Team 22 – Automated High Volume Bearing Bore Gauge

Final Report

Koyo Bearing's Automated Bearing Bore Gauge

Team Members:

Eric Allgeier
Matthew Boler
Kevin Flemming
Seth Norman
Christopher Proffett

Sponsor:

Robert Potts (Koyo Bearing)

Advisor:

Dr. Cartes

Instructors:

Dr. Amin
Dr. Frank
Dr. Shih

4-17-2014

Table of Contents

Abstract

Koyo Bearings utilizes an automated high volume bearing bore gauge to check the quality of the bearings they produce. Their current system was developed over 20 years ago and is in need of an update. After communicating with Koyo Bearings, it was decided that they are pleased with the existing air transducers set up using LVDTs (Linear Variable Differential Transformer) and were only interested in updating the electronics and interface. We have researched interfacing options and have come up with a design implementing a signal conditioner from the LVDT to the PLC (Programmable Logic Controller) and to a CPU (Central Processing Unit). The CPU stores bearing history and controls the graphical user interface. The PLC is in charge of actuating all mechanical processes in the gauge. The updated bearing bore gauge is now capable of graphical user interface and pneumatic actuation.

Acknowledgements

Team 22 would like to thank Mr. Keith Larson for continued support of this project, including the use of his machine shop and manufacturing insight. We would also like to thank GMF Industries and Pop's Painting for their timely and high quality manufacturing and painting of parts.

Project Overview

Problem statement

The objective of this project is to improve the out-of-date bearing bore gauge system used by Koyo Bearings. This improvement must advance the user interface while maintaining the quality of the measuring device and the sampling rate. The improvement should also allow for the communication of each gage to a central terminal. This will allow for multiple systems to be monitored from a single device.

Project Objectives

The main goal for this protect is to retrofit the bearing gauge testing console with a new computer, operating system, and display. In addition, the machine should later be able to connect to the network at the Koyo plant. Based on the current progress of the bearing bore gauge, the projected date of completion is May 2015.

Overall Methodology

The designing of this retrofit will be broken down into multiple phase. The first phase will be to study the behavior/controls of the air transducers. Then there will be a group decision, consisting of the team members and project advisors, to see if there is any need to replace the transducers with a different style of pneumatic transducer. Phase two will be to research the new heavy duty industrial rated computer and display. Phase three will be to design a complete working system, then submit the design to Koyo Bearing. Phase four will be to make a bill of material and then order the parts needed. Phase five will be to design and machine a new housing system for the selected electronic components. Phase six will be to diagnose the current wiring and rewire with new components. Lastly, phase seven will be to code and debug all programming.

Project Constraints

There are many constraints that must be taken into account starting this project. Money will be a large constraint. It will determine what technology we can actually use in the redesign of this bore gauge. At first, the budget was set to $2,000.00 and Koyo Bearing has requested that we do all purchasing through their company. After discussing the design, Koyo Bearings has agreed to go beyond this budget to ensure a quality machine using highly rated components. Another constraint to this project is time. We will have until the end of the fall semester to explore all design options and decide on a final working prototype. The spring semester will be reserved for construction and testing.

Assign Resources

Eric Allgeier - He has the responsibility of displaying and organizing any group information on a website. He will also need to keep record of all the deliverables, meeting minutes, personal information, critical dates or other records of information that is critical to the project. The information will need to be well organized and easily accessible to all members or inquiring parties.

Matthew Boler - He is responsible for ensuring that all deadlines are met. He is to keep track of all deliverables and assign tasks for each deliverable. He will be the one to turn in all assignments, whether digital or physical. He is also responsible for overseeing and delegating the mechanical aspects of the design.

Kevin Flemming - He will be in charge of building models and machining any parts that are needed. He will have primary access to the bearing bore gauge machine that is located on campus. Also, he will be in charge of the group's finances and will coordinate purchases with Koyo Bearing.

Seth Norman - The role of project manager is to delegate work amongst the team members accordingly to their qualification. He should oversee all work being done on the project and make sure it is being done correctly and in a timely fashion. Also, he will help coordinate with other team members on a realistic timeline for the project, so the team can set mini-goals throughout the project.

Christopher Proffett - It is his primary responsibility to take care of all scheduling for meetings on a weekly and biweekly basis. He is also in charge of communicating with all of our outside advisors and scheduling when the team can meet with them on a consistent basis. He will be in charge of contact with our sponsor, Mr. Robert Potts, and will decide when, where, and how often he would like to meet with us. He is in charge of sending memos to everyone reminding them when and where to be for all meetings.

Product Specifications

Design Specifications

The design chosen must be able to run off of a 120V power supply; this is the current power supply being implemented at Koyo Bearing. Air transducers will be used to measure each boring. The device must be able to operate at a rate of one bearing per six seconds; this includes measuring the bore and determining whether or not it passes the allowable tolerances.

Performance Specifications

The design must incorporate a large touch screen display/input. It must also be able to communicate with the network at Koyo Bearing's facility. The device must also maintain 100% accuracy on tolerance acceptance and maintain the failure alert system; this means that three consecutive failures will alert the staff.

Function Analysis

The Bearing Bore Gage is comprised of three main sections. The first section is mechanical in nature; it locks the bearing in the testing platform, inserts the pneumatic probe, and allows bearings within the tolerance to pass while dumping bearings outside of the tolerance. This is accomplished through the use of pneumatic pistons; when the pistons are activated, they cause linear motion which is the input of a mechanism. Several piston/mechanism combinations are used in conjunction with one another to achieve the desired bearing path.

The second section of the device is the mechanical/electrical interface. This is composed of the LVDTs, figure 1, pneumatic solenoids, and pneumatic cylinders. The LVDTs take the pressure from the pneumatic probe and converts it into an electrical signal, in this case, a differential voltage. The LVDT operates by utilizing three solenoids and a ferromagnetic core. The central solenoid is excited by an external, alternating, voltage source. This induces a current, and therefore a voltage, in the other two solenoids. The placement of the core in the LVDT alters the induced voltage in the secondary solenoids. The core's placement inside the LVDT is a direct result of the pressure from the pneumatic probe. The LVDT outputs the voltages from the secondary solenoids; the difference between these voltages can then be analyzed to give the bearing's bore size.

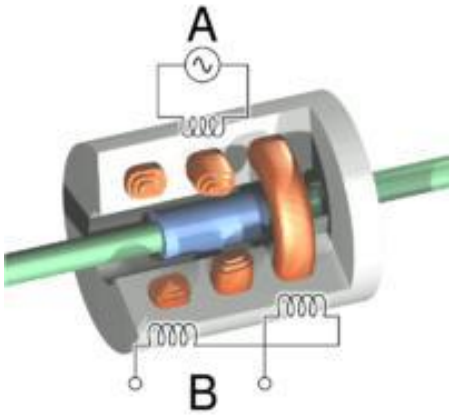


Figure 1. Schematic of an LVDT

The pneumatic solenoids, figure 2, take an electrical signal from the PLC and produces a pressure to actuate a mechanism. This device uses a solenoid as a valve; when a current flows through the solenoid, it creates a magnetic field which forces the magnetic core to open. When the valve is open, it allows the pressurized air to flow to the pneumatic piston. When the valve is closed, the pressurized air is

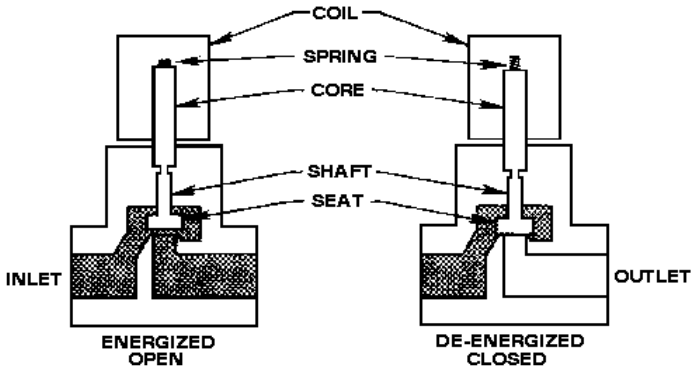


Figure 2. Schematic of a pneumatic actuator

halted. In this case, a secondary opening must be made to release the pressure; this is often done by combining two valves: one valve allows the line to be pressurized, the other acts as a pressure release valve.

The final section of the device is electrical. In this section, the electrical signals from the LVDTs are conditioned and analyzed. The information contained in the signals must be transmitted to the PLC for a decision on tolerances. This must be then transmitted to the linear actuators in order to pass or fail the bearing. The LVDT signal shall also be transmitted to the CPU. The CPU will record the past bearing values and control the user interface. A proximity switch must also be used to indicate when a bearing is present.

Design Concepts

Koyo bearing has expressed their pleasure with the current mechanical system in place and their concerns with the electrical system. Because of this, we have been instructed to update the GUI and networking components only. This means that we will be keeping the current LVDTs and pneumatic actuators while updating the CPU, PLC, display, and user input controls. Therefore, the concepts listed below are for the electrical section described above.

Each concept that was created is based off of the same template, which will be described now; the differences between each concept will be explained in the sections below.

The PLC receives a signal from the Proximity Switch, signaling that a bearing is present. The PLC will first command the actuators to lock the bearing in the testing apparatus and insert the pneumatic probe. The LVDT outputs a signal which gets transmitted to the PLC. By comparing the LVDT signal to the allowable range, the PLC will send signals to the pneumatic actuators which will send the bearing to either the pass or fail bin. An Ethernet switch will allow Koyo Bearing to communicate with the CPU. When the CPU is not communicating with Koyo bearing, it will be receiving the signal from the LVDT. The CPU will also communicate with the GUI which allows the user to see relevant information and input commands.

Concept 1: PC-104

The first design concept that we created utilized a PC-104 board and the CPU. Figure 3 shows the electrical layout in block diagram form. In this configuration, the LVDT signal is read by the PC-104 board. This board interprets the signal and forwards the information to the switch and PLC.

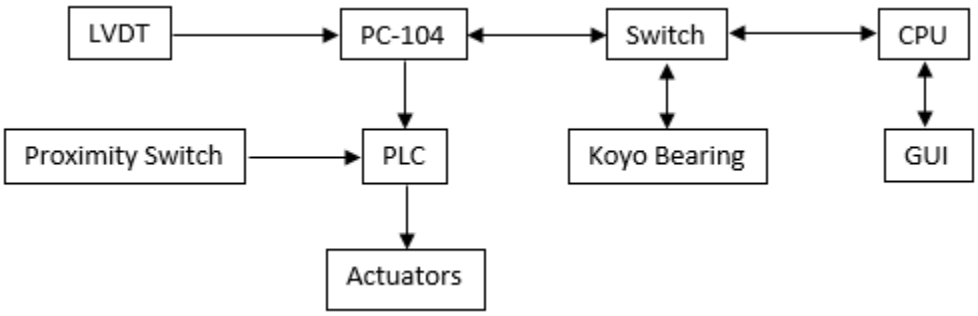


Figure 3: Communication Diagram of Concept 1

Concept 2: SC and PC-104

The second design concept utilizes a PC-104 board in conjunction with a signal conditioning module (SC). Figure 4, below, shows the electrical layout. In this configuration, the LVDT signal is send to the signal conditioning module. A clean signal is then forwarded to the PC-104 board which forwards the signal to the switch and PLC. The signal conditioner also acts as an amplifier; LVDTs usually produce a small differential voltage on the scale of millivolts.

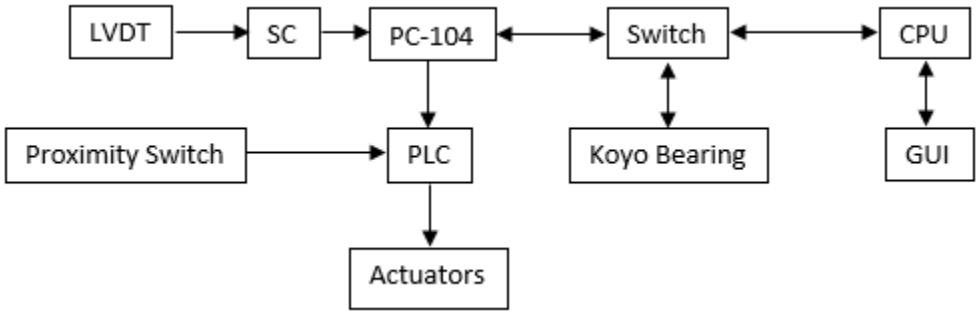


Figure 4: Communication Diagram of Concept 2

Concept 3: SC

The third design concept uses only a signal conditioning module, seen in figure 5. In this design, the LVDT signal is filtered through the SC and sent directly to the switch and PLC, bypassing the PC-104 board from design concept 2. This design is a result of research in the area of signal conditioners. Several signal conditioners are able to be programmed or tolerances. They can then export a logic high or low signal indicating whether the measurement was within specifications.

