Miami University
College of Liberal Arts and Applied Sciences
Department of Engineering Technology

ENT 497/498
Senior Project

Clippard

Remote Machine Monitoring System

Ben Arron and Joshua Lydy

Reza Abrisham Baf

May 1st, 2020

Executive Summary

Clippard, A well-known company in Cincinnati Ohio that manufactures pneumatic valves, cylinders, and fittings, provided us with a challenge to build a system for them. This system will monitor 16 CNC machines during the off hours as they only work 1st shift. They are relying on the machines to stay running thought-out the afternoon and night. When a CNC machine goes down it stays down till the morning when a technician comes in to restart it. This leaves a substantial amount of unnecessary down time for the company.

They have requested a system to be designed and built to alert a technician by text or email when either a machine is down or if the fire suppression system is triggered. Our group brainstormed several different ideas of how to get two signals to a computer through the internet to phone or e-mail. Given the company's environment around each machine, we concluded that wireless signals through the WiFi system would be the best fit.

The machine monitoring system will work with an ESP8266 NodeMCU that will be wired into each machine to communicate with the server. Using a digital input on the chip (GPIO pins) will give us our signal. The ESP8266 chip is programmable to connect to Wi-Fi then using IEE 802.11 protocol. This signal will be a publish only to the MQTT Mosquitto broker which is installed on the Raspberry pi. Node-Red will then receive and translate the signals to be displayed on the dashboard and be sent as notifications in the form of email or text message. Node-Red is a flow-based development tool for visual programming. Node-Red will process the data and if an alert is detected, it will send the email or text as it is programmed to do. The Mosquitto MQTT broker along with Node-Red will be hosted on a Raspberry Pi 4. The Raspberry Pi a low cost, credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse.

Table of Contents

1. Scope and Methodology

Clippard is a manufacturing company in Cincinnati, Ohio, that makes all different types of valves and cylinders for industrial use. The majority of what they produce are small valve assembly parts all of which are manufactured in house. They use 16 CNC machines and some run 24/7. They have come to notice the loss of production time and money due to the downtime from running the machines unattended overnight. They feel by implementing a monitor system will help fill orders and meet deadlines faster. This is why they have come to us to implement a machine monitoring system that they could check on a website to see the statues of each machine and ping the operator if a machine shuts down. The three key aspects they would like to monitor is run time, cycle count, and fire suppression. They have an assortment of three different types of CNC machines that they will be working with. The layout of the company's plant having different machines 6 Ganesh, 7 Citizens, and 2 Star Machines spread throughout one building. The plant has a large amount of oil and cutting grease so any part must hold up to the grease and grime from this industry. This production data can be used in the future to understand loss of production and miss opportunities.

