

Yushen Li

December 4th, 2019

Art 150

Final Project Documentation

Purpose of this Project:

To create an IoT (Internet of Thing) that is low cost and must be able to interact with the users. Utilizing cardboard / recycle material for enclosure to communicate the ideology of environmental protection and recycle materials can be used for art creation.

Material List:

- Raspberry Pi 3
- Arduino Mega
- Temperature & Humidity Sensor
- Kumar 3.5 TFT Touch Display
- Infrared Sensor
- IR Remote
- ¼ Size Breadboard
- Micro USB and 5V-2A Power Source
- Micro SD Card
- Wires
- Magnet
- Cardboard
- Hot Glue
- Soldering iron

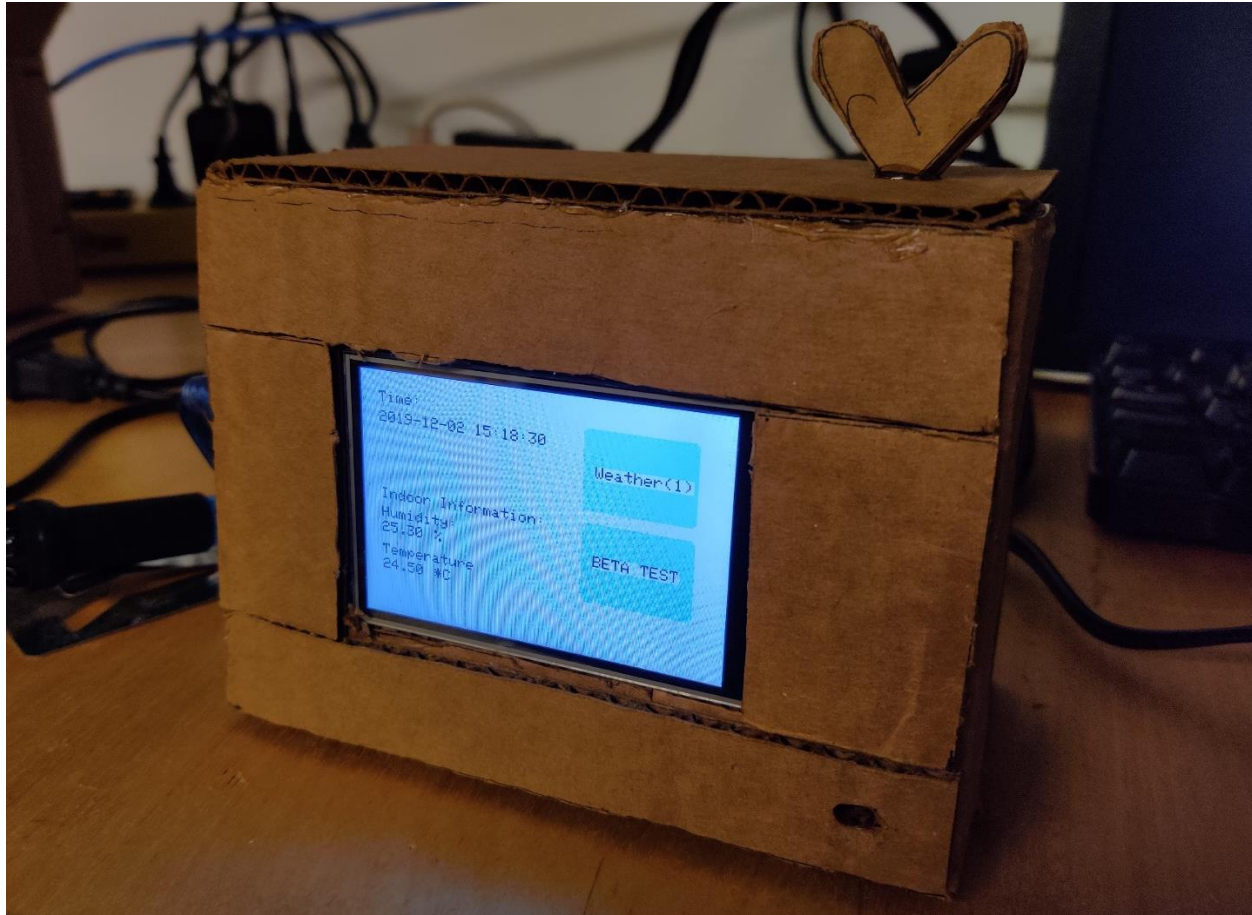
Video Explanation:

- <https://youtu.be/DRHEPip43MM>

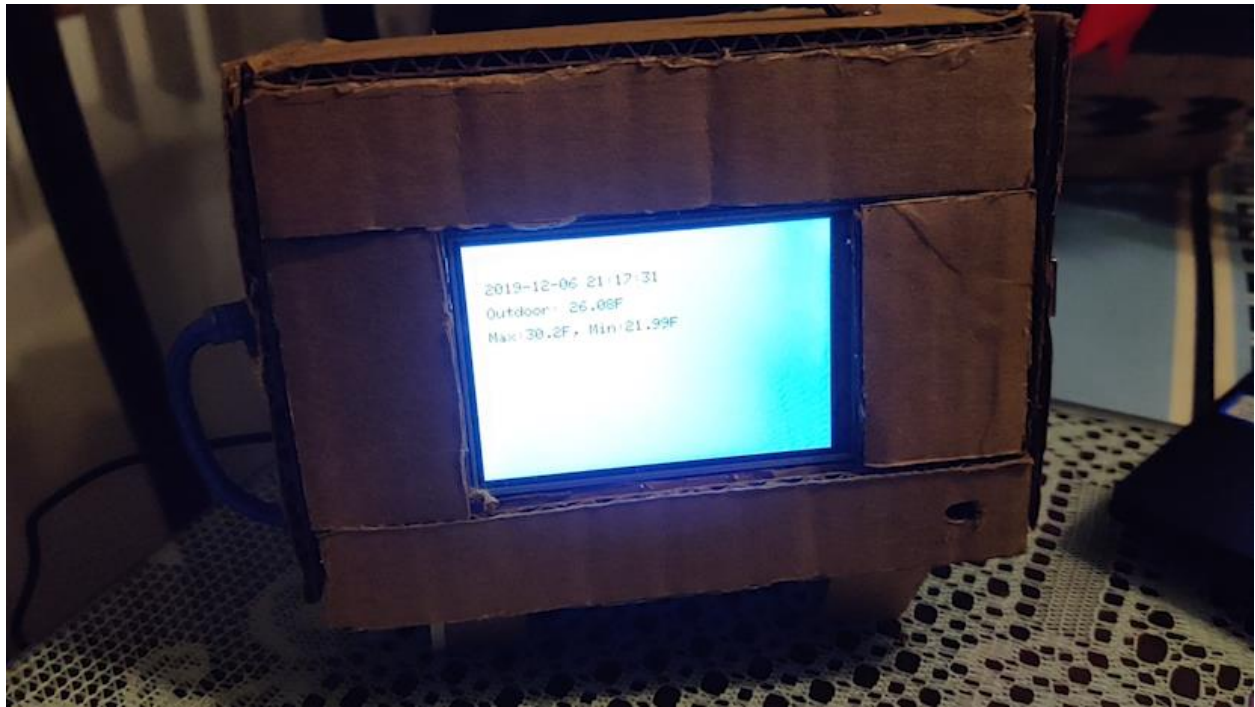
Source Code:

- <https://github.com/Yushenli1996/Retro-IoT-TV>

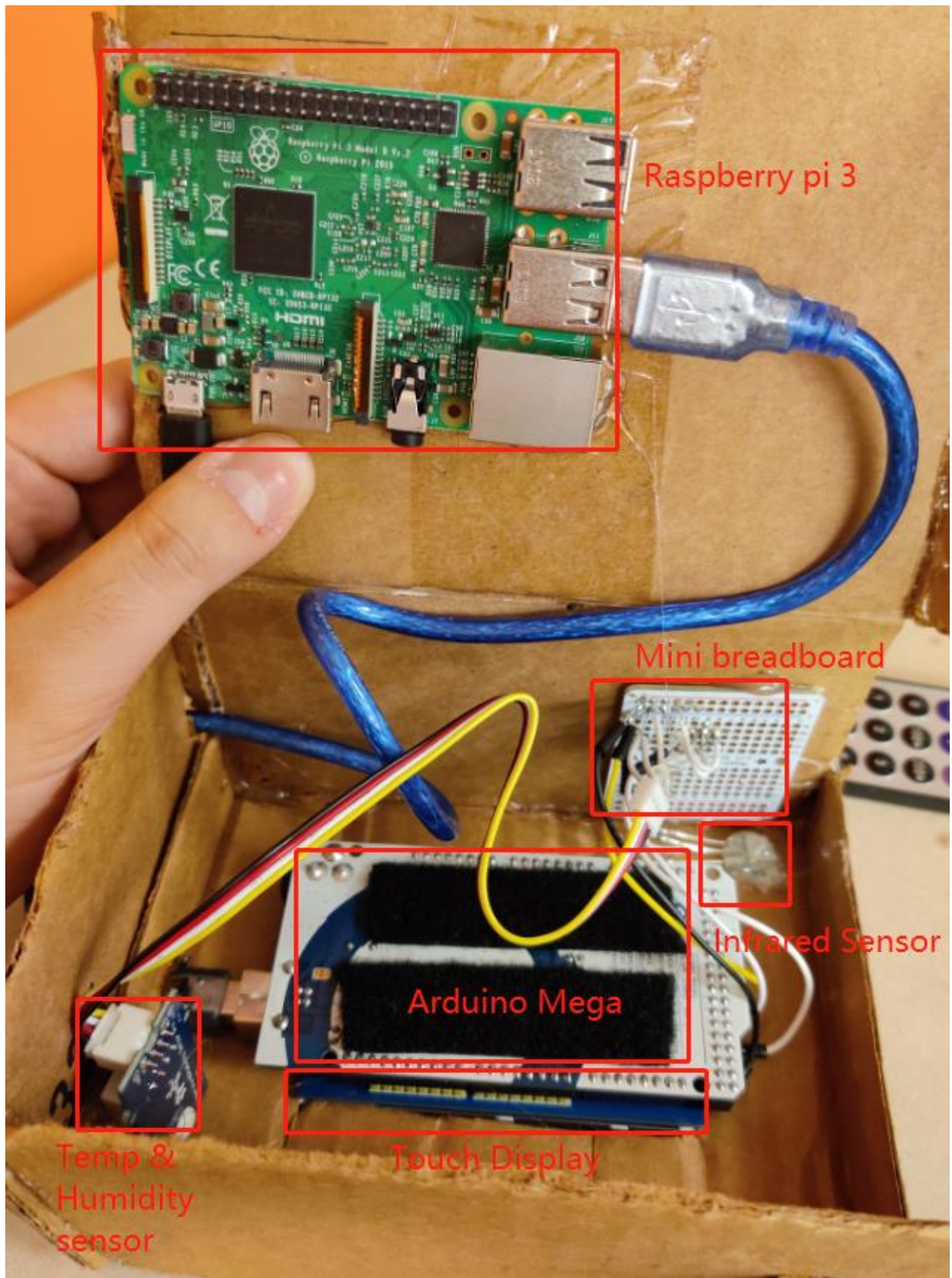
How does it work:



This is a general video of the completed product. The screen at the front shows the current time and indoor information. The indoor information contains information of current indoor humidity and temperature and updates real time. On the right side of the screen, there are two rectangles, one shows Weather (1) and one shows BETA TEST. Originally, I wanted to use the touch feature as a way for users to interact with the device; however, due to the compatible issue with the display driver, I dropped this feature at the end and change to remote interaction instead. (1) in this case representing the keypad number 1. BETA TEST is just a string I want to show user the current source code version, but users are free to reconfigure it in the Arduino code. The bottom right has a tiny hole cutout. The hole is for the infrared sensor to receive infrared information from the remote. Of course, how would a retro TV come without a pair of antennas.



