Project Number: SYS-SYS7

Real-Time Process Monitoring And Statistical Process Control For an Automated Casting Facility

A Major Qualifying Project Report

Submitted to the Faculty Of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

In Mechanical Engineering

By

Daniel Lettiere

Date: June 2012

Approved:

Prof. Satya Shivkumar, Advisor

Contents

Abstract

In the metal casting industry, defects increase cost of production, expand required labor hours, and decrease overall productivity. Better understanding of process variables allows for successful reduction of defects. Utilizing real-time data logging technology and statistical process control software creates the framework for an effective process monitoring system. To establish a correlation between mold temperature and casting defects, a unique data logging system, capable of withstanding high temperatures, was designed. The beneficial results of this project will not only impact the process today, but will help improve future innovations in the industry.

1. Introduction

Since the dawn of Industrial Revolution, the goal of manufacturing companies has been to produce high quantities of product while still being able to generate profit. Before computers and automated machines, these companies were forced to hire more and more laborers to meet production demands. With Henry Ford's invention of the assembly line, plants were able to build complex products faster. As technology advanced production methods did as well. The introduction of computers paved the way for manufacturing automation, allowing fewer laborers to perform at the same rate. As machine design improved, so did the need to control these machines. Gauges and meters on a panel with results being tallied on a clipboard worked, but this data would need to be analyzed and interpreted by an engineer. It wasn't until digital computers became small (about the size of a refrigerator) that they found a home in manufacturing as data gathering/analysis machines. These early data acquisition devices (DAQs) were simple but they worked. DAQ's also helped push machine design even further, by allowing engineers a closer look at the properties that were previously indiscernible to the human eye. Through years of research, brilliant scientists and engineers were able to correlate gathered data to properties of the material or the reaction of the machine.

Metal casting is not a simple process and many different systems and people interact with castings as they make their way from liquid wax to finished part. At each step of the process, data is gathered. One problem that engineers face when dealing with this data is what to do with it. Many times data is gathered and stored on servers never to be used again, while other times data is not gathered at all. Another problem that engineers face is scrap output caused by defects. The US foundry industry as a whole loses $1.5-3 billion a year in scrap and

rework. One reasoning for the high losses is the lack of process knowledge [Roshan et al.] The goal of this project is to develop a data analysis system to better understand this process knowledge. Leveraging existing infrastructure with new software technology, a web-based system will be put in place to analyze process data and provide process engineers with a graphical view of processes in near real-time. The data will be compared to baselines set by product engineers and will show any variability that is present. Process engineers can then make corrective actions before a small issue grows into a larger high-scrap producing problem.

Through constant monitoring of process information, engineers will be able to make improvements and changes that can optimize the process and increase overall productivity and reduce scrap output. This will lead to increased profits, efficiency, product throughput and even employee morale. To think that just some numbers on a computer somewhere can affect such a large change is amazing and makes us wonder what the manufacturing industry will look like in 10 years.

2. Background

2.1 Manufacturing of Investment Castings

The investment casting process is a multibillion dollar a year industry, producing critical parts for many top level industries. Although the demand for investment cast parts has skyrocketed over the past hundred years, the technique has remained relatively the same. Castings houses need to advance the process in order to remain relevant for the next hundred years.

2.1.1 Origins

The process of investment casting, also called lost-wax, was first developed in the early stages of the Bronze Age nearly 5000 years ago [Noble et al.]. These early castings were primarily bronze alloys containing copper and tin. These alloys were very primitive and their properties were not as well understood as they are in the present day. The investment casting process involves making a pattern of the desired part out of a material with a low melting point (wax) and then covering the wax model with clay. The clay is left to harden and then the assembly is heated enough to let the wax melt out from a hole left in the base of the clay. This leaves a negative impression of the wax part in the clay shell. This shell is then inverted and molten metal is poured into the shell via the vent hole. After the molten metal has solidified the clay shell is broken and an exact replica of the original wax part is produced. The lost-wax process has the unique ability to replicate very detailed parts, but it is not without its draw backs. As mentioned by Noble, if more than one copy of a part was desired a wax pattern had to be created for each copy.

The lost –wax process has been used throughout the ages and process has not changed much since the first cast. The materials used to produce the casting have improved as we gained a greater understanding of materials and their properties. Today modern investment casting firms use slurry consisting of fine silica particles suspended in a binder liquid. The wax parts are dipped in the slurry and then coated with stucco particles and left to dry. Depending on the size of the parts the molds may be dipped multiple times, as the shell must be strong enough to support the weight of the molten metal in the later stages of the process [Sidhu et al.].

In 1969 a family-run casting firm, Hitchiner Manufacturing Company, developed a new and innovative way to produce investment casting. The driving force behind this innovation was reducing cost. They found that they could save 40% of the cost associated with casting parts by simply inverting the mold and using vacuum pressure to draw metal into the mold [Spada]. This radical change propelled Hitchiner to become one of the largest producers of investment castings in North America. Hitchiner continued to develop new processes based of their Counter-Gravity casting method and as of 1994 had created 8 additional techniques, each one having a specific use for different alloys or parts. In order to continue growing as a company, Hitchiner constructed a brand new 90,000 ft^2 casting plant known as the Automated Casting Facility (ACF). To the layman, the plant was just another casting facility, but on the inside ACF is full of cutting edge technology aimed at producing high-volume castings [Wright]. Even with the amount of technology in place, scrap output could not be avoided. While only a fraction of the parts produced have defects, the goal of ACF (truly any manufacturer) is to eliminate defects

and increase production. Leveraging technology with engineering talent, the root cause of defects can be determined and improvements to the processes can be made.

2.1.2 Defects

Defects can occur in all methods of casting, but the types of defects differ. The most common defects in investment castings are shrinkage, followed by inclusions, gas porosity and cold shut. Other defects may occur but are much more sporadic. Solidification shrinkage is caused by the density of liquid metal being less dense than its solidified form. While it is possible to design around shrinkage, it is still common in investment casts.

Defects are undesirable for a number of reasons, mainly the cost associated with reworking or reproducing the defective parts. In a high volume environment, a common work around is to produce a certain percentage more parts than required so that after defective parts are removed the yield will be correct. This is a waste of materials, energy, and time. If engineers can determine the root cause of these defects, they can attempt to eliminate the problem and increase plant throughput. Hitchiner's approach to reducing defects is to reduce the level of operator interaction during the production cycle. The more controlled that the environmentally variables (i.e. temperature, humidity, time) are, there should be less variation in finished parts. To control variables is not a simple task however, and requires many different systems to monitor the environment and process parameters.

2.1.3 Effects of Humidity

Humidity plays a major role in the production of investment castings. After the shell is dipped in slurry and stucco, it must dry before it moved to the next stage of the process. The

time required to dry the shell is affected by the humidity and temperate of the drying room. The room must be maintained at a constant humidity to ensure even drying. If the humidity is too low the shell can crack because the outer layers dry faster than the inner layers and compressive stress at the surface causes cracks to form. If the humidity is too high, the shell may take too long to dry and moisture can become trapped in the shell. While it does not have a pronounced effect during drying, when the shell is sent to the burnout oven to remove the wax patterns the moisture will expand to steam and can cause cracks if not vented properly. Many casting houses have automated systems to maintain the humidity at a set level [Chakrabarti]. Humidity is also a major contributor to inclusion defects in the cast part [Roshan et al.].

2.1.4 Effects of Temperature

The temperature of the shell drying environment is also critical. Chakrabarti noted that the effect of temperature is quite important when combined with the relative humidity level. He observed that when shells were dried at low temperatures and high humidity the modulus of rupture (MOR) was higher than that of shells dried at high temperatures and low humidity levels. This can be attributed to the rate of evaporation, which is a function of temperature and humidity, as well as other environmentally variables. The temperature of the drying room, just like the humidity, is carefully controlled in an effort to prevent cracking the shells.

2.1.5 Effects of Time

The time that shells spend drying and the time it takes to move shells to the foundry must also be controlled. If shells spend too long in the drying room they can become too dry and brittle. If this occurs the shells become difficult to handle and must be carefully inspected

before being cast. The shells must be moved from drying to the burnout oven quickly, as the longer they remain in an uncontrolled environment, the greater the effect of the natural temperature and humidity. This is most evident when weather patterns bring high humidity and temperature with them. Hitchiner has observed greater defect rates when the weather is adverse, especially for extended periods of time.

2. Process Monitoring

The ultimate purpose of manufacturing data is to improve the quality and productivity while reducing cost [Bunkofske et al.]. Using a combination of sensors, data acquisition devices (DAQs), programmable logic controllers (PLCs), computers/servers, and software a system can be designed to provide engineers with the data that they need to drive development of existing processes and spur the invention of new ones. These systems are simple in their design but elegantly complex is their function.

A process monitoring system is built on the capabilities of four separate systems working together [Bunkofske et al.]: The process tool or the machine that is performing the operation, the statistical process control and data analysis system, real-time process monitoring system (data acquisition system), and the human interface (computer or display panel on a machine). If any one of these systems is not working properly, engineers can make the call to continue work “blind” or stop production until the system is fixed. This determination is based primarily on the variability of the process and the desired amount of process knowledge.

2.2.1 Real-Time Process Monitoring (RTPM)

Arguably the most useful way to monitor data is in real-time (or as near real-time as possible). In order to do this, data must be acquired and stored rapidly. Early data loggers stored information on rolls of paper for later analysis, but with modern advancements in networking, data can now be gathered and stored in databases within a matter of milliseconds. This speed is what makes RTPM so useful, as corrective measures can be taken early before larger problems arise. Another useful feature of RTPM is the ability to gather multiple data streams simultaneously. The major limiting factor here is the number of physical inputs on the DAQ. This can easily be fixed should this limit arise but simply replacing the DAQ with a large one or by using an integrated circuit called a shift register. Shift registers allow a DAQ to poll at least 16 different channels while only requiring 4 DAQ I/O's. This simple circuit can expand a systems potential at least twofold for less than five dollars. The only problem that engineers will face then is "What else should I monitor?"

2.2.2 Statistical Process Control (SPC)

All the data that the DAQs collect is useless without performing some analysis. Statistical Process Control (SPC) methods can be used to analyze this data at every stage of production [Motorcu et al.]. SPC is widely used in manufacturing as it provides a "simple" visual representation of process data. SPC is made up of seven basic tools, first emphasized by Kaoru Ishikawa [Tague]. These tools are: Control Charts, Histograms, Pareto Charts, Scatter Diagrams, Fishbone Diagrams, Check Sheets, and Stratification. By using some or all of these techniques, an engineer should be able to improve the quality of final products. The most common tool is the Control Chart, which was developed by W.A. Shewhart in the 1920's. Control Charts graph

data as a function of time, where the average of the data is forming the midline and the upper and lower set limits being 3σ and -3σ respectively. These are excellent at demonstrating when process data that contains variation. Many SPC software packages can even send email alerts or sound warning alarms if the data is out of the specified range (most cases $>1\sigma$). For investment casting a typical control chart would contain temperature of the ovens or drying room, although it can include any data not just temperature. Control charts are also unique as they can be constantly updating with new data and while many charts can do this, they cannot show trends nearly as well as control charts.

SPC tools can be used manually via Excel Plugins (i.e. QIMacros) or automatically via standalone programs or SQL Server Add-Ons. The advantage of using an automatically updating system is the continuity that can be achieved. When charts are generated via Excel, they can only contain the data that was stored prior to creating the chart. These “manual” charts can be useful if an engineer is looking to analyze a specific data set or would like to make modifications to the chart. SQL based SPC packages pull data directly from the SQL Server and analyze it on the fly, as new data gets added, the charts add the new data on their own. This type of SPC package would be most desirable for a general monitoring system, where perhaps the charts were displayed on a large screen so many people can see any wayward trends and start he action chain to alleviate the problem. These SPC systems can be accessed locally via a dedicated computer or remotely via a web browser (increasing in popularity).

