**A Brief Simulation of a Smart Home**

**Mikaela Cubon, Dawn Dixon, Abdullah Alotaibi**

Abstract Summary:

The topic chosen for our project was to simulate home automation by having an alarm trigger certain functions within our environment, which include: turning on lights, reading the weather, and playing music to wake the homeowner. A phone app would be used to set the alarm, and would also receive a warning if a smoke detector were to go off. The phone app would have alternate functions to manually turn the light and alarm music on and off at the users will.

Introduction:

Our project was designed with the following:

**Hardware**

* Arduino Uno x1
* Breadboard x1
* Arduino LED x1
* Arduino Speaker x1
* Android Smartphone x1 (A simulator could also be used)

**Software**

* Arduino IDE
* Node Red
* RedMobile – Node-RED on Android

Background Study:

When beginning our project, we were not sure how to approach it. We started off with our Arduino being our front end; we built a circuit and wrote code for it, and had it successfully running. It was tracking time based on the built in delay() function, which was not user friendly on its own, but converting the user inputs to work with the delay function would not have been troublesome. The problem was connecting our Arduino to read the weather.

We had completed an ICP that involved reading the weather with Node-RED. This gave us the idea to shift our front end over to Node-RED, have it control the alarm time and read the weather, and let the Arduino sit in the back end, running the circuits. We built up the Node-RED code, and had to do some studying on Text-to-Speech options that we could download, an easy way to extract the date and time, and how to set up our Node-RED to communicate with our Arduino circuits. We were able to successfully complete this goal.

Our next goal, which we were unable to meet in time, was to shift the font end again, to the phone app. Communicating the phone app to node red would be simple enough, as we discovered the RedMobile app, which could connect an android device to Node-RED. This would ultimately shift our project to being very user friendly, having both the Arduino and Node-Red being our backend, and requiring the user to only set the date/time and flip some switches to control our home automation environment.

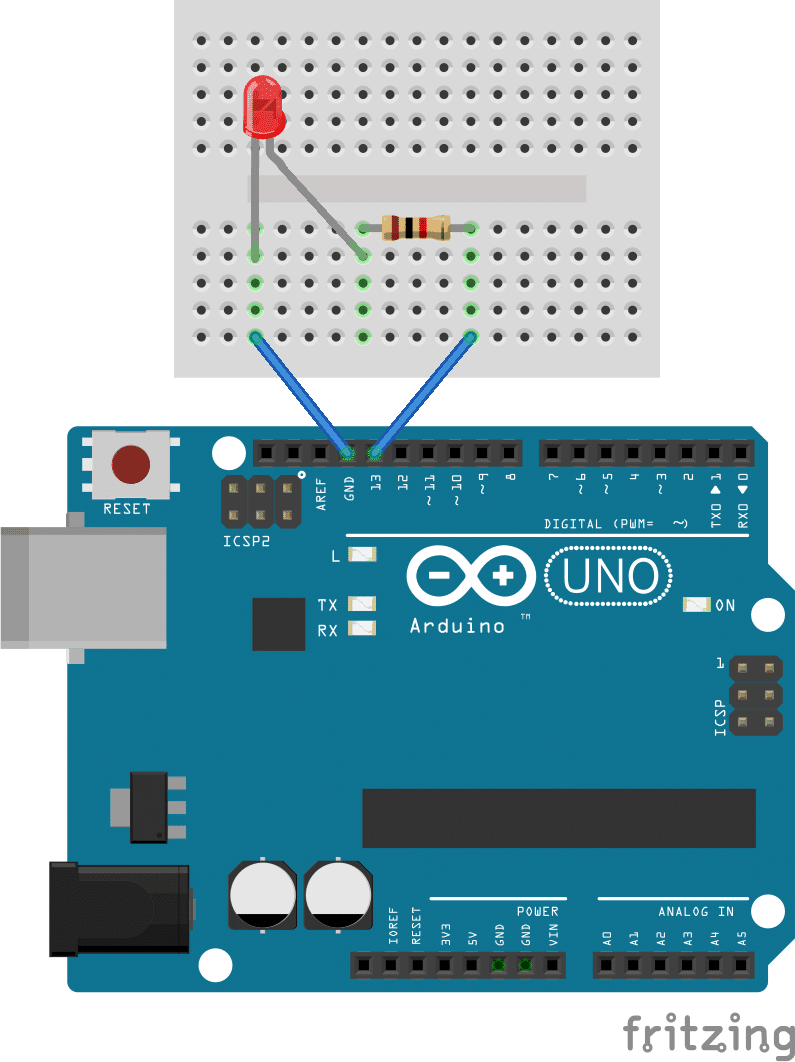
A lot of time spent trying to figure out how to allow the parts to communicate with each other, which nodes to use, and what code was going to be running on the Arduino. This study was where the majority of our time was spent.