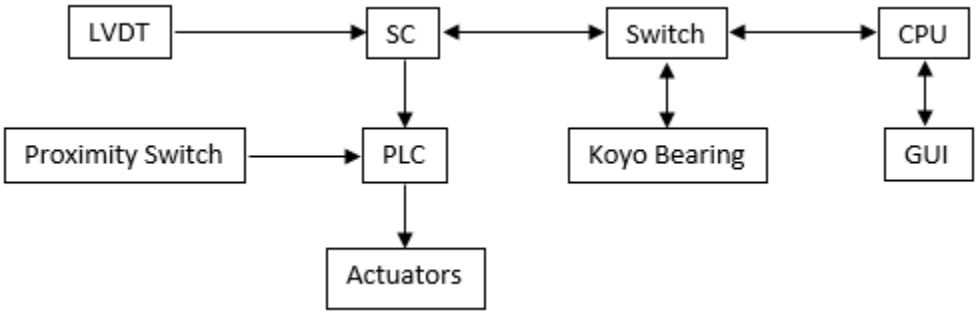


Figure 5: Communication Diagram of Concept 3

Concept 4: SC and no Switch

The last design concept, figure 6, differs from the previous one in the networking section; instead of using an Ethernet switch to prioritize communication between the CPU, SC, and Koyo's plant, the CPU talks directly with both. This can only be accomplished if the CPU has multiple RJ-45 ports.

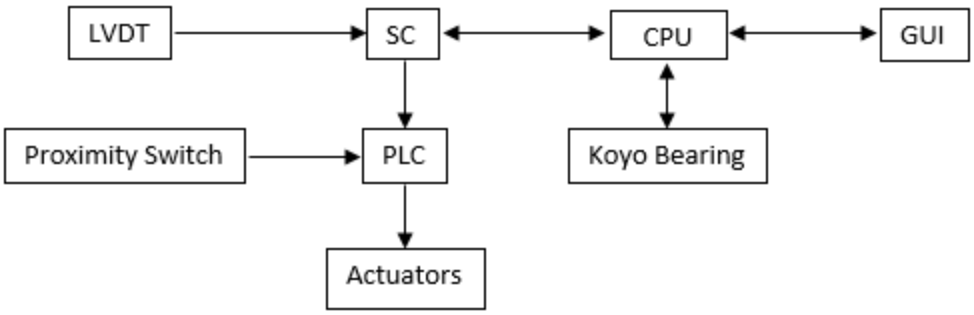


Figure 6: Communication Diagram of Concept 4

Electronic Components

CPU

A Raspberry Pi Model B was found to have all of the capabilities that were required of the CPU. It contained an RJ-45 port, two USB ports, and an HDMI port. Further research into other CPUs was halted when Koyo Bearing informed us that they are already in possession of a Lenovo ThinkCentre M92p CPU. This board is much faster and has more ports than the Raspberry Pi; and should perform all necessary tasks.

Switch

Because both the signal conditioner, or PC-104 board, and Koyo's plant must be able to communicate with the CPU, a switch must be implemented. An Ethernet switch provides a simple means of achieving this communication junction. Table 1 shows the relevant technical specification of the possible Ethernet switches. All of these are DIN rail mountable to allow for easy instillation and replacement.

Table 1: Relevant Specifications on possible Ethernet switches

Ethernet Switch	Cost	Number of Ports	Speed	Power Consumption	MTBF
N-T1005TX	$288.00	5	10/100/1000 Mbps	36 Watt	2,000,000 hr.
EISK5-GT	$148.00	5	10/100/1000 Mbps	3 Watt	N/A
IES5100	$77.24	5	10/100 Mbps	2.4 Watt	1,677,807 hr.

Power Supply

Because the devices being considered operate on a DC voltage, the 120VAC signal provided by Koyo's plant must be modified. The output power will be determined by the other components in use. For convenience, all devices will run off of the same 24V DC signal. Table 2 is a compilation of possible power supplies. Each power supply runs off of 120VAC at 60Hz which is supplied by Koyo Bearing. They are also DIN rail mountable to allow for easy installation and replacement.

Table 2: Relevant specifications on possible power supplies

Power Supply	Cost	Output Voltage	Output Power	Housing	Max Current	MTBF
PSB24-060-P	$28.00	24 VDC	60 Watts	Plastic	2.5 Amp	>800,000 hr.
PS24-050D	$99.00	24 VDC	50 Watts	Metal	2.0 Amp	2,992,000 hr.
1769-PA4	N/A	24 VDC	48 Watts	Metal	2.0 Amp	N/A

Monitor

A monitor is required to display relevant information to the user. By selecting a touch screen, it will also act as the operators input. The previous year's team has recommended the FPM-5191G-X0AE. This is a 19" touchscreen HD LCD display. Due to budget constraints and the size of the recommended monitor, we decided to find a smaller display. The ELO 1537L was chosen for its industrial strength, multiple mounting options, and low cost.

PLC

Because the PLC is the brains behind the bearing bore gauge, Koyo Bearing has specifically requested an Allen Bradley model; Allen Bradley is the industrial standard for PLCs. After communicating our needs to tech-support, we decided on the MicroLogix1200

Design Selection

In our final design we ended up choosing concept 3. We chose this because it is the simplest and most robust design. Design 3 gives us the least amount of components that are required for the machine to function. This keeps cost as low as possible and it keeps the overall design simple so there is less of a chance for component failure. We made one edit to design 3, it was decided that we would need a router instead of just an Ethernet switch. This is needed to keep the internal IP address separate from the Koyo network IP address. This is important for smooth networking between the machine and Koyo bearings factory network. Below, in table 3, is a simple decision matrix that was used to help us determine the most efficient design for this machine.

Table 3: Decision Matrix

	Cost	Complexity	Durability	Total
Design 1	7	6	7	20
Design 2	8	7	7	22
Design 3	9	10	8	27
Design 4	10	10	4	24

Programming Needs and Control

PLC (Programmable Logical Controller)

The PLC will need to be programed to accept the signal coming from the LVDT and actuate the pneumatics. The LVDT will be connected to a signal conditioner. The signal conditioner will connect to the PLC, through the router, with a single digital signal that is normally closed. Therefore, the PLC will have a nested logical algorithm that is controlled by signal coming from the signal conditioner. Once the logical algorithm has been processed by the PLC, the PLC will send a command to the pneumatic linear actuators to either accept or denied bearing, depending on the size tolerance of the bearing.

Signal Conditioner

The signal conditioner will need to be configured to work with LVDT inside of the air transducer. The LVDT works on a differential voltage between the secondary coils. The difference between the voltages will give the size of bore on the inner diameter of the bearing. The signal conditioner will need to be calibrated to accept the minimum to maximum allowable size of the bearing. In the case where the bearing is between the minimum and maximum range, the signal conditioner will be programmed to send a low logic flag to the PLC. When the LVDT is out of range, the signal conditioner will send high logical flag to the PLC.

The signal conditioner will also need to be programed to send data to the new CPU. The data coming from signal conditioner will be stored in CPU for a history of accepted and denied bearing, so the Koyo plant can have data on the efficiency of their lathes and tooling.

CPU (Central Processing Unit)

The CPU will be used in this design to store data coming from signal conditioner. This data will be used to display a histogram to the operator of the machine and to the operator of the Koyo plant. Also, the CPU will be used to calibrate the signal conditioner for the minimum and maximum range of the bearing bore size. Advance Micro Controls INC., the makers of the signal conditioner, already have a program that will load on the CPU and be used for calibrating the signal conditioner. A program will need to be written to display the current histogram of the machine. Also in this program there will be a parts counter. Which displays the number of accepted and rejected bearing during an eight hour shift. Also, the program will warn the operator when the tooling inside their lathe needs to be changed.

Design for Manufacturing

The first plan of action to manufacture this machine is to obtain all components and produce a test bench that will utilize all the logical functions of this machine before the machine is assembled. To produce this test bench we will be wiring a set of manual switches to replicate the inputs to the machine and lights to represent the outputs from the PLC. With all the inputs and outputs simulated we can program the PLC for the desired outputs without risking any damage to the machine.

To program the GUI, Visual C++ will be used. The display has a home menu with various screens that may be desired by the operator. All menu options and statistical analysis equations are provided in detail in the current user's manual. This organization of the software will be

replicated. The machine is going to utilize an Ethernet router to establish an internal network that will be utilized to pass data from the signal conditioners to the CPU and the PLC. All modules in this network will be assigned a static IP address.

To begin the manufacturing process of this machine all current electrical components, wiring, and mounting devices will be removed. To insert the new touch screen, housing must be rebuilt first. The current design utilizes a module mounting rack for the PLC, a similar style mounting will be manufactured to replace it. For the rest of the electrical components a din rail will be mounted in the rear of the enclosure to attach the components to. When all components are inserted, wiring will be run and landed for all power, logic, and communication needs. When all is assembled the final stages of testing and debugging can commence.

Final Design

The current housing, figures 7 and 8, was redesigned to house all of the electrical components needed to function the bearing bore gauge. A PLC rack was built in order to place the PLC and Ethernet module in a central location for ease of wiring set up. The front of the housing was resized so that could accommodate the touch screen monitor. The back of the housing had Din-rail and Panduit mounted to it so that we could fit the rest of the electrical components and the necessary wiring.

Figure 7: Main Housing Compartment

Figure 8: Housing Back Panel

The PLC, in figure 7, is currently wired to all inputs on the front panel, all proximity switches, all pneumatic outputs, and the power supplies. It is coded to automatically pass all bearings because the measuring device has not been implemented. The code for manual operation was implemented, but we were unable to alternate between the two modes during operation.

The GUI displays the necessary information needed from the operator and to master the signal conditioner. When the user opens the GUI they will be greeted with a menu layout that contains functions similar to that on the previous GUI displayed. The user will need to master the gauge at the start of every shift. When the machine is mastered, the GUI will log the time and return back to the home page. Utilizing a receiving C++ socket, the GUI opens up a socket to the IP 168.192.0.50 at port 502. When then information is received it will be written to a file inside the computer along with a time string to record the day, month, year, hour, minute, and second that the data was recorded. The GUI will display the information in a variety of ways for the

operator. Including a single bearing as it reads the inside middle and outside dimensions along with the bell mouth and taper, figure 9. A histogram can also be displayed showing all the data inside the current file. The GUI will also record time sensitive information in the clock menu. This will include when the gauge was mastered and when the machine failed three bearings in a row to alert the operator.

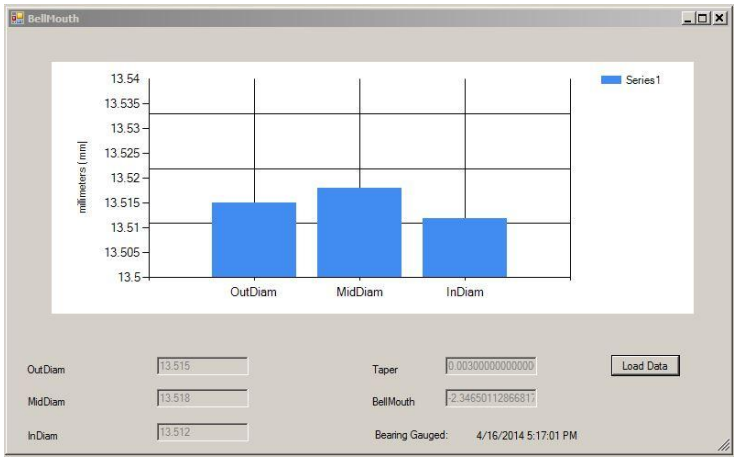


Figure 9: Bell mouth graph from GUI

Considerations for Environment, Safety, and Ethics

Environmental

The machine being updated does not contain any hazardous chemicals. So there will not be any environmental issue that deal with the containment of chemicals. The machine is designed with a low energy use, so there will less impact on the environment.

Safety

All work to be done on the machine will follow a strict lock out tag out (LOTO) procedure on all potential energy. Wiring has been completed and no exposed wires remain and all parts of the machine are grounded. In addition, all power is being supplied through a power switch and an emergency stop bottom.

