Aaron Wu
July 2019

Automated Blast Furnace Monitor System Report

Project Overview

Shenwang Pioneer Tech. Corporation Beijing specializes in the simulation and monitoring of blast furnaces to improve quality and consistency of steel production. The monitor system includes a camera which is protected by a special container with nitrogen gas.

If the temperature exceeds its limit or the pressure of the nitrogen gas drops too low, the camera is at risk of damage. In this scenario, the camera needs to be removed from the blast furnace. Manual removal involves safety hazards including gas inhalation and heat damage. An automated system for removing the camera would bypass these hazards.

The original model uses programmable logic controllers to control the monitor system. Single-board microcontrollers such as Arduinos are a more versatile and economic alternative for implementing logic.

The scope of the project was to interface an Arduino with the existent monitor system, which includes a piston for removing the camera, ball valve for sealing the chamber, and dust scraper for protecting the camera lens. A communication system between Java and the Arduino for remote control was also intended.

Planning

With only two weeks to work, time was limited. Detailed diagrams were drawn and software was outlined while waiting for the Arduino and other components to be delivered.

Since the components of the monitor system operated on higher voltages such as 24V and 220V, the Arduino needed electric relays to control them with 5V outputs. Additionally, feedback from the piston and ball valve for opening or closing needed relays so that the Arduino could safely detect the signals. The connections between the Arduino's digital pins and relays were planned out.

Connections to different power sources were also planned out. Different parts used 5V DC, 24V DC, or 220V AC. The different wires from the piston, ball valve, and scraper needed to be connected to the corresponding power or relay correctly.

Different communications between Java and the Arduino were planned out. With a Bluetooth or WiFi module, an Arduino could communicate with Java using different libraries. An RXTX serial communication also provided a direct connection.

Using JFrame, the user interface for controlling the system through Java was outlined and generated. Without any of the proper connections, a framework for sending signals using buttons and reading signals to update status lights was developed in preparation. JFreeChart was also implemented to display pressure and temperature data over time.

Construction

After the components arrived, the Arduino was connected to the rest of the circuit. However, due to a low current limit from the Arduino's digital output pins, a Arduino-specific relay module was needed to connect to the main relays. The number of wires needed to power the many relays as well as components resulted in a complicated circuit.

The pressure and temperature sensors needed to be interfaced with the Arduino. However, the Arduino only inputted 5V signals while the sensors outputted 0-24V. A circuit was used to convert the sensor output to current from 0-20 mA, which was then used with resistors to range from within the 0-5V boundaries. The Arduino software converts the raw voltage readings into the corresponding pressure or temperature values using a mapping function.

The Bluetooth and WiFi modules were unable to properly connect with Java due to library errors, so a serial connection using the RXTX Java libraries was used to establish a connection. Signalling between the Arduino and Java program began to function correctly after a few test programs. Since the serial communication requires a wired connection, it serves as a prototype for communicating between the system and a remote Java program.

The communication system was further developed, with bytes sent to the Arduino as signals while String were transmitted to Java as data. The framework already constructed allowed the communication system to be completed with relative ease.

Outgoing and incoming signals were fully integrated into the Java JFrame user interface. Accurate status lights were implemented for the position of the piston and ball valve, while buttons could be used to manually control movement. The logic was implemented for the Arduino based on a set of instructions ensuring the proper functionality, using feedback signals from the machinery.

Testing

After the circuitry was finished and software completely developed, the electrical components were connected the mechanical parts. Both physical buttons directly connected to the circuit as well as the virtual buttons in the remote Java interface were tested to output the correct signals to the machinery. Each motion, as well as combination of motions, was tested for logic errors.

Physical lights in the circuit as well as virtual in Java represented the status of the machinery. Tests using relays as well as the fully connected mechanical system were conducted to ensure the status lights were accurate. Due to the Arduino's inability to multitask, the status lights needed to be updated regularly even while waiting for machinery to finish moving.

Lastly, the pressure and temperature sensors were connected and data was transmitted to Java to be represented on charts. The readings showed expected values for both sensors. When the temperature sensor was tested using hot water, the machinery responded correctly by withdrawing the camera and closing the valve.

Conclusions

The project demonstrated impressive functionality given the short time-constraint for development. The Arduino and the rest of the circuit were fully interfaced, creating room for even more additions in the future. Although remote control was not fully realized due to the constraint of a direct electrical connection, the communication framework can be easily applied to a Bluetooth or WiFi module for the Arduino in the future. A combination of proper time-management, developed planning, and dedication allowed for completion.

Many obstacles were encountered throughout development. The Arduino's relatively low functioning voltage created many issues with connecting the machinery and sensors, but clever circuit design helped connect the Arduino. Bluetooth and WiFi connections were not working properly, so a physical serial connection was used as back-up. Perseverance and creative ideas, along with information from the Internet, were crucial in overcoming these obstacles.

The management of tasks, from planning to testing, helped ease the workload tremendously. Developing software while waiting for hardware made the process much smoother once the hardware arrived. Constructing frameworks to use was useful for the future, even if they needed to be changed drastically.

A combination of proper time-management, developed planning, and dedication allowed for the project's success. These skills will be useful for future endeavors.