EE 467 Senior Design
Final Report
Spring 2018

Smart Mirror

Submitted by:
Lucas Speer
Isaac Matzke

Table of Contents

1.0 Introduction

This Senior project’s goal is to create a “Smart Mirror”, A mirror that is also used as the screen for a computer which can display useful daily information to the user while also serving as a regular mirror. This will create a hub for all of the user’s daily information at a single glance. The intention for this design is to make it highly modular and upgradable. A locally hosted web page is used as the background for the mirror gave an easy to manage canvas to display different modules including: time, weather, news, emails and potentially more. This project takes advantage of the Raspberry Pi Zero W’s built in Bluetooth function to connect to an app for configuration, allowing the mirror to be setup without a keyboard or a mouse, but the application itself can run on any Raspberry Pi that has internet and Bluetooth connectivity.

2.0 Background

The internet of things is penetrating every aspect of the lives of the consumer and has lead to a demand for people to be more connected and to have things done automatically. This project keeps this in mind and was designed to act as a home base that can display useful information at a glance, while still being minimalist in design and space-consumption.

The modularity of this project is what makes it stand out. With a serial RFCOMM communications protocol designed from the ground up a developer can have direct control over every operation performed and can continuously add new functions to accompany any configuration or event triggering they could want in the future. They utilization of GitHub for version control also ensures that new additions are able to be rolled back in the event of a serious bug.

Using a web browser to display all the information on the mirror gave a perfect platform to have dynamic and well designed modules. HTML and CSS along with Javascript and JQuery provide nearly everything a developer needs or could want in design of this project. The jQuery library and the massive support network within the web design online community, along with a modular approach, allow even a novice web developer to more efficiently and effectively debug and improve the program as a whole.

3.0 Requirements

The two basic hardware requirements to produce a smart mirror using these programs are some type of backlit display (e.g. PC monitor, television) and an internet-connected device/computer that can run a web browser on the display device. The implementation for this project was done using a raspberry pi zero W and a Sceptre LCD 19” monitor.
Additional hardware components are optional that will increase the functionality of the mirror, as well as the price to produce (e.g. bluetooth dongle and smartphone that allow easier customization). This project was designed around using a photoresistor (light-level sensor) and a push on/off button.

3.1 Constraints

Constraints	Hardware Components	Hardware System	Software
Economic	Affordable, widely available	Affordable, widely available	Free
Environmental	Low power	Recycled monitor and Handmade frame	Automatically dim or shut off monitor with a light sensor
Social	No hazardous components	No hazardous components	Open-Source and User friendly
Political	Parts primarily sourced from America	Parts primarily sourced from America	Open-Source and spyware free
Health	No Health impact	No Health Impact	N/A
Safety	Stable and Low power	No high currents	Secure, no high-level system permissions in case of corruption.

[Fig 3.1: Real World Constraints Table]

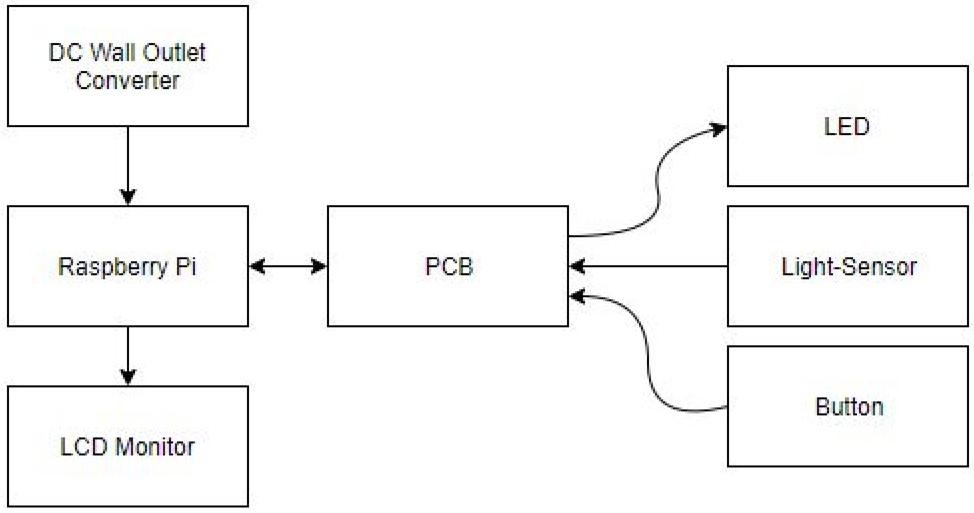
3.2 Functions

The purpose of this mirror is to be highly upgradable and customizable. This means that the functionality of the mirror will grow as more peripherals and widgets are developed. The goal was to produce a base that functioned without a keyboard and mouse and had an intuitive and useful design.

Currently several simple modules are completed that perform various functions. The first is a greeting title that changes dynamically according to the time of day. The mirror has a module to display the weather in the configured location (can be set using the app via zip code). This utilizes an open-source plugin called Simpleweather.js. In addition, the mirror uses the built-in functionality of the time/date in the Raspbian operating system to display the time and date.

