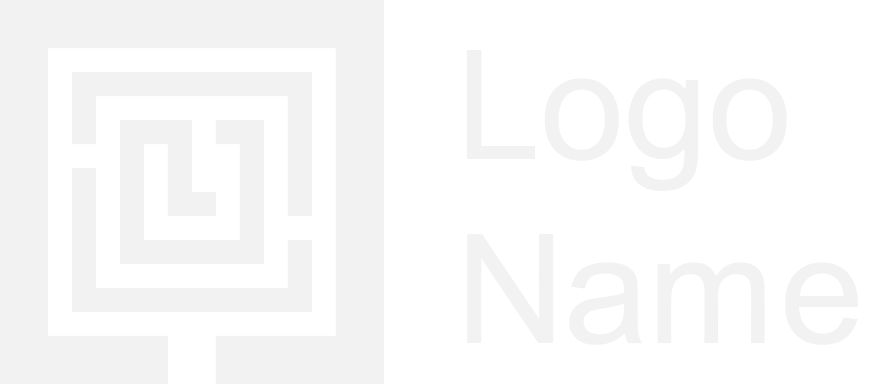


|  |
| --- |
| Automation  2021  text divider |
|  |
| august  text divider  Authored by:  Mason  Cory  Eamon  Royce |



|  |
| --- |
| Automated door part 1 Door design For our assessment our group decided to design an automated sliding door. This door will be like what you would see in a commercial or shop set up you see around modern buildings.   How we want the door to operate. Our door design consists of what would be a standard modern automatic door would work. With a combination of infrared sensors, timeouts and contact sensors so that the door knows when objects and the door itself is in the way.    The door will be attached via a glass to slider attachment.    the actual mechanism we want to be hidden in a mechanical space removable and accessible to a tradesperson or service professional   User initiated door commands We want our door to operate even if some major functions like the collision sensors or digital lock features decide to not work. A good door design will have manual settings to lock, open, close or stop that a user can use on the secure side of the automated door. Motor operation for open and close cycle The motor will be one of the most important components of the door. Not only will it need to be powerful enough to pull and push a glass plane, it will also need to be able to withstand 100’s of uses a day. It must stand the test of time and be low cost. We also want to add infrared sensors to the door so that there is feedback for collision events and sensors so that the door knows when it's wide open. This will allow for minimization of injury and digital errors if you have an analog sensor to tell the software that the door has stopped rather than relying on timers that could become misaligned. |
| *“Find even more easy-to-use tools on the Insert tab, such as to add a hyperlink or insert a comment”* |
| Operation flow diagram |

## Ladder Logic Diagram

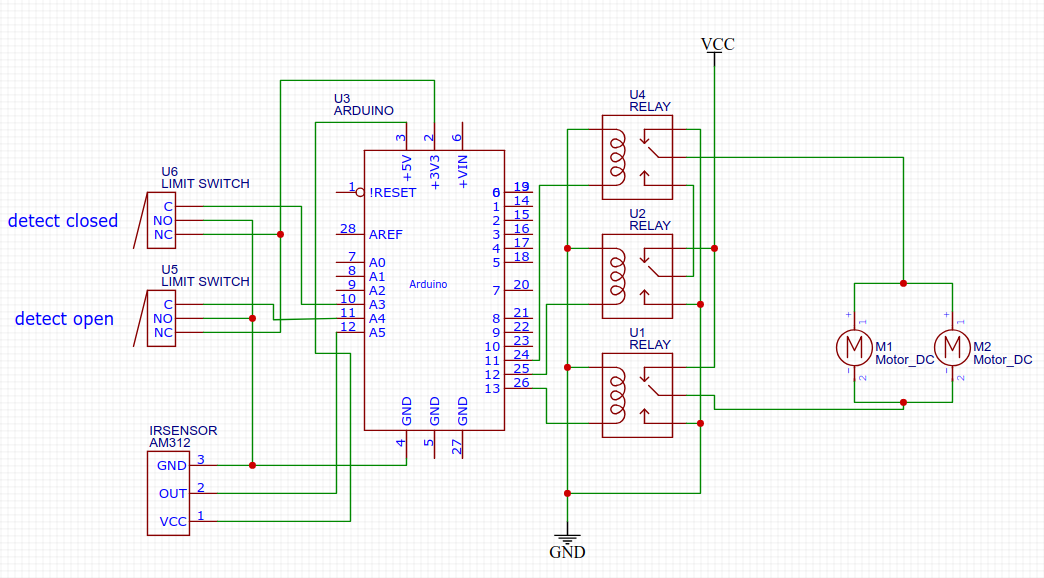


Our plan as a group was to make sure that the door opens and closes when safe conditions are met from both the users of the door and the doors safety functions. This meant that we wanted the door to know when it was fully open, fully closed, when someone was in proximity to the door and let the door make its “logical decision” based on the rules we set.