The machine has been design with a latching mechanism on the electrical housing. This way once the machine is completed and placed on the manufacturing floor at the Koyo plant there will be no access to the electrical components by unauthorized personnel.

Ethics

The machine was designed after modeling the original machine. We are not using any of the original electrical components in the design nor are we doing any reverse engineering. Therefor there is no infringement on the previous design.

Conclusion

This year, our team made major progress in the renovation of the bearing bore gauge system given to us by Koyo Bearings. We were tasked with updating the user interface and adding in networking capabilities. We were also required to maintain the sampling rate and quality of measurement currently on the device. We came up with 4 design schemes eventually settling on a design utilizing a new PLC, an Ethernet switch, and a Signal Conditioner to accurately filter the signal sent from the LVDT and pneumatic gauge. We designed and manufactured a housing for

the electronics and wired existing, and new, components into the housing. We also programmed the foundation for a graphical user interface or GUI that will display the data accumulated by measuring the bearings. The finale bearing bore gauge can be seen in figure 10.

There were several project issues that came up over the year, the largest being the procurement of parts. Our sponsor was preoccupied during the ordering process resulting in large procurement delays. This led to our team needing to take initiative and procure parts that were not in our design, but would meet the requirements. Once we had all the parts we needed we successfully programmed the PLC to actuate the machine and were able to display the GUI and implement it on a touch screen monitor.

Figure 10: Finalized Bearing Bore Gauge

In conclusion, large procurement delays made it impossible to complete the project as fully intended. That being said, our team accomplished a great deal in the redesign of the machine. We came up with a solid design, procured all the parts needed to complete the renovation, and programmed a software platform for collecting and displaying data.

For the continuation of this project, the next team will be tasked with integrating the software and hardware we have left for them and should be fully capable of delivering a functioning machine. Below you will find our recommendations for future work for the individual components of the machine.

Recommendations for Future Work

Based on the nature of this project and what still needs to be accomplished, we recommend that next year this becomes an electrical project. As a minimum, the next team should contain one electrical engineer, one mechanical engineer, one computer engineer, and one computer science graduate student.

Housing

The current housing system was designed to house the desired electrical components. The PLC mounting plate should be built taller; this will allow for more wiring room. It is also recommended that the front panel is redesigned with a hinge. This would allow for maintenance to be done easily without hanging wires. Lastly, a locking mechanism should be installed on the Bearing Bore Gauge to keep out unwanted personnel.

PLC

The PLC will need some modifications to the programming. As the machine sits now it is programmed so that it always accepts the bearing. The PLC will need to be programmed to accept or reject the bearing once measurement has been made. Also the red light on top of the

machine will need to be programmed. If there are three consecutive rejected bearings, the light needs to flash at 1Hz.

The PLC is programmed using RSlogix 500 software which uses ladder logic. Whoever starts the programming PLC needs know that ladder logic is simple form logical programming (do not over complicate it). RSlogix 500, and RSLinx (used for communicating to the PLC) is already programmed on the computer that is located in Keith Larson's lab. Also, the installation disk and installation key will be left with Mr. Larson. Do not lose this disk and key, it cost Koyo Bearings $2,000.

There is a PLC Ethernet device that will connect to the Ethernet switch. The Ethernet switch will connect the CPU, PLC, LVDT Signal Conditioner, and Koyo's Plant together so the all can network. The PLC Ethernet device will need to be configured to work with the CPU and the signal conditioners.

Measuring Device

When we first were designing the Automated High Volume Bearing Bore Gauge, we found that the new Edmunds CAG's use a LVDT (Linear Variable Differential Transformer). So we design the machine around these devices. In the last two weeks, we found that the measuring device are not LVDT, but solid state pressure device. From this information, we came up with a solution. The bearing bore gauge will need to incorporate a pressure sensor amplifier. This amplifier will take the signal from the solid state pressure device and turn it useable data. The Edmunds Gage Company makes these amplifier, so it would probably be best to use their amplifier. It is unnecessary to inform them of the project; they need only know that you are measuring air pressure. These pressure amplifiers will need to be installed in the housing.Alternatively, the solid state pressure device housing can be redesigned to hold an LVDT. This would enable the signal conditioners to take measurements.

GUI

Currently, the GUI can receive and store data. However, communication needs to be established with the measuring device. All errors and irregularities in the Windows OS should be recorded on the clock menu for the operator's convenience. The GUI must be programmed to count the number of bearings tested, passed, and failed. The graph layout should be submitted to Koyo Bearings for approval.

Schedule, Recourses, and Budget

Budget

Table 4: Bill of Materials

Device	Part Number	Price Per Unit	Quantity	Price
PLC – MicroLogix1200	1762-L24AWA	$566.20	1	$566.20
PLC – Ethernet Module	1761-NET-ENI	$950.00	1	$950.00
PLC - Software	RSLogix 500	$2000.00	1	$2000.00
Signal Conditioner	ANR2	$895.00	2	$1790.00
Power Supply 24V	PSB24-060-P	$28.00	1	$28.00
Power Supply 12V	PSB12-060	$37.25	1	$37.25
Router	CTR-Link EIPR-E	$299.00	1	$299.00
Monitor	ELO 1537L	$527.00	1	$527.00
Circuit Breakers	QUO110	$30.65	1	$30.65
Misc. (DIN Rail…)				~ $200.00
Total			**11**	**$6428.10**

Fall Schedule

Team 22 - Gantt Chart																
	Date that Week Starts															
	26-Aug	2-Sep	9-Sep	16-Sep	23-Sep	30-Sep	7-Oct	14-Oct	21-Oct	28-Oct	4-Nov	11-Nov	18-Nov	25-Nov	2-Dec	9-Dec
Project Assignment/ Ice Breaking																
Need Assessment																
Code of Conduct																
Bi-Weekly Reports																
Staff Meetings																
Project Plans/ Product Specs																
Analyze Device																
Pneumatic Transducers																
PLC																
Pneumatic Actuators																
Research																
PC104 Boards																
Alternate PLC Devices																
CPU																
Interfaceing																
Concept Generation and Selection																
Design Development																
Design Selection																
Submit Design																
Team Evaluation Report																
Interim Presentation / Report																
Create Report																
Presentation to MEAC																
Ordering Parts																
Final Presentation / Report																

Spring Schedule

Team 22 - Gantt Chart - Spring																
	Day that Week Begins															
	6-Jan	13-Jan	20-Jan	27-Jan	3-Feb	10-Feb	17-Feb	24-Feb	3-Mar	10-Mar	17-Mar	24-Mar	31-Mar	7-Apr	14-Apr	21-Apr
Establish regular schedule																
Confirm status of parts																
Create test platform																
Disassemble current electronics																
Write and test code																
Integrate electrical components																
Final debugging and testing																
Calibrate final product																
Demonstrate bearing bore gage																

Product Specifications

The bearing bore gauge will require the following for operation:

24 in. x 30 in. of floor space / 76 in. high

120 VAC / 15 Amps

100 - 150 psi

Standard Procedure

This procedure was written for the completed bearing bore gauge. The current machine is not yet able to perform all of these tasks.

Start Up

For first time start up, see Additional Assembly section prior to the following procedure.

1. Turn the power switch to the on position.
2. Follow calibration instructions.

Data Acquisition

The bearing bore gauge will automatically be acquiring data from the LVDTs and the proximity switches. To access the histogram and running statistics, follow these steps.

1. Click “SPC Charts” button.
2. To view a histogram, click “histogram.”
3. To view running statistics for the current shift, click “statistics.”

Mode of Running

Manual Operation

1. To enter manual operation, turn the bypass knob to the manual position.
2. Turn the “gage plug” knob to “adv” to advance the plug into the bearing.
3. Press the “meter part” button.
4. Turn the “gage plug” knob to “ret” to retreat the plug from the bearing.

Automatic Operation

1. To enter automatic operation, turn the bypass knob to the automatic position.

Shutdown

1. Click the “gauge master” icon on the GUI.
2. Click the ANR2 Module 1 tab.
3. Click the measurement sub-tab.
4. Click “Stop Measurement”.
5. Turn the power switch to the off position.

Calibration

1. Click the “gauge master” icon on the GUI.
2. Click the ANR2 Module 1 tab.

3. Go to the configuration sub-tab.
4. Choose a two-point linear alignment type.
5. Set the excitation voltage to 4000 mV.
6. Set the excitation frequency to 4000 Hz.
7. Set Channel 1 wire mode to “four wire.”
8. Set Channel 1 displacement to 1000.
9. Set Channel 1 Sensitivity to 1000.
10. Repeat steps 7 - 9 for Channels 2 and 3.
11. Go to the alignment sub-tab.
12. Set channels 1 - 3 to a two point linear alignment.
13. Click “Enter Alignment Mode.”
14. Place minimum bearing into the bearing track.
15. Set minimum values for channels 1 – 3.
16. Replace minimum bearing with maximum bearing on the bearing track.
17. Set maximum values for channels 1- 3.
18. Co to the measurement sub-tab.
19. Enable all channels.
20. Click “Start Measurement.”

Additional Assembly

Upon reception of the bearing bore gage follow these steps for first time assembly:

1. Place in desired location – must have access to compressed air and electricity.
2. Connect 100 – 150 psi air source to the air input line on bearing bore gauge.
3. Connect the 120 VAC bearing bore gage plug into nearby 120 VAC port.

Trouble Shooting

This section is organized so that all lines with the same numbering pertain to the same issue.

Potential Problems

1. Bearings are backed up.
2. Broken actuation.
3. Monitor is unresponsive to touch.
4. Total machine shutdown.

Possible Cause

1. Bearing jammed in track, debris in actuating components, faulty proximity switch
2. Pneumatic leaks (loose hose, fittings), bad solenoid, deformed rods
3. Greasy fingers, wet screen, loose USB cable
4. Emergency stop button is engage, power is disconnected

Recommended Solution Procedure

1. Remove jammed bearing, clean bearing track/actuating components, replace proximity switch
2. Replace bad hose/fittings, replace bad solenoid, replace deformed rods

3. Wash hands, clean screen, check USB cable connection
4. Disengage emergency stop button, check that power supply is plugged in

Routine Maintenance

- Clean screen
- Clear bearing track
- Lubricate moving parts
- Oil air fittings

Future Repair

- All design components selected for significantly high MTBF (Mean time between failures), no internal component failure expected in the foreseeable future.
- Mechanical components not evaluated in this project; we are not qualified to comment on part status.

Suggested Spare Parts

The following parts are recommended to have on hand to reduce down time in case of minor failure.

- Air hose, hose fittings, chucks
- Set of Master bearings
- Various connection cables (USB, mini-din)
- Buttons and switches

References

http://eng.fsu.edu/me/senior_design/2014/team22

Appendix

Components

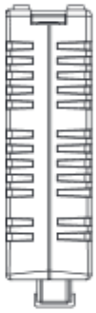
These are the component diagrams for the optimum design.