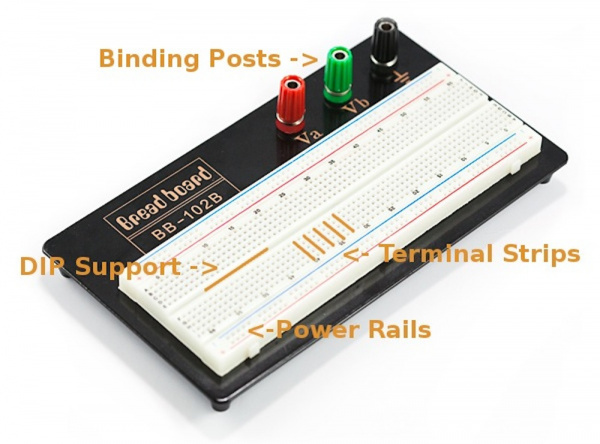
Methodology:

We first started with an Arduino based alarm that would work on its own.

To begin, we had to figure out what the circuit would look like and what parts we needed.

We started simple, wire our light.



Looking up the circuit of a light like the one shown above was one thing, but to build our circuit entirely, we needed to understand how a breadboard carried its current.

It had terminal strips, power rails, and a DIP support ravine. The terminal strips carried current along the same row, while the power rails offered easy access to voltage and ground. The DIP support provided easy isolation for each leg of our components. So our light circuit ended up looking something like this:

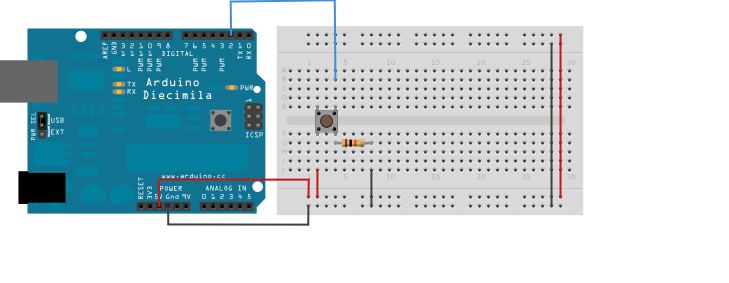
digital pin 13 ground

C:\Users\Mikaela\Documents\Robotics\Project\LEDNOLEG.PNG

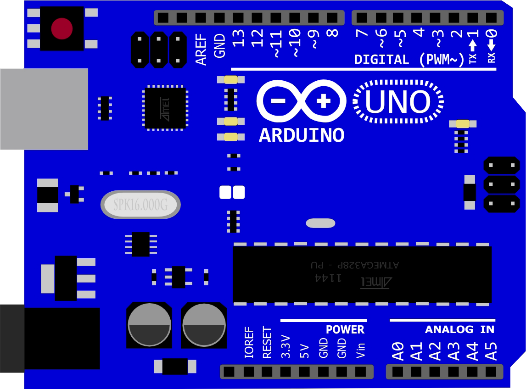
C:\Users\Mikaela\Documents\Robotics\Project\RESISTORNOLEG.PNG

The light would be turned on and off according to HIGH and LOW inputs from the Arduino IDE to pin 13. The light would receive a HIGH command once our delay() function reached its input time.

Next we had to add a button. This would let the user turn the light back to LOW and reset the alarm



So we put it together.