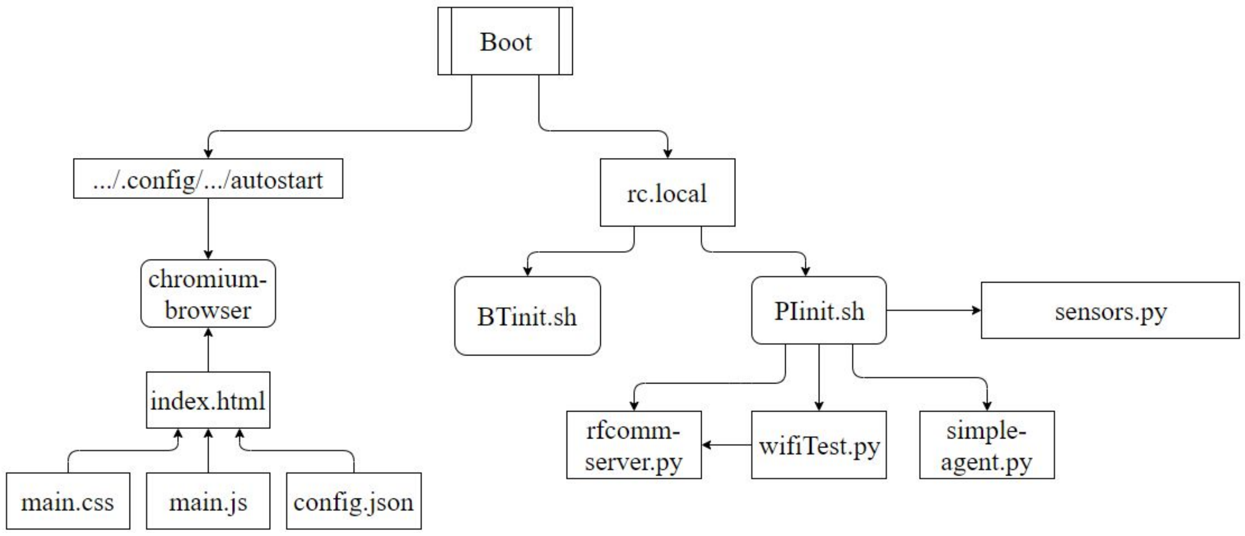
4.0 System

4.1.1 System Block Diagram



[Fig 4.1: Basic Block Diagram]

4.1.2 Software Block Diagram



[Fig 4.2: Raspberry Pi Software Block Diagram]

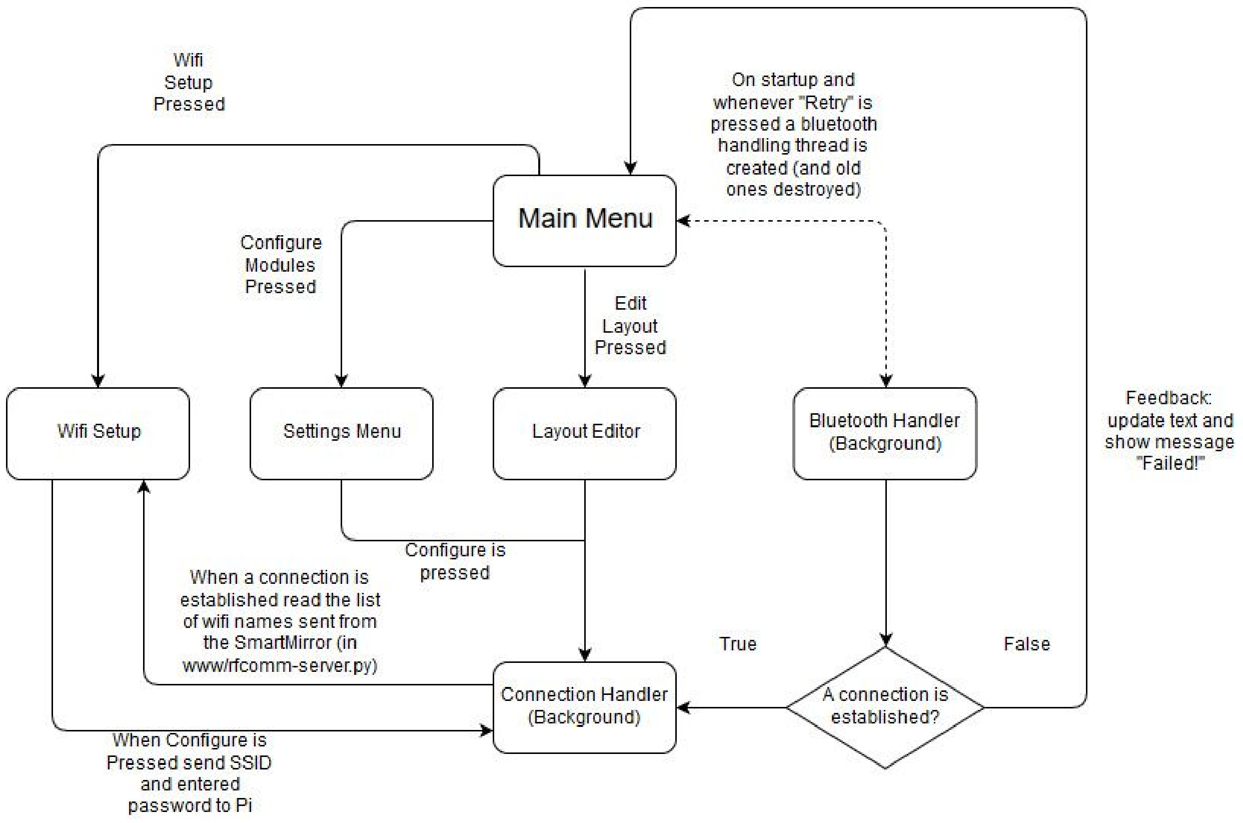

Wifi Setup Pressed
On startup and whenever "Retry" is pressed a bluetooth handling thread is created (and old ones destroyed)
Main Menu
Configure Modules Pressed
Edit Layout Pressed
Wifi Setup
Settings Menu
Layout Editor
Bluetooth Handler (Background)
Feedback: update text and show message "Failed!"
Configure is pressed
When a connection is established read the list of wifi names sent from the SmartMirror (in www/rfcomm-server.py)
True
False
Connection Handler (Background)
A connection is established?
When Configure is Pressed send SSID and entered password to Pi

Fig 4.3: Android App Block Diagram

4.2 Block Description

The left side of the block diagram in Figure 4.2 are the software pieces that run the html page which will be used to display the mirror information. The autostart program will automatically open index.html from a chromium browser, making it full screen. Index.html is run by 3 different files. Main.css handles the styling of the page, making sure that data text is white on a black background. The on/off status of a widget is also controlled here. Test.js holds all of the javascripts; these scripts are used to handle information displayed to the html page. Test.js is also used to read in the data from the config.json and use the information to adjust what is displayed to the page. Config.json is written to by rfcomm-server.py based on the data it receives from the app.