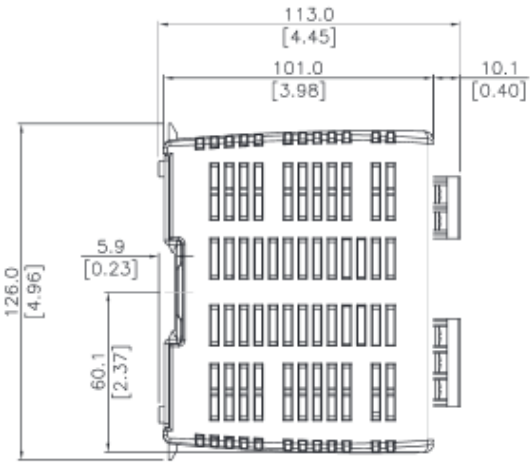
Power Supplies

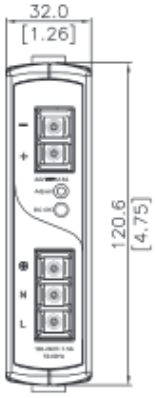
PSB24-060-P (24V)

PSB24-060-P

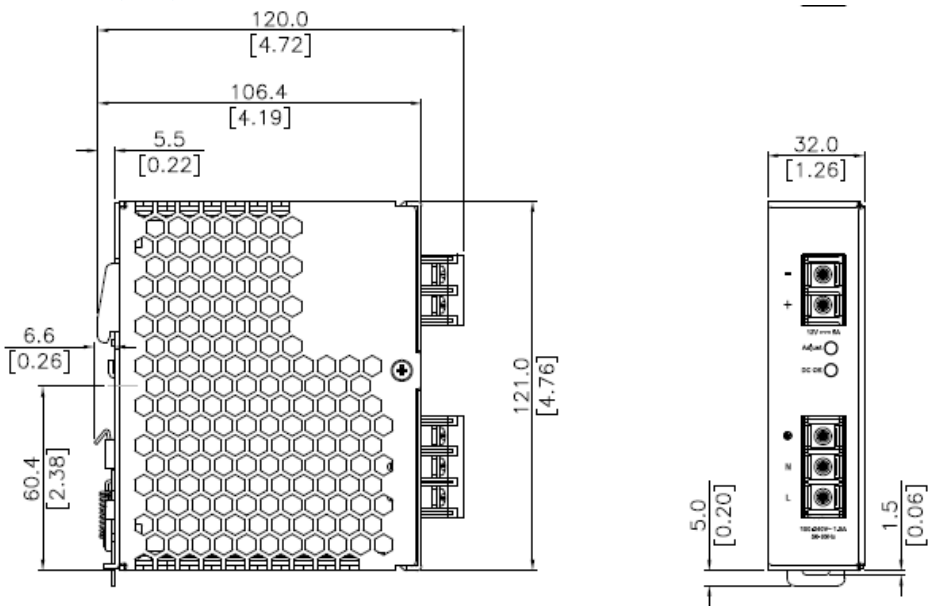
Wiring Connection			
Input		*Output*	
L	Line	+	Out +
N	Neutral	-	Out -
⏚	AC Ground		





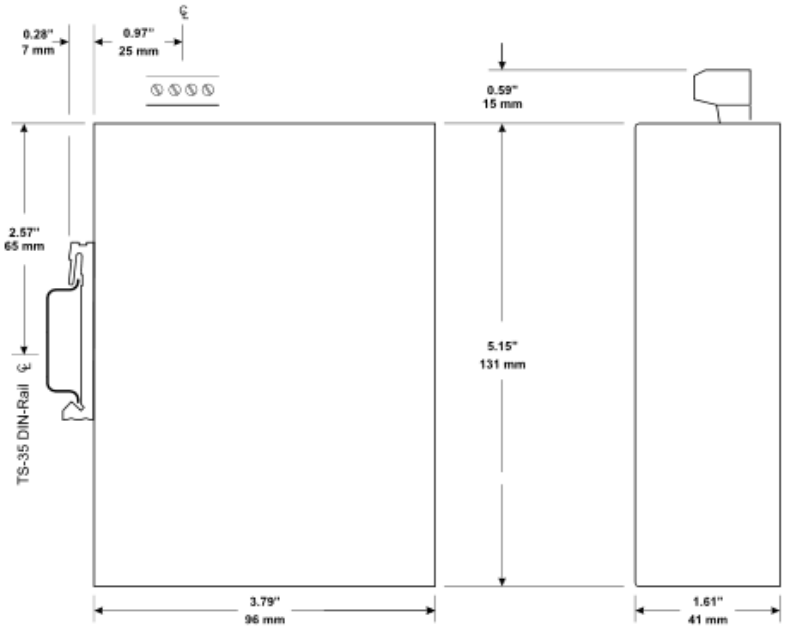


PSB12-060 (12V)



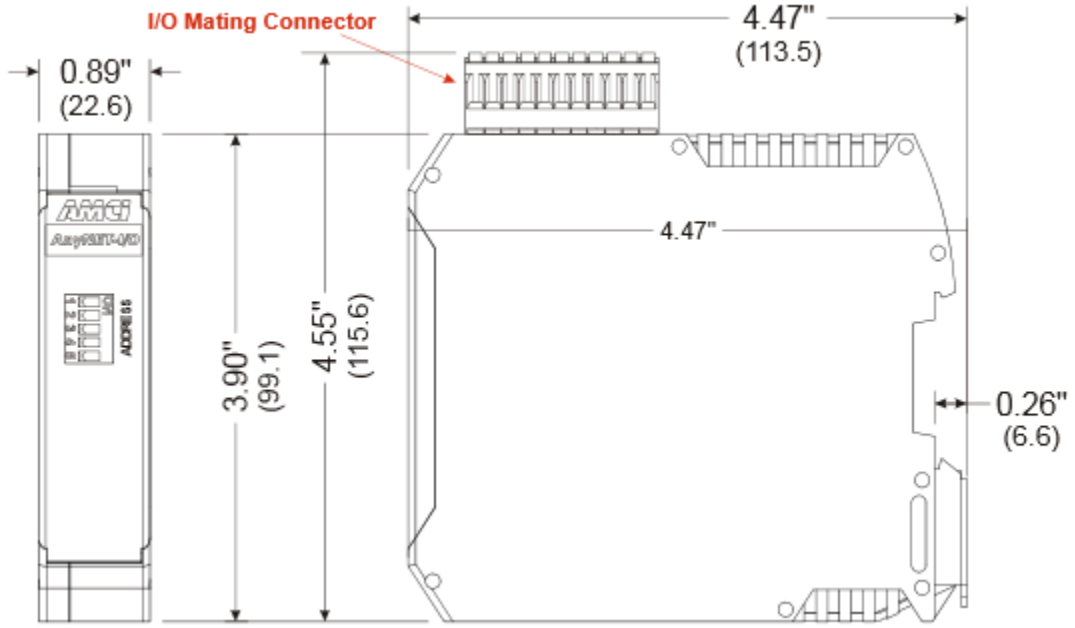
Router

CTR-Link EIPR-E



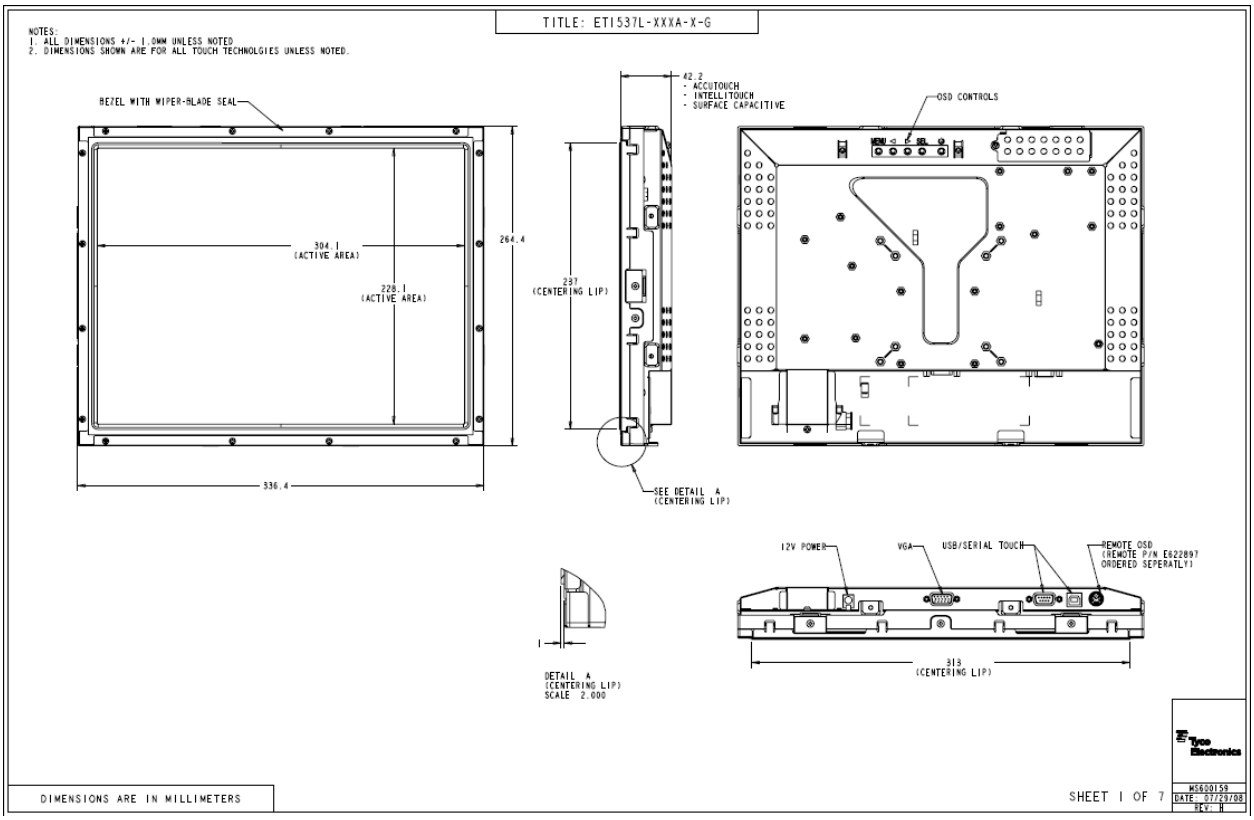
Signal Conditioner

AnyNET I/O ANR2 Network Compatible LVDT/RVDT Signal Conditioner



Monitor

ELO 1537L



Mechanical

Back Panel Components

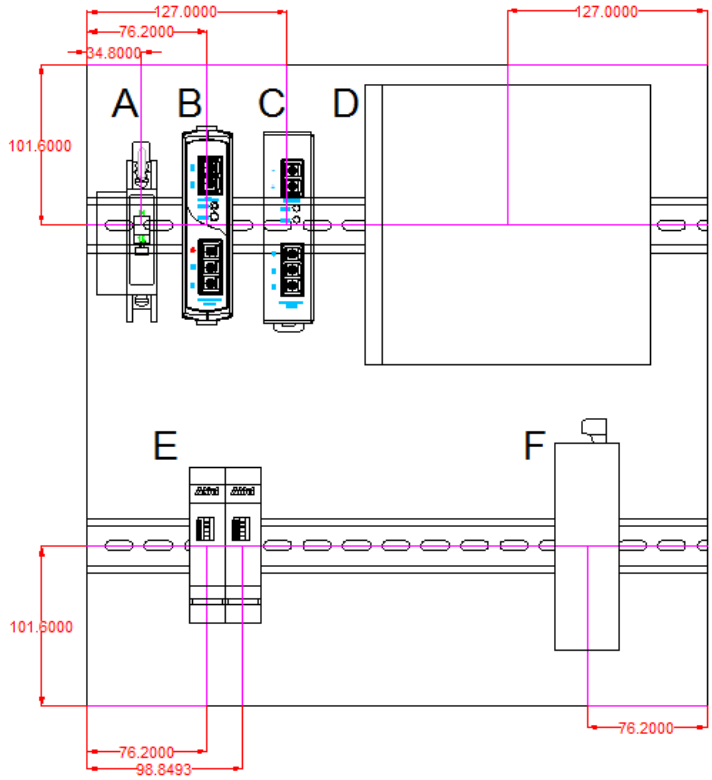
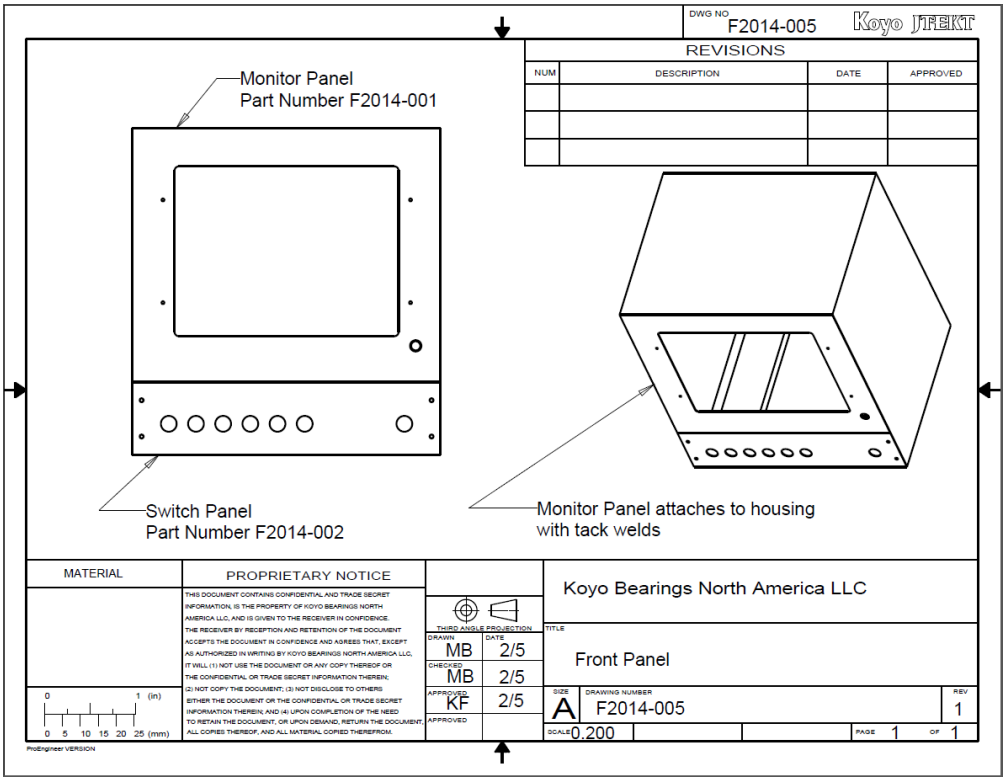