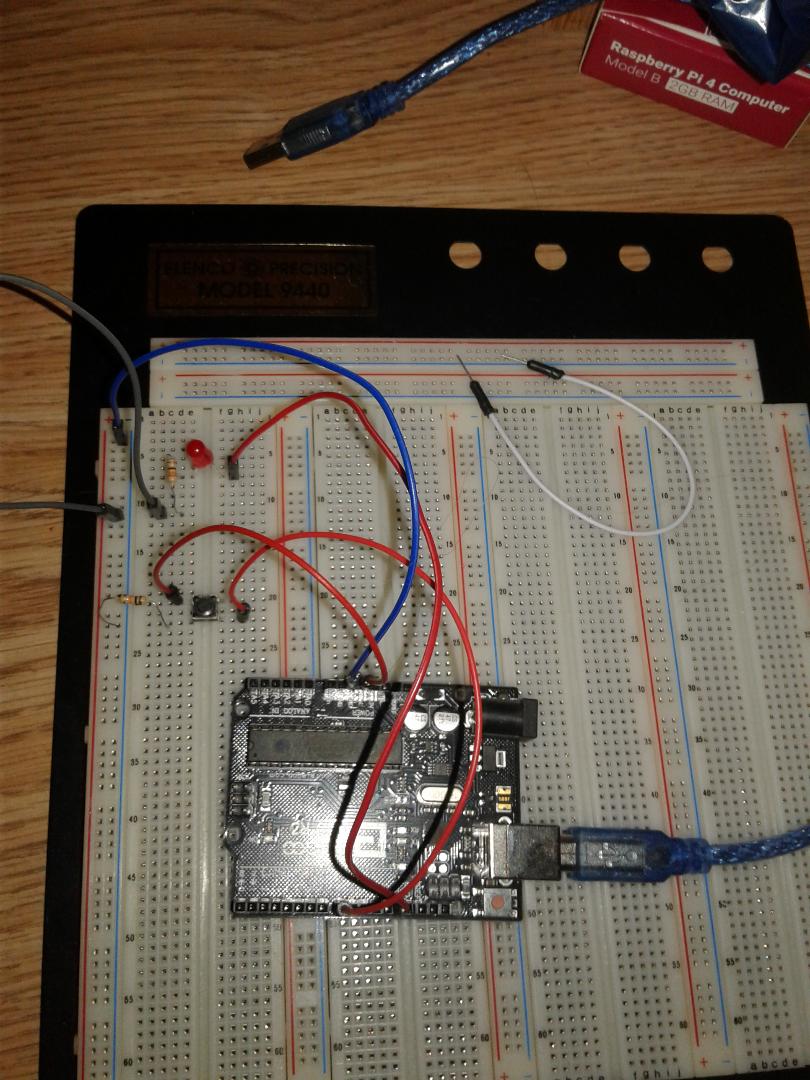


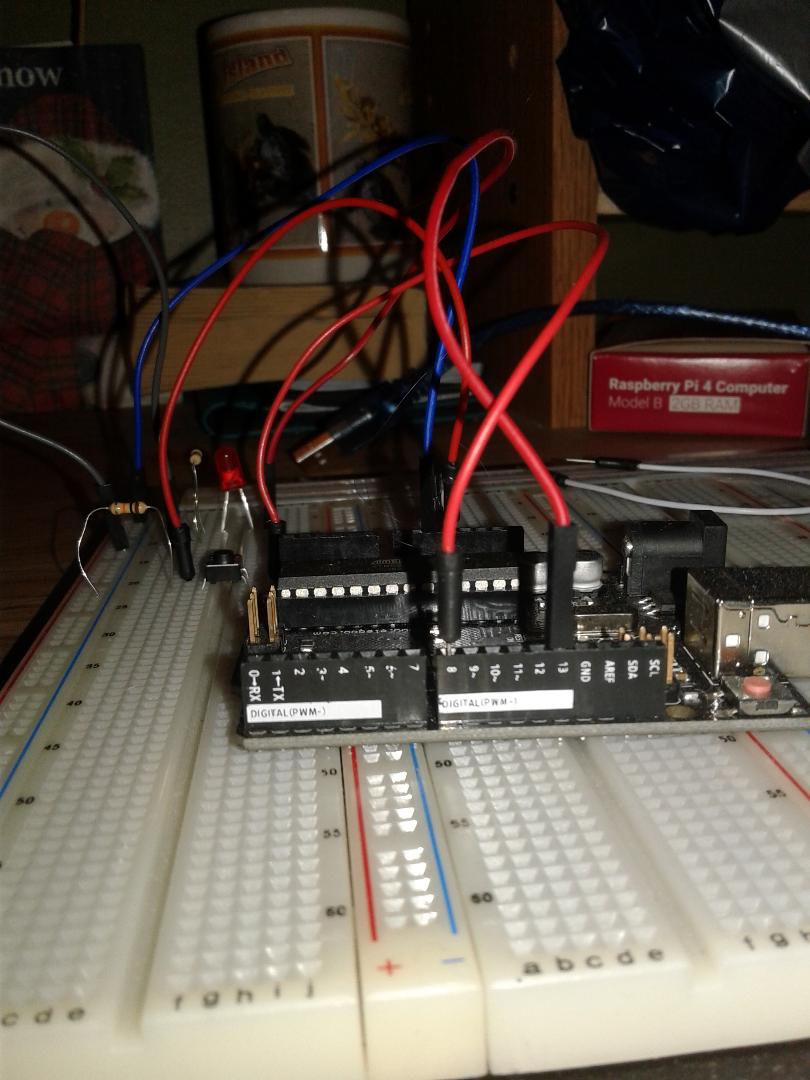
C:\Users\Mikaela\Documents\Robotics\Project\LEDNOLEG.PNG