Figure 4.3 shows the block diagram for the Android app. On the main menu there are 4 buttons, one for the layout, one for general settings, one to attempt to connect and one to setup the wifi. The app creates a background task called bluetooth handler whenever the “Retry” button is pressed which attempts to connect to the SmartMirror. Once a connection is established the settings can be sent to the Pi which receives and interprets them

5.0 Hardware and Software Elements
5.1 Code

```
Raspberry Pi
        SmartMirror/
                setup.sh
                Update.sh
                README.md
                www/
                        config.json
                        defaultconfig.json
                        index.html
                        main.css
                        rfcomm-server.py
                        saveToSever.sh
                        sensors.py
                        test.js
                        time.js
                        Wifitest.py
                lib/
                        autostart
                        dbus-org.bluez.service
                        machine-info
                        main.conf
                        panel
                        rc.local
                        scr/
                                BTinit.sh
                                PIinit.sh
                                bluezutils.py
                                rfcommset.py
                                Simple-agent
```

Fig 5.1: Raspberry Pi Software List

```
Android App
        main/java/
                AndroidManifest.xml
                BluetoothHandler.java
                ConnectedThread.java
                LayoutConfig.java
                MainActivity.java
                MyAdapter.java
                Settings.java
                WifiSetup.java
        Layout
                activity_layout_config.xml
                activity_main.xml
                default_settings.xml
                item_simple_itemview.xml
```
[Fig 5.2: Android App Software list]

*full code implementation in Appendix C

5.2 Code Description

The files shown in Figure 5.1 are all located on the Raspberry Pi and handle everything from listening for a bluetooth connection to configuring certain system settings necessary for operation. Index.html, main.css, and test.js define the web page and take care of the scripting behind the scenes of the modules. Test.js is where the code takes advantage of JQuery to update the html elements from inside the program while referencing the dynamic config files. Rfcomm-server.py does most of the heavy lifting on the Pi side. It is triggered from PIinit.sh which runs every boot. Rfcomm-server.py starts listening for incoming connections and handles any data as it arrives. It is also responsible for sending the list of seen wifi networks to the app. The list of wifi networks is generated every boot, first through a shell command in PIinit.sh, which saves a rough list, and then through wifitest.py which formats that list into a clean list of only wifi network SSIDs.
All the files in SmartMirror/lib/ are system files that are placed in their necessary locations via update.sh. Autostart starts chrome and a program called unclutter which hides the mouse cursor. The other files are necessary for bluetooth to function properly.
In Figure 5.2 the Android app files are broken into two main categories, the java files and the layout files. This list does not contain all the generated files which can be found on the GitHub page. There is one java file for each ‘Activity’ which is just a page within the app, these files are MainActivity.java, Settings.java, LayoutConfig.java, and WifiSetup.java. The BluetoothHandler.java class has a background function that attempts to connect to the Raspberry Pi and on creates an instance of the ConnectThread.java class on a successful connection. ConnectedThread.java handles all the incoming and outgoing data. MyAdapter.java is a custom ListAdapter for displaying the wifi networks in WifiSetup.java.

6.0 Build

The physical build of this project is hardware-minimal. The main components are the Raspberry Pi and the LCD monitor acting as the mirror. The circuitry design was for the peripheral sensor, button, and LED connections to the raspberry Pi.

6.1 Bill of Materials

Item No.	Item Description	Manufacturer	Retailer	Quantity	Price
0	Raspberry Pi Zero W	Raspberry Pi	Adafruit	1	10.00
1	19" LCD Monitor	Sceptre	PC Liquidations	1	49.99
2	8 Gb Micro SD	SanDisk	Amazon	1	9.95
3	2.4A Wall converter and cable	Kanex	Amazon	1	9.99
4	One-way Reflective Film	Smart Chabon	Amazon	2'/5' roll	10.00
5	HDMI to DVI M/M cable	Amazon	Amazon	1	5.99
6	Wall Outlet Power cable AWG standard	CableCreation	Amazon	1	7.99
7	PCB	Seed	Seed	1	0.99
8	Assorted MtF Jumper Wires	Kuman	Amazon	1	7.49
9	TACT Button/Switch	CO-RODE	Amazon	1	0.68
10	LED	Adafruit	Amazon	1	0.23
11	PhotoResistor	Sunfounder	Amazon	1	6.99
12	MCP3008 Analog-Digital Converter	Adafruit	Adafruit	1	3.75
				Total	125.04

[Fig 6.1]

6.2 Essential Parts Description

SunFounder Photoresistor light sensor module:

This is used by the Raspberry Pi to detect a rise or fall in light levels in a room to trigger an auto-sleep function. This is connected to the analog to digital converter.

Adafruit MCP3008 Analog to Digital Converter:

Because the Raspberry Pi used in this build does not come with a built-in ADC, one was needed. This particular ADC uses a 10-bit carriage. The power and clock are supplied by the Raspberry PI.

TACT Button:

This is used as an alternative power switch to the light sensor. The user may desire the system to not be running at all times while the room is lit. This button acts as a “master” switch, with higher priority than the light sensor.

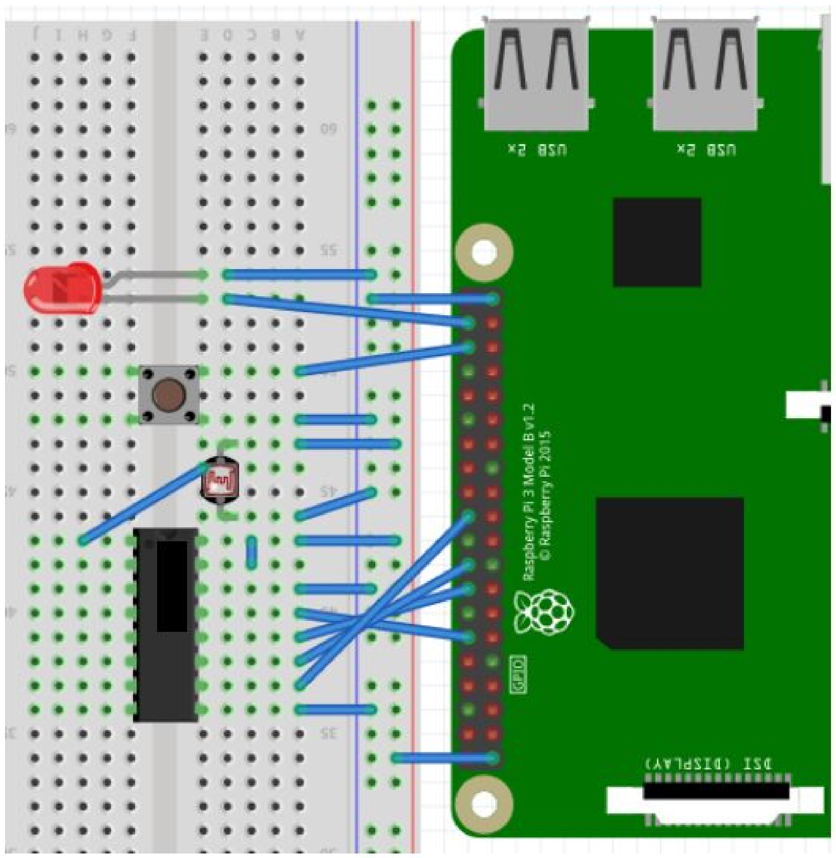
Raspberry Pi 3 Zero W:

This microcontroller was chosen because of the massive open-source operating system Raspbian. Raspbian is widely supported across Raspberry Pi devices and contains many of the functionalities of a full-sized OS. Mainly, Raspbian already has a graphic interface with web browser support. For this project specifically, the Pi 3 Zero W was chosen because of its low power consumption and its pre-installed wifi and bluetooth modules.

PCB:

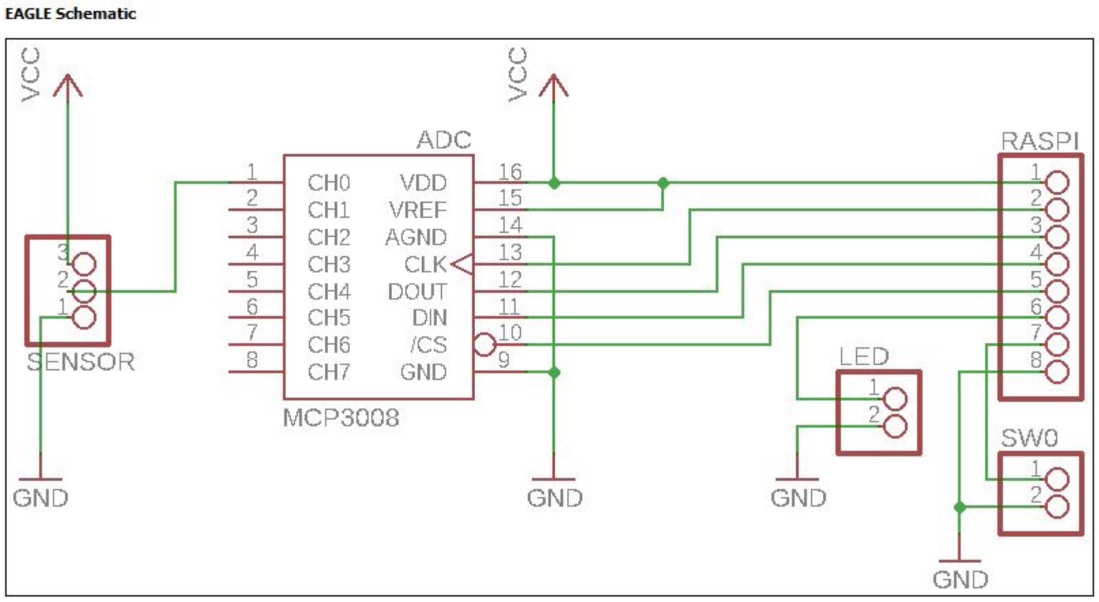
The PCB is primarily used as an interface between the ADC and the Raspberry Pi. It cleans up the wire management between the Raspberry Pi and its peripherals as well as limiting the number of connections needed to the Pi’s pins.

6.3 Wiring Diagram

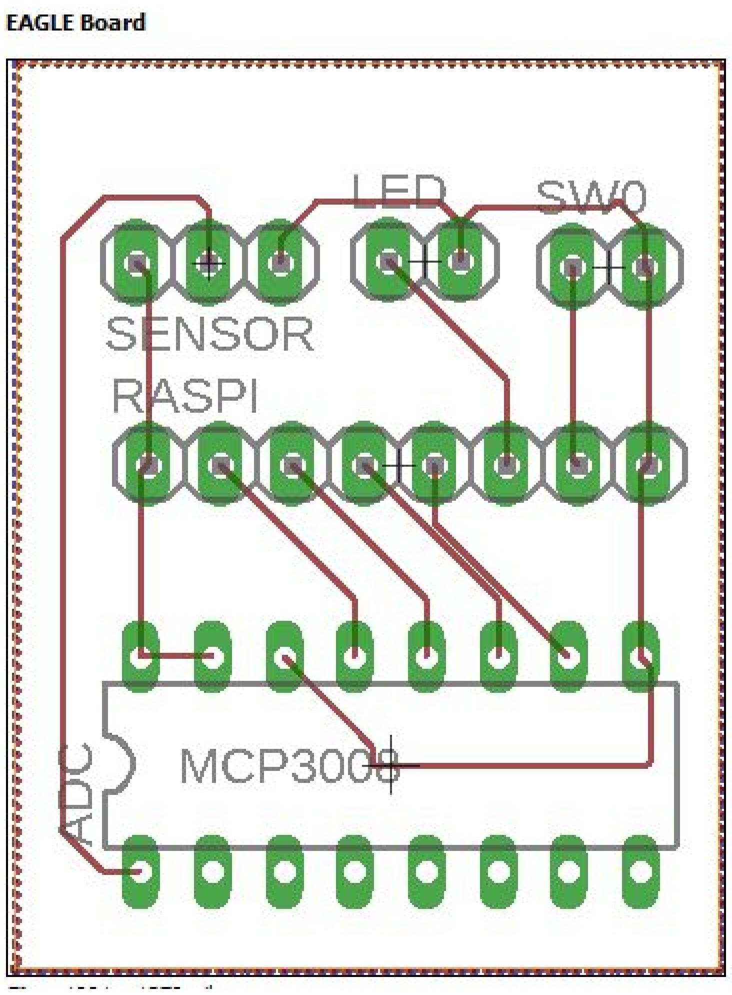


[Fig 6.2: Wiring Diagram of Peripherals]

6.4 Board Layout



[Fig 6.3: PCB Schematic]



[Fig 6.4: PCB Board Layout]

7.0 Test

7.1 Completed Tests

Monitor:

So far, the mirror system has been tested on three different monitors. The demo monitor is the 19” sceptre monitor described in the parts list. The application was able to run on this screen with little problem, the display bright is lower than desired but text is readable in decent lighting. Application was also run through the VGA port on the 20” campus LED monitors in the lab. These monitors had issues with the output levels of the Raspberry Pi’s HDMI port and would flicker. The third monitor test was on a 40” LED Samsung television. This display came through the clearest and brightest.