2.2.3 Web-Based Data Interface

Before the boom in low cost PCs many companies could only afford computers for dedicated purposes and it was rare to see a PC on an employee's desk. As prices started to decrease and employers began to see the value computers could have in the work place, PCs found their way to everyone's desk. While the early computers could not do nearly what they can today, they were still incredibly powerful tools. The only downside at the time was the networking capabilities, as Local Area Networks (nearly a utility today) were still in their primitive form. Modems could be used to connect to remote resources. The caveat of this system was that a company would have to have an individual phone line and modem for each simultaneous user [Nahale]. This meant 2 engineers could not connect to the same location and view data. This may or may not have been a problem depending on the size of the company. As technology advanced, Ethernet-based networks became more readily availed and "easy" to install (relative to modem based networks). Ethernet networks have one major advantage over modem networks; they can support multiple simultaneous users. This feature allows for multiple engineers to all connect to the same resources without encountering any of the delays that plagued modems. A web-based data interface is ideal for a manufacturing environment as anyone (with permissions) can view the page. Since the page can be permission controlled, system admins need not worry about someone accidently deleting files or corrupting the data. For some companies web-based interfaces are used to allow engineers to work on the floor, away from their desktop, from a laptop, tablet, or mobile device. This freedom of movement can provide the engineer with a unique insight to the process as they can see the data and the process at the same time.

3. Objectives

- Upgrade existing data collection infrastructure to ensure the precision of collected data
- Develop data collection system capable of withstanding conditions inside 1800°F oven
- Correlate scrap defects to temperature gradients in oven
- Design and implement a web-based front-end to monitor and analyze process parameter data
- Implement display system for each departments critical process variables

4. Methodology

4.1 Real Time Process Monitoring

The investment casting process is a complex process that must be carefully controlled to produce the highest quality castings. A great deal of sensing equipment from thermocouples and rheostats to viscometers and scales are used to make observations throughout the process. These devices all operate at a certain polling frequency ranging from 1 sample per minute to 1 sample every 4 hours. While it would be possible to poll every device every second, this would result in huge databases and decreased analysis performance. Since the devices used in this system all poll at different rates, the "Real Time" nature of this monitoring system is most evident in the fact that data gathered and stored electronically and instantly, no need for an operator to transcribe observations made on paper. This means the system always has the most current data available.

4.1.1 Critical Process Variables

It is crucial to identify the process parameters that have the biggest impact on the process itself. For this system to provide useful feedback, the Critical Process Variables (CPV's) had to be carefully chosen. The system has two different types of CPV's, process driven and part driven. Process driven CPV's are variables that are applied to every mold regardless of its part while part driven CPV's are variables that are changed dependent on each parts work instructions. For example, the humidity in the drying room is process driven as every mold is subjected to the same humidity, while the melt temperature that a mold is cast at is different depending on the part specifications. For the most part, all of the process variables that will be

monitored by this system will be process driven, as part driven variables are more closely monitored by operators throughout the process. A listing of each department's CPV's that will be monitored, can be seen in the table below.

Wax	Shell	Foundry
Distribution Loop Temperatures	Slurry Viscosity	Mold Temperature
	Slurry Temperature	
	Slurry Plate Weight	
	Slurry Zahn5 Value	Melt Temperature
	QC Temperature	
	QC Humidity	

4.1.2 Centralization of Collected Data

The existing data collection system has been working and functioning the same for the past 10 years. Each department has an individual computer out on the floor that monitors the process in that area. The data is stored locally on each machine and is not readily accessible by engineers. For this system to function, the data must be housed in one central SQL database. Since each computer is gathering data via a VijeoCitect application the most logical choice for database consolidation is implementing VijeoHistorian. VijeoHistorian is a SQL based data warehousing application. It can easily be configured to seamlessly pull process data from selected VijeoCitect applications. By implementing the VijeoHistorian system, the Statistical Process Control (SPC) software will be able to access the process data in one location and not have to query each individual machine. While it would not be impossible to query each VijeoCitect application machine, many of them are running older hardware that would cause a significant decrease in performance when compared to running the same queries against a high power SQL server with the same information.

4.1.3 Data Cleaning

Each VijeoCitect application has been in place for many years and has been changed multiple times during the course of other projects. The current applications needed to be cleaned before they could be linked to the VijeoHistorian. The reason for cleaning before is to keep the VijeoHistorian as neat and streamlined as possible. This will make it easier for future projects to access the data and to know what everything is. This will also have a twofold effect on the network as it will keep the size of the database under control and reduce network bandwidth usage as useless tables and data will not be transmitted to the VijeoHistorian database.

4.2 Statistical Process Control

There are many statistical analysis tools that could be used to analyze the captured process data. Many of these packages would suffice to analyze the data and are used by engineers at Hitchiner before this project. They all have the limitation of requiring user input to process data. Three different packages; Qi Macros (Excel Add-In), Minitab (standalone), and Statistica QC (standalone) were identified as possible analysis packages. QIMacros was eliminated as Hitchiner did not want to rely on Excel for this project. Minitab was also ruled out as it does not possess the automation flexibility that was needed for this system. The Statistica QC software package, developed by StatSoft, was chosen. Statistica is a standalone application that contains powerful automation capabilities and rich API allowing for other software packages to interface with the Statistica client.

4.2.1 Statistica QC Software

As mentioned before Statistica QC is a statistical analysis package developed by StatSoft, leaders in the statistical analysis software industry for years. Statistica is a powerful tool that can perform nearly any type of statistical analysis imaginable. These analyses can be performed as spreadsheets, charts, or graphs, allowing for a greater amount of flexibility in terms of displaying output for users.

Another feature that solidifies Statistica's role in this project is the ability to reference not only local data but also SQL databases, which is how the VijeoHistorian (as well as Hitchiner's ERP, IQMS) stores data. Although the Statistica QC package is a desktop standalone application, its connectivity features make it useful as a networked analysis tool.

Statistica QC will be playing a crucial role in this system. It will be used to generate statistical process control charts and Runs Tests analysis. This can be accomplished in one of two ways, either manually or programmatically. Since the goal of the project is to automate these analyses Statistica, the system will use Statistica Visual Basic (SVB) macros to call analysis routines on a schedule. Macros are created one of two ways, either by hand using syntax guidelines, or using Statistica's built-in macro recorder. This system will be programmed using the macro recorder to lay a foundation and then customizing the recorded code.

Statistica will provide outputs of graphs as PNG files, and spreadsheets can be output as HTML files. This makes it simple to link these outputs to a webpage for display. The only spreadsheets that will be output are results of Western Electric Runs Rules, a set of rules that can be used to determine trending patterns or out of control process data. The results output

from Statistica will be stored on Hitchiner's servers, allowing the webpage to pull the results from a static file location. This means the webpage will only have to be coded once and Statistica will just change the result files as it processes the latest data.

While the Statistica QC package will be able to handle the workload of this project, if future capabilities are desired, the system is easily upgradable to include Statistica modules such as Web Based Analysis Tools, Data Acquisition and Data Mining. StatSoft also makes modules that can be used in other aspects of business, not just manufacturing.

4.3 Web Based Display System

The data that is processed through Statistica must be accessible to anyone at the company. To meet this requirement, an internal webpage was created to make it easy to access the image files with only a web browser. The webpage must be accessible from a direct link or via an icon in IQMS, Hitchiner's existing ERP software.

4.3.1 Main Page

The webpage will be divided into 3 departments: Wax, Shell, and Foundry. Each department is split into sections titled with the graph or spreadsheet that is generated, providing large and clear sections that will be visible from a distance. This is critical as the web page will be displayed not just on engineer's computers but also on large screens in each department.

From the main page, each section is marked with visual indicators in the form of red, yellow, and green backgrounds when process data violates different Western Electric runs tests. These indicators will make it very easy for operators and engineers to see when there is a problem with the process and to make corrective actions.

The webpage must be simple and easy to change should the process change or be modified. Since the page is written in Hypertext Markup Language (HTML), edits can be made without any special software. HTML is easy to write and even easier to modify once the framework is laid.

4.3.2 Main Page Features

The main page will be where most users will land, so it must have the most current data at all times. Since Statistica runs its macros every 15 minutes, the page needs to refresh automatically. While it would be easy to have the page refresh every 15 minutes, this page must be set to refresh every 15 seconds. The reason for this is that data is processed nearly continually over the 15 minutes between the start and finish of the Statistica macros. By refreshing so frequently, the visual indicators will always be indicating the latest data.

4.3.3 Departmental Display Pages

The main page displays all the different departments at one time. Each department needs an individual page to display the data that pertains to that department. Large monitors will be mounted in a central location for each department powered by small Thin Client computers. Since the displays will not have user input devices, such as mice or keyboards, attached they departmental pages must be able to continually cycle through all the graph and spreadsheet results for that department. This is accomplished by simply putting the individual graph and spreadsheet HTML pages inside of an IFrame that is programmed to switch pages every 15 seconds. Each individual page is set to refresh every 5 seconds and the cycle page is set to refresh approximately at the end of the cycle, bringing it back to the beginning displaying any new data.

4.3.2 Bandwidth and Storage Considerations

Since Statistica will be storing data on a shared directory on Hitchiner's network, it is important to keep file and folder sizes under control. The graphs and spreadsheets that are displayed via the web are continually overwritten so they don't pile up on the server. When the webpage is refreshed, new images and spreadsheets must be transmitted to the requesting browser. Transferring images over a network uses a lot of bandwidth every time a page is refreshed. Since this systems pages auto-refresh constantly, it would be assumed that data would be constantly flowing to computers requesting the pages. To avoid having so many transfers constantly taking place, the pages are set to allow caching by the browser. Caching works by saving images and spreadsheets from the page locally. When the page refreshes, the browser first checks if the file changed, if it did it grabs the latest copy, if it doesn't then it uses the locally stored copy of the image. This will speed up page performance and provide users with the smoothest experience possible.

4.4 Thermal Survey of Oven

Throughout the investment casting process the shells go through many different areas. The environmental conditions in these areas are monitored carefully as they can have a significant effect on the shells if they are not controlled. Most of the stages of the process are relatively easy to monitor with simple equipment and techniques, and are already configured and acquiring information. The last step before the molds are cast, pre-heat, is the most difficult to monitor and possibly the most critical. Monitoring the pre-heat oven presents some unique challenges that must be designed for.

4.4.1 Oven Design

The pre-heat oven at Hitchiner's Automated Casting Facility is called G-Oven. The oven was custom built for Hitchiner and is the only one of its kind. The oven is 100 feet long and opens on both ends allowing carts to be loaded in one side and removed from the other. The oven is a continuous loop containing 24 mold carts. There are two sets of tracks in the oven with a thermal barrier separating the halves. There is also a third track, outside the oven, that is used for cooling the carts and then reloading them for the next run. The carts ride through the oven for about 2 hours to heat the molds to the target temperature of 1800±25°F.

4.4.2 Design Constraints

The continuous nature of this oven presents a unique engineering problem, "How do we gather temperature data from the molds as they ride through the oven?" If just one mold was to be monitored, a long length of ceramic thermocouple wire could be run through the oven and then pulled out the other side, but for this project the goal is to monitor 35 type K

thermocouples. It would be quite difficult, dangerous and expensive to create a run of 35, 100 foot long thermocouple cables, so the data logger has to ride with the cart as it travels through the oven. The data logger would have to be thermally insulated to protect it from the extreme temperatures in the oven. Fortunately, the underside of each mold cart only reaches 500 F, meaning the data logger has to resist less heat transfer. The major design requirements of the data logger are as follows:

- Must be able to withstand 500 F temperature found under mold cart.
- Must be able to last up to 6 hours at 500 F.
- Must be able to sample 35 Type K Thermocouples for the entire time the logger is in the oven.
- Must be a standalone, battery powered system.
- Thermal protection system must not extend past the bounds of the underside of the cart.
- Thermal protection system must be removable from the cart for future use in other ovens.
- Thermal protection system must keep electronics at or below 85 C for at least 6 hours.

4.4.3 Custom Built Data Logger Design

Since a data logger with the desired capabilities does not already exist, a custom device has to be built. To meet the design requirements for the data logger, the system will be built on the Arduino platform. The Arduino Uno R3 is low cost, easy to program, low power, and highly expandable. This was chosen as the base not only because of the aforementioned benefits, but

also due to the fact that OceanControls.com.au, based in Australia, makes Type K thermocouple breakout shield (KTA-259) for the Arduino. Each shield will provide 8 thermocouple inputs. By connecting 5 shields in parallel, the system can monitor 35 type K thermocouples. The data will be logged to a microSD card that is also connected in a parallel to the Arduino. The data logger must be designed first as the thermal protection system will be built around the logger.

4.4.3 Thermal Protection System

To protect the data logger from the heat found under the cart, a unique design must be implemented. The data logger will be housed inside a water filled shell, fitted with a steam vent. The water jacketed design will use the high heat capacity of water to absorb heat from the oven and use it to boil off the water in the jacket. As long as there is water in the jacket, the inner chamber will not get above 100 C. Since the thermal limit of the data logger circuitry is only 85 C it will be critical to provide a large enough volume of water to capture the heat without letting it boil within 6 hours.