This picture shows the weather information of my current location. I hard coded the location in the connection.py file. If users want to change the location, please check down below in the **How to run** section.



Raspberry pi 3



Mini breadboard



Infrared Sensor



Arduino Mega

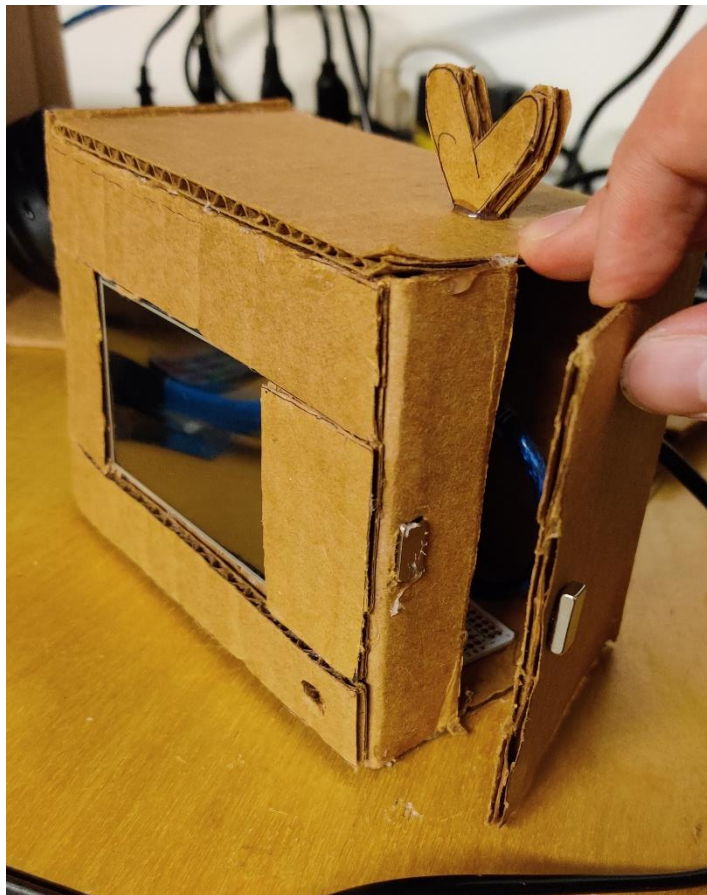


Touch Display

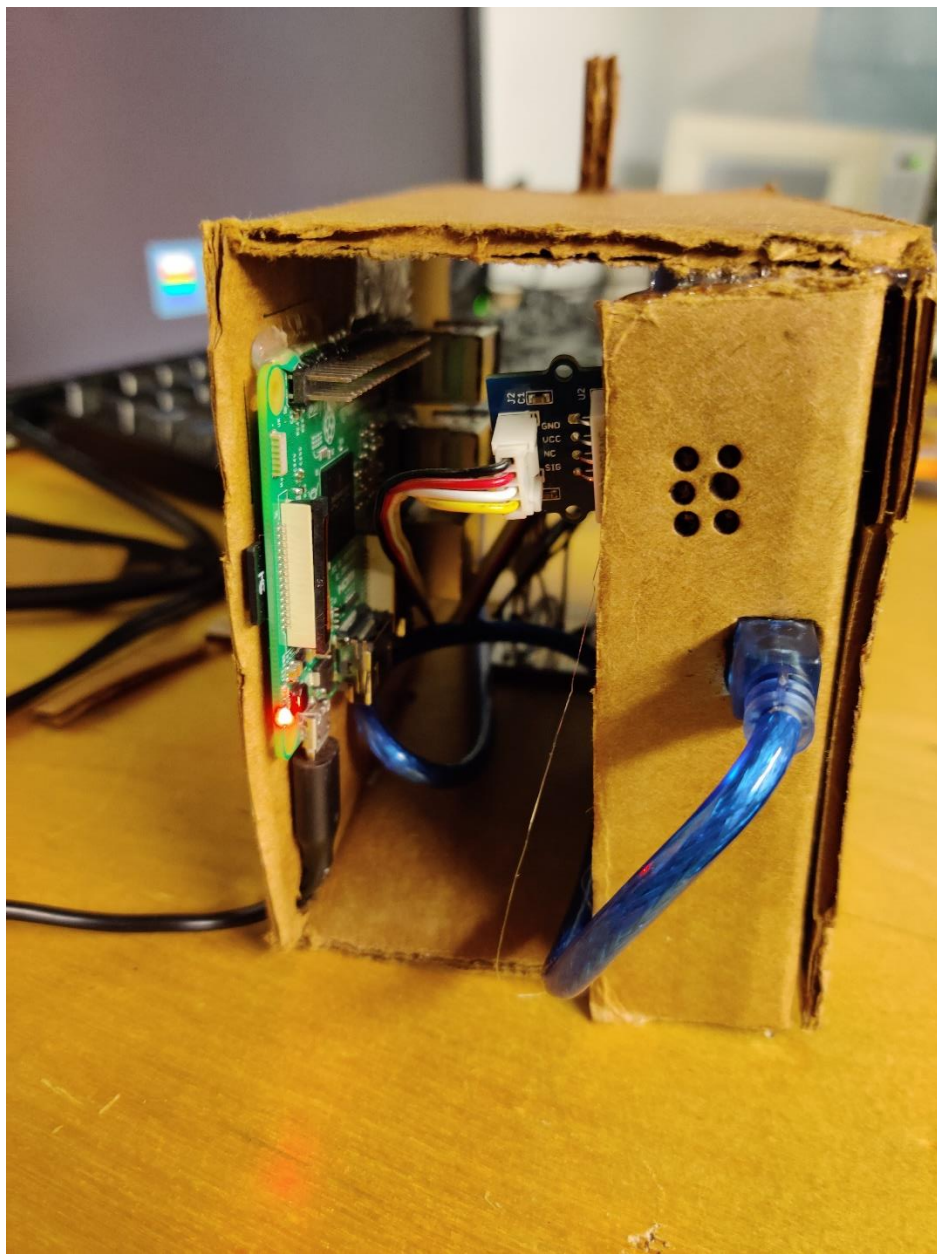


Temp &