Light Sensor:

The photoresistor is the main hardware component for the setup used. The data is interpreted through the 10-bit converter connected to the raspberry pi. The sensor was tested in a variety of lighting conditions in order to find and optimal test value for the sensing code. First the sensor was tested for maximum values by sending waves of logic

1’s and logic 0’s. The maximum theoretical range was confirmed to be 0-1023. Then the resistor was connected and tested in normal room conditions. On average, the sensor value ranged from 200 (well-lit) to 800 (lights-off/dim). Under normal usage, the sensor tested between 550 and 650 (due to the shadow of the user in front of the mirror).

CPU:

The Raspberry Pi has been thoroughly tested for both response time and CPU usage. From the moment it is plugged in to when the display appears takes about 1 minute and 30 seconds. The CPU usage once the Pi has settled hovers between 20-30%. Comparing this to the average power consumption for a Pi Zero W shows that this project is only using about 120mW, not including the screen. Once the Pi opens the display it takes another 30 seconds or so to become responsive to layout and setting changes but once it does new changes are applied within 4 seconds of pressing the button in the app.

Android App:

The app itself has been tested in a number of different scenarios and utilizes background

tasks to stay responsive even when attempting to connect. Slight stuttering can be seen immediately after trying to connect but clears up within a second.

8.0 Follow On

8.1 Near Future Additions

The goal of the Project was to make a set of software that can be easily adapted and modified/upgraded. The next immediate plan is to simplify the installation and behavior even more. This means that code will be modified to be more easily understood by a new user. There will be documentation added that will instruct a user on how to add their own code to the program to create widgets. There will be dedicated sections of code in the install, python scripts, and javascript that will guide a user on where to place their new code so that it can cooperate with the rest of the application

8.2 Far Future Additions

Once the code is made to be more easily-edited, the next plan is to create more widgets. These will be many new sections of the screen that can give the user access to even more information. Current ideas for additional widgets are news scrolls, email notifications, and voice activation.

Appendix

A.0 Schedule

week of	1/8	1/15	1/22	1/29	2/5	2/12	2/19	2/26	3/5	3/12	3/19	3/26	4/2	4/9	4/16	4/23	4/30
Screen Layout Config	▨	▨	▨	▨													
Bluetooth server hosting				▨	▨												
Android phone communication					▨	▨	▨										
App Write to Config.json								▨	▨	▨							
BIOS automatic boot config		▨	▨	▨													
Automatic Wifi setup														▨	▨	▨	
PCB Design and Ordering											▨	▨					
PCB circuit testing													▨				
peripheral coding/control														▨	▨		
testing														▨	▨	▨	
Final Assembly																▨	▨

B.0 Issues, Delays, Improvements

In implementing the code, there are errors encountered when transitioning from a desktop development environment to the raspberry Pi. The Pi does not support as many requests and library functions as web browsers on the more powerful machines being used to write the code. This means that some of the methods needed to be rewritten into a format that only used functions available to the weaker browser on the Pi to ensure that the program runs.
In addition to library errors, the Raspberry Pi is not able to connect to some internet setups. Secondary authentication (WLAN, etc.) is sometimes difficult to setup and bypass on the device. An internet connection is required for every element of this project because of the jQuery libraries that they use. This project solves this by connecting via bluetooth to an Android device first which can then be used to select a network and enter a password.
The goal is for the project to be universally usable, to not every function/library can all be used in every machine. Error returns are being worked into modules to ensure that the programs will still run even if not every widget does. This will also make debugging modules for certain machines much easier because the broken widget will be easier to identify.

C.0 Code

PIinit.sh

```
#!/bin/sh
cd /home/pi/SmartMirror-1.0/SmartMirror/www/
#./saveToServer.sh
cd /home/pi/SmartMirror-1.0/SmartMirror/lib/scr/
sudo hciconfig hci0 up
echo "starting bluetooth device hci0"
sudo cp simple-agent /var/www/html/
cd /var/www/html/
sudo ifconfig wlan0 up
sudo rm wifilist.save
iwlist wlan0 scan | grep ESSID | sudo nano wifilist
sudo python wifitest.py &
sudo python simple-agent &
sudo python rfcomm-server.py &
Sudo python sensors.py &
sudo hciconfig hci0 piscan &
echo "making SmartMirror Discoverable and starting rfcomm server and bluetooth agent"
```

rfcommset.py

```
#This file creates the file /etc/bluetooth/rfcomm.conf with the MAC adress of the machine its run on
#Auth: Lucas Speer
#created 2/7/18


addrFile = open('addr.save', 'r')
roughStr = addrFile.read()

def MACfromSTR(str):
	addr = ""
	for i in range(13,30):
		addr += str[i]
	return addr

if __name__ == "__main__":
	firstHalf = "rfcomm1 {\n# Automatically bind the device at startup\nbind yes;\n# Bluetooth address
of the device\ndevice "
	secondHalf = "\n# RFCOMM channel for the connection\nchannel 1;\n# Description of the
connection\ncomment \"SmartMirror\";\n}"
	bdaddr = MACfromSTR(roughStr)
	toSet = firstHalf + bdaddr + secondHalf
	rfcommConfFile = open('rfcomm.conf', 'w')
	rfcommConfFile.write(toSet)
	rfcommConfFile.close()
	addrFile.close()
	print toSet
```

