

“Smart” (Parental Control) Outlet

The Problem:

These days many of us spend way more time watching tv, or playing video games than we should. However this is not something that needs to continue unchecked, with proper monitoring and allocated “screen time” this becomes less and less of a problem.

The Solution:

My project is the perfect solution to those looking to control and monitor what their kids, loved ones, or selves are doing with their free time. Using websockets, a raspberry pi, an arduino, and a number of sensors and components I've developed a comprehensive system to control and monitor the use of a smart outlet. The project meets these needs with the capability of setting, starting and checking the status of a timer both in person, and over the web.

Components:

- For this project the following hardware was used:
 - Raspberry Pi 4
 - Arduino Uno
 - RFID reader/writer Module
 - (3) Push Buttons
 - (3) Leds
 - (3) 220 Ohm Resistors
 - (1) 1000 Ohm Resistor
 - Active Buzzer
 - Photoresistor
 - 5v Relay Module

Diagram:

Figure 1.1 is the wiring and component diagram for the entire project (excluding serial connection to Raspberry Pi) This is the diagram of the project that was used for testing and submission, needless to say there are a number of changes that are needed before taking this project into its next (prototype) phase in development.

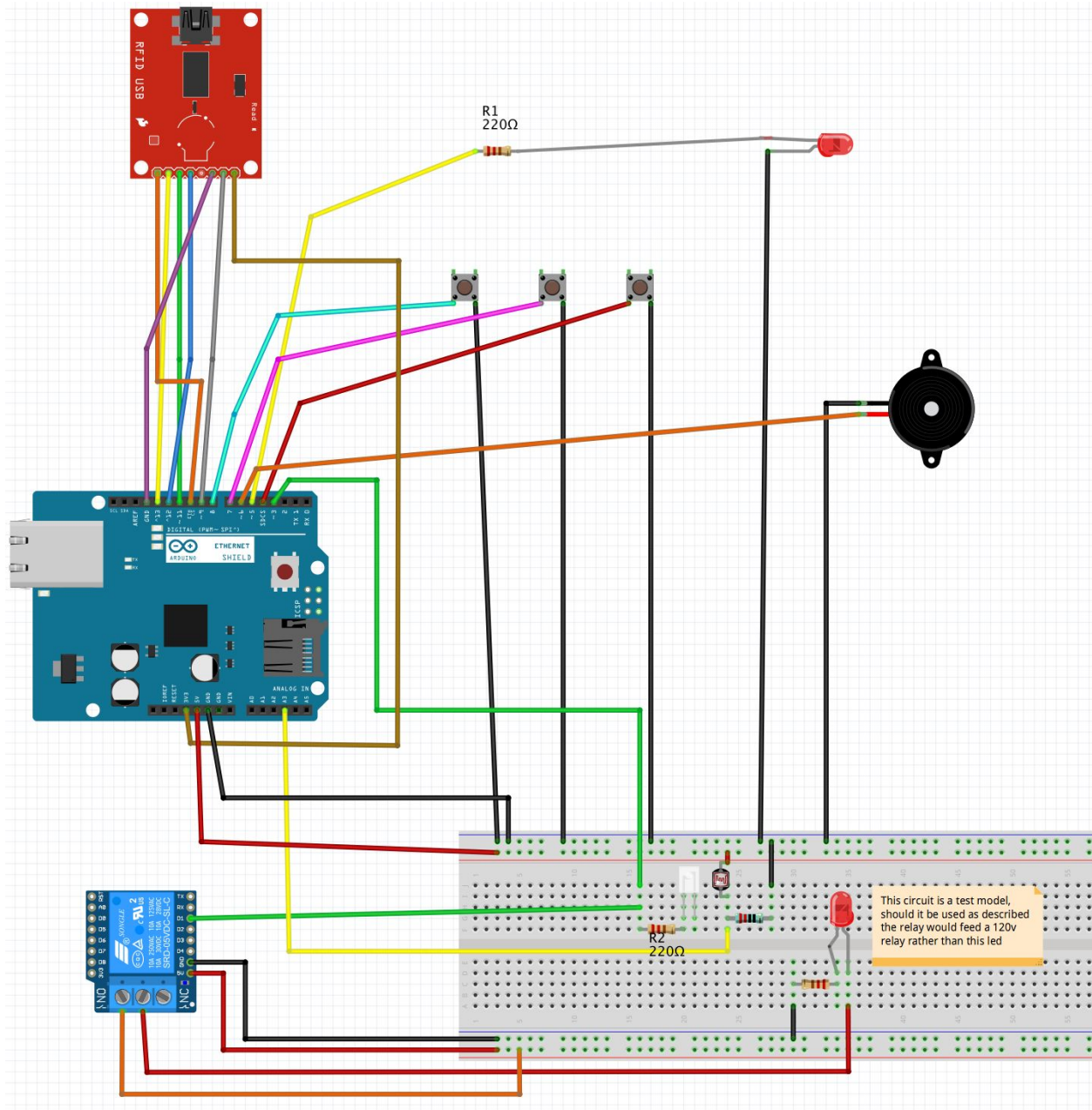


Figure 1.1

Next Steps:

As mentioned before there are a couple of changes that would need to be made on both the software and hardware, in order to continue this product through development.