C:\Users\Mikaela\Documents\Robotics\Project\PICS\BUTTON.PNGC:\Users\Mikaela\Documents\Robotics\Project\RESISTORNOLEG.PNG

C:\Users\Mikaela\Documents\Robotics\Project\RESISTORNOLEG.PNG

And this was our final circuit.





The next part was the code. We needed to know:

* How to read in the pin inputs.
  + We used digitalRead(13) for the light and digitalRead(pushButton) for the button, where pushButton was set to 8 for our pin code. We wanted to keep the light pin as a constant, immuatable 13, because we could see immediately once the Arduino was plugged in if the light was wired correctly or not. These were basic concepts we already knew from our ICPs
  + We needed to know how to stall our loop until our button was pressed. We used the statement:

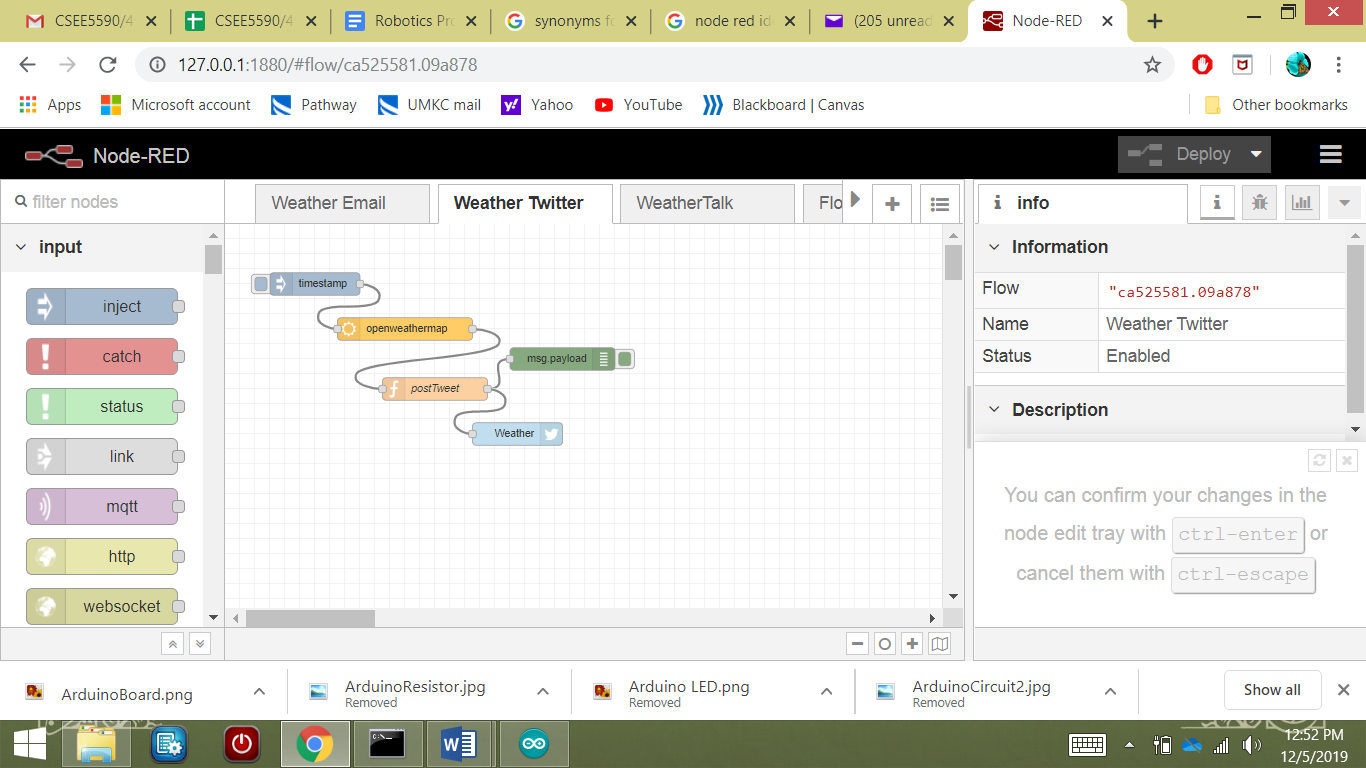
while (digitalRead(pushButton) == LOW)