Board Design

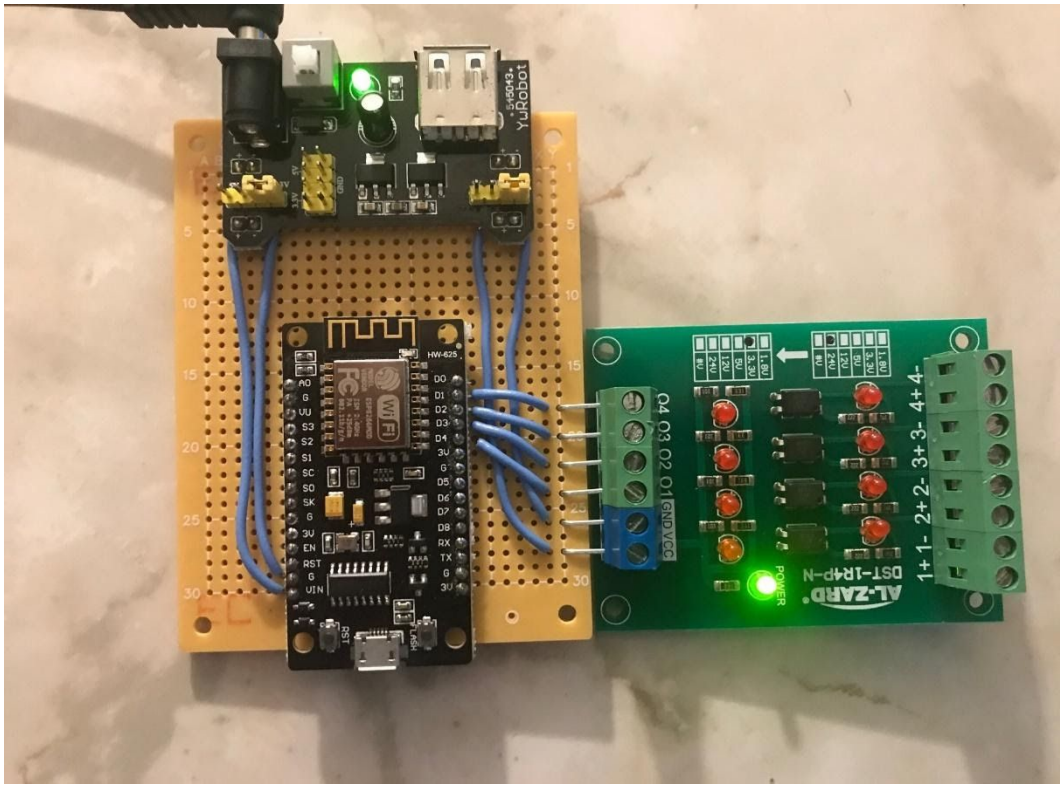

Figure 1

2. Design and Development

The design aspect that we planned on implementing is a Wi-Fi connection to monitor the status of 2 inputs. We were thinking about installing a breakout relay board to handle all the input from each machine. But given the environment around the machines we felt that is would be the best option. The ESP8266 is a little chip that can be installed in any little compartment that has 110V AC to power the chip and optocoupler board. This makes it very nice when dealing with a dirty environment. The power supply helps regulate a constant 5Volt DC to power the chip. We did a current analysis for each part and found that we are only using 90mA of the 700mA supplied by the power supply.

Chart 1

Power Consumption of Circuit			
Current total Power supply can handle.			
Power Supply	**700 mA**	**0.700**	700mA
Circuit Analysis total Current draw			
Node MCU Chip	70mA	0.070	200mA
Led	20mA	0.020	20mA
DST-1R4P-N breakout board	5mA	0.005	5A
	Total Used	**0.095**	90mA
	(Watts)-(Circuit) LEFT mA	**0.605**	605mA

There was the task of not having our signals connected to each machine. To solve this problem, we used a optocoupler board. The board be coupled by light and not electricity helps isolate the machine signal form the chips due to any damage from either. From the Machine

Design Board drawing there are four inputs. However, we are only using two of them the other are intended for future updates.