4.4.4 Water Jacket Design

The size of the water jacket was driven by the available space underneath the mold cart. Utilizing all of the available space under the cart provides just over 5 gallons of water to protect the logger. The jacket is designed to be a chamber within a chamber with an opening on one side for the logger to slide in. The water jacket also is fitted with a stainless steel steam vent to allow air to escape as the water heats up and potentially boils. Without the vent the system would become pressurized and explode, resulting in damaged equipment and injured personnel. The water jacket and every other metal component of the logger system must be

made of stainless steel as humidity caused by the steam, coupled with the extreme heat, would cause rapid oxidation of carbon steels.

4.5 G-Oven Thermal Survey Experimental Design

In order to obtain useful process information about the oven, an experiment must be designed to relate ambient air temperature in the oven to mold temperatures and defect output. To accomplish this, the mold cart will be fitted with 9 stainless steel rods; each fitted with 3 adjustable heights Type K armored thermocouples. The adjustable height is needed in order to set the height of the ambient air thermocouples equal to the vertical position of the thermocouples that will be embedded in the molds. Each of the 4 molds on the cart will be outfitted with 3 Type K thermocouples, inserted through channels built onto the parts during the wax assembly stage.

To relate the vertical temperature gradient to defects in the cast part, the parts on the sprue must be categorized for later analysis. Since the parts are all the same, it is critical to split them into sub groups. To make it easier to select these sub groups, wax patterns are marked with a raised wax letter from A to F. Each letter group will be place in rows on the sprue with matching letters. The lettering will allow for defect analysis with respect to height in oven. This data can be correlated to the temperature data from the data logger. Ideally, the results will provide support for installing a hot air circulation system to keep the entire oven at same temperature independent of location in the oven.

5. Results

5.1 Real Time Process Monitoring System

Hitchiner already had a complex data acquisition system in place that was gathering process data, the data was not easy to retrieve or interpret however. By consolidation the data and ensuring the quality of sensor observations, the system has increased its effectiveness. Data is only as good as the analysis being performed on it.

5.1.1 Database Consolidation

To aggregate the process data that is collected on the floor, a VijeoHistorian SQL database was installed. The Historian has the ability to fetch new data from selected VijeoCitect applications as soon as it is collected on the floor. This database is being run on a powerful server with more than enough computing power and bandwidth to support the goals of this project.

5.1.2 Sensing Equipment Evaluation

Before any of the data could be considered accurate, each sensor had to be verified. This meant checking each sensors location against the building schematics, confirming that it was in good working condition, determining if calibration was needed, and replacing any parts that were worn out or broken, regardless of their performance. By replacing sensors before they fail, the system will never suffer from a lapse in data for that specific measurement dimension. Most of the sensors on the floor were in excellent condition, with the exception of long probe style thermocouples, which were replace from on hand supplies. Hitchiner has a schedule in place to evaluate this equipment at more frequent intervals as a result.

5.2 Statistical Process Control System

The majority of the data analysis system is centered on Statistical Process Control (SPC). A license of Statistica QC Desktop Edition version 10 was purchased from StatSoft. The software was installed on a simple desktop computer in the lead process engineer's office. Originally the computer was slated to go in a computer rack in a closet maintained by IT, but since multiple instances of Statistica can be run a one time, the process engineer can use the software for complex data analysis, without interrupting the macro automation features.

5.2.1 Statistica Package Setup

Given the advanced capabilities of Statistica, the software is fairly intuitive and easy to learn. The user interface is styled like Microsoft Office 2010 products, reducing the learning curve and intimidation factor.

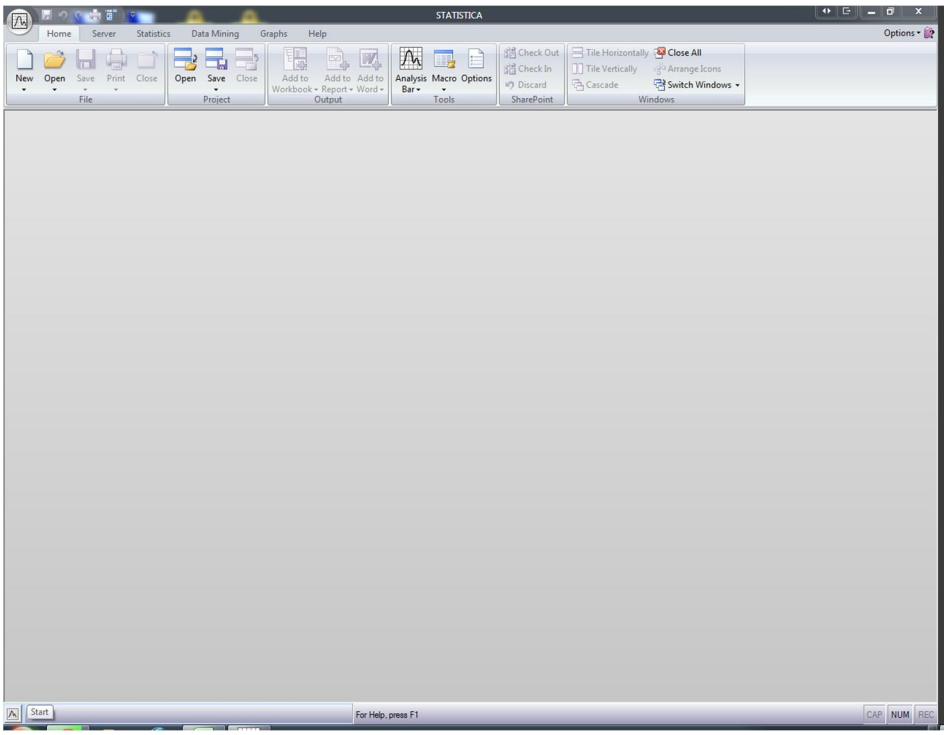
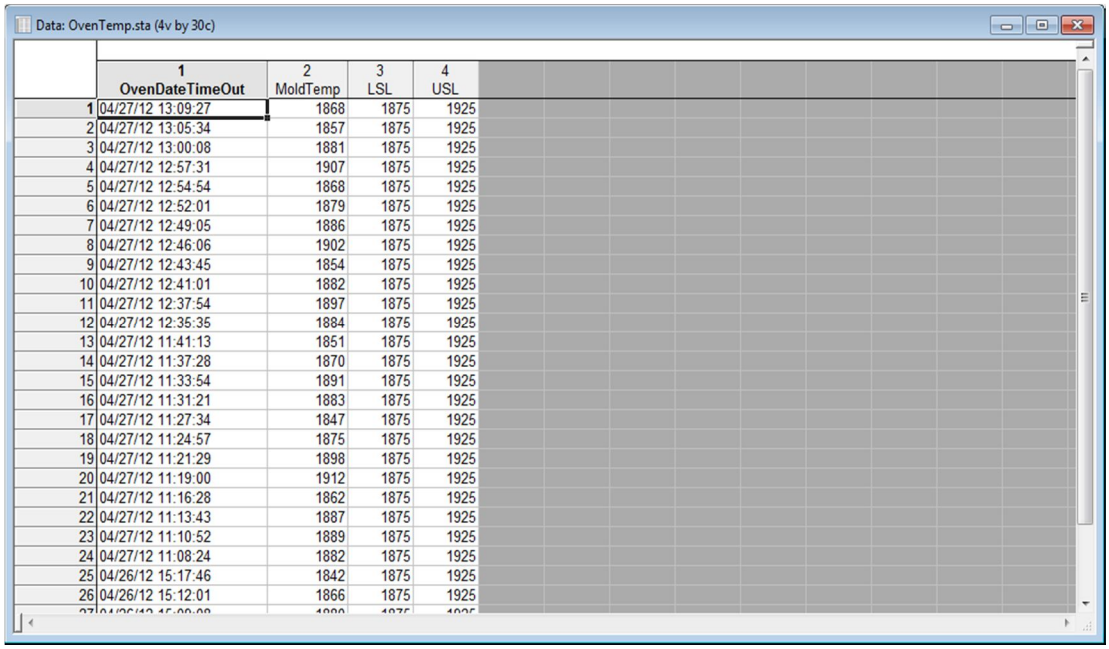


Figure 1 Statistica QC Main User Interface

Statistica was then configured to be able to access the VijeoHistorian SQL database as well as an Oracle SQL database containing slurry information that is acquired by operators and input via IQMS. Statistica was also connected to an additional 2 SQL databases containing part information and specifications and foundry data.

Once the connections were in place, data could be retrieved from the databases via SQL SELECT statements. These statements are used to select the specific data that is desired out of the entire database. Since the data is located in different tables and locations, the whole data set is cannot be downloaded and sorted locally. The SELECT statements harness the power of the SQL server to find only the data that is desired and then sends that back to the requesting computer. The figure below shows an example Statistica spreadsheet that is populated by a SELECT statement. The select statement can be seen as well

SELECT [OvenDateTimeOut],[MoldTemp] FROM [HMOpsDataCollector].[dbo].[Shells]
WHERE MoldTemp > 0 And OvenDateTimeOut >= DateAdd(Day,-1,GETDATE())"


Data: OvenTemp.sta (4v by 30c)

	1 OvenDateTimeOut	2 MoldTemp	3 LSL	4 USL
1	04/27/12 13:09:27	1868	1875	1925
2	04/27/12 13:05:34	1857	1875	1925
3	04/27/12 13:00:08	1881	1875	1925
4	04/27/12 12:57:31	1907	1875	1925
5	04/27/12 12:54:54	1868	1875	1925
6	04/27/12 12:52:01	1879	1875	1925
7	04/27/12 12:49:05	1886	1875	1925
8	04/27/12 12:46:06	1902	1875	1925
9	04/27/12 12:43:45	1854	1875	1925
10	04/27/12 12:41:01	1882	1875	1925
11	04/27/12 12:37:54	1897	1875	1925
12	04/27/12 12:35:35	1884	1875	1925
13	04/27/12 11:41:13	1851	1875	1925
14	04/27/12 11:37:28	1870	1875	1925
15	04/27/12 11:33:54	1891	1875	1925
16	04/27/12 11:31:21	1883	1875	1925
17	04/27/12 11:27:34	1847	1875	1925
18	04/27/12 11:24:57	1875	1875	1925
19	04/27/12 11:21:29	1898	1875	1925
20	04/27/12 11:19:00	1912	1875	1925
21	04/27/12 11:16:28	1862	1875	1925
22	04/27/12 11:13:43	1887	1875	1925
23	04/27/12 11:10:52	1889	1875	1925
24	04/27/12 11:08:24	1882	1875	1925
25	04/26/12 15:17:46	1842	1875	1925
26	04/26/12 15:12:01	1866	1875	1925

Figure 2 Statistica Spreadsheet of G-Oven Temperatures for 24 hours

5.2.2 Statistica Macro Configuration

Microsoft Office uses Visual Basic for Applications (VBA) when writing macros. Statistica uses a highly customized version of VBA called Statistica Visual Basic (SVB) for its internal macros. Macros can be written from scratch, which is difficult, or recorded by Statistica as actions are performed. For this project, macros were recorded as graphs and charts were generated, and then modified by hand if needed. This made it possible to create complex macros and change small aspects of the output if a mistake was made during the recording. Statistica has a built in SVB macro editor that provides an easy to follow user interface with syntax highlighting and debugging features. Below is a sample macro, created to graph G-Oven temperatures, open for editing. As can be seen, the code is very clean and easy to follow.