This caused the program to stop until it was pressed

* How to take in proper user input
  + We used a simple string search with isDigit(). If any input the user made was not a number, it was not taken.
* How to reset the alarm for new input once it was turned off
  + Flag logic was used here. The flag started false, and when false, would ask for user input for the alarm. Once serial input was read, the flag would be set to true if it was valid (meaning it would not ask for input again until the alarm was finished and the button was pressed), or would remain false with improper input, meaning it would loop again back to asking for input so that the user could try again.

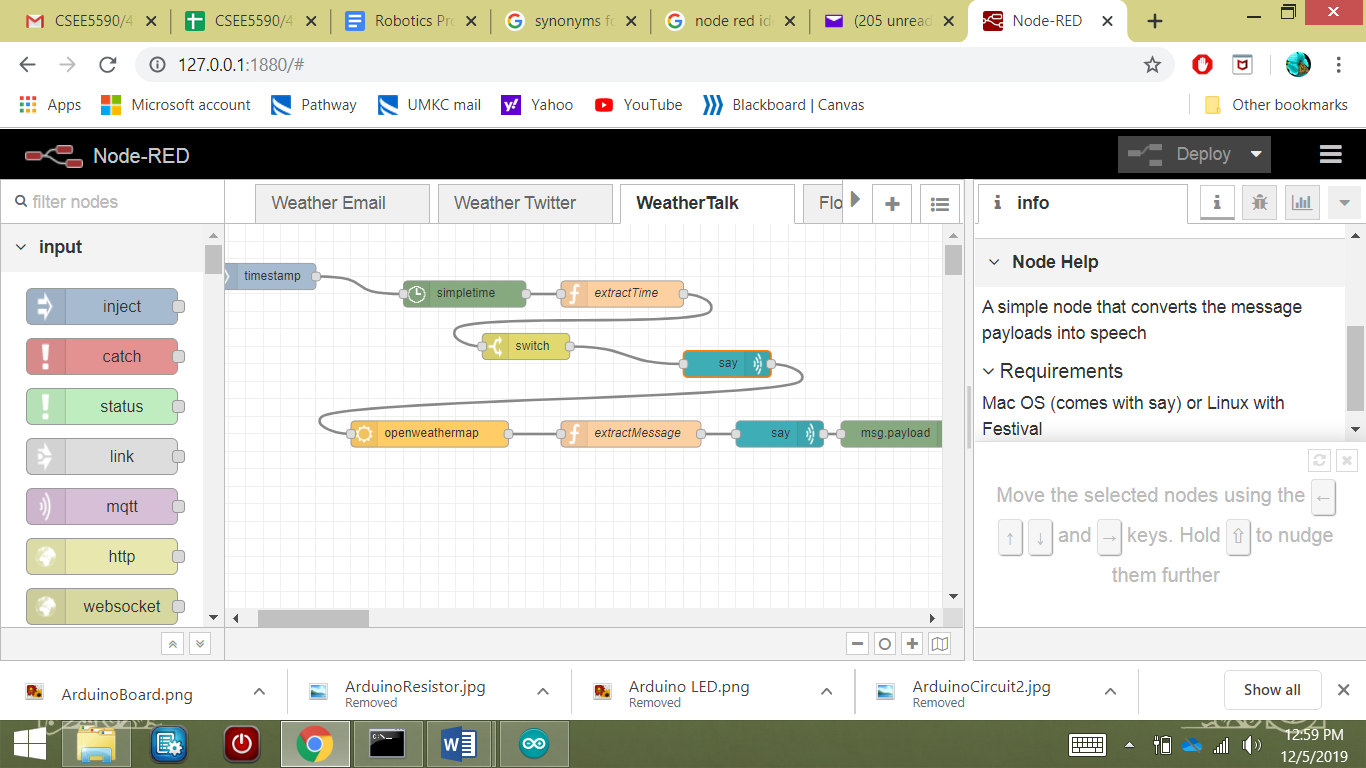
Once the code was completed, our alarm was done! At this point we did not have the Arduino speaker, so we couldn’t test it with sound playing, but the logic would have been simple enough. We just needed to play the sound when the light was on, and turn it off when the light was off. We now needed to figure out how to read the weather.