Wiring Diagram

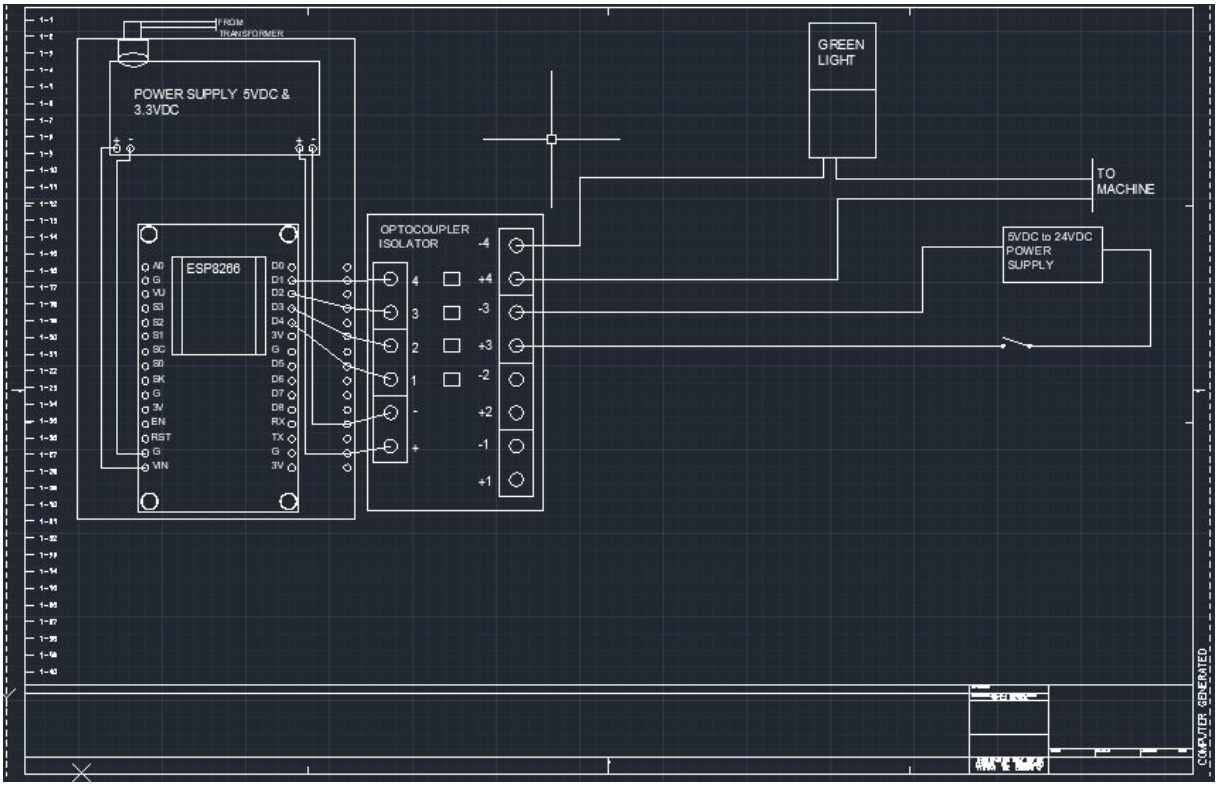

Figure 2

There are two signals that we felt that we can use for our inputs,the first one being the run light status. This signal will give us the run status of the machine and once that signal is published to the MQTT broker, we will have a date and time stamp sent along with it. The time will be kept in the graph in the Node-Red the longest that it can be set is 12 hours. Last is the fire suppression system, they will provide a relay that is normally closed for their fire system. Once this signal reads, the relay will open up and the input will be sent to the internet. We have noticed that the distance of the machine spread though-out the shop might induce a problem for the Wi-Fi. We original thought the signal distance between each machine might be a problem, But Lifewire

states “A general rule of thumb in home networking says that Wi-Fi routers operating on the 2.4 GHz band can reach up to 150 feet indoors and 300 feet outdoors. The machine in the plant are no more than ten feet from each other. Knowing that the signal reaches 150feet we will have no problem in the shop area.

3. ESP8266 NodeMCU

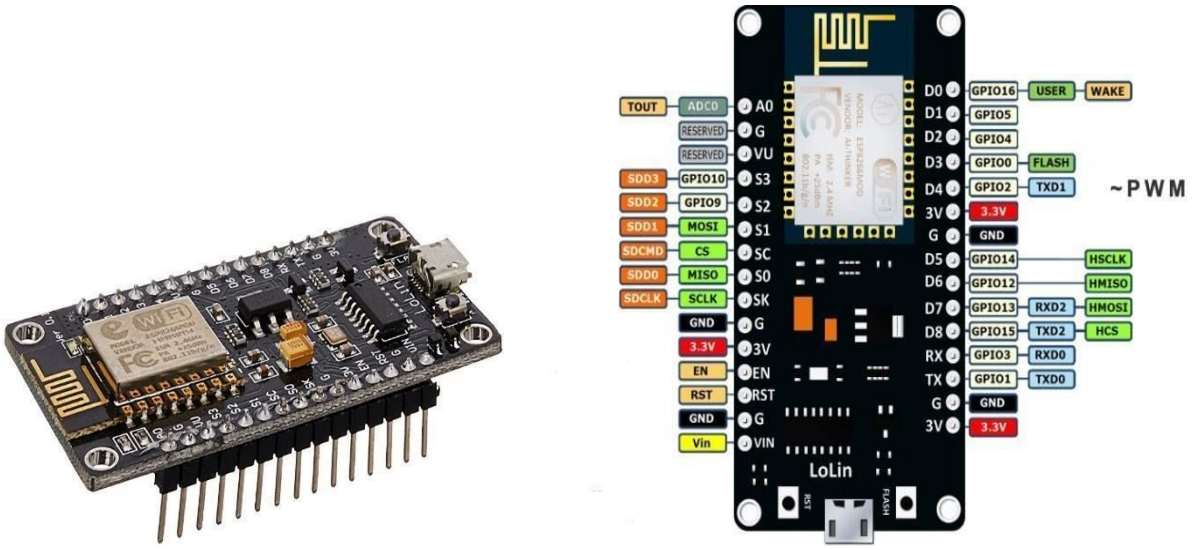

Figure 3

The ESP8266 Node-MCU is a programmable input/output that uses Arduino Uno programing software. We are using the D1-4 GPIO pins the GPIO number is what is used in the program. The pins only need 5V dc to change state. The software uses a Client.publish protocol in the program to connect to through Wi-Fi to the server. The chip has 12 input /outputs that can programed either way using a mini USB connection. To establish connection with the ESP8266 there are instructions in the Appendix.