simple-agent

```
#!/usr/bin/python

from __future__ import absolute_import, print_function, unicode_literals

from gi.repository import GObject

import sys
import dbus
import dbus.service
import dbus.mainloop.glib
from optparse import OptionParser
import bluezutils

BUS_NAME = 'org.bluez'
AGENT_INTERFACE = 'org.bluez.Agent1'
AGENT_PATH = "/test/agent"

bus = None
device_obj = None
dev_path = None

def ask(prompt):
	try:
		return raw_input(prompt)
	except:
		return input(prompt)

def set_trusted(path):
	props = dbus.Interface(bus.get_object("org.bluez", path),
					"org.freedesktop.DBus.Properties")
	props.Set("org.bluez.Device1", "Trusted", True)

def dev_connect(path):
	dev = dbus.Interface(bus.get_object("org.bluez", path),
							"org.bluez.Device1")
	dev.Connect()

class Rejected(dbus.DBusException):
	_dbus_error_name = "org.bluez.Error.Rejected"

class Agent(dbus.service.Object):
	exit_on_release = True

	def set_exit_on_release(self, exit_on_release):
		self.exit_on_release = exit_on_release

	@dbus.service.method(AGENT_INTERFACE,
					in_signature="", out_signature="")
	def Release(self):
		print("Release")
		if self.exit_on_release:
			mainloop.quit()

	@dbus.service.method(AGENT_INTERFACE,
                                        in_signature="os", out_signature="")
    def AuthorizeService(self, device, uuid):
        return

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="o", out_signature="s")
    def RequestPinCode(self, device):
        print("RequestPinCode (%s)" % (device))
        set_trusted(device)
        return ask("Enter PIN Code: ")

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="o", out_signature="u")
    def RequestPasskey(self, device):
        print("RequestPasskey (%s)" % (device))
        set_trusted(device)
        passkey = ask("Enter passkey: ")
        return dbus.UInt32(passkey)

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="ouq", out_signature="")
    def DisplayPasskey(self, device, passkey, entered):
        print("DisplayPasskey (%s, %06u entered %u)" %
                                                (device, passkey, entered))

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="os", out_signature="")
    def DisplayPinCode(self, device, pincode):
        print("DisplayPinCode (%s, %s)" % (device, pincode))

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="ou", out_signature="")
    def RequestConfirmation(self, device, passkey):
        print("RequestConfirmation (%s, %06d)" % (device, passkey))
        confirm = ask("Confirm passkey (yes/no): ")
        if (confirm == "yes"):
            set_trusted(device)
            return
        raise Rejected("Passkey doesn't match")

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="o", out_signature="")
    def RequestAuthorization(self, device):
        return

    @dbus.service.method(AGENT_INTERFACE,
                                        in_signature="", out_signature="")
    def Cancel(self):
        print("Cancel")

def pair_reply():
    print("Device paired")
    set_trusted(dev_path)
    dev_connect(dev_path)
```

```
        mainloop.quit()

def pair_error(error):
        err_name = error.get_dbus_name()
        if err_name == "org.freedesktop.DBus.Error.NoReply" and device_obj:
                print("Timed out. Cancelling pairing")
                device_obj.CancelPairing()
        else:
                print("Creating device failed: %s" % (error))

        mainloop.quit()

if __name__ == '__main__':
        dbus.mainloop.glib.DBusGMainLoop(set_as_default=True)

        bus = dbus.SystemBus()

        capability = "NoInputNoOutput"

        parser = OptionParser()
        parser.add_option("-i", "--adapter", action="store",
                                        type="string",
                                        dest="adapter_pattern",
                                        default=None)
        parser.add_option("-c", "--capability", action="store",
                                        type="string", dest="capability")
        parser.add_option("-t", "--timeout", action="store",
                                        type="int", dest="timeout",
                                        default=60000)
        (options, args) = parser.parse_args()
        if options.capability:
                capability  = options.capability

        path = "/test/agent"
        agent = Agent(bus, path)

        mainloop = GObject.MainLoop()

        obj = bus.get_object(BUS_NAME, "/org/bluez");
        manager = dbus.Interface(obj, "org.bluez.AgentManager1")
        manager.RegisterAgent(path, capability)

        print("Agent registered")

        # Fix-up old style invocation (BlueZ 4)
        if len(args) > 0 and args[0].startswith("hci"):
                options.adapter_pattern = args[0]
                del args[:1]

        if len(args) > 0:
                device = bluezutils.find_device(args[0],
                                                options.adapter_pattern)
                dev_path = device.object_path
                agent.set_exit_on_release(False)
                device.Pair(reply_handler=pair_reply, error_handler=pair_error,
                                                        timeout=60000)
                device_obj = device
        else:
                manager.RequestDefaultAgent(path)

        mainloop.run()

        #adapter.UnregisterAgent(path)
        #print("Agent unregistered")
```

autostart

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@point-rpi
@/usr/bin/chromium-browser --kiosk --disable-notifications --disable-infobars --incognito http://localhost/
@unclutter -idle 0.1 -root
```

macine-info

```
PRETTY_HOSTNAME=SmartMirror
```

dbus-org.bluez-service

```
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)
ConditionPathIsDirectory=/sys/class/bluetooth

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib/bluetooth/bluetoothd --compat
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1
ProtectHome=true
ProtectSystem=full

[Install]
WantedBy=bluetooth.target
Alias=dbus-org.bluez.service
```

main.conf

```
[General]
```

```
# Default adapter name
# Defaults to 'BlueZ X.YZ'
Name = SmartMirror

#Disable PNAT
DisablePlugins = pnat

# Default device class. Only the major and minor device class bits are
# considered. Defaults to '0x000000'.
#Class = 0x000100

# How long to stay in discoverable mode before going back to non-discoverable
# The value is in seconds. Default is 180, i.e. 3 minutes.
# 0 = disable timer, i.e. stay discoverable forever
# The Smart mirror is discoverable for 5 minutes after it boots
DiscoverableTimeout = 300

# How long to stay in pairable mode before going back to non-discoverable
# The value is in seconds. Default is 0.
# 0 = disable timer, i.e. stay pairable forever
#PairableTimeout = 0

# Automatic connection for bonded devices driven by platform/user events.
# If a platform plugin uses this mechanism, automatic connections will be
# enabled during the interval defined below. Initially, this feature
# intends to be used to establish connections to ATT channels. Default is 60.
#AutoConnectTimeout = 60