Moving over to Node-RED we recalled our ICP where we had to tweet the weather using Node-RED. Openweathermap.org was used, along with the openweathermap node.

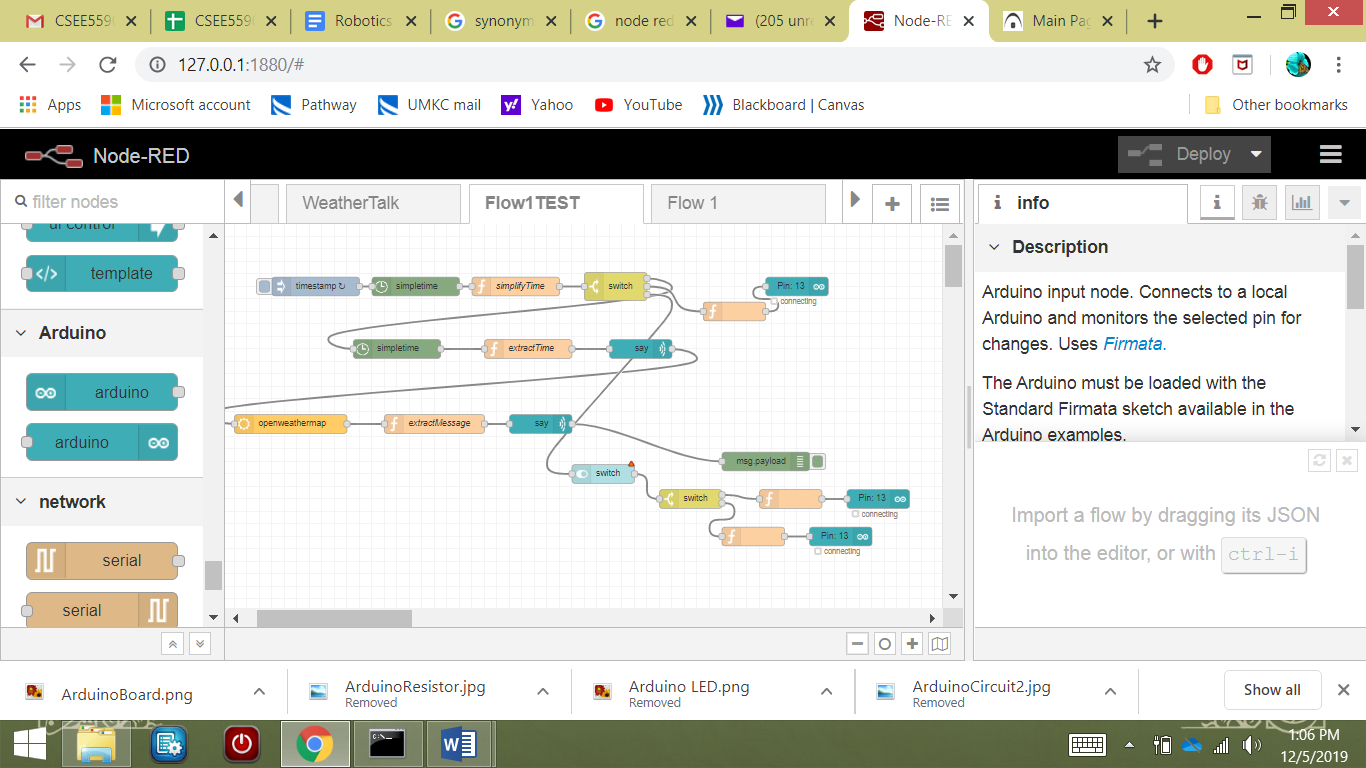


But we needed to get our Node-RED to speak out the information that was getting printed in our debug window. We also needed to find a way to grab the date and time and also have those spoken.

For the date/time extraction we used simpletime. This node gave us all the functionality we needed, and allowed us to test easily. We had a function node following it that