It is our plan to make sure that the door knows its own state of operation at all times and also knows what to do when a human is involved in the decision making of its day to day use.

You can see more of this in the code and diagrams provided in the “OpenPLC section of this report

## Circuit Diagram



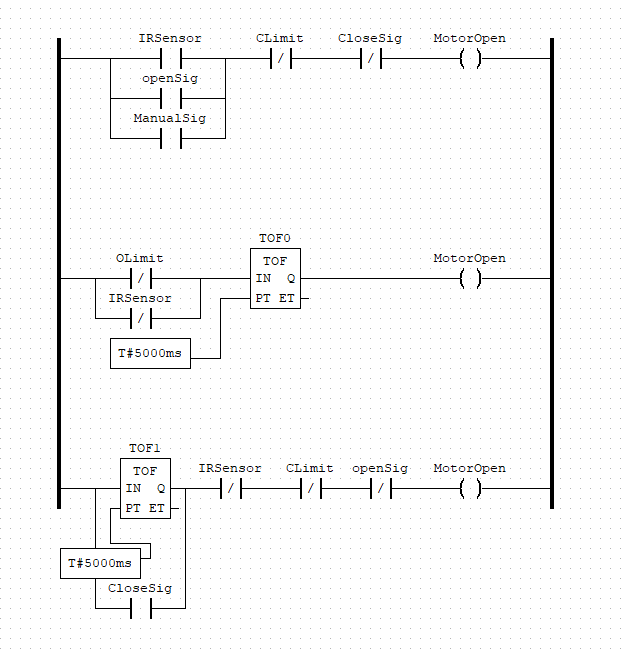
This diagram was created using the agreed requirements for a working door. This diagram functions the same as the above diagrams but also details what components will be required to turn those diagrams into a functioning prototype. The relay switches and arduino board simulate the inputs and outputs of the unipi, it is also using an incorrect input/output configuration that will be later changed to meet the hardware requirements when finalizing the OpenPLC diagram, because this configuration will not be what we submit as our working prototype it has been left in this configuration.

## 

## 

## OpenPLC design and code

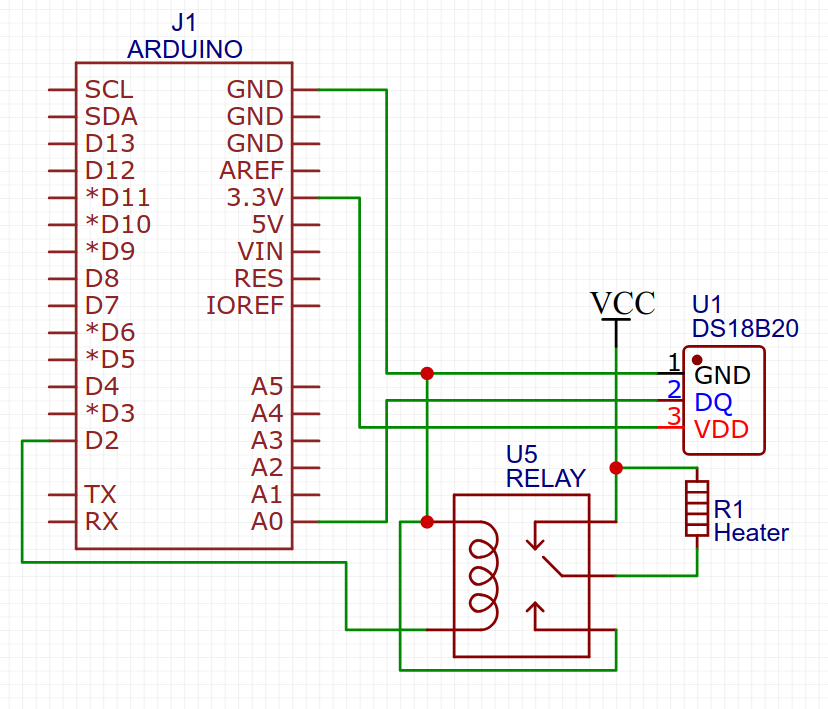
With OpenPlC being new software focused outside of class we are optimistic about how we went about making the code for our PLC. This was a direct extension of the ladder logic that we designed and simulated online.