Humidity
sensor

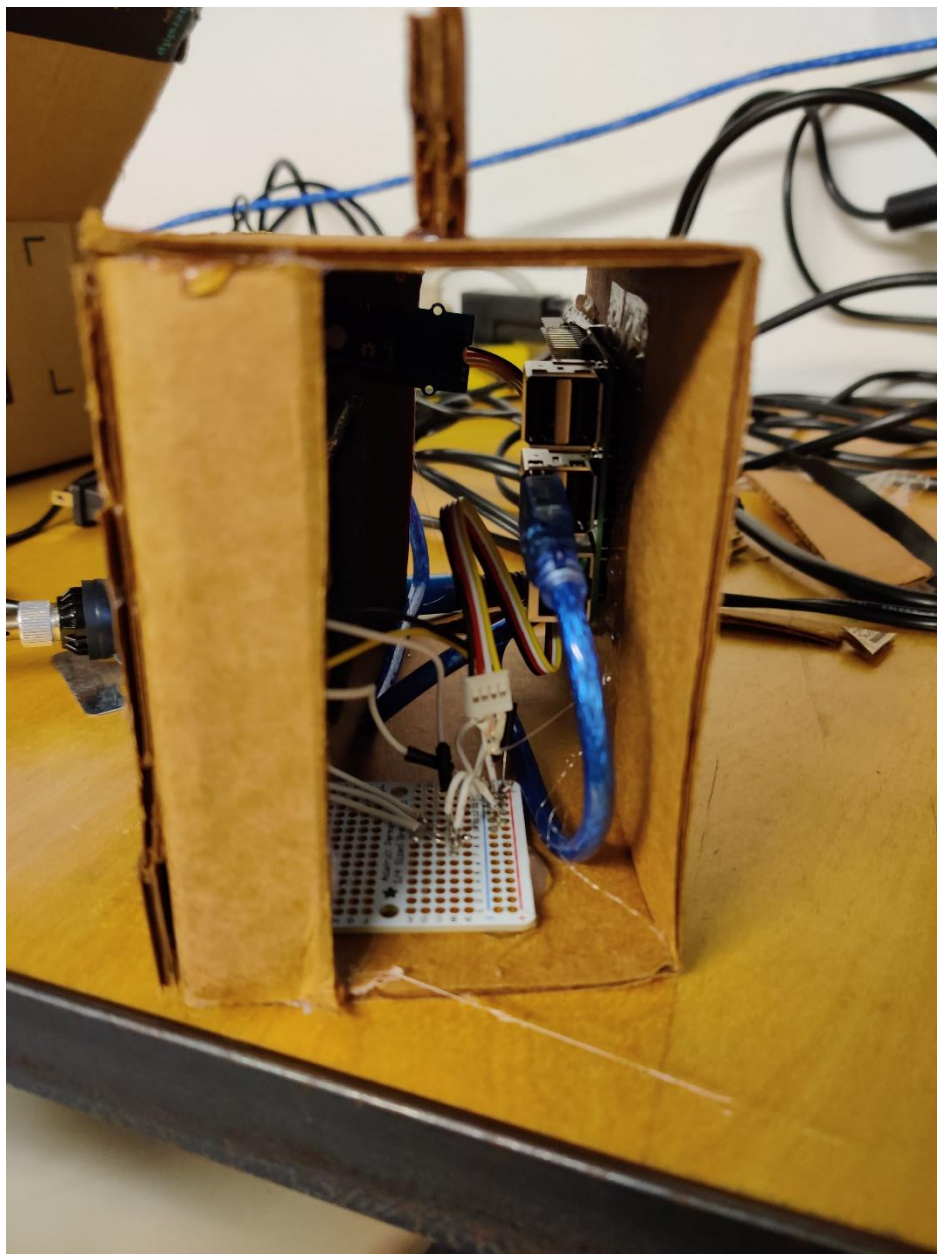
And this is the internal view of the TV box. The reason why I chose Arduino Mega is that the touch display uses all the pin connection on the Uno edition; as a result, in order for me to continue to add more sensor or other functionalities, I need more pin holes. And Mega does exactly what I wanted. The temperature sensor is