grabbed the data fields and added them to our msg.payload. Now we needed a text-to-speech node. We used the say node for this. It would convert the msg.payload to speech on our computer speakers.

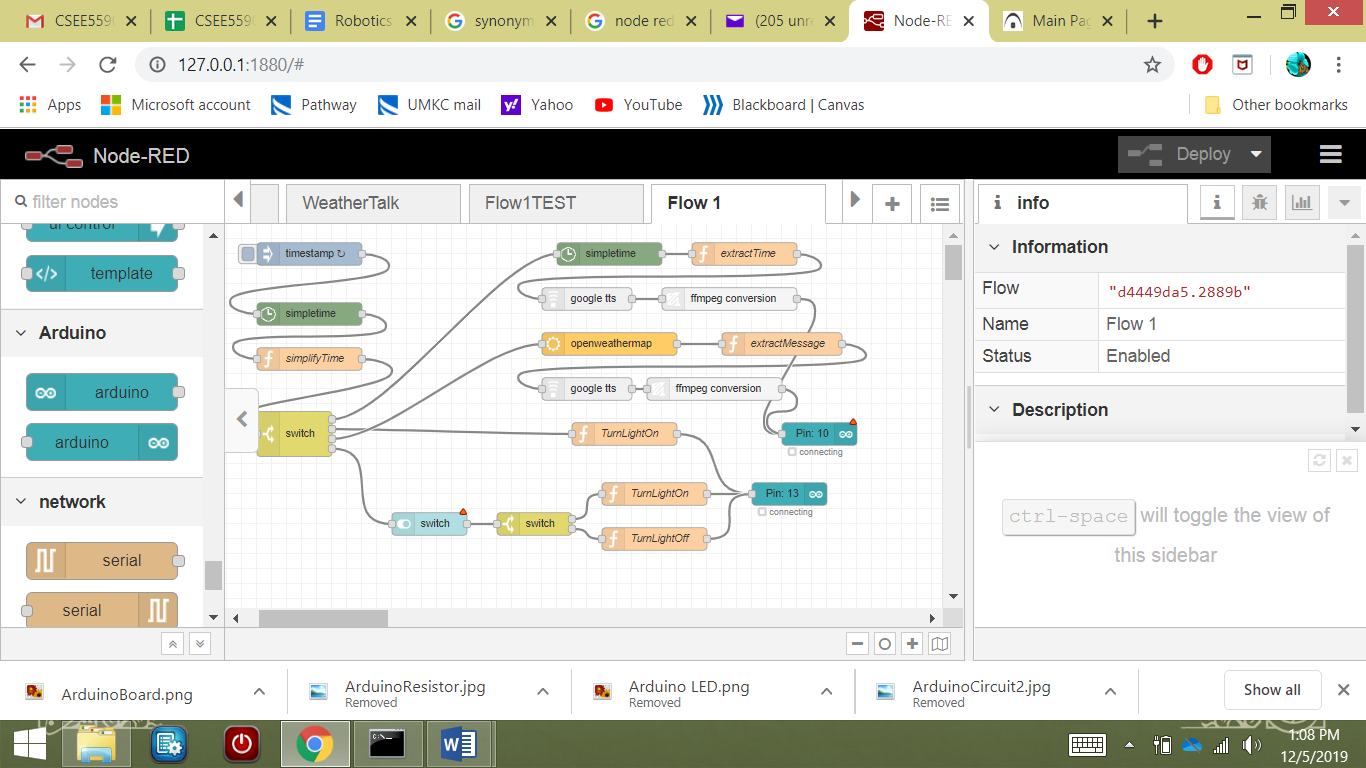


Now we had to figure out how to connect our Arduino to our Node-RED, which proved to be very simple. Arduino I/O nodes can be downloaded onto node red. They use a program called Firmata to run with Arduino. We simply had to upload the code onto Arduino, and then Node-RED would have access to our Arduino pins and circuit.