This is the OpenPlc programming logic that we are running from the original ladder logic design, the only thing that we changed is the structure conditions for the opening gate to still be working.

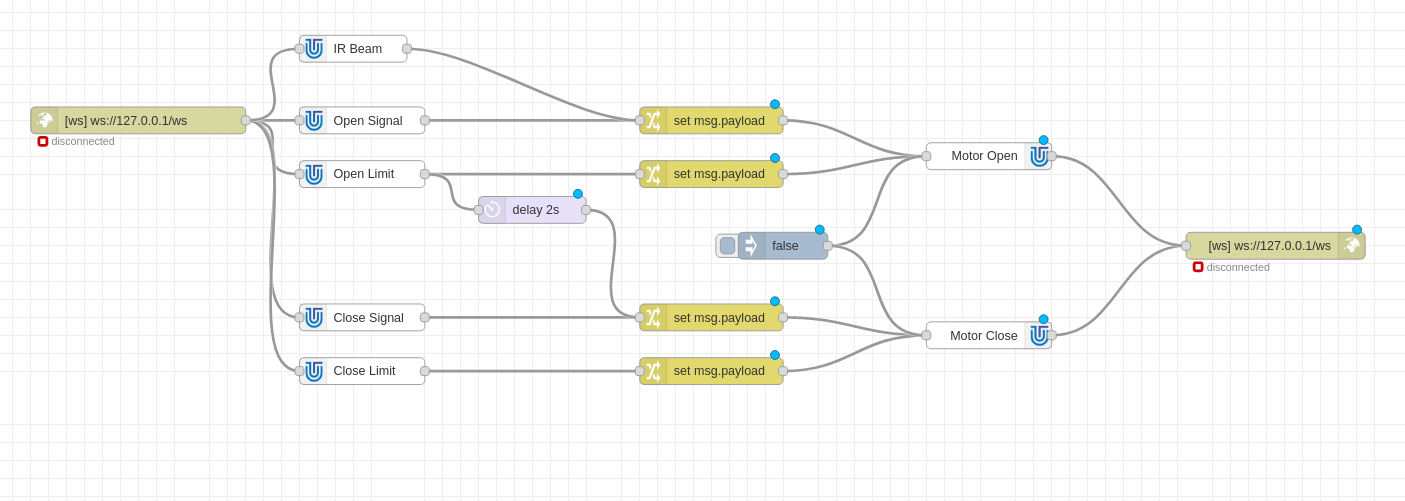
Our code will be attached to the email submission. if it does not work please don’t hesitate to ask for working code as we are not sure what is suitable for testing.

## Temperature circuit



This diagram uses an Arduino and relay circuit in place of a raspberry pi and unipi, and a DS18B20 temperature sensor connected via GPIO as opposed to an rj45 port (which is the connector used for building the circuit). The pins used to content the components to the raspberry pi are examples as the finished circuit will use the pins listed in the node red code.

# Node Red Gate Control



The node red program was a little intimidating at first, as we were doing this over a remote session and it made it hard to have that visual feedback.