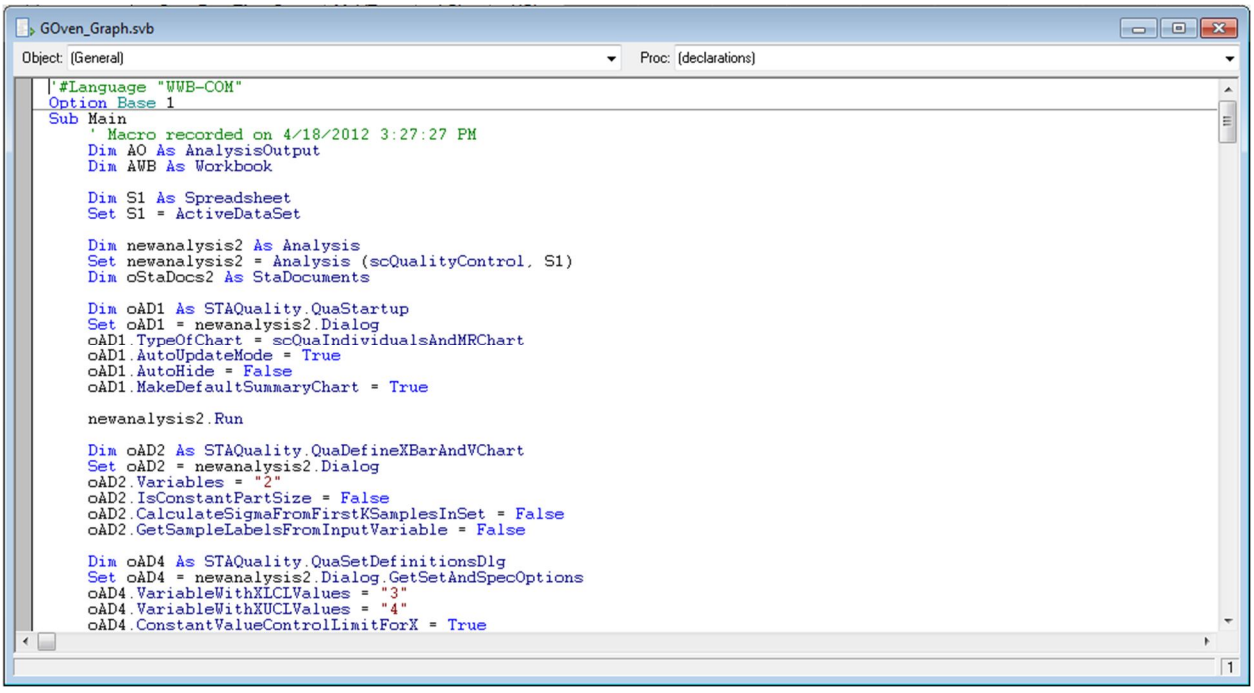


Figure 3 Statistica SVB Macro Editor

Although many macros can be used to make the desired graphs and spreadsheets, it was important to try and fit each department's graphs into one macro to reduce file operations.

5.2.3 Statistica Output

Statistica can output analysis as graphs or spreadsheets (other outputs are possible, but not used for this project). Like macros, these graphs are highly customizable and can provide nearly any visual information you want. The data for this project is all analyzed as Individual and Moving Range graphs, due to the sampling characteristics of the datasets. By default individual and moving range charts are output as seen in the figure below. As can be seen, this graph includes histograms of observations and a moving R chart. These are not needed for the web page display system.

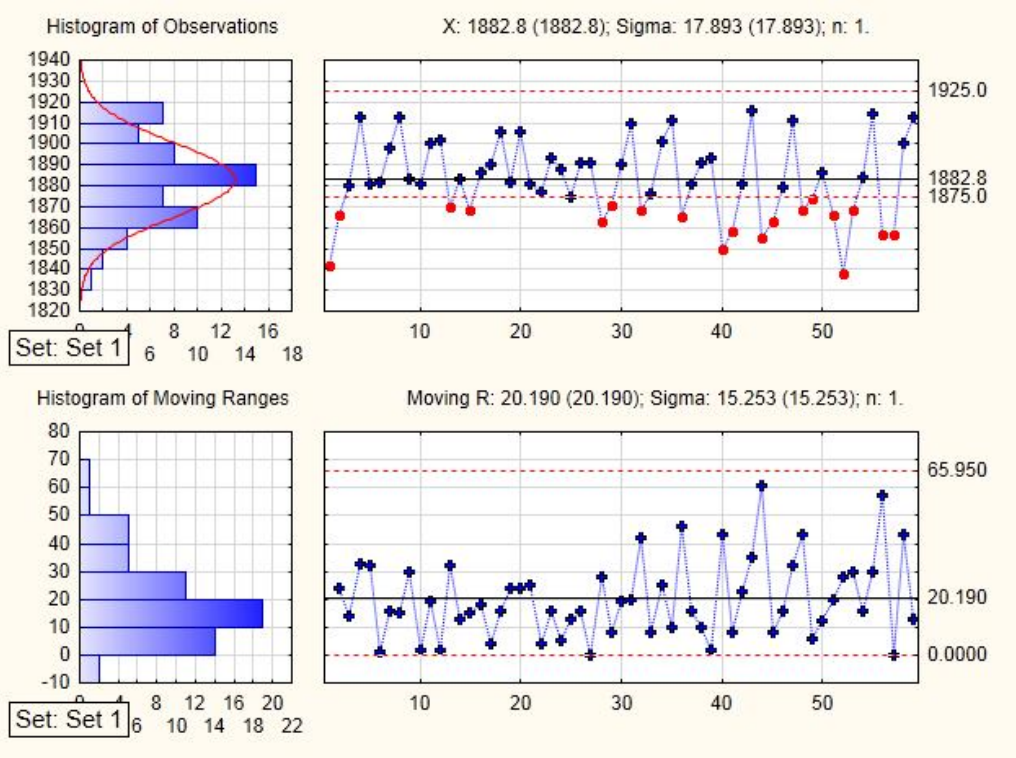


Figure 4 Default Individual and Moving Range

The graphs can be configured to remove the histograms and the moving r chart. This provides a neater, larger graph that can more easily interpreted by operators on the floor. A sample of a pure Individuals chart can be seen below.

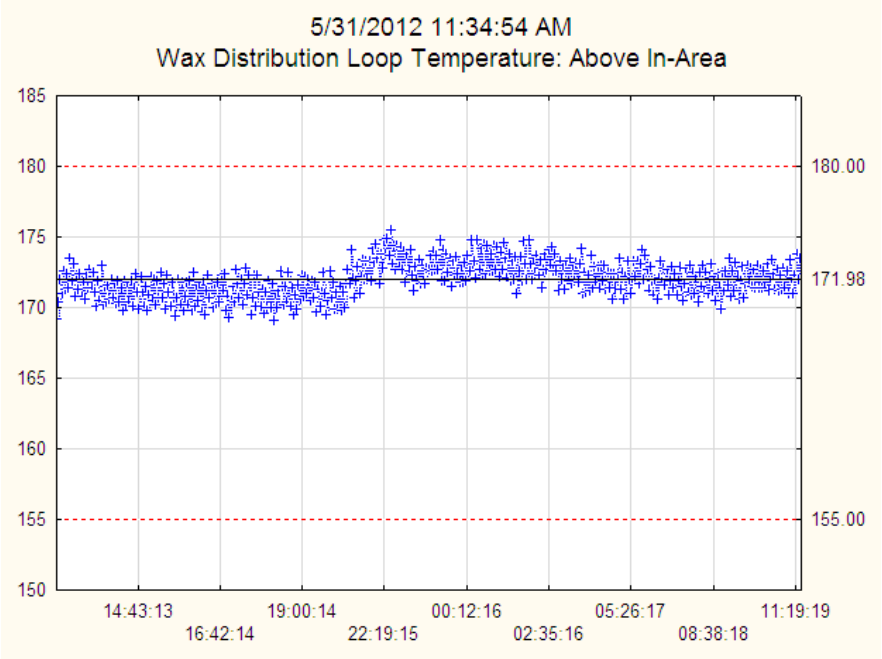


Figure 5 Pure Individuals Chart

In order to make the data relevant, Statistica applies Western Electric Runs Tests on the data that is graphed. These rules can identify when samples are out of control limits, or trending toward limits. These rules are set from a dialog and can be executed from a macro after the graph is generated. The Runs Tests output their results as spreadsheets with the rules that were validated and the result. If rules are violated the points that violated are listed in the spreadsheet as well as highlighted in red on the graphs.

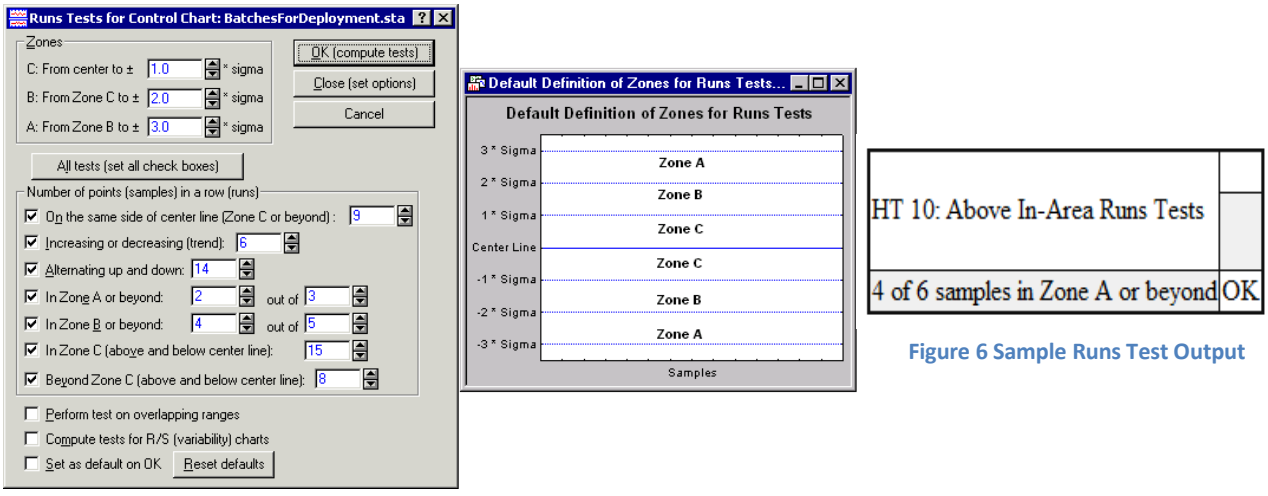


Figure 6 Sample Runs Test Output

Figure 7 Runs Tests Options and Zone Definitions

5.2.4 Output Result Files to Network

When Statistica runs a macro to graph data and perform runs tests, the output remains inside the Statistica window. To get the results out of Statistica, a macro was written that merges all the outputs to a single workbook. That workbook is then parsed by a second macro that saves graphs as PNG files and spreadsheets as HTML files. These files are saved right to a network directory accessible from anywhere inside Hitchiner.

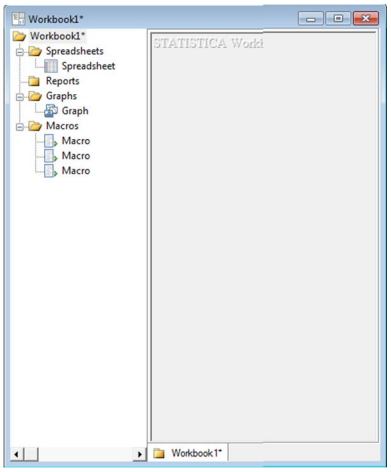


Figure 8 Sample Merged Workbook

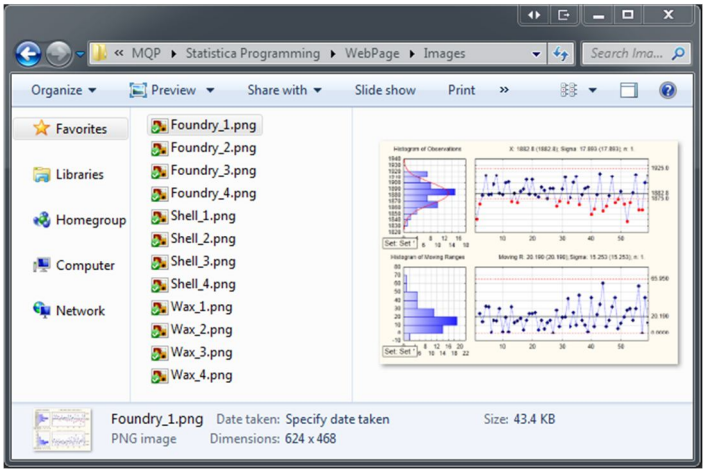


Figure 9 Network Directory of Result Outputs

5.2.4 Network Impact

The IT department at Hitchiner was concerned with the amount of files that would be stored on the network for display and how much data would be used up. Their concerns were alleviated by overwriting the existing graphs and spreadsheets as to only have the most current one available. The raw data is still safely stored in the databases if additional processing is needed. The total size of the network directory including all graphs, spreadsheets, macros, and webpages is less than 10 megabytes; it would take 10,000 copies of this folder to even approach the storage capacity of the network location.

5.3 Web Based Display System

After configuring Statistica's macros to generate the graphs and spreadsheets that were desired, the next step was to create a way to display this data for users to see. The system has to be "thin", meaning users do not need any additional software. This was accomplished by creating a series of HTML webpages that reside in the same network directory as the Statistica output. Since HTML is a basic language, it will be easy for future developers to make changes to the webpages from any text editor, without having any special web design software. The nature of HTML pages allows for them to be run with nothing but the single file, whereas more complex web design packages can generate thousands of supplementary files and folders.