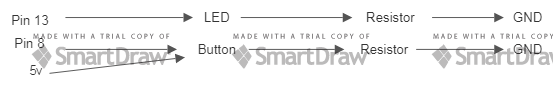
Our light would now turn on according to the node-red time! Our weather was also being reported aloud on our computer speakers.

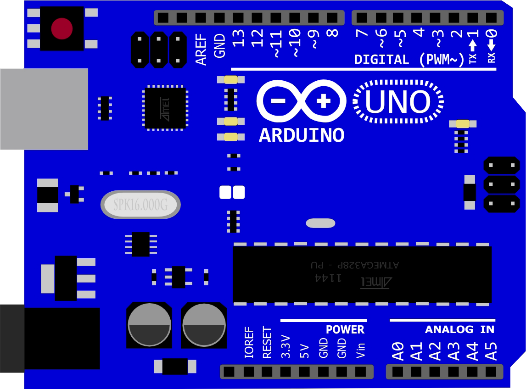
At this point we obtained our Arduino speaker. We were ideally going to have the speaker say the weather instead. We designed a flow for this, but ran out of time to integrate it on the Arduino side.



This was our stopping point in our work. We did not reach the app development part of our project.

Circuit Diagram:







C:\Users\Mikaela\Documents\Robotics\Project\LEDNOLEG.PNG

C:\Users\Mikaela\Documents\Robotics\Project\PICS\BUTTON.PNGC:\Users\Mikaela\Documents\Robotics\Project\RESISTORNOLEG.PNG

C:\Users\Mikaela\Documents\Robotics\Project\RESISTORNOLEG.PNG

Results and Evaluation:

We successfully simulated the following:

* Turning on lights automatically
* Setting an alarm
* Reading the weather

We left the following incomplete:

* Playing sound automatically with our Arduino Speaker
* Detecting smoke
* Controlling the simulation via an app

Conclusion and Future Work:

Using the RedMobile app was our assumed next step. We weren’t able to test it at all in an appropriate amount of time, so it’s possible there are better alternatives out there. We weren’t able to set up the Arduino Speaker in time, so our Final flow in node red would need to be looked at and edited. A different tts node would probably need to be used.

Working with a fuller team, we probably would have been able to accomplish more. We were ultimately satisfied with what we were able to do.

References and Resources;

**BUTTON:**

<https://www.arduino.cc/en/Tutorial/DigitalReadSerial>

<https://www.arduino.cc/en/tutorial/button>

**FIRMATA:**

\*\* Make sure node red is closed before uploading

<http://firmata.org/wiki/Main_Page>

File > examples > firmata > standard firmata

**Node Red:**

Arduino Nodes: <https://iotdesignpro.com/projects/interface-arduino-with-node-red-to-send-sensor-data-on-webpage>

Node-RED App: <https://www.youtube.com/watch?v=Xoe5Hf9Janw>

**Taking input from a user:**

Check if digit:

<https://www.arduino.cc/en/Tutorial/CharacterAnalysis>

Call serial once:

<https://forum.arduino.cc/index.php?topic=82603.0>

Serial inputs:

<https://www.norwegiancreations.com/2017/12/arduino-tutorial-serial-inputs/>

**Calling Functions:**

<https://www.arduino.cc/en/Reference/FunctionDeclaration>

**Current flow of a breadboard:**

<https://forum.arduino.cc/index.php?topic=305100.0>

<https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/all>

**Arduino Speaker:**

<https://www.programmingelectronics.com/an-easy-way-to-make-noise-with-arduino-using-tone/>

<https://maker.pro/arduino/projects/arduino-speaker>

**Weather:**

<https://home.openweathermap.org/>