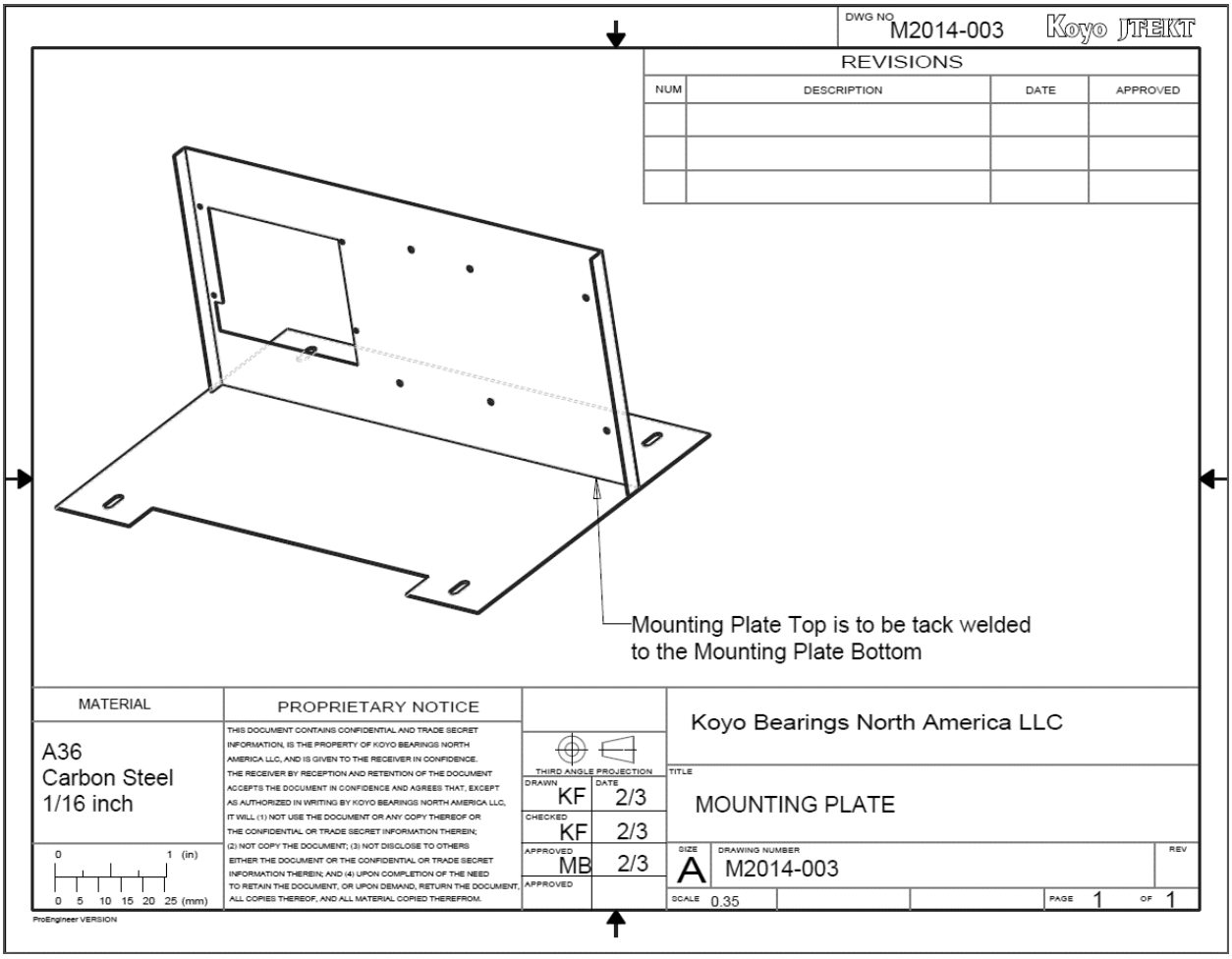
Table 5: Component Legend

	Part
A	10A circuit breaker
B	12VDC Power Supply
C	24VDC Power Supply
D	CPU
E	Signal Conditioner
F	Ethernet Router

Front Panel



PLC mounting plate



Sample Program Code

The program code has been collected for next year. Here is a sample of two functions.

Socket

```
#define WIN32_LEAN_AND_MEAN
#include <winsock2.h>
#include <Ws2tcpip.h>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <string>
#include <string.h>
#include <time.h>
using namespace std;
// Link with ws2_32.lib
#pragma comment(lib, "Ws2_32.lib")

#define DEFAULT_BUFLEN 512
#define DEFAULT_PORT "23"

int __cdecl main() {

   //----------------------
   // Declare and initialize variables.
   WSADATA wsaData;
   int iResult;

   SOCKET ConnectSocket = INVALID_SOCKET;
   struct sockaddr_in clientService;

   char recvbuf[DEFAULT_BUFLEN];
   int recvbuflen = DEFAULT_BUFLEN;

   //----------------------
   // Initialize Winsock
   iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
   if (iResult != NO_ERROR) {
     printf("WSAStartup failed: %d\n", iResult);
     return 1;
   }

   //----------------------
   // Create a SOCKET for connecting to server
   ConnectSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
   if (ConnectSocket == INVALID_SOCKET) {
      printf("Error at socket(): %ld\n", WSAGetLastError() );
      WSACleanup();
      return 1;
```

```
    }

    //---------------------
    // The sockaddr_in structure specifies the address family,
    // IP address, and port of the server to be connected to.
    clientService.sin_family = AF_INET;
    clientService.sin_addr.s_addr = inet_addr( "144.174.126.195" );
    clientService.sin_port = htons(23);

    //---------------------
    // Connect to server.
    iResult = connect( ConnectSocket, (SOCKADDR*) &clientService, sizeof(clientService) );
    if ( iResult == SOCKET_ERROR) {
       closesocket (ConnectSocket);
       printf("Unable to connect to server: %ld\n", WSAGetLastError());
       WSACleanup();
       return 1;
    }

    // Receive until the peer closes the connection
    do {

       iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
       if ( iResult > 0 )
                        {
          printf("Bytes received: %d\n", iResult);
                                time_t t = time(0);   // get time now
                                struct tm * now = localtime( & t );
                                int year = now->tm_year +1900;
                                int month = now->tm_mon +1;
                                int day = now->tm_mday;
                                int hour = now->tm_hour;
                                int minute = now->tm_min;
                                int second = now->tm_sec;

                                ofstream myfile ("bearingSize.txt",ios::out |ios::app);
                                myfile << month << "/" << day << "/" << year  << " " << hour << ":" << minute <<
":"  << second << " ";
                                myfile.write(recvbuf,iResult);
                                myfile << "\n";
                                myfile.close();

                        }
       else if ( iResult == 0 )
          printf("Connection closed\n");
       else
       printf("recv failed: %d\n", WSAGetLastError());
    } while( iResult > 0 );
```

```
    // cleanup
    closesocket(ConnectSocket);
    WSACleanup();

    return 0;

}
```

Form 1

```
//FORM1//


#pragma once
#include "GaugeMastering.h"
#include "Limits.h"
#include "SPC.h"
#include "DataReset.h"
#include "GaugeReadings.h"
#include "InputSelection.h"
#include "PartCounters.h"
#include "ClockMenu.h"
#include "SystemAlarm.h"
#include "Utilities.h"
#include <iostream>
#include <Windows.h>
using namespace std;
namespace KoyoGui {

        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;

        /// <summary>
        /// Summary for Form1
        /// </summary>
        public ref class Form1 : public System::Windows::Forms::Form
        {
        public:
                Form1(void)
                {
                        InitializeComponent();
                        //
                        //TODO: Add the constructor code here
```

```
                //
            }

        protected:
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            ~Form1()
            {
                if (components)
                {
                    delete components;
                }
            }
        private: System::Windows::Forms::Button^  btnGaugeMaster;
        private: System::Windows::Forms::Button^  btnLimits;
        private: System::Windows::Forms::Button^  btnSPC;


        private: System::Windows::Forms::Button^  btnPartCounters;

        private: System::Windows::Forms::Button^  btnSystemAlarm;
        private: System::Windows::Forms::Button^  btnClockMenu;

        public: System::String^ Mtime;
        protected:

        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                this->btnGaugeMaster = (gcnew System::Windows::Forms::Button());
                this->btnLimits = (gcnew System::Windows::Forms::Button());
                this->btnSPC = (gcnew System::Windows::Forms::Button());
                this->btnPartCounters = (gcnew System::Windows::Forms::Button());
                this->btnSystemAlarm = (gcnew System::Windows::Forms::Button());
                this->btnClockMenu = (gcnew System::Windows::Forms::Button());
                this->SuspendLayout();
                //
```

```
                        // btnGaugeMaster
                        //
                        this->btnGaugeMaster->Location = System::Drawing::Point(12, 12);
                        this->btnGaugeMaster->Name = L"btnGaugeMaster";
                        this->btnGaugeMaster->Size = System::Drawing::Size(100, 100);
                        this->btnGaugeMaster->TabIndex = 0;
                        this->btnGaugeMaster->Text = L"Gauge Mastering";
                        this->btnGaugeMaster->UseVisualStyleBackColor = true;
                        this->btnGaugeMaster->Click += gcnew System::EventHandler(this,
&Form1::btnGaugeMaster_Click);
                        //
                        // btnLimits
                        //
                        this->btnLimits->Location = System::Drawing::Point(12, 118);
                        this->btnLimits->Name = L"btnLimits";
                        this->btnLimits->Size = System::Drawing::Size(100, 100);
                        this->btnLimits->TabIndex = 1;
                        this->btnLimits->Text = L"Limits";
                        this->btnLimits->UseVisualStyleBackColor = true;
                        this->btnLimits->Click += gcnew System::EventHandler(this,
&Form1::btnLimits_Click);
                        //
                        // btnSPC
                        //
                        this->btnSPC->Location = System::Drawing::Point(12, 224);
                        this->btnSPC->Name = L"btnSPC";
                        this->btnSPC->Size = System::Drawing::Size(100, 100);
                        this->btnSPC->TabIndex = 2;
                        this->btnSPC->Text = L"SPC";
                        this->btnSPC->UseVisualStyleBackColor = true;
                        this->btnSPC->Click += gcnew System::EventHandler(this,
&Form1::btnSPC_Click);
                        //
                        // btnPartCounters
                        //
                        this->btnPartCounters->Location = System::Drawing::Point(118, 12);
                        this->btnPartCounters->Name = L"btnPartCounters";
                        this->btnPartCounters->Size = System::Drawing::Size(100, 100);
                        this->btnPartCounters->TabIndex = 5;
                        this->btnPartCounters->Text = L"Part Counters";
                        this->btnPartCounters->UseVisualStyleBackColor = true;
                        this->btnPartCounters->Click += gcnew System::EventHandler(this,
&Form1::btnPartCounters_Click);
                        //
                        // btnSystemAlarm
                        //
                        this->btnSystemAlarm->Location = System::Drawing::Point(118, 224);
                        this->btnSystemAlarm->Name = L"btnSystemAlarm";
```

```
                this->btnSystemAlarm->Size = System::Drawing::Size(100, 100);
                this->btnSystemAlarm->TabIndex = 7;
                this->btnSystemAlarm->Text = L"System Alarm";
                this->btnSystemAlarm->UseVisualStyleBackColor = true;
                this->btnSystemAlarm->Click += gcnew System::EventHandler(this,
&Form1::btnSystemAlarm_Click);
                //
                // btnClockMenu
                //
                this->btnClockMenu->Location = System::Drawing::Point(118, 118);
                this->btnClockMenu->Name = L"btnClockMenu";
                this->btnClockMenu->Size = System::Drawing::Size(100, 100);
                this->btnClockMenu->TabIndex = 8;
                this->btnClockMenu->Text = L"Clock Menu";
                this->btnClockMenu->UseVisualStyleBackColor = true;
                this->btnClockMenu->Click += gcnew System::EventHandler(this,
&Form1::btnClockMenu_Click);
                //
                // Form1
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(775, 565);
                this->Controls->Add(this->btnClockMenu);
                this->Controls->Add(this->btnSystemAlarm);
                this->Controls->Add(this->btnPartCounters);
                this->Controls->Add(this->btnSPC);
                this->Controls->Add(this->btnLimits);
                this->Controls->Add(this->btnGaugeMaster);
                this->Name = L"Form1";
                this->Text = L"Home";
                this->WindowState = System::Windows::Forms::FormWindowState::Maximized;
                this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
                this->ResumeLayout(false);

            }
#pragma endregion
             private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e) {
             }
    private: System::Void btnGaugeMaster_Click(System::Object^  sender, System::EventArgs^  e) {
                    WinExec("C:\\OTGUI4-9-
2014\\KoyoGui\\KoyoGui\\AmciEthernetConfigurator15.exe", SW_SHOW);
                    DateTime datetime = DateTime::Now;
                    Mtime = datetime.ToString();
                    System::IO::File::AppendAllText("LOGTIME.txt","Mastered at: " +
Mtime +Environment::NewLine);

                }
```

```
private: System::Void btnLimits_Click(System::Object^  sender, System::EventArgs^  e) {
                Limits ^ form = gcnew Limits;
                form->ShowDialog();
            }

private: System::Void btnSPC_Click(System::Object^  sender, System::EventArgs^  e) {
                SPC ^ form = gcnew SPC;
                form->ShowDialog();
            }
private: System::Void btnDataReset_Click(System::Object^  sender, System::EventArgs^  e) {
                DataReset^ form = gcnew DataReset;
                form->ShowDialog();
            }
private: System::Void btnInputSelection_Click(System::Object^  sender, System::EventArgs^  e) {
                InputSelection ^ form = gcnew InputSelection;
                form->ShowDialog();
            }
private: System::Void btnPartCounters_Click(System::Object^  sender, System::EventArgs^  e) {
                PartCounters ^ form = gcnew PartCounters;
                form->ShowDialog();
            }
private: System::Void btnGaugeReadings_Click(System::Object^  sender, System::EventArgs^  e) {
                GaugeReadings ^ form = gcnew GaugeReadings;
                form->ShowDialog();
            }
private: System::Void btnSystemAlarm_Click(System::Object^  sender, System::EventArgs^  e) {
                SystemAlarm ^ form = gcnew SystemAlarm;
                form->ShowDialog();
            }
private: System::Void btnClockMenu_Click(System::Object^  sender, System::EventArgs^  e) {
                ClockMenu ^ form = gcnew ClockMenu;
                form->ShowDialog();
            }
private: System::Void btnUtilities_Click(System::Object^  sender, System::EventArgs^  e) {
                Utilities ^ form = gcnew Utilities;
                form->ShowDialog();
            }
};
}
```