5.3.1 Main Page

The main page of the display system was designed to give users a quick way to see the status of the process. Each monitored aspect of the system is represented on the main page with a visual indicator. When a rule is violated, the indicator changes from green for good to yellow for warning to red for bad. The color is determined by what rule is violated. A piece of custom VB.NET software, same one that calls Statistica macros on a schedule, is used to parse the HTML file generated by the Runs Tests. If the correct string is found, the software compares it against its library of acceptable values and then overwrites the visual indicator file on the network with a stock image that is red, yellow, or green. The main page also contains links to the individualized graphs and runs tests results, as well as departmental auto-cycle pages. Every webpage associated with this project has auto-refresh built in, so it can always be sure it has the freshest information available. The main page is shown in the figure below, with 2 of the indicators identifying that there are out of control data points for that dataset. The page is

quite simple in its design as it is comprised of only tables inside of tables. The tabular nature of the page lends to its ease in modifying it, as squares can be broken up into many more squares without changing the whole page layout.

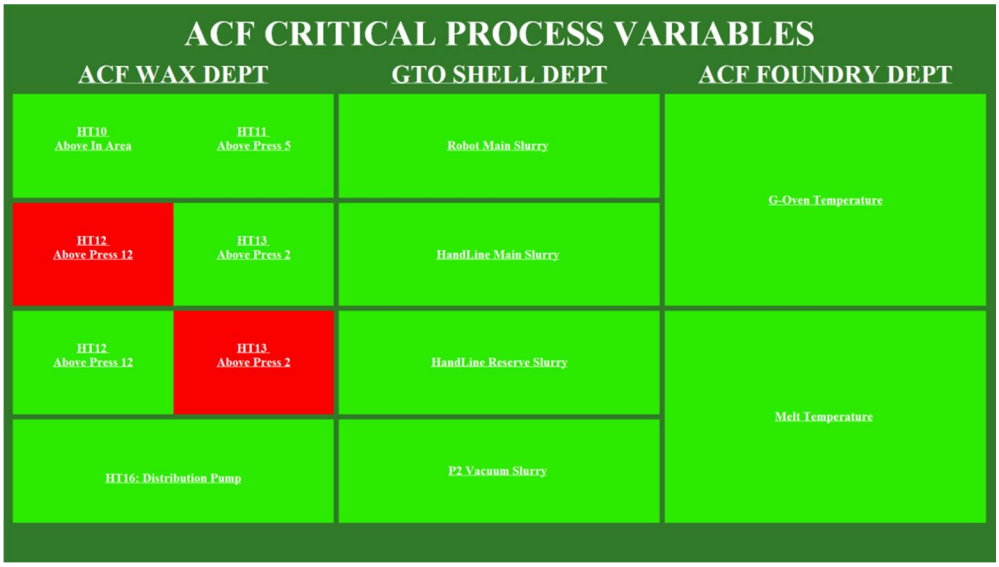


Figure 10 Critical Process Variable Dashboard

5.3.2 Individual Data Set Pages

Each square on the main page links to a subpage containing graphs and Runs Test relevant to that page. This page, like the main page, is also table based allowing for future expansion. As can be seen at right, on the individual page for the wax distribution loop heat trace 10,

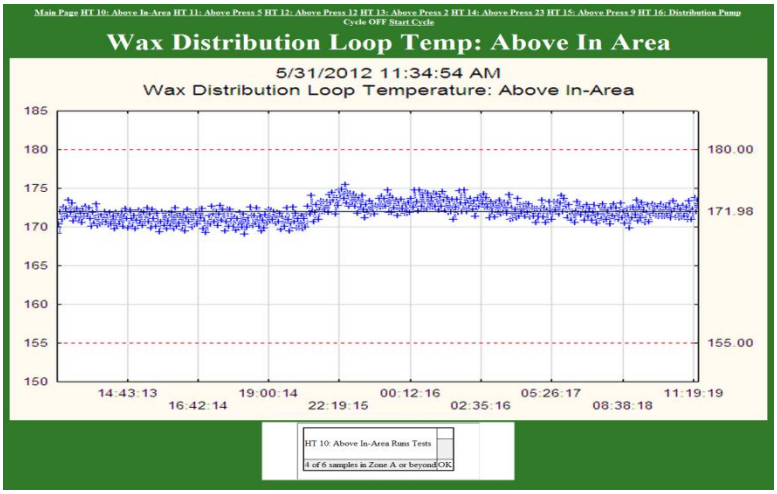


Figure 11 Wax Distribution Loop Heat Trace 10 Temperatures

the data is normal and the process is in control. This is validated by the runs tests results displayed right below the graph.

5.3.3 Departmental Pages

Each department will be installing large flat panel monitors over the next few months as construction occurs. These monitors will eventually display data from this system full time. Since the displays will be located in extreme environments they must be able to display data continually and without operator input. To accomplish this, the same pages used to display graphs by name will be cycled through using JavaScript to call each page in infinite succession. The departmental page looks nearly identical to the individual page but it is actually two sections, one section contains the static links at the top, and the other is an IFrame element running the JavaScript loop. This is most clearly evident when the individual page is missing and the script cannot locate the page.

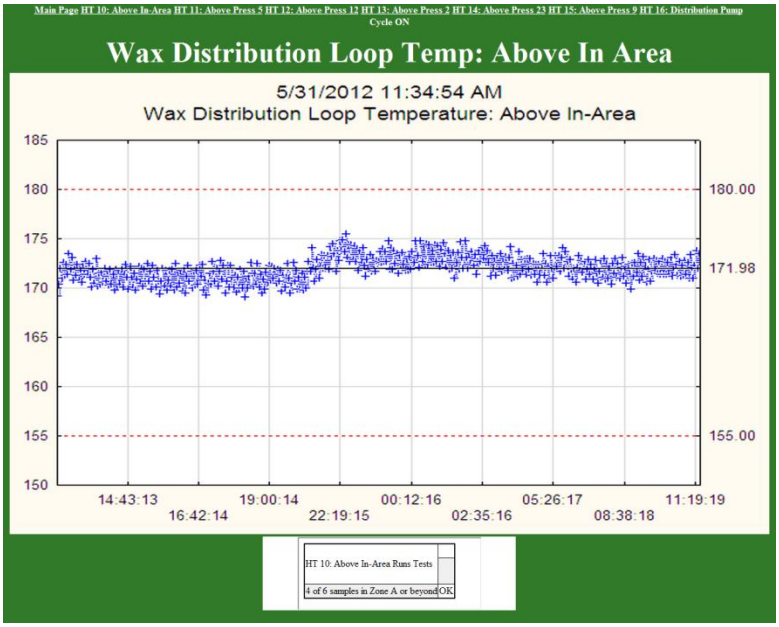


Figure 12 Wax Department Cycle

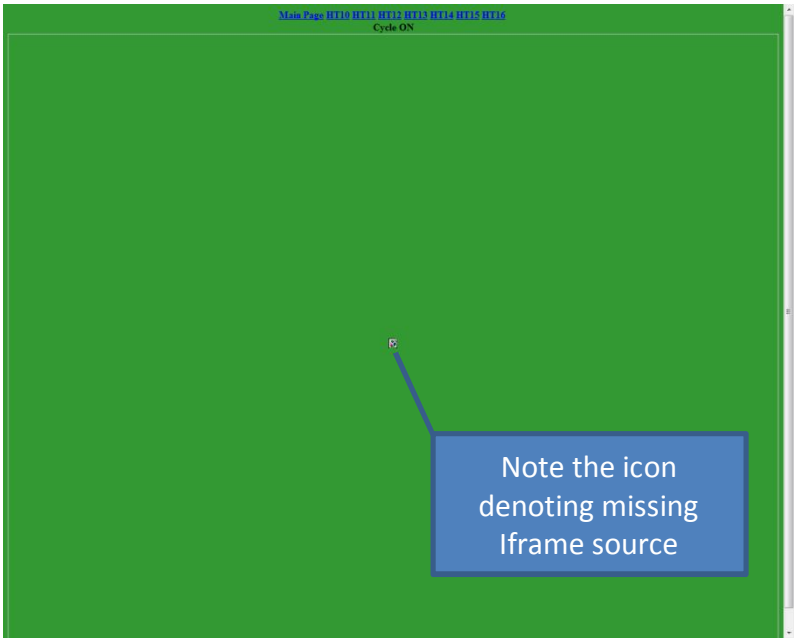


Figure 13 Wax Department Cycle Page Suffering from Missing Page Error

5.4 Thermal Survey of Oven

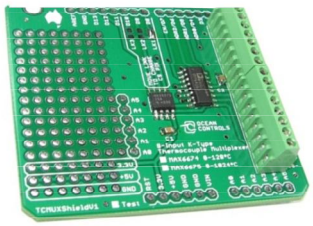
The data logger system is a fairly simple concept, but incredibly complex in its implementation. The system proved to be much more complex than originally planned and suffered because of it.

5.4.1 Data Logger Build

To have an electronics design house fabricate a device capable of monitoring 35 Type K thermocouples inside of an oven without any cables would cost more than $10,000. The total cost of this data logger is under $1000. The data logger is built on an Arduino Uno R3 microcontroller. The Arduino is a low cost ($70) small, fast, and open source, microcontroller. What makes the Arduino special is the myriad of “shields” (add on circuit boards) that exist for this device. Many of those shield come with pre built libraries making coding even easier. The Arduino was chosen because OceanControls, an Australian electronics company, produced the KTA-259 thermocouple shield. This device is capable of connecting 8 Type K thermocouples amplifying the reading and reporting that reading as a digital value between 0 and 1024. This digital value is the temperature in degrees Celsius. The boards are designed to stack together as many as required, as long

Figure 14 Arduino Uno R3 Microcontroller

Figure 15 OceanControls KTA-259



as the Chip Select pins are all connected to individual digital I/O's on the Arduino. The data logger required 35 inputs so 6 shields were purchased, the last 6 shields in stock in the US, along with an Arduino and a MicroSD card breakout shield. The MicroSD card shield has a large prototyping area for wiring custom circuits. The seven shields were stacked 2 high and two deep with the MicroSD Shield tripled stacked on the Arduino. The thermocouple shields are connected by additional perforated circuit board to give the system a rigid backbone. The support structure circuit board is not electrically connected to the shields at any point, however there are 3 power cables running through the channel made by the structure. Only the upper level boards are connected together the lower boards are held in by the two rows of headers. The Arduino and MicroSD shield snap under the first shield to complete the set. The protoshield is used to run wires between the SPI (Serial Parallel Interface) pins on each stack. The Arduino is able to individually address each board using this shared connection. This is because each thermocouple shield has a chip select pin allowing for the Arduino to select which board it would like to communicate with.

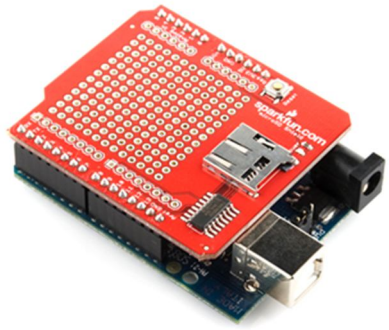


Figure 16 MicroSD Shield

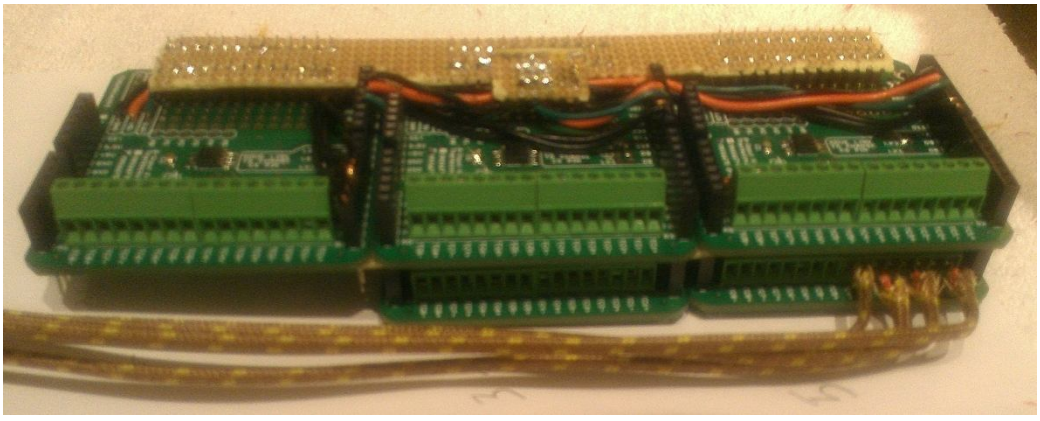


Figure 17 Thermocouple shield array

The thermocouple shields need to be traced to a more robust connection plate to allow for high temperature thermocouple plugs to be connected. This was accomplished by purchasing a 36 plug faceplate and wiring each plug to a port on the data logger. This wiring job alone took 36 hours due to the strong nature of type K thermocouple wire.