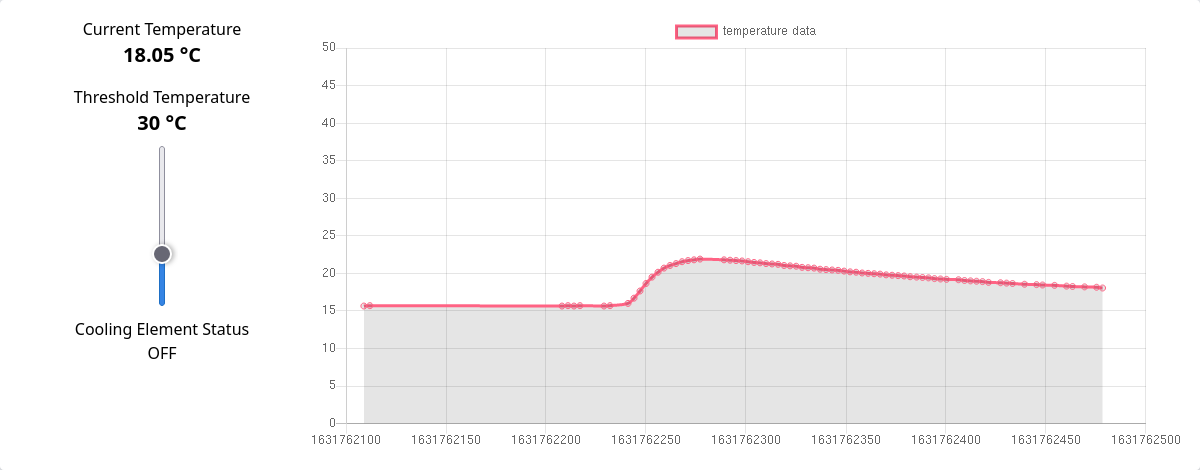
However, when we got the correct ports set up and the correct nodes working with the motor signals we got the gate working without problems.

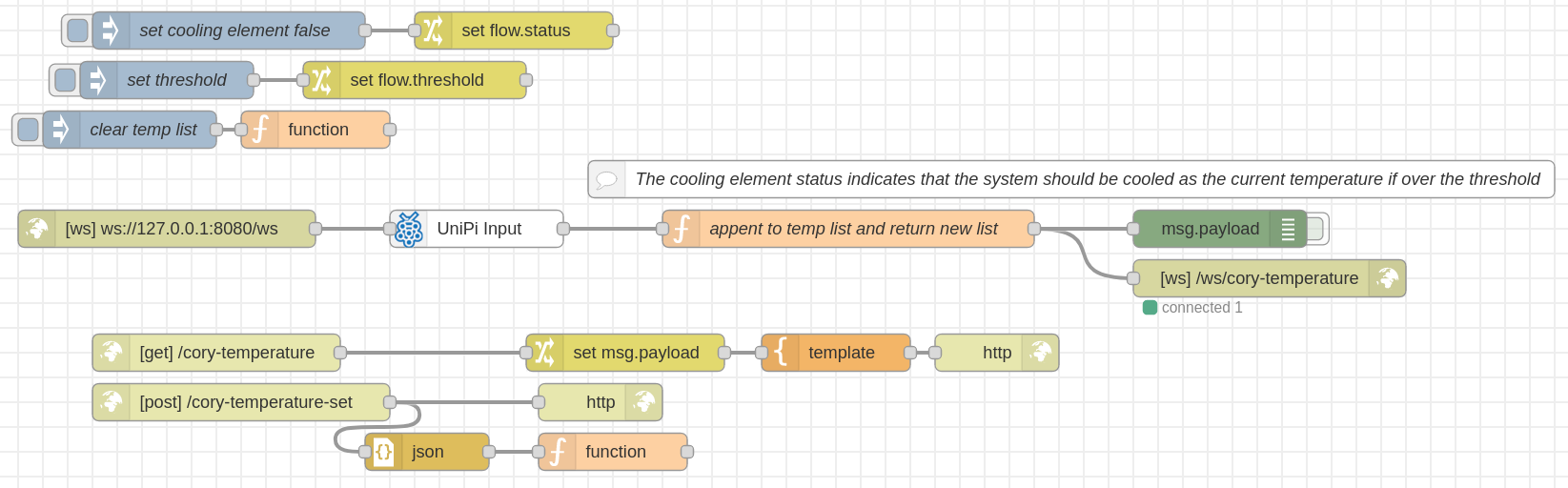
The process we used was walking through the steps we wanted the gate to go through when an event was triggered. This “logic ” was translated into node red and worked according to our wants and needs specified in the assessment.

## 

﻿

# Temperature Monitoring and Control





I (cory) made the interface for the temperature control with html, css and javascript. The chart is displayed using <https://www.chartjs.org/>. The historical data and current threshold is stored in node red under the *flow* object in javascript and sent to the client over a websocket. When the current temperature (which is the last entry in the temperature list) rises above the set threshold, the cooling element is enabled. The cooling element is not disabled until the current temperature drops 5℃ below the threshold. This is done so that the cooling element is not being spammed on and off if an equilibrium is met at the threshold temperature.