The ESP8266 chip needs to be programmed before connecting.

The ESP8266 will connect using an IP address to send input data wirelessly in the local network. The input data will be published though an MQTT broker online that will be installed on the Raspberry pi. The program has a call back function in the program so if the plant loses power at any time, the chip will reset when powered back up and reestablish connection. We have not done any further testing to see how long the chip can be off without disrupting the Wifi connection. If this happens, the program may need to be reloaded on the chip.

4. Using the Broker (Mosquitto)

The ESP8266 works on a Publish/Subscribe command in the program. We are publishing the Topic and status to the MQTT. We are using the MQTT Mosquitto that will be installed on the Raspberry Pi. The Mosquitto broker then recognizes the topic and status for feather use. This all works be hinds the seams communicating with the ESP8266 to send and receive. This makes it suitable for Internet of Things messaging such as with low power sensors or mobile devices such as phones, embedded computers or microcontrollers. The Raspberry pi will Subscribe its data from the ESP8266 while the Chip will Publish. This make it a little confusing but if you think of it as playing catch. One is sending the ball and the other is receiving. Our data flow only works in one but all the sensor is doing is looking at the input. Below is a diagram of the publish being in Blue and the Subscribe being in green.

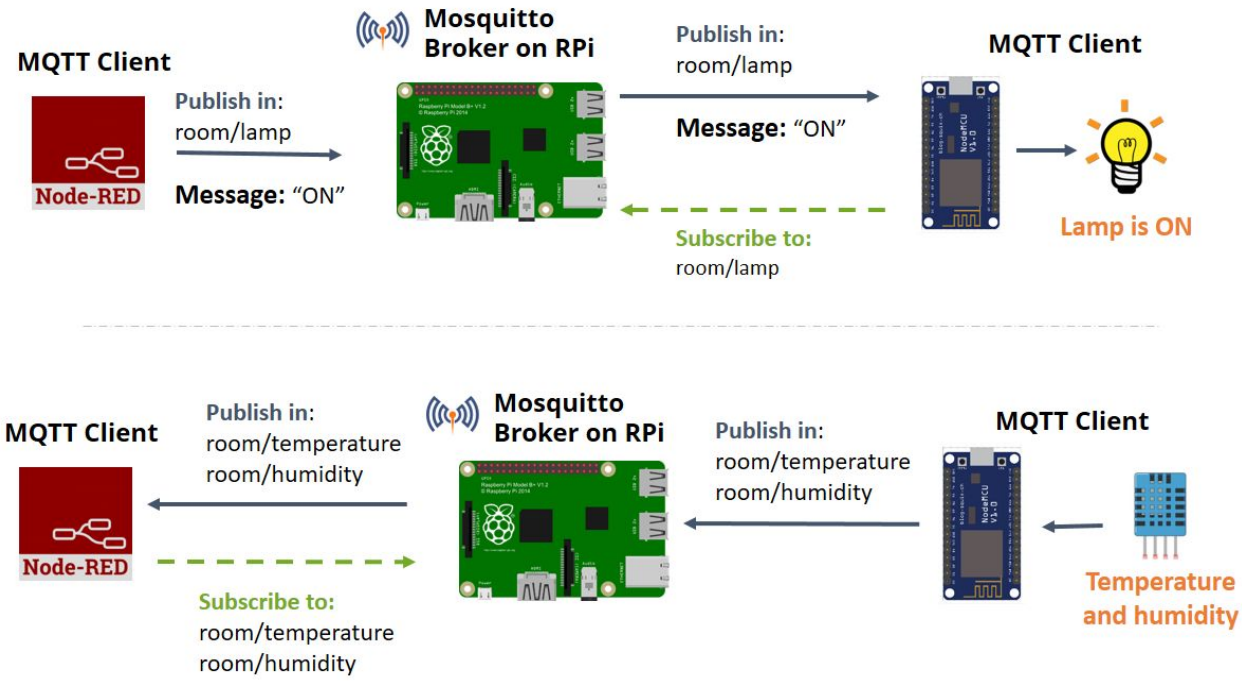

Figure 4

5. Node-Red

The Node-Red is a flow-based development tool for visual programming. Node-Red can be installed on Windows or on a mobile device like Raspberry Pi, Arduino, and Android. It is very easy to learn and use, especially for people with very little programming experience. The input into Node-Red can be from a website, social media, serial, web sockets or a microcontroller like we are using in our project. Node-Red processes the input data through the flow created by the user and onto the output. Outputs are what send the processed data to the outside world. These outputs can send data to twitter, email addresses, text messages, or simply deploy the data or message to the Node-Red console.

You will see in figure 4 the programming interface for node-red.

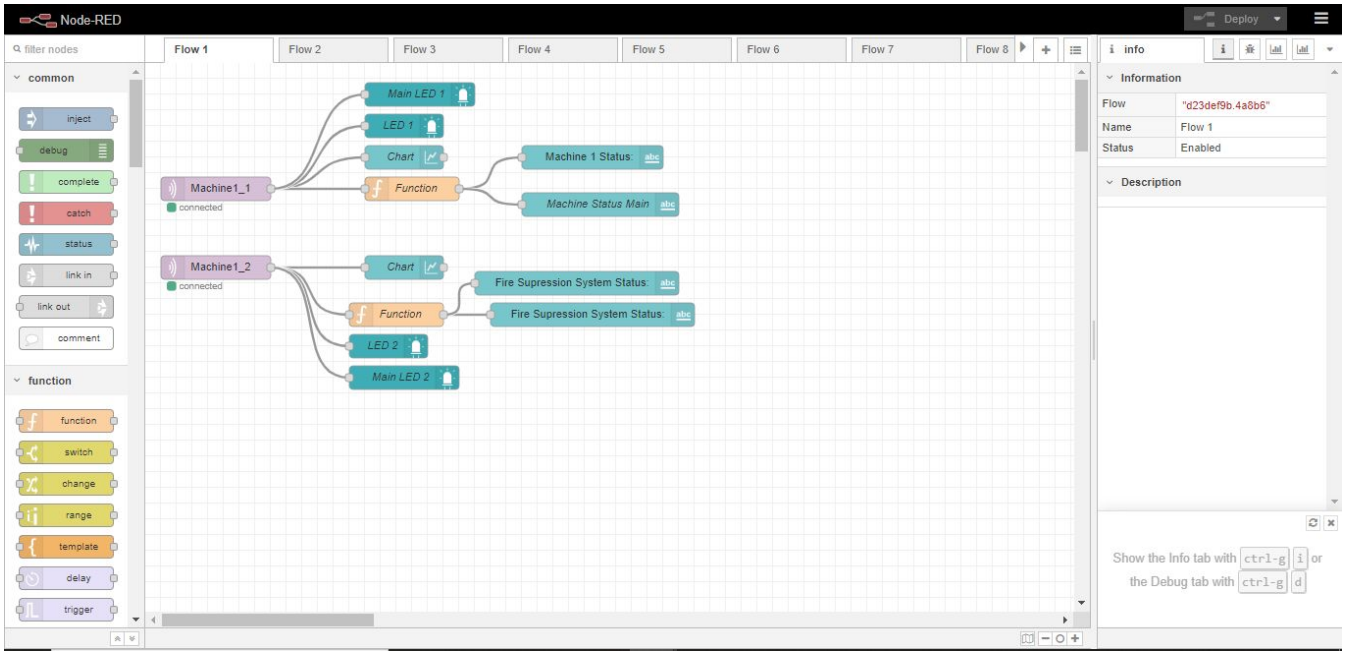

Figure 4

Node-Red flow boards can be confusing but are actually quite simple. The purple node that say “Machine1-1” means the incoming information is from machine 1 and input 1, which in our case, input 1 is machine status and input 2 is the fire suppression system. The LED nodes are the red and green lights for the dashboard. We have 2 nodes for each input that way there is a light for both the main dashboard and each machine page. The chart node is used to show the history which we have set up to display the past 12 hours of run time. The function node contains a section of code to take the incoming input and convert it say display on the dashboard whether the machine is running or not.