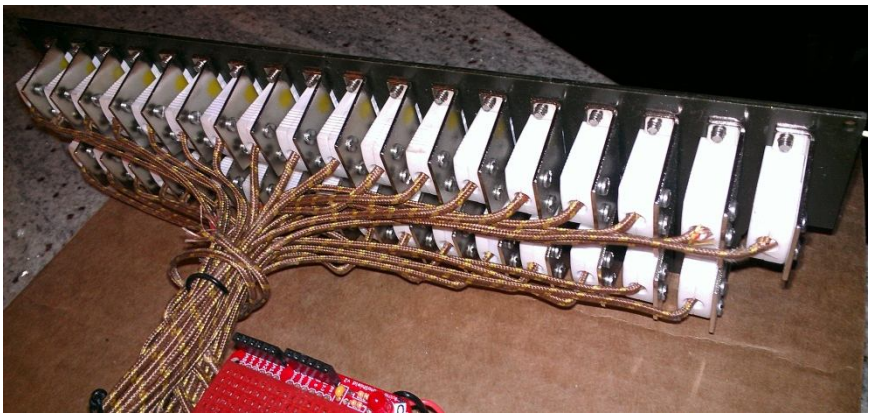


Figure 18 Thermocouple Plug Faceplate

5.4.2 Thermal Protection System

To protect the data logger from the intense heat found underneath the mold cart a water jacket was built. The jacket holds 5 gallons of water which absorbs the excess heat and tries to prevent it from reaching the data logger. The heat capacity of water is great enough that this volume of water in a 400 F environment can keep the inner chamber below 85 C for 6 hours. A stainless steel shell was fabricated and painstakingly TIG welded to be 100% leak-proof. The jacket was designed to fit underneath a mold cart. The cart had to be modified to accept the jacket

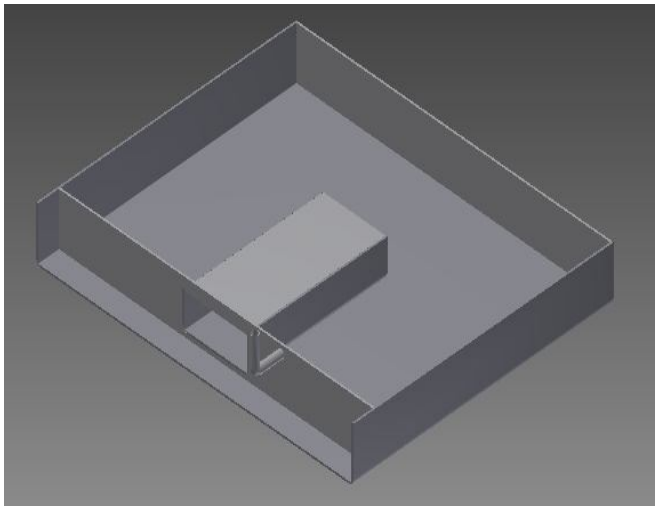


Figure 19 Water Jacket Design

however. The wheels had to be mounted a different way as to not impede the water jacket from sliding in. One side of the carts frame had to be removed and angle iron rails had to be welded on to support the water jacket. A notch was cut in the face of the cart to allow for the steam vent to slide in. The last modification to the cart was to weld on nine short upright tubes to hold the ambient air thermocouple rods. These modifications were all made on an extra mold cart so as to not disturb the production cycle.

Figure 20 Steam Vent Notch

Figure 21 Wheel Modification and Angle Iron

Figure 22 Ambient Air Thermocouple

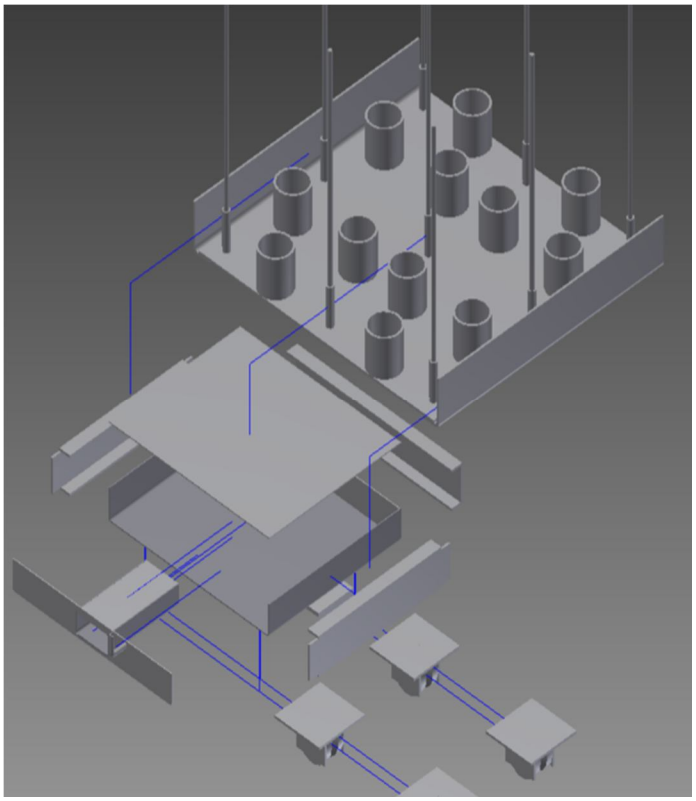


Figure 23 Thermal Survey Cart Design

5.4.3 Thermal Testing Of Water Jacket

To prove the water jacket could keep the inner chamber with the electronics specified thermal limits, it was put inside a box oven at 400F filled with water and wrapped in kaowool refractory blanket. Those conditions match those found under the cart in G-Oven. The results of two trials can be seen below. The temperature was monitored on a hand held thermocouple meter every over the course of 6 hours. The first test was the worst case, with the underside of

Figure 24 Worst Case Scenario Testing

the cart being 500 F and not wrapping the water jacket in kaowool. The second test was the normal case scenario that should be expected under operation conditions.

Figure 25 Normal Operating Conditions

Oven condition temperatures were determined by taking a temperature reading with an infrared thermometer from under the cart as soon as the oven door opened. The average reading was about 400°F.

The results of these two trials confirm that the water jacket can successfully protect the data logger from the temperatures found under the oven.

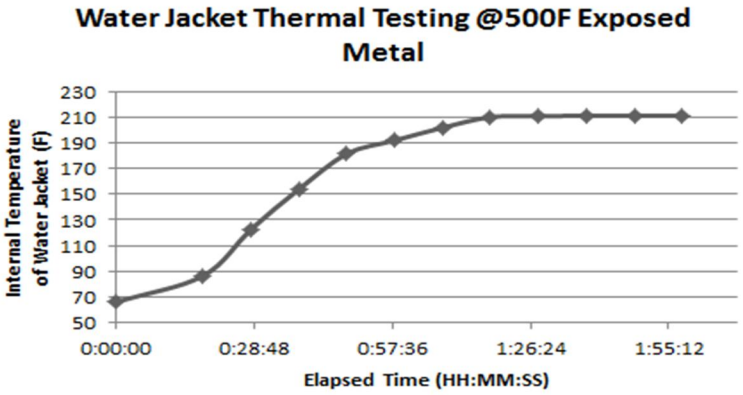


Figure 26 Worst Case Scenario Temperature Profile

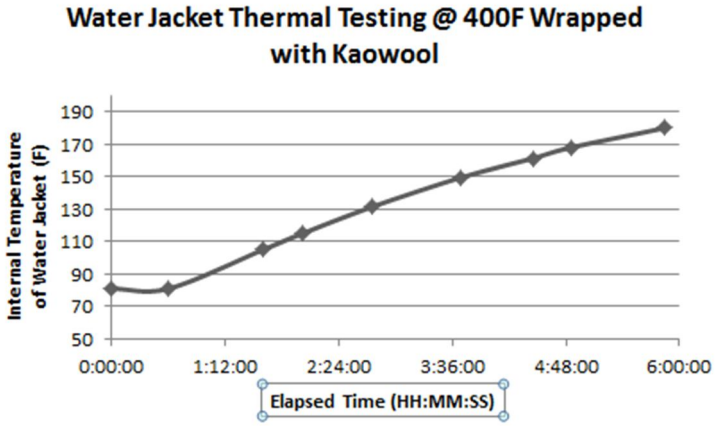


Figure 27 Normal Operating Condition Temperature Profile

5.4.4 Thermal Survey

When it came time to run the trials with the data logger in the oven, the data logger would not work as it did during building and testing. Each thermocouple shield works perfectly on its own, but when they are connected the entire SPI system becomes grounded together. OceanControls technicians were baffled by the situation and think that some of the integrated

circuits may be bad. There are no more shields in stock and will take one month to produce more. A secondary system is being designed using only 16 thermocouples but it is not expected be to tested until mid-June. The results of the thermal survey will provide priceless information about the pre-heat process and allow Hitchiner to possible change things to improve the process.

6. Conclusions

To goal of this project was to create a system that can analyze real-time process data and display the results, automatically. Harnessing the power of SQL Server 2008, VijeoCitect, VijeoHistorian, Statistica 10 QC, and open source HTML web design; a system has been put in place at Hitchiner that makes it possible for an engineer or operator to summon this data and react to changes much faster than before. By making Hitchiner's process data more accessible, this system has eliminated the time spent tracking down data and processing it manually using conventional tools like Excel and Minitab.

The system is currently available on any computer inside the Hitchiner network. Hitchiner is actively deploying large screen displays for each department to display this systems data analysis for operators and engineers to see. This new system provides the analytical capabilities needed to help catch defects and defect causing conditions early. Additionally, having users also analyzing the data will further improve the company's ability to prevent defects. Reduction in defects will not only have a significant impact on production but also on employee morale as employees perform better when they can take pride in the quality of their work.

The thermal survey of the oven was a unique experience, fraught with challenge. The water jacket designed to protect the electronics was a success. The mold cart modifications were also successful. However, due to circuitry problems beyond the control of this project, the data logger was unable to function as required. Replacement parts that may be able to restore functionality are on backorder from the manufacturer. Moving forward, the data logger should

be rebuilt using shift registers, instead of using enable pins, to expand the digital outputs from the Arduino. Shift registers will eliminate the grounding problems experienced when the thermocouple shields are all assembled into the data logger.

The future holds a great deal for the investment casting industry. Even though Hitchiner is on the forefront of investment casting technology, there is still room for improvement. As materials advance and machine design advances, the industry too will have to advance their practices. This will only be possible by learning and understanding every single aspect of the process. Hitchiner has taken one step closer to ultimate process understanding. Technology will take the industry further than ever, but the industry must embrace it with open arms.

7. Works Cited

Ali Rıza Motorcu, A. G. (2006). Statistical process control in machining, a case study for machine. *Materials and Design, 27*, 364-372.

Balwinder Singh Sidhu, P. K. (2008, August). Effect of Slurry Composition on Plate Weight. *Journal of Materials Engineering and Performance, 17*(4), 489-499.

Chakrabarti, B. K. (2002, August). Drying conditions and their effect on ceramic. *Materials Science and Technology, 18*, 935-941.

H. Roshan, R. R. (2010, October). Process knowledge and product characteristics gained using casting process optimization. *Foundry Management and Technology*, 14-18.

Ivan Miletic, F. B. (2008, October). Experiences in Applying Data-Driven Modelling. *The Canadian Journal of Chemical Engineering, 86*, 937-947.

Nahale, A. W. (2002, October). Why Web-based monitoring? *Contracting Business, 1*(1), 10-18.

Noble, J. V. (1975, October). The Wax of the Lost Wax Process. *American Journal of Archaeology, 79*(4), 368-369.

Raymond J. Bunkofske, N. T. (1996). Real-Time Process Monitoring. *IEEElSEMl Advanced Semiconductor Manufacturing Conference*, (pp. 382-341).

S. Jones, M. R.-C. (2003, July). Effect of moisture upon mechanical properties of. *Materials Science and Technology, 19*, 907-915.