Bell Mouth

```
//BELLMOUTH//

#pragma once
#include <ctime> //for time
```

```
#include <cstdlib> //for srand and rand
#include <string.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <fstream>

using namespace std;

namespace KoyoGui {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for BellMouth
	/// </summary>
	public ref class BellMouth : public System::Windows::Forms::Form
	{
	public:
		BellMouth(void)
		{
			InitializeComponent();


			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~BellMouth()
		{
			if (components)
			{
				delete components;
			}
		}

	private: System::Windows::Forms::TextBox^  txtIDA1;
	private: System::Windows::Forms::TextBox^  txtIDA3;
```

```
	protected:


	private: System::Windows::Forms::TextBox^  txtIDA2;

	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::Label^  label4;
	private: System::Windows::Forms::Label^  label5;
	private: System::Windows::Forms::TextBox^  txtTaper;
	private: System::Windows::Forms::TextBox^  txtBellMouth;
	private: System::Windows::Forms::DataVisualization::Charting::Chart^  chart1;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Timer^  timer1;
	private: System::Windows::Forms::Label^  label6;
	private: System::Windows::Forms::Label^  label7;
	private: System::ComponentModel::IContainer^  components;



	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>


#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			System::Windows::Forms::DataVisualization::Charting::ChartArea^  chartArea1
= (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());
			System::Windows::Forms::DataVisualization::Charting::Legend^  legend1 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Legend());
			System::Windows::Forms::DataVisualization::Charting::Series^  series1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
			this->txtIDA1 = (gcnew System::Windows::Forms::TextBox());
			this->txtIDA3 = (gcnew System::Windows::Forms::TextBox());
			this->txtIDA2 = (gcnew System::Windows::Forms::TextBox());
			this->label1 = (gcnew System::Windows::Forms::Label());
			this->label2 = (gcnew System::Windows::Forms::Label());
			this->label3 = (gcnew System::Windows::Forms::Label());
			this->label4 = (gcnew System::Windows::Forms::Label());
			this->label5 = (gcnew System::Windows::Forms::Label());
```

```
                this->txtTaper = (gcnew System::Windows::Forms::TextBox());
                this->txtBellMouth = (gcnew System::Windows::Forms::TextBox());
                this->chart1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
                this->button1 = (gcnew System::Windows::Forms::Button());
                this->label6 = (gcnew System::Windows::Forms::Label());
                this->label7 = (gcnew System::Windows::Forms::Label());
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->chart1))-
>BeginInit();
                this->SuspendLayout();
                //
                // txtIDA1
                //
                this->txtIDA1->Enabled = false;
                this->txtIDA1->Location = System::Drawing::Point(162, 358);
                this->txtIDA1->Name = L"txtIDA1";
                this->txtIDA1->Size = System::Drawing::Size(100, 20);
                this->txtIDA1->TabIndex = 1;
                //
                // txtIDA3
                //
                this->txtIDA3->Enabled = false;
                this->txtIDA3->Location = System::Drawing::Point(162, 430);
                this->txtIDA3->Name = L"txtIDA3";
                this->txtIDA3->Size = System::Drawing::Size(100, 20);
                this->txtIDA3->TabIndex = 2;
                //
                // txtIDA2
                //
                this->txtIDA2->Enabled = false;
                this->txtIDA2->Location = System::Drawing::Point(162, 393);
                this->txtIDA2->Name = L"txtIDA2";
                this->txtIDA2->Size = System::Drawing::Size(100, 20);
                this->txtIDA2->TabIndex = 3;
                //
                // label1
                //
                this->label1->AutoSize = true;
                this->label1->Location = System::Drawing::Point(18, 365);
                this->label1->Name = L"label1";
                this->label1->Size = System::Drawing::Size(48, 13);
                this->label1->TabIndex = 4;
                this->label1->Text = L"OutDiam";
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(18, 400);
```

```
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(48, 13);
this->label2->TabIndex = 5;
this->label2->Text = L"MidDiam";
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(18, 437);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(40, 13);
this->label3->TabIndex = 6;
this->label3->Text = L"InDiam";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(394, 365);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(35, 13);
this->label4->TabIndex = 7;
this->label4->Text = L"Taper";
//
// label5
//
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(394, 400);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(54, 13);
this->label5->TabIndex = 8;
this->label5->Text = L"BellMouth";
//
// txtTaper
//
this->txtTaper->Enabled = false;
this->txtTaper->Location = System::Drawing::Point(477, 358);
this->txtTaper->Name = L"txtTaper";
this->txtTaper->Size = System::Drawing::Size(100, 20);
this->txtTaper->TabIndex = 9;
//
// txtBellMouth
//
this->txtBellMouth->Enabled = false;
this->txtBellMouth->Location = System::Drawing::Point(477, 393);
this->txtBellMouth->Name = L"txtBellMouth";
this->txtBellMouth->Size = System::Drawing::Size(100, 20);
this->txtBellMouth->TabIndex = 10;
//
```

```
                        // chart1
                        //
                        chartArea1->AxisX->MajorGrid->Interval = 0;
                        chartArea1->AxisX->ScaleBreakStyle->Enabled = true;
                        chartArea1->AxisX->ScaleBreakStyle->MaxNumberOfBreaks = 1;
                        chartArea1->AxisX->ScaleBreakStyle->StartFromZero =
System::Windows::Forms::DataVisualization::Charting::StartFromZero::No;
                        chartArea1->AxisX->ScaleView->MinSize = 13.4;
                        chartArea1->AxisX->ScaleView->SizeType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;
                        chartArea1->AxisY->IntervalAutoMode =
System::Windows::Forms::DataVisualization::Charting::IntervalAutoMode::VariableCount;
                        chartArea1->AxisY->MajorGrid->Interval = 0.011;
                        chartArea1->AxisY->MajorGrid->IntervalOffset = 0;
                        chartArea1->AxisY->MajorGrid->IntervalOffsetType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;
                        chartArea1->AxisY->MajorGrid->IntervalType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;
                        chartArea1->AxisY->Maximum = 13.54;
                        chartArea1->AxisY->Minimum = 13.5;
                        chartArea1->AxisY->ScaleBreakStyle->Enabled = true;
                        chartArea1->AxisY->ScaleBreakStyle->StartFromZero =
System::Windows::Forms::DataVisualization::Charting::StartFromZero::No;
                        chartArea1->AxisY->ScaleView->MinSize = 13.4;
                        chartArea1->AxisY->ScaleView->MinSizeType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;
                        chartArea1->AxisY->ScaleView->Position = 13.4;
                        chartArea1->AxisY->ScaleView->SizeType =
System::Windows::Forms::DataVisualization::Charting::DateTimeIntervalType::Number;
                        chartArea1->Name = L"ChartArea1";
                        this->chart1->ChartAreas->Add(chartArea1);
                        legend1->Name = L"Legend1";
                        this->chart1->Legends->Add(legend1);
                        this->chart1->Location = System::Drawing::Point(47, 37);
                        this->chart1->Name = L"chart1";
                        series1->ChartArea = L"ChartArea1";
                        series1->Legend = L"Legend1";
                        series1->Name = L"Series1";
                        this->chart1->Series->Add(series1);
                        this->chart1->Size = System::Drawing::Size(701, 274);
                        this->chart1->TabIndex = 11;
                        this->chart1->Text = L"chart1";
                        this->chart1->Click += gcnew System::EventHandler(this,
&BellMouth::chart1_Click);
                        //
                        // button1
                        //
                        this->button1->Location = System::Drawing::Point(658, 356);
```

```
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(75, 23);
            this->button1->TabIndex = 12;
            this->button1->Text = L"button1";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click += gcnew System::EventHandler(this,
&BellMouth::button1_Click);
            //
            // label6
            //
            this->label6->AutoSize = true;
            this->label6->Location = System::Drawing::Point(508, 437);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(35, 13);
            this->label6->TabIndex = 13;
            this->label6->Text = L"label6";
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Location = System::Drawing::Point(397, 436);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(90, 13);
            this->label7->TabIndex = 14;
            this->label7->Text = L"Bearing Gauged: ";
            //
            // BellMouth
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(787, 461);
            this->Controls->Add(this->label7);
            this->Controls->Add(this->label6);
            this->Controls->Add(this->button1);
            this->Controls->Add(this->chart1);
            this->Controls->Add(this->txtBellMouth);
            this->Controls->Add(this->txtTaper);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->txtIDA2);
            this->Controls->Add(this->txtIDA3);
            this->Controls->Add(this->txtIDA1);
            this->ForeColor = System::Drawing::SystemColors::ControlText;
            this->Name = L"BellMouth";
            this->Text = L"BellMouth";
```

```
                this->WindowState = System::Windows::Forms::FormWindowState::Maximized;
                this->Load += gcnew System::EventHandler(this, &BellMouth::BellMouth_Load);
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->chart1))-
>EndInit();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion

private: System::Void BellMouth_Load(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void chart1_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {

                DateTime datetime = DateTime::Now;
                this->label6-> Text = datetime.ToString();

                    ifstream inFile;
                    double Diam1=0.0,Diam2=0.0,Diam3=0.0,T=0.0,BM=0.0;

                    chart1->Series["Series1"]->Points->Clear();

                    inFile.open("BMData.txt");

                        inFile>>Diam1;
                        inFile>>Diam2;
                        inFile>>Diam3;

                        T = Diam1 - Diam3;
                        BM = (Diam1+Diam3)/(2 - Diam2);



                        txtIDA1->Text = Convert::ToString(Diam1);
                        txtIDA2->Text = Convert::ToString(Diam2);
                        txtIDA3->Text = Convert::ToString(Diam3);
                        txtTaper->Text = Convert::ToString(T);
                        txtBellMouth->Text = Convert::ToString(BM);

                this->chart1->Series["Series1"]->Points->AddXY("OutDiam",Diam1);
                this->chart1->Series["Series1"]->Points->AddXY("MidDiam",Diam2);
                this->chart1->Series["Series1"]->Points->AddXY("InDiam",Diam3);


            }
```

```
};
}
```

