the local node.

User commands are processed in the local node and sent to the remote node. User input is received by the local node through an RS232 connection. Before being sent to the remote node, the local node checks that the user input is a valid command.

Valid commands are received by remote node through an I2C connection and executed. Depending on the command, the motor speed will be adjusted appropriately. The remote node also monitors the error state of the system and measures the current motor speed. The error state is calculated if any of the motor speeds go beyond the following ranges:

±5.0% - Level 0 - severe

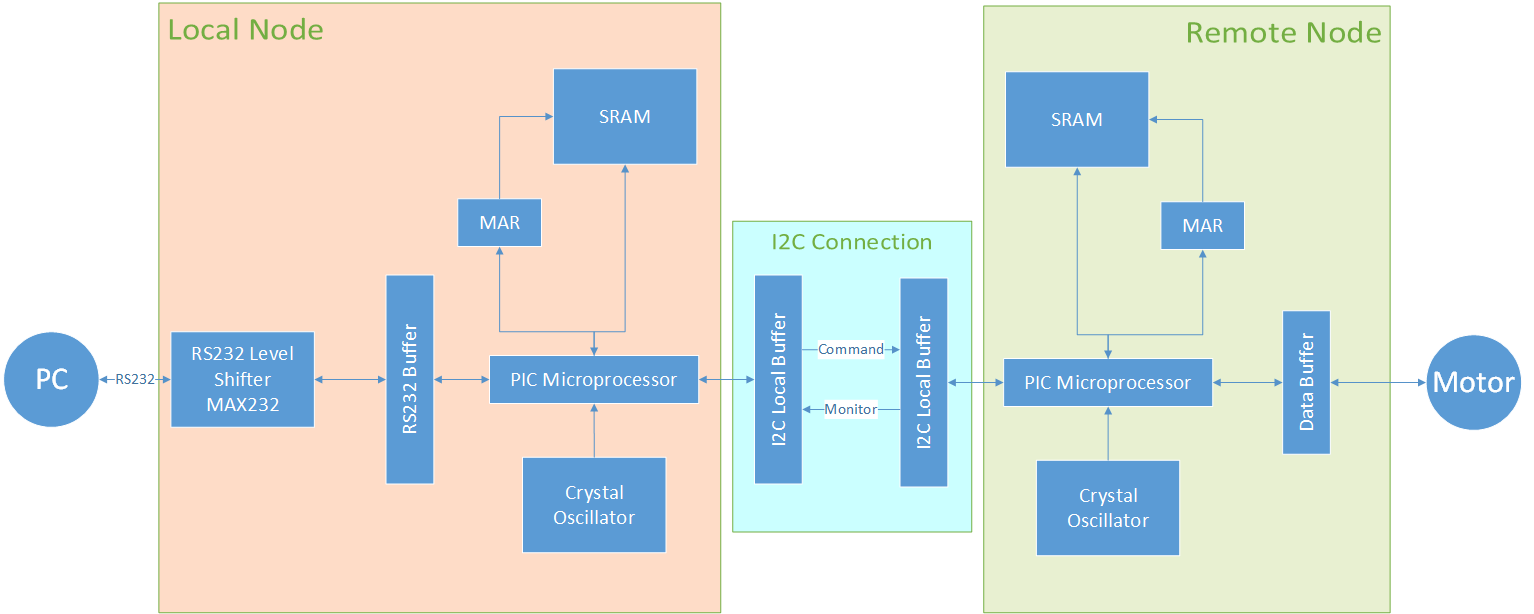
±2.0% - Level 1 - moderate

±1.0% - Level 2 – of concern

The error state is sent back to the local node through the same I2C connection.

The local node then receives data on the current motor speed and error state from the remote node – this data is stored in the local node’s SRAM. The data is then read from the SRAM and outputted through an RS232 connection to the PC terminal, making it viewable to the user.

The system comprises of following block diagram.

****

### Operating Specifications

The system shall operate in a sterile medical environment.

Temperature Range 19-23 C

Humidity Range is 20-60%

Power 5V

### Reliability and Safety Specification

The robot surgery system shall comply with the following safety standards

* Meet government regulations regarding medical devices as outlined by the FDA’s CFR (Code of Federal Regulations)
* Continue to safely function in the absence of power (blackout, etc.)
* All outer parts of must be sterilized before use
* Properly alert personnel of any error states and take correct course of action

1. **DESIGN PROCEDURE**
2. **SYSTEM DESCRIPTION**
3. **HARDWARE IMPLEMENTATION**
   1. Local Node

The Local Node consisted of the following hardware:

|  |  |  |  |
| --- | --- | --- | --- |
| Part Number | Description | Quantity | Price |
| PIC18F25K22 | 28-pin microcontroller | 1 |  |
| OSC-20MHz | Crystal oscillator | 1 |  |
| CY7C128A | SRAM | 1 |  |
| GAL22V10D | Programmable logic chip | 1 |  |
| MAX232 | RS232 level shifter | 1 |  |
| 1 uF Capacitors | ---------------------------- | 5 |  |

The PIC microcontroller was the central unit of the local node. Every other part in the node is connected to and controlled by the PIC.

