My progress so far in chronological order…

* Initial problem was connectivity issues because of the security protocols on Tufts wireless networks; I ended up using a separate router to get around these issues
* also, the linux system on the chip added to the ev3 was a newer version than what another graduate student at the CEEO was working on (with name changes, function edits, additions and fixes) so it was new territory
* Took some time to become familiar with the debian linux system, monobrick, ev3dev code, and the uses of and way to call many functions for both the sensors and the motors; focused particularly on the touch sensor and the motor
* Documentation for using the ev3dev material (and syntax for functions) was often sparse or hidden but I grew more familiar with it over the days
* Learned how to use python along the way; not terribly difficult but slowed me down in syntax
* Created a single thread Useless Machine program
* Used Pool from the python multi-processing library in an attempt to make the program concurrent or parallel (whichever was simpler)
* I initially tried to manually implement a concurrent form with a Queue and manual management of Workers but this option lacked many of the features I needed, such as the ability to join and close threads, so I switched to the automatic handling of Pool.
* After implementing that, I changed the program from being based on current state and instead made it rely on state changes
* The original code/program was created in Labview which I was not familiar with so I had to prod and speak with some Labview users and fellow students to see what was behind the logic (the implementation) and adapt that to python, the linux system, and Legoo sensor/motor functions.
* Got that to work successfully
* Then, began to add scanning for sensors and motors so the user does not have to input this information for the program to work. This is almost completely implemented.
* There are levels of goals. The first is to take an ev3 and have a physical input lead to a physical output. Since this is nearly implemented in python code, I am planning on adding another ev3 and having a button on one run a motor on the other. Then the next goal is to have a button press cause a web page to be opened or a message to be sent/posted to social media.
* Throughout the process so far my goal was to make the program as modular as possible so that it could be easily built upon, adapted for new uses and capabilities, and easily translated into a back end for the IFTTT group if/when our group interests were to merge partially at some future point. My code boils down to a main function that calls a ‘get info’ function for either sensors or motors (you can specify; I call it twice, once for each) which gathers all sensor and motor information for ports and types of sensors and motors.

This information is then sent to ‘interpret command,’ a function that contains commands (and with IFTTT could contain user input) in ‘if then’ format. All commands the user inputs (or in this case me) go directly into making this custom function. Nothing else needs to change.

* (On the side, I was also considering changing the logic slightly to allow motors and sensors to be added and removed during a program’s running by checking the ports repeatedly. This is a kid-friendly feature.)
* <https://docs.google.com/document/d/1IkEcJhH4C3PUL8MLB-D52iTWV8N_v-aHuJ1aZkMxkmA/edit?usp=sharing> – Here is the link to my comments, notes, sources used, etc. for this project so far; bolded text by a source indicates it was important and actively used