connected to the Arduino directly, while the Arduino is powered and communicate directly to the Raspberry Pi 3.



On the left side there is a small door I made for the internal structure and cable checking. And the small door is able to close by using magnet.



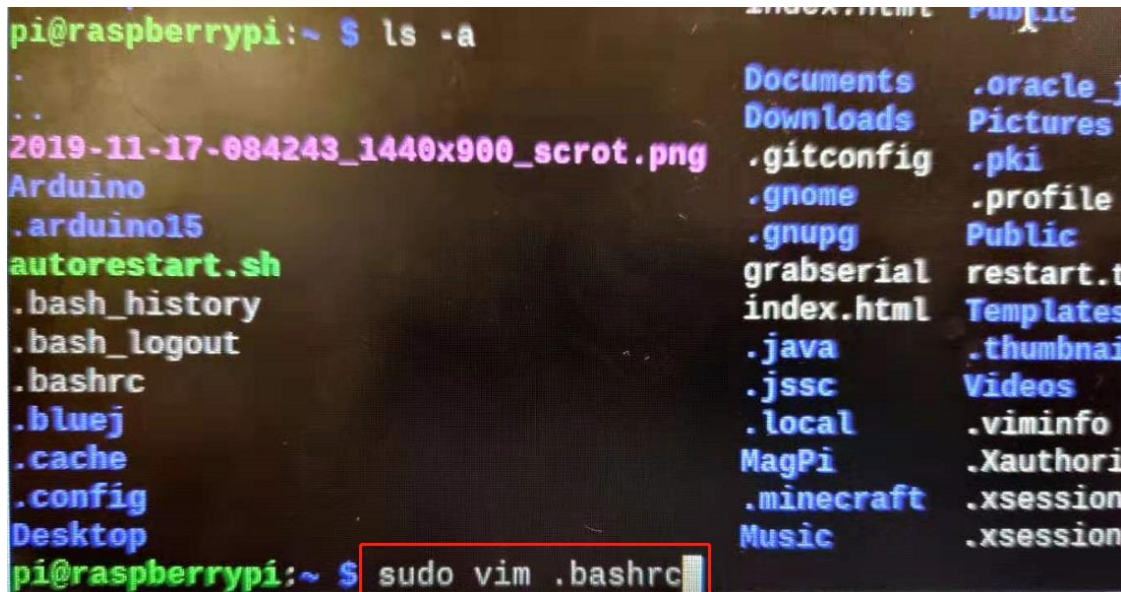
The left side has cut out for cables and breathing holes for the temperature sensors.