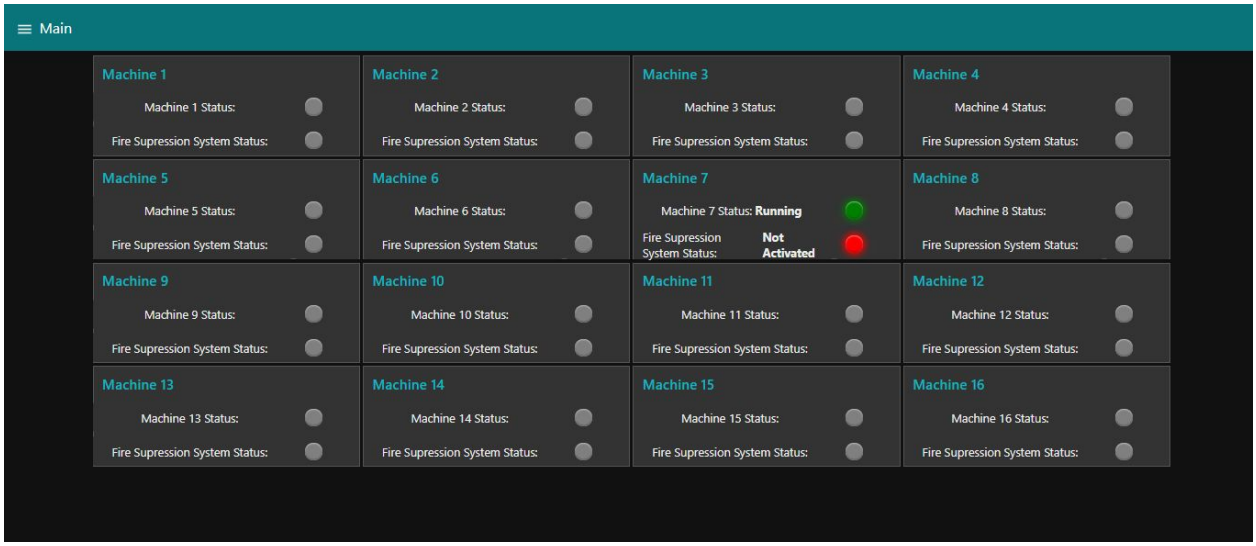

Figure 5

The HMI or Human machine Interface helps bring the user a better understanding of what the programing is controlling without all the technical know how. The HMI is designed to help the user change certain parameters though-out the program without changing the structure of the program. In our case, there is no need to change parameters as this project will be set-up as a monitoring system only.