Spada, A. T. (1998, July). Hitchiner Manufacturing Co.--Turning the casting world upside down. *Modern Casting, 88*(7), 39-44.

Swenson, L. (2002, January). Improving investment casting through innovations. *Modern Casting, 92*(1), 32-35.

Tague, N. R. (2004). *The Quality Toolbox.* Asq Quality Press.

Wright, J. R. (2000, Februrary). Hitchiner's foundry of the future. *Foundry Management & Technology;, 128*(2), 26-30.

8. Appendix

8.1 HTML Code for Main Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="en-us" http-equiv="Content-Language" />
<meta http-equiv="refresh" content="25"
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>ACF Data Monitoring System</title>
<style type="text/css">
.auto-style3 {
	text-align: center;
	font-size: 50pt;
	height:5%;
}.auto-style7 {
	text-align: center;
	font-size: xx-large;
}.auto-styleWax1 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_2.png');
}.auto-styleWax2 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_4.png');
}.auto-styleWax3 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_6.png');
}.auto-styleWax4 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_8.png');
}.auto-styleWax5 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_10.png');
}.auto-styleWax6 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_12.png');
}.auto-styleWax7 {
	text-align: center;
	font-size: x-large;
	background-image: url('Indicators/Wax_14.png');
}.auto-style9 {
```

```
        border: 5px solid #000000;
        background-color: #A4A4A4;
}.auto-style10 {
        text-align: center;
        font-size: large;
        background-image: url('Indicators/Shell1.png');
}.auto-style11 {
        text-align: center;
        font-size: x-large;
        background-image: url('Indicators/Foundry1.png');
}.auto-style12 {
        background-image: url('Indicators/Wax_4.png');
        text-align: center;
        font-size: x-large;
}.auto-style14 {
        background-image: url('Indicators/Wax_6.png');
        text-align: center;
        font-size: x-large;
}.auto-style15 {
        text-align: center;
        background-image: url('Indicators/Wax4.png');
        font-size: x-large;
}.auto-style16 {
        text-align: center;
        background-image: url('Indicators/Shell2.png');
}.auto-style17 {
        text-align: center;
        font-size: x-large;
        background-image: url('Indicators/Shell3.png');
}.auto-style18 {
        text-align: center;
        background-image: url('Indicators/Shell4.png');
}.auto-style19 {
        text-align: center;
        background-image: url('Indicators/Foundry2.png');
}.auto-style20 {
        font-size: x-large;
        text-align: center;
        background-image: url('Indicators/Foundry3.png');
}.auto-style21 {
        font-size: x-large;
        text-align: center;
        background-image: url('Indicators/Foundry4.png');
}.auto-style22 {
        font-size: x-large;
}.auto-style23 {
        background-image: url('Indicators/Good.png');
        text-align: center;
        font-size: x-large;
}.auto-style24 {
        background-image: url('Indicators/bad.png');
        text-align: center;
        font-size: x-large;
```

```
}.auto-style25 {
	background-image: url('Indicators/bad.png');
	text-align: center;
	font-size: x-large;
}.auto-style26 {
	background-image: url('Indicators/good.png');
	text-align: center;
	font-size: x-large;
}
</style>
</head>

<body class="body" style="background-color: #317a29"  link="white" alink="white" vlink="white" visible="true">
<!-- <p font="size:1"><br></p> -->
<table cellpadding="0" border="0" cellspacing="10" style="width: 100%; height:70px">
	<tr height="5%">
	<td colspan="3" class="auto-style3"><strong><font color="white">ACF CRITICAL PROCESS
VARIABLES</font></strong></td>
	</tr>
		<tr height="5%">

		<td class="auto-style7" style="width: 33%; height:100%;"><font size="10" color="white"><a
href="./WaxPages/CycleWax.html"><strong>ACF WAX DEPT</strong></font></td>
		<td class="auto-style7" style="width: 33%; height:100%;"><font size="10" color="white"><a
href="./ShellPages/CycleSlurry.html"><strong>GTO SHELL DEPT</strong></a></font></td>
		<td class="auto-style7" style="width: 33%; height:100%;"><font size="10" color="white"><a
href="./FoundryPages/CycleFoundry.html"><strong>ACF FOUNDRY DEPT</strong></a></font></td>
	</tr>
		<tr height="300px">
		<td style="width: 33%; height: 200px">
			<table cellpadding="0" cellspacing = "0" style="width: 100%; height: 100%">
				<tr>
				<td class="auto-styleWax1" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax1.html"><strong>HT10 <br> Above In Area</strong></a><strong><br><br></td>
				<td class="auto-styleWax2" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax2.html"><strong>HT11 <br> Above Press 5</strong></a><strong><br><br></td>
				</tr>
			</table>
		</td>

		<td class="auto-style10" style="width: 33%; height: 100%"><a
href="./ShellPages/slurry1.html"><strong><span class="auto-style22">Robot Main
Slurry</span></strong></a><span class="auto-style22"> </span></td>
		<td rowspan="2" class="auto-style11" style="width: 33%; height: 100%"><strong><a
href="./FoundryPages/Foundry1.html">G-Oven Temperature</a></strong></td>
	</tr>
		<tr height="300px">
		<td class="auto-style12" style="height: 200px; width: 33%">
			<table cellpadding="0" cellspacing = "0" style="width: 100%; height: 100%">
				<tr>
				<td class="auto-styleWax3" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax3.html"><strong>HT12 <br> Above Press 12</strong></a><strong><br><br></td>
```

```
                                        <td class="auto-styleWax4" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax4.html"><strong>HT13 <br> Above Press 2</strong></a><strong><br><br></td>
                                        </tr>
                                </table>
                        </td>


                        <td style="width: 33%; height: 25%" class="auto-style16">
                        <a href="./ShellPages/slurry2.html"><strong><span class="auto-style22">HandLine Main
Slurry</span></strong></a><span class="auto-style22"> </span></td>


        </tr>
                        <!--Main Row 3-->
        <tr height="300px">
                        <td style="height: 200px; width: 33%" class="auto-style14">
                                <table cellpadding="0" cellspacing = "0" style="width: 100%; height: 100%">
                                        <tr>
                                        <td class="auto-styleWax5" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax5.html"><strong>HT12 <br> Above Press 12</strong></a><strong><br><br></td>
                                        <td class="auto-styleWax6" style="width: 50%; height: 100%"><a
href="./WaxPages/Wax6.html"><strong>HT13 <br> Above Press 2</strong></a><strong><br><br></td>
                                        </tr>
                                </table>
                        </td>

                        <td style="width: 33%; height: 100%" class="auto-style17">
                        <a href="./ShellPages/Slurry3.html"><strong>HandLine Reserve Slurry</strong></a></td>
                        <td rowspan="2" style="width: 33%; height: 25%" class="auto-style19">
                        <a href="./FoundryPages/Foundry2.html"><span class="auto-style22"><strong>Melt
                        Temperature</strong></span></a><span class="auto-style22"> </span></td>
                        <!--<td class="auto-style23" style="height:25%"><iframe src="Test.htm" width="52%"
height="100%"></iframe></td>
                        <!-- <td style="width: 33%; height: 25%" class="auto-style20">
                        <a href="Images/Foundry_3.png"><strong>RoR</strong></a> </td> -->
        </tr>
        <tr height="300px">
                        <td style="height: 200px; width: 33%" class="auto-styleWax7"><a
href="./WaxPages/Wax7.html"><strong><br>HT16: Distribution Pump<br></strong></a></td>
                        <td style="width: 33%; height: 100%" class="auto-style18">
                        <a href="./ShellPages/Slurry4.html"><strong><span class="auto-style22">P2 Vacuum
Slurry</span></strong></a><span class="auto-style22"> </span></td>
                        <!--
                        <td style="width: 100%; height: 25%" >
                                <table cellpadding="0" cellspacing="0" style="width: 100%; height: 100%; border: 0px
solid #FFFFFF;">
                                        <tr>
                                                <td style="width: 50%; height: 50%" class="auto-style23">3.00 hr</td>
                                                <td style="width: 50%; height: 50%" class="auto-style24">6.00 hr</td>
                                        </tr>
                                        <tr>
                                                <td style="width: 50%; height: 50%" class="auto-style25">12.0 hr</td>
                                                <td style="width: 50%; height: 50%" class="auto-style26">24.0 hr</td>
```

```
                                    </tr>
                                </table>
                            </td>
                            <!--<td style="width: 33%; height: 25%" class="auto-style21">
                            <a href="Images/Foundry_4.png"><strong>Time in Oven</strong></a> </td>-->
                    </tr>
</table>
</body>
</html>
```

8.2 Wax Distribution Loop Single Heat Trace Individual Display Page

```
<html lang="en">
<head>
<meta http-equiv="refresh" content="30"
</head>
<body style="background-color: #317a29" link="white" alink="white" vlink="white">
<center>
<a href="../Default.html"><strong>Main Page</strong></a>
<a href="./Wax1.html"><strong>HT 10: Above In-Area</strong></a>
<a href="./Wax2.html"><strong>HT 11: Above Press 5</strong></a>
<a href="./Wax3.html"><strong>HT 12: Above Press 12</strong></a>
<a href="./Wax4.html"><strong>HT 13: Above Press 2</strong></a>
<a href="./Wax5.html"><strong>HT 14: Above Press 23</strong></a>
<a href="./Wax6.html"><strong>HT 15: Above Press 9</strong></a>
<a href="./Wax7.html"><strong>HT 16: Distribution Pump</strong></a>
<br>
<font color="white"><b>Cycle OFF</b></font> <a href="./CycleWax.html"><strong>Start Cycle</strong></a>
</center>
<table border="0"  width="100%" height="95%" style="background-color: #317a29">
  <tr>

  <td colspan="3" height="60px"><center><b><font size="10" color="white">Wax Distribution Loop Temp: Above
In Area</font></b></center></td>

</tr>

  <tr height=95%><!-- Row 1 -->
    <td colspan="3"><img src="../Images/Wax_2.png" width = 100% height=100%  alt="" border="0"></td><!-- Col
1 -->
  </tr>
<tr height=10%>
            <td width=33% bgcolor="#317a29"> </td>
            <td border="0" width=33% align="middle" bgcolor="white"><iframe id="frame"
src="../Images/Wax_2.htm" align="middle" Height = "80%"></iframe> </td><!-- Col 2 -->
            <td width=33% bgcolor="#317a29"> </td>
            </tr>
 </table>
  </body>
</html>
```

8.3 Departmental Auto-Cycle Page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
<head>
<meta http-equiv="refresh" content="7">
<title>Wax Cycle</title>
<style type='text/css'>
   #your_div img {
      display: none;
   }
</style>
</head>
<body style="background-color: #339933" link="blue" alink="blue" vlink="blue">

<center>
<a href="./Default.html"><strong>Main Page</strong></a>
<a href="./Images/Wax_2.png"><strong>HT10</strong></a>
<a href="./Images/Wax_4.png"><strong>HT11</strong></a>
<a href="./Images/Wax_6.png"><strong>HT12</strong></a>
<a href="./Images/Wax_8.png"><strong>HT13</strong></a>
<a href="./Images/Wax_10.png"><strong>HT14</strong></a>
<a href="./Images/Wax_12.png"><strong>HT15</strong></a>
<a href="./Images/Wax_14.png"><strong>HT16</strong></a>
<br>
<center><b>Cycle ON</b></center>
<div id='your_div'>
<img src='./Images/Wax_2.png' alt='' height="1000" width="100%" />
<img src='./Images/Wax_4.png' alt='' height="1000" width="100%" />
<img src='./Images/Wax_6.png' alt='' height="1000" width="100%" />
<img src='./Images/Wax_8.png' alt='' height="1000" width="100%" />
<img src='./Images/Wax_10.png' alt=''height="1000" width="100%"  />
<img src='./Images/Wax_12.png' alt='' height="1000" width="100%" />
<img src='./Images/Wax_14.png' alt='' height="1000" width="100%" />
</div>
<!--<a href="./CycleSlurry.html"><strong>Cycle</strong></a>-->
</center>
<script type="text/javascript">
(function () {
   var imgs = document.getElementById('your_div').getElementsByTagName('img'),
      index = 0;
   imgs[0].style.display = 'block';
   setInterval(function () {
      imgs[index].style.display = 'none';
      index = (index + 1) % imgs.length;
      imgs[index].style.display = 'block';
   }, 1000);
}());
</script>
</body>
</html>
```