Limits

```
#pragma once
#include "ErrorForm.h"
#include <iostream>
#include <Windows.h>

using namespace std;
extern double Nmin, Nmax, TolMin, TolMax, RLow, RHigh;

namespace KoyoGui {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for Limits
	/// </summary>
	public ref class Limits : public System::Windows::Forms::Form
	{
	public:
		Limits(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Limits()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Label^  lblNominal;
```

```
private: System::Windows::Forms::Label^  lblPartTol;
protected:

protected:

private: System::Windows::Forms::Label^  label3;
private: System::Windows::Forms::Label^  lblNmin;
private: System::Windows::Forms::Label^  lblNmax;
private: System::Windows::Forms::Label^  lblTolMin;
private: System::Windows::Forms::Label^  lblTolMax;
private: System::Windows::Forms::Label^  lblRLow;
private: System::Windows::Forms::Label^  lblRHigh;
private: System::Windows::Forms::TextBox^  txtNmin;
private: System::Windows::Forms::TextBox^  txtTolMin;
private: System::Windows::Forms::TextBox^  txtNmax;
private: System::Windows::Forms::TextBox^  txtTolMax;

private: System::Windows::Forms::TextBox^  txtRLow;

private: System::Windows::Forms::TextBox^  txtRHigh;




private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;
        //Variables for Limits Form



private: System::Windows::Forms::TextBox^  txtTestnmin;
private: System::Windows::Forms::TextBox^  txtTestNmax;
private: System::Windows::Forms::TextBox^  textBox3;
private: System::Windows::Forms::TextBox^  textBox4;
private: System::Windows::Forms::TextBox^  textBox5;
private: System::Windows::Forms::TextBox^  textBox6;
private: System::Windows::Forms::Button^  button1;
private: System::Windows::Forms::Button^  btnStore;



#pragma region Windows Form Designer generated code
```

```
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->lblNominal = (gcnew System::Windows::Forms::Label());
			this->lblPartTol = (gcnew System::Windows::Forms::Label());
			this->label3 = (gcnew System::Windows::Forms::Label());
			this->lblNmin = (gcnew System::Windows::Forms::Label());
			this->lblNmax = (gcnew System::Windows::Forms::Label());
			this->lblTolMin = (gcnew System::Windows::Forms::Label());
			this->lblTolMax = (gcnew System::Windows::Forms::Label());
			this->lblRLow = (gcnew System::Windows::Forms::Label());
			this->lblRHigh = (gcnew System::Windows::Forms::Label());
			this->txtNmin = (gcnew System::Windows::Forms::TextBox());
			this->txtTolMin = (gcnew System::Windows::Forms::TextBox());
			this->txtNmax = (gcnew System::Windows::Forms::TextBox());
			this->txtTolMax = (gcnew System::Windows::Forms::TextBox());
			this->txtRLow = (gcnew System::Windows::Forms::TextBox());
			this->txtRHigh = (gcnew System::Windows::Forms::TextBox());
			this->txtTestnmin = (gcnew System::Windows::Forms::TextBox());
			this->txtTestNmax = (gcnew System::Windows::Forms::TextBox());
			this->textBox3 = (gcnew System::Windows::Forms::TextBox());
			this->textBox4 = (gcnew System::Windows::Forms::TextBox());
			this->textBox5 = (gcnew System::Windows::Forms::TextBox());
			this->textBox6 = (gcnew System::Windows::Forms::TextBox());
			this->btnStore = (gcnew System::Windows::Forms::Button());
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// lblNominal
			//
			this->lblNominal->AutoSize = true;
			this->lblNominal->Location = System::Drawing::Point(21, 32);
			this->lblNominal->Name = L"lblNominal";
			this->lblNominal->Size = System::Drawing::Size(73, 13);
			this->lblNominal->TabIndex = 0;
			this->lblNominal->Text = L"Nominal Sizes";
			this->lblNominal->Click += gcnew System::EventHandler(this,
&Limits::label1_Click);
			//
			// lblPartTol
			//
			this->lblPartTol->AutoSize = true;
			this->lblPartTol->Location = System::Drawing::Point(21, 121);
			this->lblPartTol->Name = L"lblPartTol";
			this->lblPartTol->Size = System::Drawing::Size(77, 13);
```

```
            this->lblPartTol->TabIndex = 1;
            this->lblPartTol->Text = L"Part Tolerance";
            this->lblPartTol->Click += gcnew System::EventHandler(this,
&Limits::label2_Click);
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(21, 215);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(95, 13);
            this->label3->TabIndex = 2;
            this->label3->Text = L"Reasonalble Limits";
            this->label3->Click += gcnew System::EventHandler(this, &Limits::label3_Click);
            //
            // lblNmin
            //
            this->lblNmin->AutoSize = true;
            this->lblNmin->Location = System::Drawing::Point(21, 78);
            this->lblNmin->Name = L"lblNmin";
            this->lblNmin->Size = System::Drawing::Size(31, 13);
            this->lblNmin->TabIndex = 3;
            this->lblNmin->Text = L"Nmin";
            //
            // lblNmax
            //
            this->lblNmax->AutoSize = true;
            this->lblNmax->Location = System::Drawing::Point(159, 78);
            this->lblNmax->Name = L"lblNmax";
            this->lblNmax->Size = System::Drawing::Size(34, 13);
            this->lblNmax->TabIndex = 4;
            this->lblNmax->Text = L"Nmax";
            //
            // lblTolMin
            //
            this->lblTolMin->AutoSize = true;
            this->lblTolMin->Location = System::Drawing::Point(21, 166);
            this->lblTolMin->Name = L"lblTolMin";
            this->lblTolMin->Size = System::Drawing::Size(39, 13);
            this->lblTolMin->TabIndex = 5;
            this->lblTolMin->Text = L"TolMin";
            this->lblTolMin->Click += gcnew System::EventHandler(this,
&Limits::label6_Click);
            //
            // lblTolMax
            //
            this->lblTolMax->AutoSize = true;
            this->lblTolMax->Location = System::Drawing::Point(159, 166);
```

```
            this->lblTolMax->Name = L"lblTolMax";
            this->lblTolMax->Size = System::Drawing::Size(42, 13);
            this->lblTolMax->TabIndex = 6;
            this->lblTolMax->Text = L"TolMax";
            this->lblTolMax->Click += gcnew System::EventHandler(this,
&Limits::label7_Click);
            //
            // lblRLow
            //
            this->lblRLow->AutoSize = true;
            this->lblRLow->Location = System::Drawing::Point(21, 260);
            this->lblRLow->Name = L"lblRLow";
            this->lblRLow->Size = System::Drawing::Size(35, 13);
            this->lblRLow->TabIndex = 7;
            this->lblRLow->Text = L"RLow";
            this->lblRLow->Click += gcnew System::EventHandler(this,
&Limits::lblRLow_Click);
            //
            // lblRHigh
            //
            this->lblRHigh->AutoSize = true;
            this->lblRHigh->Location = System::Drawing::Point(159, 260);
            this->lblRHigh->Name = L"lblRHigh";
            this->lblRHigh->Size = System::Drawing::Size(37, 13);
            this->lblRHigh->TabIndex = 8;
            this->lblRHigh->Text = L"RHigh";
            //
            // txtNmin
            //
            this->txtNmin->Location = System::Drawing::Point(25, 52);
            this->txtNmin->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtNmin->Name = L"txtNmin";
            this->txtNmin->Size = System::Drawing::Size(100, 19);
            this->txtNmin->TabIndex = 9;
            this->txtNmin->Click += gcnew System::EventHandler(this,
&Limits::txtNmin_Click);
            this->txtNmin->TextChanged += gcnew System::EventHandler(this,
&Limits::txtNmin_TextChanged);
            //
            // txtTolMin
            //
            this->txtTolMin->Location = System::Drawing::Point(25, 142);
            this->txtTolMin->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtTolMin->Name = L"txtTolMin";
            this->txtTolMin->Size = System::Drawing::Size(100, 19);
            this->txtTolMin->TabIndex = 10;
            this->txtTolMin->Click += gcnew System::EventHandler(this,
&Limits::txtTolMin_Click);
```

```
            this->txtTolMin->TextChanged += gcnew System::EventHandler(this,
&Limits::txtTolMin_TextChanged);
            //
            // txtNmax
            //
            this->txtNmax->Location = System::Drawing::Point(163, 52);
            this->txtNmax->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtNmax->Name = L"txtNmax";
            this->txtNmax->Size = System::Drawing::Size(100, 19);
            this->txtNmax->TabIndex = 11;
            this->txtNmax->Click += gcnew System::EventHandler(this,
&Limits::txtNmax_Click);
            this->txtNmax->TextChanged += gcnew System::EventHandler(this,
&Limits::txtNmax_TextChanged);
            //
            // txtTolMax
            //
            this->txtTolMax->Location = System::Drawing::Point(163, 142);
            this->txtTolMax->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtTolMax->Name = L"txtTolMax";
            this->txtTolMax->Size = System::Drawing::Size(100, 19);
            this->txtTolMax->TabIndex = 12;
            this->txtTolMax->Click += gcnew System::EventHandler(this,
&Limits::txtTolMax_Click);
            this->txtTolMax->TextChanged += gcnew System::EventHandler(this,
&Limits::txtTolMax_TextChanged);
            //
            // txtRLow
            //
            this->txtRLow->Location = System::Drawing::Point(25, 235);
            this->txtRLow->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtRLow->Name = L"txtRLow";
            this->txtRLow->Size = System::Drawing::Size(100, 19);
            this->txtRLow->TabIndex = 13;
            this->txtRLow->Click += gcnew System::EventHandler(this,
&Limits::txtRLow_Click);
            this->txtRLow->TextChanged += gcnew System::EventHandler(this,
&Limits::txtRLow_TextChanged);
            //
            // txtRHigh
            //
            this->txtRHigh->Location = System::Drawing::Point(163, 235);
            this->txtRHigh->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtRHigh->Name = L"txtRHigh";
            this->txtRHigh->Size = System::Drawing::Size(100, 19);
            this->txtRHigh->TabIndex = 14;
            this->txtRHigh->Click += gcnew System::EventHandler(this,
&Limits::txtRHigh_Click);
```

```
            this->txtRHigh->TextChanged += gcnew System::EventHandler(this,
&Limits::txtRHigh_TextChanged);
            //
            // txtTestnmin
            //
            this->txtTestnmin->Enabled = false;
            this->txtTestnmin->Location = System::Drawing::Point(527, 32);
            this->txtTestnmin->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtTestnmin->Name = L"txtTestnmin";
            this->txtTestnmin->Size = System::Drawing::Size(100, 19);
            this->txtTestnmin->TabIndex = 15;
            //
            // txtTestNmax
            //
            this->txtTestNmax->Enabled = false;
            this->txtTestNmax->Location = System::Drawing::Point(527, 78);
            this->txtTestNmax->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->txtTestNmax->Name = L"txtTestNmax";
            this->txtTestNmax->Size = System::Drawing::Size(100, 19);
            this->txtTestNmax->TabIndex = 16;
            //
            // textBox3
            //
            this->textBox3->Enabled = false;
            this->textBox3->Location = System::Drawing::Point(527, 118);
            this->textBox3->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->textBox3->Name = L"textBox3";
            this->textBox3->Size = System::Drawing::Size(100, 19);
            this->textBox3->TabIndex = 17;
            //
            // textBox4
            //
            this->textBox4->Enabled = false;
            this->textBox4->Location = System::Drawing::Point(527, 166);
            this->textBox4->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->textBox4->Name = L"textBox4";
            this->textBox4->Size = System::Drawing::Size(100, 19);
            this->textBox4->TabIndex = 18;
            //
            // textBox5
            //
            this->textBox5->Enabled = false;
            this->textBox5->Location = System::Drawing::Point(527, 215);
            this->textBox5->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
            this->textBox5->Name = L"textBox5";
            this->textBox5->Size = System::Drawing::Size(100, 19);
            this->textBox5->TabIndex = 19;
            //
```

```
                        // textBox6
                        //
                        this->textBox6->Enabled = false;
                        this->textBox6->Location = System::Drawing::Point(527, 260);
                        this->textBox6->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
                        this->textBox6->Name = L"textBox6";
                        this->textBox6->Size = System::Drawing::Size(100, 19);
                        this->textBox6->TabIndex = 20;
                        //
                        // btnStore
                        //
                        this->btnStore->Location = System::Drawing::Point(304, 279);
                        this->btnStore->Margin = System::Windows::Forms::Padding(3, 2, 3, 2);
                        this->btnStore->Name = L"btnStore";
                        this->btnStore->Size = System::Drawing::Size(147, 34);
                        this->btnStore->TabIndex = 21;
                        this->btnStore->Text = L"GOGOSTOREVAR";
                        this->btnStore->UseVisualStyleBackColor = true;
                        this->btnStore->Click += gcnew System::EventHandler(this,
&Limits::btnStore_Click);
                        //
                        // button1
                        //
                        this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif",
12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                                static_cast<System::Byte>(0)));
                        this->button1->Location = System::Drawing::Point(575, 310);
                        this->button1->Name = L"button1";
                        this->button1->Size = System::Drawing::Size(100, 49);
                        this->button1->TabIndex = 22;
                        this->button1->Text = L"BACK";
                        this->button1->UseVisualStyleBackColor = true;
                        this->button1->Click += gcnew System::EventHandler(this,
&Limits::button1_Click);
                        //
                        // Limits
                        //
                        this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                        this->ClientSize = System::Drawing::Size(687, 371);
                        this->Controls->Add(this->button1);
                        this->Controls->Add(this->btnStore);
                        this->Controls->Add(this->textBox6);
                        this->Controls->Add(this->textBox5);
                        this->Controls->Add(this->textBox4);
                        this->Controls->Add(this->textBox3);
                        this->Controls->Add(this->txtTestNmax);
                        this->Controls->Add(this->txtTestnmin);
```

```
                this->Controls->Add(this->txtRHigh);
                this->Controls->Add(this->txtRLow);
                this->Controls->Add(this->txtTolMax);
                this->Controls->Add(this->txtNmax);
                this->Controls->Add(this->txtTolMin);
                this->Controls->Add(this->txtNmin);
                this->Controls->Add(this->lblRHigh);
                this->Controls->Add(this->lblRLow);
                this->Controls->Add(this->lblTolMax);
                this->Controls->Add(this->lblTolMin);
                this->Controls->Add(this->lblNmax);
                this->Controls->Add(this->lblNmin);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->lblPartTol);
                this->Controls->Add(this->lblNominal);
                this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 7.8F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(0)));
                this->Margin = System::Windows::Forms::Padding(4);
                this->Name = L"Limits";
                this->Text = L"Limits";
                this->WindowState = System::Windows::Forms::FormWindowState::Maximized;
                this->Load += gcnew System::EventHandler(this, &Limits::Limits_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
        private: System::Void label1_Click(System::Object^  sender, System::EventArgs^  e) {
                 }
private: System::Void label2_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void label3_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void label6_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void label7_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void lblRLow_Click(System::Object^  sender, System::EventArgs^  e) {
            }
private: System::Void txtNmin_TextChanged(System::Object^  sender, System::EventArgs^  e) {

                double iRetVal;
                 iRetVal = validatekeypress(txtNmin->Text);

                 if(iRetVal > -9999.9)
                        Nmin = iRetVal;
                 else
```

```
                {
                  txtNmin->Text = Convert::ToString(Nmin);
                        txtNmin->SelectAll();
                        Nmin = 0;
                }

        }
private: System::Void txtNmax_TextChanged(System::Object^  sender, System::EventArgs^  e) {
                double iRetVal;
                 iRetVal = validatekeypress(txtNmax->Text);

                 if(iRetVal > -9999.9)
                         Nmax = iRetVal;
                 else
                 {
                   txtNmax->Text = Convert::ToString(Nmax);
                         txtNmax->SelectAll();
                         Nmax = 0;
                 }

        }
private: System::Void txtTolMin_TextChanged(System::Object^  sender, System::EventArgs^  e) {
                        double iRetVal;
                 iRetVal = validatekeypress(txtTolMin->Text);

                 if(iRetVal > -9999.9)
                         TolMin = iRetVal;
                 else
                 {
                   txtTolMin->Text = Convert::ToString(TolMin);
                         txtTolMin->SelectAll();
                         TolMin = 0;
                 }


        }
private: System::Void txtTolMax_TextChanged(System::Object^  sender, System::EventArgs^  e) {
                double iRetVal;
                 iRetVal = validatekeypress(txtTolMax->Text);

                 if(iRetVal > -9999.9)
                         TolMax = iRetVal;
                 else
                 {
                   txtTolMax->Text = Convert::ToString(TolMax);
                         txtTolMax->SelectAll();
                         TolMax = 0;
                 }
```

```
		}
private: System::Void txtRLow_TextChanged(System::Object^  sender, System::EventArgs^  e) {
					double iRetVal;
				iRetVal = validatekeypress(txtRLow->Text);

				if(iRetVal > -9999.9)
					 RLow = iRetVal;
				else
				{
				  txtRLow->Text = Convert::ToString(RLow);
					txtRLow->SelectAll();
					RLow = 0;
				}

		}

private: System::Void txtRHigh_TextChanged(System::Object^  sender, System::EventArgs^  e) {
				double iRetVal;
				iRetVal = validatekeypress(txtRHigh->Text);

				if(iRetVal > -9999.9)
					 RHigh = iRetVal;
				else
				{
				  txtRHigh->Text = Convert::ToString(RHigh);
					txtRHigh->SelectAll();
					RHigh = 0;
				}

		}

private: System::Void btnStore_Click(System::Object^  sender, System::EventArgs^  e) {
				txtTestnmin->Text = Convert::ToString(Nmin);
				txtTestNmax->Text = Convert::ToString(Nmax);
				textBox3->Text = Convert::ToString(TolMin);
				textBox4->Text = Convert::ToString(TolMax);
				textBox5->Text = Convert::ToString(RLow);
				textBox6->Text = Convert::ToString(RHigh);

		}
private: System::Void Limits_Load(System::Object^  sender, System::EventArgs^  e) {
		}

private: double validatekeypress(System::String^ sControlText){
				 double iTest;
```

```
                                        if(Double::TryParse(sControlText,iTest))//verify itest is double
                                                return iTest;
                                        else
                                        {
                                        MessageBox::Show("You have typed and invalid value!!!","Your in the
DANGERZONE!!"); //you have entered a zone of danger
                                        return -9999.9; //use -9999.9 as a flag for invalid value
                                        }
                                        }


private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {

                                this->Close();
                        }

private: System::Void txtNmin_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
private: System::Void txtNmax_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
private: System::Void txtTolMin_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
private: System::Void txtTolMax_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
private: System::Void txtRLow_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
private: System::Void txtRHigh_Click(System::Object^  sender, System::EventArgs^  e) {
                                system("osk.exe");
                        }
};
}
```