**SRAM Interface**

The following diagram shows the pin assignments for the SRAM interface.

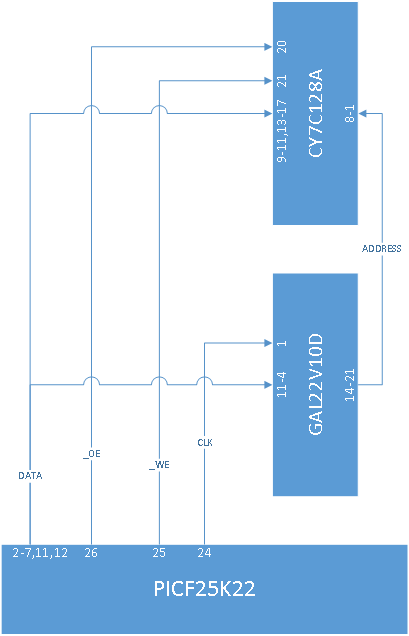


Figure 1 - Hardware block diagram for the SRAM interface

Pins 2-7, 11, and 12 of the PIC were connected to pins 11-4 of the GAL chip, and to the I/O ports of the SRAM on pins 9-11, and 13-17. Pin 2 on the PIC was the LSB of the data or address, while pin 12 was the MSB. Pins 14-21 on the GAL chip were connected to the address lines of the SRAM on pins 8-1. There are tristate drivers in both the SRAM and the PIC that control whether data is going into the SRAM and out of the PIC, or if data is coming from the PIC and going to the GAL or SRAM. On the PIC, these tristates are controlled by software and there is one for each I/O pin on the PIC. On the SRAM, the tristates are controlled by the status of the Output Enable input signal.

The SRAM and GAL chip are controlled by three signals coming out of the PIC on pins 24-26. Pin 24 was the clock signal for the GAL chip, which connected to the GAL’s pin 1. Pin 25 was the write enable signal for the SRAM, which was sent to the SRAM’s pin 21. Pin 26 of the PIC was the SRAM’s output enable signal, which connected to the SRAM’s pin 20. The last two signals for the SRAM and GAL is the GAL’s reset signal, which was connected to ground, and the SRAM’s chip enable signal, which was also connected to ground. The status of these signals was controlled by software on the PIC.

**External Clock**

The 20MHz crystal oscillator was the primary clock for the PIC microcontroller. The output pin of the clock was sent to pin 6 of the microcontroller, and the pin was configured with software to allow the clock to be used as the system clock.

**RS232 Communications**

Communication to a PC via a serial communications port was done using the PIC’s USART interface and a MAX232 level shifter. Pin 25 of the PIC was the RS232 transmit pin, and was connected to pin 11 on the MAX232. Pin 26 of the PIC was the RS232 receive pin, and was connected to pin 12 on the MAX232. The receive pin of the RS232 cable was connected to pin 14 of the MAX232, and the transmit pin was connected to pin 13 of the RS232. The MAX had capacitors connected between pins 1 and 3, 4 and 5, 16 and ground, 2 and ground, and ground and pin 6.

**I2C Communication**

An I2C connection was established between the local and remote nodes by connecting pins 11 and 12 on the local PIC to pins 19 and 20 on the remote PIC. The local node is the master of the control channel, where the local node is sending information about the current motor speed to the remote node. The I2C protocol is controlled by software.

* 1. Remote Node

The Remote Node consisted of the following hardware:

|  |  |  |  |
| --- | --- | --- | --- |
| Part Number | Description | Quantity | Price |
| PIC18F25K22 | 28-pin microcontroller | 1 |  |
| OSC-20MHz | Crystal oscillator | 1 |  |
| CY7C128A | SRAM | 1 |  |
| GAL22V10D | Programmable logic chip | 1 |  |

The PIC microcontroller was the central unit of the remote node. Every other part in the node is connected to and controlled by the PIC.

**SRAM Interface**

Please see the local node section about the SRAM Interface, both the local node and remote node use the exact same interface.

**ADC Measurements**

Pin 10 of the PIC was used as input for the ADC that was used for measuring the current voltage of the motor. The motor was simulated with an RC low-pass filter circuit with a 20kHz cutoff filter consisting of a 10k Ohm resistor and a 220nF capacitor.

**PWM Output**

Pin 21 of the PIC was used as PWM output that would control the motor. It was configured to output a 20KHz modulated square wave by software.

**I2C Communication**

An I2C connection was established between the remote and local nodes by connecting pins 11 and 12 on the remote PIC to pins 19 and 20 on the local PIC. The remote node is the master of the monitor channel, where the remote node is sending information about the current measured motor speed and warning state to the local node. The I2C protocol is controlled by software.