8.4 Arduino Data Logger Code

```
/*
Arduino Uno R3
Datalogger for Himco
by
Daniel Lettiere
 */
#include <SPI.h>
#include <SD.h>
#define MUX_EN   7 // ADG608 MUX Enable
#define MUX_A0   4 // ADG608 Addr0
#define MUX_A1   5 // ADG608 Arrd1
#define MUX_A2   6 // ADG608 Addr2
void setup()
{
  pinMode(10,OUTPUT); // Hardware /SS, need to be
output
  pinMode(11,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(1,OUTPUT);
  pinMode(0,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(7,OUTPUT); // Enable pin on ADG608
  pinMode(4,OUTPUT); // A0 on ADG608
  pinMode(5,OUTPUT); // A1 on ADG608
  pinMode(6,OUTPUT); // A2 on ADG608
  digitalWrite(0,HIGH);
    digitalWrite(1,HIGH);
      digitalWrite(2,HIGH);
        digitalWrite(3,HIGH);
         digitalWrite(11,HIGH);
  digitalWrite(7,HIGH); // Enable on

  Serial.begin(9600);
  Serial.print("Initializing SD card....");
  if (!SD.begin(8)) {
    Serial.println("Card failed, or not present");
    // don't do anything more:
    return;
  }
  Serial.println("card initialized.");

  SPI.begin(); // Init SPI
}
//-------------------------
//-------------------------
void Select_Board_Number(unsigned char board)
{
  switch(board)
  {
case 1:  //Deselect All Boards, wait 25ms
  digitalWrite(9,LOW);
  digitalWrite(3,LOW);
  digitalWrite(2,LOW);
  digitalWrite(1,LOW);
  digitalWrite(0,LOW);
  digitalWrite(8,LOW);
  delay(25);
  digitalWrite(9,HIGH); //Select Board 1
  delay(25);


  break;
case 2:  //Deselect All Boards, wait 25ms
  digitalWrite(9,LOW);
  digitalWrite(3,LOW);
  digitalWrite(2,LOW);
  digitalWrite(1,LOW);
  digitalWrite(0,LOW);
  digitalWrite(8,LOW);
  delay(25);
  digitalWrite(3,HIGH); //Select Board 2
  delay(25);
  break;
case 3:  //Deselect All Boards, wait 25ms
  digitalWrite(9,LOW);
  digitalWrite(3,LOW);
  digitalWrite(2,LOW);
  digitalWrite(1,LOW);
  digitalWrite(0,LOW);
  digitalWrite(8,LOW);
  delay(25);
  digitalWrite(2,HIGH); //Select Board 3
  delay(25);
  break;
case 4:  //Deselect All Boards, wait 25ms
  digitalWrite(9,LOW);
  digitalWrite(3,LOW);
  digitalWrite(2,LOW);
  digitalWrite(1,LOW);
  digitalWrite(0,LOW);
  digitalWrite(8,LOW);
  delay(25);
  digitalWrite(1,HIGH); //Select Board 4
  delay(25);
  break;
case 5:  //Deselect All Boards, wait 25ms
  digitalWrite(9,LOW);
  digitalWrite(3,LOW);
  digitalWrite(2,LOW);
```

```
   digitalWrite(1,LOW);
   digitalWrite(0,LOW);
   digitalWrite(8,LOW);
   delay(25);
   digitalWrite(0,HIGH); //Select Board 5
   delay(25);
   break;
  case 6:  //Deselect All Boards, wait 25ms
   digitalWrite(9,LOW);
   digitalWrite(3,LOW);
   digitalWrite(2,LOW);
   digitalWrite(1,LOW);
   digitalWrite(0,LOW);
   digitalWrite(8,LOW);
   delay(25);
   digitalWrite(8,HIGH); //Select SD Breakout
   delay(25);
   break;
  }
}
void Set_Mux_Channel(unsigned char chan)
{
  switch(chan)
  {
    //----------------------
    // MUX Channel 1
  case 1:
    digitalWrite(MUX_A0,LOW);
    digitalWrite(MUX_A1,LOW);
    digitalWrite(MUX_A2,LOW);
    break;
    //----------------------
    // MUX Channel 2
  case 2:
    digitalWrite(MUX_A0,HIGH);
    digitalWrite(MUX_A1,LOW);
    digitalWrite(MUX_A2,LOW);
    break;
    //----------------------
    // MUX Channel 3
  case 3:
    digitalWrite(MUX_A0,LOW);
    digitalWrite(MUX_A1,HIGH);
    digitalWrite(MUX_A2,LOW);
    break;
    //----------------------
    // MUX Channel 4
  case 4:
    digitalWrite(MUX_A0,HIGH);
    digitalWrite(MUX_A1,HIGH);
    digitalWrite(MUX_A2,LOW);
    break;
    //----------------------
    // MUX Channel 5
  case 5:
    digitalWrite(MUX_A0,LOW);
    digitalWrite(MUX_A1,LOW);
    digitalWrite(MUX_A2,HIGH);
    break;
    //----------------------
    // MUX Channel 6
  case 6:
    digitalWrite(MUX_A0,HIGH);
    digitalWrite(MUX_A1,LOW);
    digitalWrite(MUX_A2,HIGH);
    break;
    //----------------------
    // MUX Channel 7
  case 7:
    digitalWrite(MUX_A0,LOW);
    digitalWrite(MUX_A1,HIGH);
    digitalWrite(MUX_A2,HIGH);
    break;
    //----------------------
    // MUX Channel 8
  case 8:
    digitalWrite(MUX_A0,HIGH);
    digitalWrite(MUX_A1,HIGH);
    digitalWrite(MUX_A2,HIGH);
    break;
    //----------------------
  default:
    break;
  }
}
//-------------------------
//-------------------------
int Read_Temperature_Board_1(void)
{
  unsigned int temp_reading1;
  // force conversion now..
  delay(5);
  digitalWrite(9,LOW); // Set MAX7765 /CS Low
  delay(5);
  digitalWrite(9,HIGH); // Set MAX7765 /CS High
  delay(250); // wait for conversion to finish..
  // read result
  digitalWrite(9,LOW); // Set MAX7765 /CS Low
  delay(1);
  temp_reading1 = SPI.transfer(0xff) << 8;
  temp_reading1 += SPI.transfer(0xff);
  digitalWrite(9,HIGH); // Set MAX7765 /CS High
  delay(1);
  // check result
  if(bitRead(temp_reading1,2) == 1) // No Connection
  {
```

```
   return(-1); // Failed / NC Error
  }
  else
  {
   return((int)(temp_reading1 >> 5)); //Convert to
Degc
  }
}
//-------------------------
//-------------------------
int Read_Temperature_Board_2(void)
{
  unsigned int temp_reading2;
  // force conversion now..
  delay(5);
  digitalWrite(3,LOW); // Set MAX7765 /CS Low
  delay(5);
  digitalWrite(3,HIGH); // Set MAX7765 /CS High
  delay(250); // wait for conversion to finish..
  // read result
  digitalWrite(3,LOW); // Set MAX7765 /CS Low
  delay(1);
  temp_reading2 = SPI.transfer(0xff) << 8;
  temp_reading2 += SPI.transfer(0xff);
  digitalWrite(3,HIGH); // Set MAX7765 /CS High
  delay(1);
  // check result
  if(bitRead(temp_reading2,2) == 1) // No Connection
  {
   return(-1); // Failed / NC Error
  }
  else
  {
   return((int)(temp_reading2 >> 5)); //Convert to
Degc
  }
}
//-------------------------
//-------------------------
int Read_Temperature_Board_3(void)
{
  unsigned int temp_reading3;
  // force conversion now..
  delay(5);
  digitalWrite(2,LOW); // Set MAX7765 /CS Low
  delay(5);
  digitalWrite(2,HIGH); // Set MAX7765 /CS High
  delay(250); // wait for conversion to finish..
  // read result
  digitalWrite(2,LOW); // Set MAX7765 /CS Low
  delay(1);
  temp_reading3 = SPI.transfer(0xff) << 8;
  temp_reading3 += SPI.transfer(0xff);
  digitalWrite(2,HIGH); // Set MAX7765 /CS High
  delay(1);
  // check result
  if(bitRead(temp_reading3,2) == 1) // No Connection
  {
   return(-1); // Failed / NC Error
  }
  else
  {
   return((int)(temp_reading3 >> 5)); //Convert to
Degc
  }
}
//-------------------------
//-------------------------
int Read_Temperature_Board_4(void)
{
  unsigned int temp_reading4;
  // force conversion now..
  delay(5);
  digitalWrite(1,LOW); // Set MAX7765 /CS Low
  delay(5);
  digitalWrite(1,HIGH); // Set MAX7765 /CS High
  delay(250); // wait for conversion to finish..
  // read result
  digitalWrite(1,LOW); // Set MAX7765 /CS Low
  delay(1);
  temp_reading4 = SPI.transfer(0xff) << 8;
  temp_reading4 += SPI.transfer(0xff);
  digitalWrite(1,HIGH); // Set MAX7765 /CS High
  delay(1);
  // check result
  if(bitRead(temp_reading4,2) == 1) // No Connection
  {
   return(-1); // Failed / NC Error
  }
  else
  {
   return((int)(temp_reading4 >> 5)); //Convert to
Degc
  }
}
//-------------------------
//-------------------------
int Read_Temperature_Board_5(void)
{
  unsigned int temp_reading5;
  // force conversion now..
  delay(5);
  digitalWrite(0,LOW); // Set MAX7765 /CS Low
  delay(5);
  digitalWrite(0,HIGH); // Set MAX7765 /CS High
  delay(250); // wait for conversion to finish..
```

```
 // read result
 digitalWrite(0,LOW); // Set MAX7765 /CS Low
 delay(1);
 temp_reading5 = SPI.transfer(0xff) << 8;
 temp_reading5 += SPI.transfer(0xff);
 digitalWrite(0,HIGH); // Set MAX7765 /CS High
 delay(1);
 // check result
 if(bitRead(temp_reading5,2) == 1) // No Connection
 {
  return(-1); // Failed / NC Error
 }
 else
 {
  return((int)(temp_reading5 >> 5)); //Convert to
Degc
 }
}
//-------------------------
//-------------------------
void loop()
{
 int temperature = 0;
 while(1)
 {
  Select_Board_Number(1);
  for(int i = 1; i < 9; i++)
  {
   delay(25);
   Set_Mux_Channel(i);
   temperature = Read_Temperature_Board_1();
   if(temperature == -1)
   {
    Serial.println("N/C Board 1 Channel ");
   }
   else
   {
    Serial.print(temperature,DEC);
    Serial.println(" DegC B1 C ")
     ;
   }
  }
  Select_Board_Number(2);
  for(int i = 1; i < 9; i++)
  {
   delay(25);
   Set_Mux_Channel(i);
   temperature = Read_Temperature_Board_2();
   if(temperature == -1)
   {
    Serial.println("N/C B2");
   }
   else
   {
    Serial.print(temperature,DEC);
    Serial.println(" DegC B2");
   }
  }
  Select_Board_Number(3);
  for(int i = 1; i < 9; i++)
  {
   delay(25);
   Set_Mux_Channel(i);
   temperature = Read_Temperature_Board_3();
   if(temperature == -1)
   {
    Serial.println("N/C B3");
   }
   else
   {
    Serial.print(temperature,DEC);
    Serial.println(" DegC B3");
   }
  }
  Select_Board_Number(4);
  for(int i = 1; i < 9; i++)
  {
   delay(25);
   Set_Mux_Channel(i);
   temperature = Read_Temperature_Board_4();
   if(temperature == -1)
   {
    Serial.println("N/C B4");
   }
   else
   {
    Serial.print(temperature,DEC);
    Serial.println(" DegC B4 C");
   }
  }
  Select_Board_Number(5);
  for(int i = 1; i < 9; i++)
  {
   delay(25);
   Set_Mux_Channel(i);
   temperature = Read_Temperature_Board_5();
   if(temperature == -1)
   {
    Serial.println("N/C B5");
   }
   else
   {
    Serial.print(temperature,DEC);
    Serial.println(" DegC B5");
   }
  }
```

```
   // open the file. note that only one file can be
open at a time,
   // so you have to close this one before opening
another.
   File dataFile = SD.open("datalog.txt", FILE_WRITE);
   // if the file is available, write to it:
   if (dataFile) {
     dataFile.println("hello world!!!");
     dataFile.close();
     // print to the serial port too:
     Serial.println("Hello World!!");
   }
   // if the file isn't open, pop up an error:
   else {
     Serial.println("error opening datalog.txt");

   }
  }
}
```