# Use vendor id source (assigner), vendor, product and version information for
# DID profile support. The values are separated by ":" and assigner, VID, PID
# and version.
# Possible vendor id source values: bluetooth, usb (defaults to usb)
#DeviceID = bluetooth:1234:5678:abcd

# Do reverse service discovery for previously unknown devices that connect to
# us. This option is really only needed for qualification since the BITE tester
# doesn't like us doing reverse SDP for some test cases (though there could in
# theory be other useful purposes for this too). Defaults to 'true'.
#ReverseServiceDiscovery = true

# Enable name resolving after inquiry. Set it to 'false' if you don't need
# remote devices name and want shorter discovery cycle. Defaults to 'true'.
#NameResolving = true

# Enable runtime persistency of debug link keys. Default is false which
# makes debug link keys valid only for the duration of the connection
# that they were created for.
#DebugKeys = false

# Restricts all controllers to the specified transport. Default value
# is "dual", i.e. both BR/EDR and LE enabled (when supported by the HW).
# Possible values: "dual", "bredr", "le"
#ControllerMode = dual
```

```
# Enables Multi Profile Specification support. This allows to specify if
# system supports only Multiple Profiles Single Device (MPSD) configuration
# or both Multiple Profiles Single Device (MPSD) and Multiple Profiles Multiple
# Devices (MPMD) configurations.
# Possible values: "off", "single", "multiple"
#MultiProfile = off

# Permanently enables the Fast Connectable setting for adapters that
# support it. When enabled other devices can connect faster to us,
# however the tradeoff is increased power consumptions. This feature
# will fully work only on kernel version 4.1 and newer. Defaults to
# 'false'.
#FastConnectable = false

# Default privacy setting.
# Enables use of private address.
# Possible values: "off", "device", "network"
# "network" option not supported currently
# Defaults to "off"
# Privacy = off

[Policy]
#
# The ReconnectUUIDs defines the set of remote services that should try
# to be reconnected to in case of a link loss (link supervision
# timeout). The policy plugin should contain a sane set of values by
# default, but this list can be overridden here. By setting the list to
# empty the reconnection feature gets disabled.
#ReconnectUUIDs=00001112-0000-1000-8000-00805f9b34fb,0000111f-0000-1000-8000-00805f9b34fb,
0000110a-0000-1000-8000-00805f9b34fb

# ReconnectAttempts define the number of attempts to reconnect after a link
# lost. Setting the value to 0 disables reconnecting feature.
#ReconnectAttempts=7

# ReconnectIntervals define the set of intervals in seconds to use in between
# attempts.
# If the number of attempts defined in ReconnectAttempts is bigger than the
# set of intervals the last interval is repeated until the last attempt.
#ReconnectIntervals=1,2,4,8,16,32,64

# AutoEnable defines option to enable all controllers when they are found.
# This includes adapters present on start as well as adapters that are plugged
# in later on. Defaults to 'false'.
AutoEnable=true
```

<u>panel</u>

```
# lxpanel <profile> config file. Manually editing is not recommended.
# Use preference dialog in lxpanel to adjust config when you can.

Global {
  edge=top
  allign=left
  margin=0
  widthtype=percent
  width=100
  height=36
  transparent=0
  tintcolor=#000000
  alpha=0
  autohide=0
  heightwhenhidden=2
  setdocktype=1
  setpartialstrut=1
  usefontcolor=0
  fontsize=12
  fontcolor=#ffffff
  usefontsize=0
  background=0
  backgroundfile=/usr/share/lxpanel/images/background.png
  iconsize=36
}
Plugin {
  type=space
  Config {
    Size=4
  }
}
Plugin {
  type=menu
  Config {
    image=launch
    system {
    }
    separator {
    }
    item {
      name=Run...
      image=system-run
      command=run
    }
    separator {
    }
    item {
      name=Shutdown...
      image=system-shutdown
      command=logout
    }
  }
}
Plugin {
  type=space
  Config {
    Size=8
  }
```

```
}
Plugin {
  type=launchbar
  Config {
    Button {
      id=lxde-x-www-browser.desktop
    }
    Button {
      id=pcmanfm.desktop
    }
    Button {
      id=lxterminal.desktop
    }
    Button {
      id=wolfram-mathematica.desktop
    }
    Button {
      id=wolfram-language.desktop
    }
  }
}
Plugin {
  type=space
  Config {
    Size=8
  }
}
Plugin {
  type=taskbar
  expand=1
  Config {
    tooltips=1
    IconsOnly=0
    ShowAllDesks=0
    UseMouseWheel=1
    UseUrgencyHint=1
    FlatButton=0
    MaxTaskWidth=200
    spacing=1
    GroupedTasks=0
  }
}
Plugin {
  type=space
  Config {
    Size=2
  }
}
Plugin {
  type=tray
  Config {
  }
}
Plugin {
  type=space
  Config {
    Size=2
  }
}
Plugin {
  type=dhcpcdui
  Config {
  }
}
Plugin {
  type=space
  Config {
    Size=2
  }
}
Plugin {
  type=volumealsabt
  Config {
  }
}
Plugin {
  type=space
  Config {
    Size=2
  }
}
Plugin {
  type=cpu
  Config {
    ShowPercent=1
    Foreground=#a9a9a9a9a9a9
    Background=#d3d3d3d3d3d3
  }
}
Plugin {
  type=dclock
  Config {
    ClockFmt=%R
    TooltipFmt=%A %x
    BoldFont=0
    IconOnly=0
    CenterText=1
  }
}
Plugin {
  type=space
  Config {
    Size=2
  }
}
Plugin {
  type=ejecter
  Config {
  }
}
```

rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This is the rc.local file that will run everytime the SmartMirror boots and will trigger the initialization script
#
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
cd /home/pi/SmartMirror-1.0/SmartMirror/lib/scr
./PIinit.sh &
exit 0
```

defaultconfig.json

```
{
"layout":
        {
        "l1": "greeting",
        "l2": "time",
        "l3": "",
        "r1": "weather",
        "r2": "",
        "r3": ""
        },
"weather":
        {
        "useC": false,
        "zipcode": "56001"
        },
"general":
        {
        "military": false,
        "showSec": false,
        "color": "white",
        "size": 500
        }
}
```