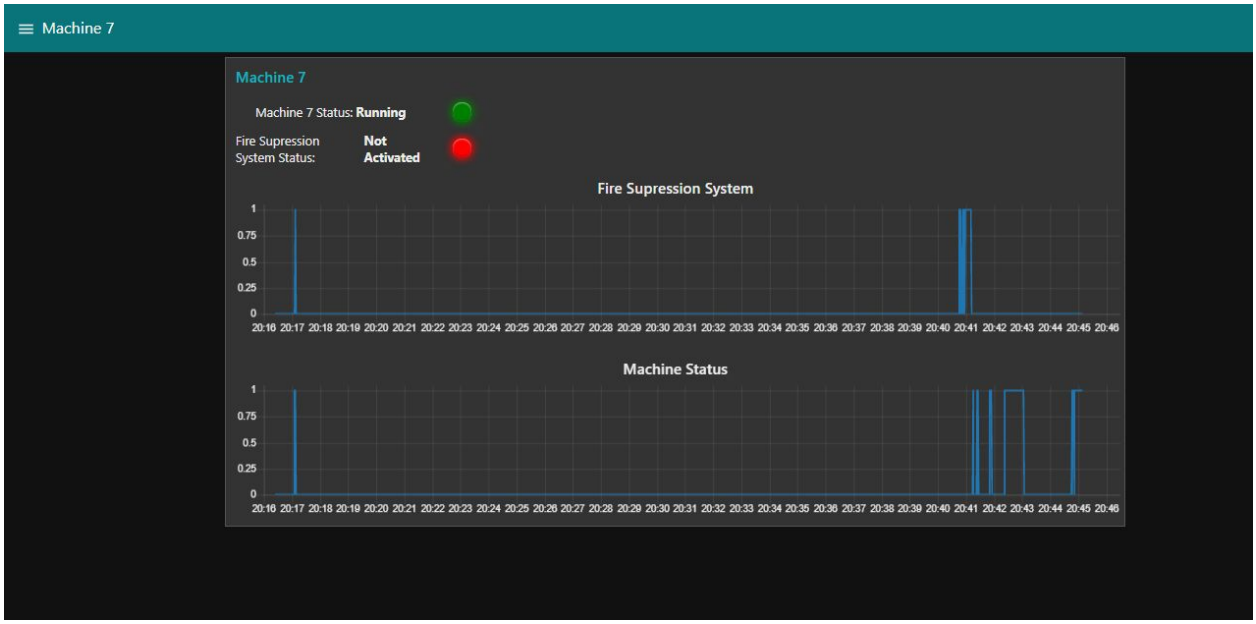

Figure 6

If you refer to figure 7 below you will see that the dashboard will be split into sections for each individual machine and then there will be a main page with the status of each machine for a quick glance.

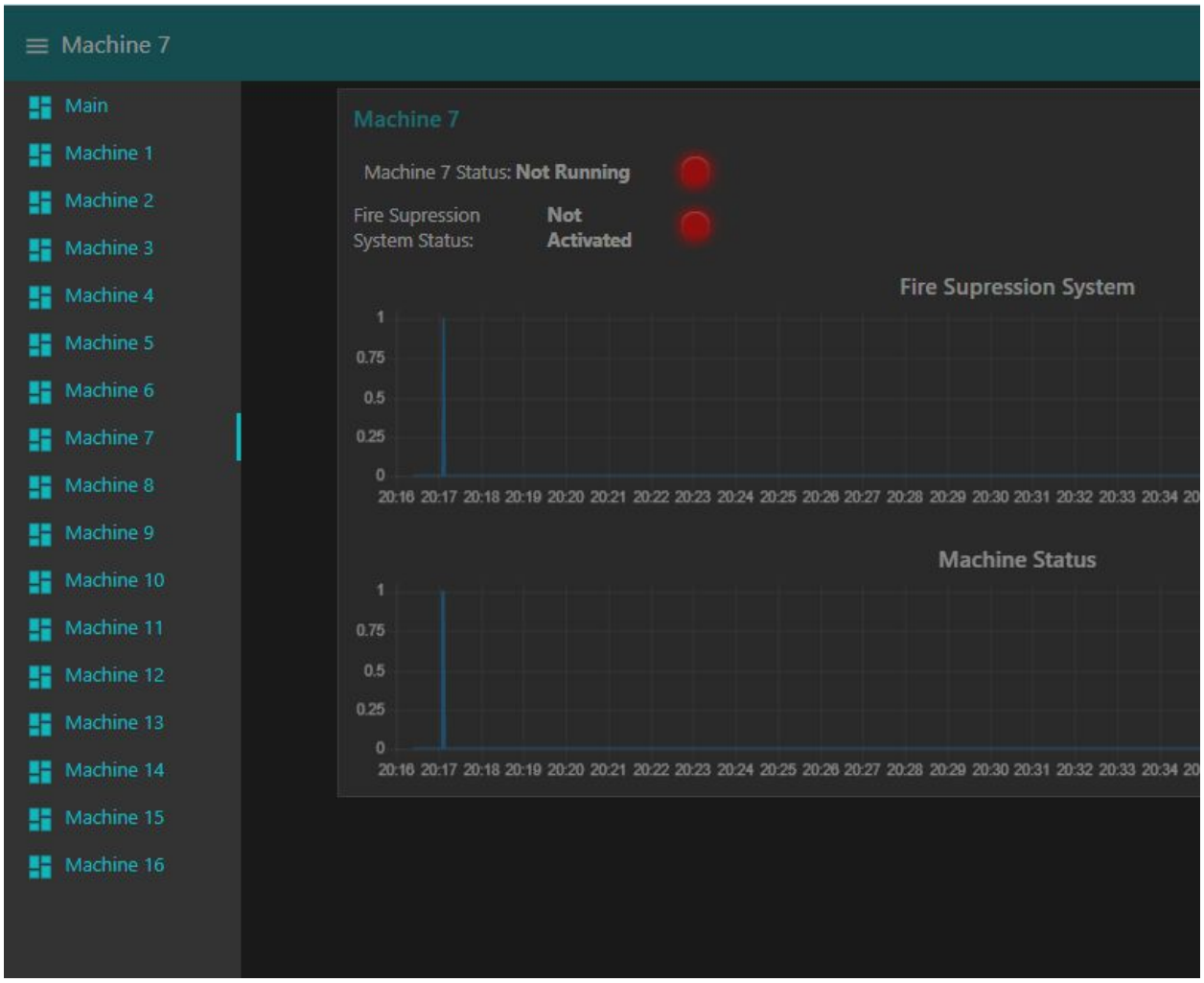
Machine 7
Main
Machine 1
Machine 2
Machine 3
Machine 4
Machine 5
Machine 6
Machine 7
Machine 8
Machine 9
Machine 10
Machine 11
Machine 12
Machine 13
Machine 14
Machine 15
Machine 16
Machine 7
Machine 7 Status: Not Running
Fire Supression System Status: Not Activated
Fire Supression System
1
0.75
0.5
0.25
0
20:16 20:17 20:18 20:19 20:20 20:21 20:22 20:23 20:24 20:25 20:26 20:27 20:28 20:29 20:30 20:31 20:32 20:33 20:34
Machine Status
1
0.75
0.5
0.25
0
20:16 20:17 20:18 20:19 20:20 20:21 20:22 20:23 20:24 20:25 20:26 20:27 20:28 20:29 20:30 20:31 20:32 20:33 20:34