Clock Menu

```
#pragma once
#include <string.h>
#include <stdlib.h>
#include <string>
#include <iostream>
#include <fstream>
using namespace std;
```

```
namespace KoyoGui {
	using namespace System::IO;
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for ClockMenu
	/// </summary>
	public ref class ClockMenu : public System::Windows::Forms::Form
	{
	private: System::Windows::Forms::Timer^  datetime;


	public:


		ClockMenu(void)
		{
			InitializeComponent();

			datetime -> Start();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~ClockMenu()
		{
			if (components)
			{
				delete components;
			}
		}

	private: System::Windows::Forms::Label^  dtime;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::ListBox^  listBox1;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Button^  button2;
```

```
        private: System::ComponentModel::IContainer^  components;
        protected:

        private:
                /// <summary>
                /// Required designer variable.
                /// </summary>


#pragma region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel::Container());
                        this->datetime = (gcnew System::Windows::Forms::Timer(this->components));
                        this->dtime = (gcnew System::Windows::Forms::Label());
                        this->label1 = (gcnew System::Windows::Forms::Label());
                        this->listBox1 = (gcnew System::Windows::Forms::ListBox());
                        this->button1 = (gcnew System::Windows::Forms::Button());
                        this->button2 = (gcnew System::Windows::Forms::Button());
                        this->SuspendLayout();
                        //
                        // datetime
                        //
                        this->datetime->Tick += gcnew System::EventHandler(this,
&ClockMenu::timer1_Tick);
                        //
                        // dtime
                        //
                        this->dtime->AutoSize = true;
                        this->dtime->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12,
System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                                static_cast<System::Byte>(0)));
                        this->dtime->Location = System::Drawing::Point(102, 31);
                        this->dtime->Name = L"dtime";
                        this->dtime->Size = System::Drawing::Size(57, 20);
                        this->dtime->TabIndex = 1;
                        this->dtime->Text = L"label1";
                        //
                        // label1
                        //
                        this->label1->AutoSize = true;
```

```
            this->label1->Location = System::Drawing::Point(85, 9);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(95, 13);
            this->label1->TabIndex = 2;
            this->label1->Text = L"Current Date/Time";
            this->label1->Click += gcnew System::EventHandler(this,
&ClockMenu::label1_Click);
            //
            // listBox1
            //
            this->listBox1->FormattingEnabled = true;
            this->listBox1->Location = System::Drawing::Point(39, 54);
            this->listBox1->Name = L"listBox1";
            this->listBox1->Size = System::Drawing::Size(293, 95);
            this->listBox1->TabIndex = 3;
            this->listBox1->SelectedIndexChanged += gcnew System::EventHandler(this,
&ClockMenu::listBox1_SelectedIndexChanged);
            //
            // button1
            //
            this->button1->Location = System::Drawing::Point(39, 155);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(75, 23);
            this->button1->TabIndex = 4;
            this->button1->Text = L"Clear";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click += gcnew System::EventHandler(this,
&ClockMenu::button1_Click_3);
            //
            // button2
            //
            this->button2->Location = System::Drawing::Point(120, 155);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(75, 23);
            this->button2->TabIndex = 5;
            this->button2->Text = L"Back";
            this->button2->UseVisualStyleBackColor = true;
            this->button2->Click += gcnew System::EventHandler(this,
&ClockMenu::button2_Click);
            //
            // ClockMenu
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(371, 403);
            this->Controls->Add(this->button2);
            this->Controls->Add(this->button1);
            this->Controls->Add(this->listBox1);
```

```
                        this->Controls->Add(this->label1);
                        this->Controls->Add(this->dtime);
                        this->Name = L"ClockMenu";
                        this->Text = L"Clock Menu";
                        this->WindowState = System::Windows::Forms::FormWindowState::Maximized;
                        this->Load += gcnew System::EventHandler(this,
&ClockMenu::ClockMenu_Load);
                        this->Shown += gcnew System::EventHandler(this,
&ClockMenu::ClockMenu_Shown);
                        this->ResumeLayout(false);
                        this->PerformLayout();

                    }
#pragma endregion
        private: System::Void dateTimePicker1_ValueChanged(System::Object^  sender,
System::EventArgs^  e) {
                         }
        private: System::Void ClockMenu_Load(System::Object^  sender, System::EventArgs^  e) {
                        StreamReader ^ sr = gcnew StreamReader("LOGTIME.txt");
                        String ^ temp;
                        while(temp = sr->ReadLine())
                         listBox1->Items->Add(temp);
                        sr->Close();
                         }
        private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e) {

                                DateTime datetime = DateTime::Now;
                           this -> dtime -> Text = datetime.ToString();
                         }
        private: System::Void label1_Click(System::Object^  sender, System::EventArgs^  e) {
                        }
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                    }
private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {

                    }
private: System::Void listBox2_SelectedIndexChanged(System::Object^  sender, System::EventArgs^  e) {
                    }
private: System::Void listBox1_SelectedIndexChanged(System::Object^  sender, System::EventArgs^  e) {

                    }
private: System::Void ClockMenu_Shown(System::Object^  sender, System::EventArgs^  e) {

                    }
private: System::Void button1_Click_2(System::Object^  sender, System::EventArgs^  e) {

                    }
private: System::Void button1_Click_3(System::Object^  sender, System::EventArgs^  e) {
```

```
                                        //read contents of the LOGTIME file
                                        std::ofstream ofs;
                                        ofs.open("LOGTIME.txt", std::ofstream::out | std::ofstream::trunc);
                                        ofs.close();

                                        listBox1->Items->Clear(); // clear list box
                        }
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
                                this->Close();
                        }
};
}
```