The right-side view without the small door.

How to setup and run the code:

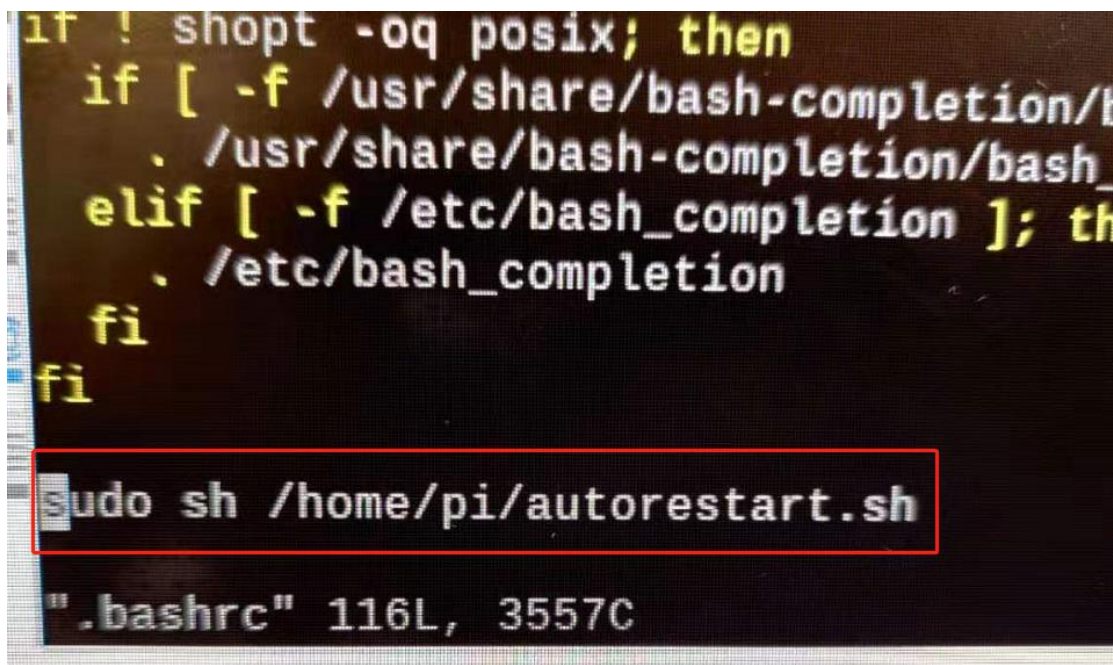
1. <https://www.raspberrypi.org/forums/viewtopic.php?t=66206>, follow this forum to setup the terminal on boot, you can choose anything. I used the built-in lxtermial.
2. Setup the self-execute script on boot. Use text editor to edit the hidden file named .bashrc in the /home/pi directory and add the command in the screenshot down below.