Figure 7

Conclusion and Recommendation for Further Study

Remote machine monitoring is becoming an increasingly useful tool in most manufacturing operations in this competitive market. But with technology getting smarter and cheaper these days there is always a way to be more successful in manufacturing. Clippard manufacturing wants to stay ahead of its competition by running their CNC machines over multiple shifts. Without having technicians present during off shifts, this would cause excessive unnecessary downtime. Machine monitoring systems can help reduce the down time by having a technician on call to start them up. When a critical machine goes down and sits idle for any amount of time, that time is wasted and causes loss of money and missed deadlines. However, while having a technician notified of the issue will increase production and reduce down time. My partner and I believe that this project will improve productivity and stop unwanted downtime for the future.

This project has come a long way in the last 8 months but the possibilities are endless. We now have the system set-up with 2 inputs to display on a dashboard when the machine is running and if the fire suppression system has been activated. The main addition and probably an easier one is an email/messaging system to notify the technicians but the tricky part will be turning off during operating hours while the techs are in the shop. A part counter on the dashboard could be useful.

References

NodeMCU ESP8266 chip data:

https://nodemcu.readthedocs.io/en/master/#nodemcu-documentation

NodeMCU ESP8266 chip picture:

https://circuit-diagramz.com/wp%20content/uploads/2018/11/ESP8266-12e-Pinout-13.jpg/

MQTT web Broker: https://randomnerdtutorials.com/esp8266-and-node-red-with-mqtt/

https://mosquitto.org/

The evolution of WiFi standards: a look at 802.11 a/b/g/n/ac:

https://www.actiontec.com/wifihelp/evolution-wi-fi-standards-look-802-11abgnac/

Wi-Fi data : https://www.lifewire.com/range-of-typical-wifi-network-816564

Node- Red:

https://blog.techdesign.com/wp-content/uploads/2017/04/Node-RED-the-visual-wiring-tool.png

Node-Red: https://nodered.org/users/go-iot/NodeRedDashboard.png

Appendices

Clipper Machine Project Cost List				
# of parts	Discription	Cost $	total	
17	optocoupler 4 channel 12v/24v to 3.3v	$5.99	101.83	Amazon
18	3.3 - 5.5Volt power supply Module MB102 (Pack of 3)	$5.99	35.94	Amazon
17	Power supply cord (MC141019 - MULTICOMP AC/DC	$2.50	42.5	Newwark
17	ESP8266 NodeMcu Wifi Development Board	$5.00	85	Amazon
17	BreadBoard 2-3/4 3-1/16 #2760158	$3.50	59.5	RadioShack
1	Rasberry PI 4 (Using for a MQTT broker)	$67.99	67.99	Besybuy
	Software Node-Red	free		Download
	wire scrap wire	$ -		
		Total		
		392.76		