1. Relay Connectivity - Rather than an led (strictly for test purposes) an outlet and plug should be wired to the relay as shown in figure 2.1

2. Time Spans - This software solution is also simply a matter of transitioning this project out of its testing stage. All that needs to be done is change the span of time that the system is incrementing by. This is done by changing the following line (fig 2.2) in the main python file from 10 to 60, thus changing 10 second increments into 1 min increments.
3. AWS - unfortunately to operate this project one needs to have a couple files already on their device in order to open, run and communicate through the web page. The next step here is to complete the websocket to operate through an AWS link so that the webpage can be accessed through any device (after the link is shared)
4. PCB - Because of the large number of components and connections another step that should be taken is to transfer as much of this layout as possible onto a pcb. This would allow for fewer errors, higher reliability and a more compact end device.

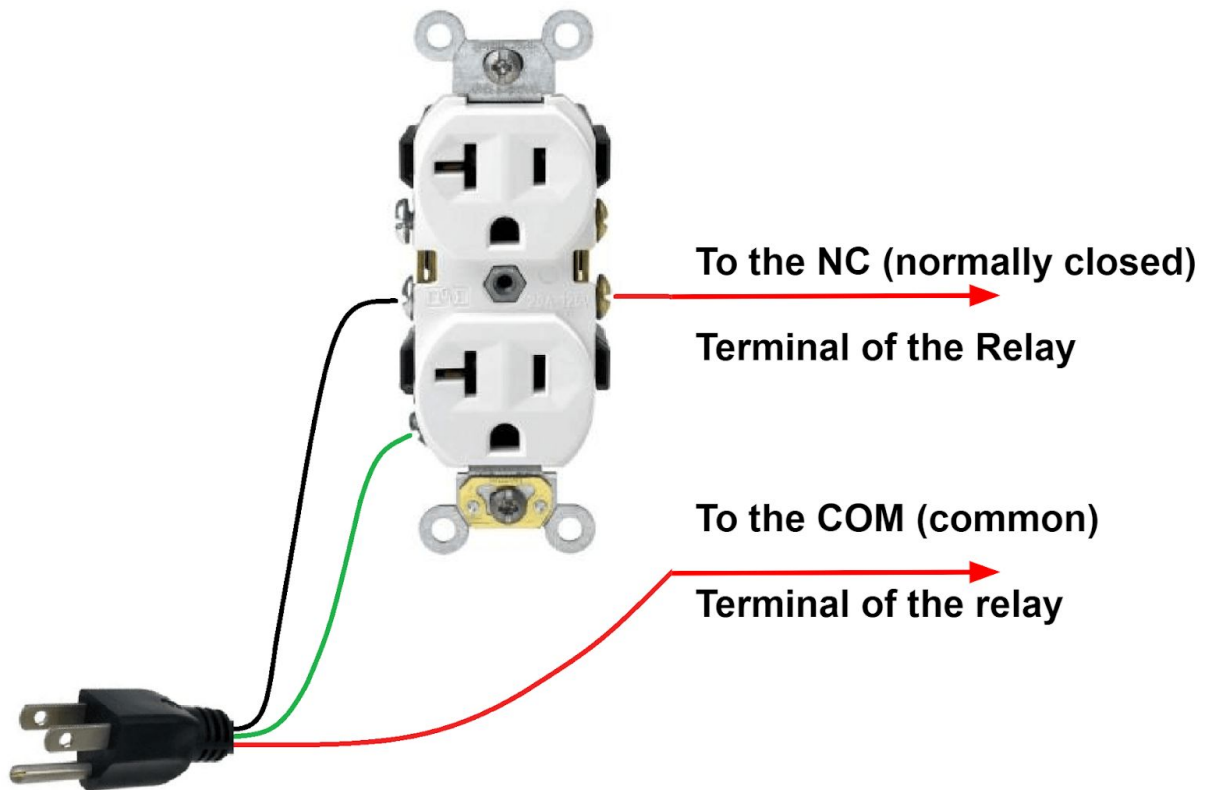
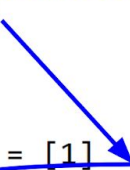


Figure 2.1

This "10" should be changed into "60" to represent 1 min increment rather than a 10 sec increment



```
—
..
pulse = [1]
full_time = 10*check_time()
while (time.time() - now_time) < (full_time):
    if int(full_time-(time.time() - now_time))%2 == 1:
        pulse = [1]
        self.locoIoT.setData(self.msg.SUBTYPE_DO_2, pulse)
    elif int(full_time-(time.time() - now_time))%2 == 0:
        pulse = [0]
        self.locoIoT.setData(self.msg.SUBTYPE_DO_2, pulse)
```

Figure 2.2

Conclusion:

Overall this project was a success, this isn't to say that there weren't a number of issues that needed to be solved, changing the webpage and its functionality using Javascript and HTML are just one example of the many challenges that had to be overcome. Moving forward I would definitely implement the four changes previously described in order to have a smaller, more functional, and cleaner prototype. None of these changes are inconceivable, or outside the scope of the project, however completing them within the original timeline of the project would not be possible. With another 5 - 10 days it is reasonable to assume that all of these changes could be made, with exception of obtaining the PCBs which could take additional time to obtain given shipping and manufacturing lead time.