A terminal window on a Raspberry Pi. The prompt is 'pi@raspberrypi:~ \$'. The user has run 'ls -a', which lists files in two columns. The first column contains: '..', '2019-11-17-084243_1440x900_screenshot.png', 'Arduino', '.arduino15', 'autorestart.sh', '.bash_history', '.bash_logout', '.bashrc', '.bluej', '.cache', '.config', 'Desktop'. The second column contains: 'Documents', 'Downloads', '.gitconfig', '.gnome', '.gnupg', 'grabserial', 'index.html', '.java', '.jssc', '.local', 'MagPi', '.minecraft', 'Music', '.oracle_j', 'Pictures', '.pki', '.profile', 'Public', 'restart.t', 'Templates', '.thumbnail', 'Videos', '.viminfo', '.Xauthori', '.xsession', '.xsession'. At the bottom, the user has entered 'sudo vim .bashrc' and the cursor is at the end of the command.

```
pi@raspberrypi:~ $ ls -a
.
..
2019-11-17-084243_1440x900_screenshot.png
Arduino
.arduino15
autorestart.sh
.bash_history
.bash_logout
.bashrc
.bluej
.cache
.config
Desktop
Documents
Downloads
.gitconfig
.gnome
.gnupg
grabserial
index.html
.java
.jssc
.local
MagPi
.minecraft
Music
.oracle_j
Pictures
.pki
.profile
Public
restart.t
Templates
.thumbnail
Videos
.viminfo
.Xauthori
.xsession
.xsession

pi@raspberrypi:~ $ sudo vim .bashrc
```

It's important to pre-configure the raspberry pi if you want to do custom script on boot.

A terminal window showing the contents of the .bashrc file. The text is: 'if ! shopt -oq posix; then', 'if [-f /usr/share/bash-completion/bash_completion]; then', ' . /usr/share/bash-completion/bash_completion', 'elif [-f /etc/bash_completion]; then', ' . /etc/bash_completion', 'fi', 'fi'. At the bottom, the user has entered 'sudo sh /home/pi/autorestart.sh' and the prompt has changed to 'pi@raspberrypi:~ \$'. Below the command, it says '" .bashrc" 116L, 3557C'.

```
if ! shopt -oq posix; then
if [ -f /usr/share/bash-completion/bash_completion ]; then
  . /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
  . /etc/bash_completion
fi
fi

pi@raspberrypi:~ $ sudo sh /home/pi/autorestart.sh
pi@raspberrypi:~ $
".bashrc" 116L, 3557C
```

This command is required in order to make the script to automatically execute the command. The autorestart.sh script is added to the github repository (Link is above).


```
Executable File | 18 lines (15 sloc) | 249 Bytes

1  #!/bin/sh
2
3  COMMAND='python ./Desktop/connection.py'
4  LOGFILE=restart.txt
5
6  writelog()
7  {
8      now=`date`
9      echo "$now $" >> $LOGFILE
10 }
11
12 writelog "Starting"
13 while true ; do
14     sleep 2s
15     $COMMAND
16     #writelog "Exited with status $?"
17     #writelog "Restarting"
18 done
```

Make sure put the connection.py on desktop. After doing all the things above, the code should be ready to execute on boot. **If users want to modify the arduino code, make sure comment out the [#sudo sh /home/pi/autorestart.sh] command and reboot the system. Otherwise, there will be a serial connection time out issue in the Arduino IDE. And uncomment it when finish updating the Arduino code.**

```
25     url = 'https://api.openweathermap.org/data/2.5/weather?q=Chicago,usa&appid=1c865cbb1dd3b0a02e6764a9a1ce0355&units=imperial'
26     res = requests.get(url)
```

The first red box is the location I picked for this project. The second red box is the openweathermap API key, please signup for one if you need to use it in other ways. The third one is the unit I choose for result returning.

What I learned and what can be improved

I learned that it is important to search parts that is easy to work with because the display I used this time gave me a lot of troubles and forced me to give up on a lot of functionalities. For example, I want to use the touch feature, and the display I got is not very well supported. Secondly, the display uses a lot of power; thus, it's very hard to add additional sensors onto the board because it doesn't have enough power to run other sensors. I also learned how to make custom boot script for raspberry on the go, which this is very interesting in my opinion.