1. **SOFTWARE IMPLEMENTATION**
   1. Local Node

### SRAM Control

### RS-232 Communication

### I2C Communication

The hardware configuration created two separate I2C communication channels, greatly simplifying the software implementation. Just like the hardware setup, the software was divided into an outgoing and incoming channel.

#### Outgoing –Control Channel

The outgoing channel sent the speed defined by the user as an 8-bit word. The first Master Synchronous Serial Port (MSSP1) is configured as a master with a baud rate of 400kHz and no slewing. All control of the MSSP1 is done using a simple polling scheme with no interrupts. After waiting for an idle connection, the local node begins communication according to the I2C protocol. The slave address (set to 0x00 for convenience) and a write instruction are sent followed by the speed. The local node then closes the connection with a STOP signal.

#### Incoming – Monitor Channel

The incoming channel utilizes an interrupt-based scheme because the arrival of incoming data is arbitrary. Incoming data is received through Master Synchronous Serial Port 2 (MSSP2), which is configured as a slave device. When an address is sent on the monitor channel, the MSSP2 module compares it to the slave address (0x00 chosen again for convenience). If the address matches, an interrupt flag is asserted and the interrupt handler runs. The interrupt handler takes one of three actions depending on the state of the communication. If the data is an address, the interrupt handler clears the address from the MSSP2 data buffer, deasserts the interrupt flag and exits. If the data is actual information, the interrupt handler checks which bye of the sequence it is expecting. If it is the first byte, the MSSP2 data buffer contents correspond to the motor speed; otherwise the ISR assumes the data is an error state. For each option, the data is saved to the appropriate global variable, the byte counter adjusted appropriately, the interrupt flag is deasserted, and the display flag is set.

* 1. Remote Node

### SRAM Control

### ADC voltage measurement

The analog to digital module measures the actual voltage across the motor. The setup for the ADC involves configuring the ADC clock source, input channel, and acquisition time. These parameters are defined as follows.

* ADC clock source is 1/64th the primary oscillator frequency
* ADC acquisition time is 12 time periods
* Input channel is channel 14 (Pin 10)
* ADC interrupt is disabled
* ADC result is right justified
* ADC voltage references are the internal voltage references (5V and GND)

Since no interrupts are used, the microcontroller polls the ADC status flag until a computation is complete. The result is read into a temporary register and converted to a zero to 200 scale by multiplying the result by 50, bit shifting to the right by 8 and adding 1. To reduce the unnecessary memory writes, and because the precision of the ADC is greater than the noise from the input signal, the current motor speed is only updated if the new measurement is 1% greater or less than the value for the actual speed held in memory. If this is the case, a flag is set to send this value to the local node.

### PWM output

The pulse width modulation (PWM) output has only a minimal software implementation –the majority of the work is handled by the hardware module. Within software, the PWM has two primary components: setup and adjustment of the duty cycle. Setup occurs once each time the microcontroller is reset but adjustment occurs continuously while the PIC18 is running. When PWM setup occurs, the following registers are set:

* PR4 is set for 20kHz frequency
* T4CON is set for a 1:1 pre-scaling factor
* CCP4CON is set to select PWM mode
* CCP4CON and ccPR4L are set to an initial 50% duty cycle
* The Timer4 interrupt flag is cleared
* The GPIO pin B4 (pin 21) is configured as an output

To adjust the PWM duty cycle, the PIC18 calls the compiler function SetDCPWM4(). To pass an appropriately scaled value to the PWM module, the global control variable is multiplied by 5, giving a range from 0 to 1023. The PWM duty cycle is adjusted each time the main program loop is executed.

### I2C Communication

Much of the remote node I2C software operates using the same code as the local node. For a general description of operation refer to the local node section. Specifics for the remote node are provided below.

#### Outgoing –Monitor Channel

The outgoing channel sent two bytes of data to the local node. The first byte encodes the actual motor speed following the coding convention used throughout the system. The second byte contains the error state encoded as an integer. The MSSP1 module is configured just like the local node using the same polling scheme.

#### Incoming – Control Channel

The incoming channel utilizes an interrupt-based scheme in the same manner as the local node. However, since the remote only receives a single byte containing the set speed, the interrupt service routine merely check whether the MSSP data buffer contains an address or actual data: actual data is stored to the set speed global variable; an address is simply cleared from the buffer and discarded.

1. **TESTING**
   1. Test Plan
   2. Test Specification
   3. Test Cases
2. **RESULTS**
   1. Presentation
   2. Discussion
   3. Analysis
3. **ANALYSIS OF ERRORS**
4. **WHY PROJECT DIDN’T WORK**
5. **SUMMARY**
6. **CONCLUSION**
7. **APPENDIX**
   1. Charts and Tables
   2. Graphs
   3. Code