index.html

```
<!DOCTYPE html>
<html>
        <link href="./main.css" rel="stylesheet" type="text/css">
```

```
			<head>
				<meta charset="utf-8"/>
		</head>
		<body>
			<div class="grid">
				<div class="l1">No internet connection detected. Please use app to
configure</div>
				<div class="r1"></div>
				<div class="l2"></div>
				<div class="r2"></div>
				<div class="l3"></div>
				<div class="r3"></div>
			</div>
		</body>
		<div class="weatherContainer">
				<p class="temperature mainText"></p>
				<p class="forecast subText"></p>
				<p class ="location subText"></p>
		</div>
		<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
		<script> $( ".l1" ).text("Loading...");</script>
		<script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery.simpleWeather/3.1.0/jquery.simpleWeather.min.js"></scr
ipt>
		<script src="http://localhost/test.js"></script>
		<script src="http://localhost/time.js"></script>
</html>
```

main.css

```
body {
   background: black;
}
.grid {
		display: grid;
   grid-template-columns: 50% 50%;
}
.mainText {
   font-size: 60px;
   margin: 0px auto;
}
.subText {
   font-size: 20px;
   margin: 0px auto;
}
h1 {
   padding-bottom: 16px;
}
.Greeting{
   font-size: 80px;
		font-family: Arial;
}
.l1,
.l2,
.l3{
            text-align: left;
            justify-self: start;
            grid-column-start: 1;
            grid-column-end: 2;
            align-self: start;
            margin-left: 8px !important;
}
.r1,
.r2,
.r3{
            text-align: right;
            justify-self: end;
            z-index: -1;
            grid-column-start: 2;
            align-self: start;
            margin-right: 8px !important;
}
.none{
            color: black !important;
}
div{
            color: white;
}
.mainText,
.subText,
.Greeting{
            color: white;
            font-family: Arial;
}
```

<u>test.js</u>

```
/*globals $:false */
function loadWeather(location, ForC) {
    'use strict';
            $.simpleWeather({
        location: location,
        unit: ForC,
        success: function (weather) {
            var temp = weather.temp + '&deg;' + ForC.toUpperCase(),
                                            forecast = weather.forecast[3].text,
                city = weather.city + ', ' + weather.region;
            $(".location").text(city);                          //assign .location class the text in city
            $(".temperature").html(temp);
                                $(".forecast").text(forecast);
        },
        error: function (error) {
            $(".location").text("Mankato, MN");
            $(".temperature").html("69 f");
        }
    });
```

```
}
function readTextFile(file,mystring)
{
	var rawFile = new XMLHttpRequest();
	rawFile.open("GET", file, false);
	rawFile.onreadystatechange = function ()
	{
		if(rawFile.readyState === 4)
		{
			if(rawFile.status === 200)
			{
				var allText = rawFile.responseText;
				mystring.contents += allText;
			}
		}
	}
	rawFile.send(null);
}
function Greeting(config)
{
	var greeting = $(".Greeting");
	var today = new Date();

	if ((today.getHours() < 24) && (today.getHours() > 16))
		{greeting.text("Good Evening");}
	if ((today.getHours() < 17) && (today.getHours() > 10))
		{greeting.text("Good Afternoon");}
	if (today.getHours() < 11)
		{greeting.text("Good Morning");}

}
function removeModules(spot){
	$(spot).removeClass("None");
	$(spot).removeClass("Weather");
	$(spot).removeClass("Greeting");
	$(spot).removeClass("Time");
	$(spot).removeClass("Email");
	$(spot).removeClass("News");
	//$(spot).removeClass("");
}
function assignSpots(config){
			var spots = [".l1",".r1",".l2",".r2",".l3",".r3"];
			//assign each spot it's module from config.JSON
			for(var i = 0; i < 6; i++){
				removeModules(spots[i]);			//clear all module classes
			}
			$(spots[0]).addClass(config.layout.l1);
			$(spots[1]).addClass(config.layout.r1);
			$(spots[2]).addClass(config.layout.l2);
			$(spots[3]).addClass(config.layout.r2);
			$(spots[4]).addClass(config.layout.l3);
			$(spots[5]).addClass(config.layout.r3);
		}
function update(config){
	var zip = config.weather.zipcode;			//get zipcode for loadweather()
	var oldWeather = $(".Weather");
	assignSpots(config);
	var newWeather = $(".Weather");
	if(!oldWeather.hasClass("Weather")){
		newWeather.text("");
		newWeather.append( $( ".weatherContainer" ));
	}
	var ForC;
	if(config.weather.useC === true){
		ForC = 'c';
	}
	else{
		ForC = 'f';
	}
	loadWeather(zip,ForC);		//set weather
	var emptySpots = $( ".None" );
	emptySpots.text("");
	$( ".Time" ).addClass( "mainText" );
	Greeting(config);			//set dynamic greeting
}
$( document ).ready(function () {
	'use strict'
	var mytext = {contents: ""};
	readTextFile("http://localhost/config.json",mytext);			//read config file every time in
case of changes
	var oldConfig = JSON.parse(mytext.contents);		//get a JSON array from the raw file
contents
	var config = oldConfig;
	setInterval(function () {
		mytext = {contents: ""};
		readTextFile("http://localhost/config.json",mytext);			//read config file every
time in case of changes
		config = JSON.parse(mytext.contents);					//get a JSON
array from the raw file contents
		if(config != oldConfig){
//if the config file has not been changed don't update everything
			update(config);
			oldConfig = config;
		}
	}, 4000); //Update everything but time(handled in time js) every this many ms
})
```

time.js

```
/*globals $:false */
function time(config)
{
	var time = $(".Time");
	var today = new Date();
	//t = setTimeout(startTime, 500);
	var hours = today.getHours();
	var minutes = today.getMinutes();
	var seconds = today.getSeconds();
	if (minutes < 10)
	{
		minutes = "0" + minutes;
	}
	if(seconds < 10){
		seconds = "0" + seconds;
	}
	if (config.general.military === false){
		if (hours > 12)
		{
			hours = hours - 12;
		}
	}
	var str = hours + ":" + minutes;
	if (config.general.showSec === true){
		str = str + ":" + seconds;
	}
	if (config.general.military === false){
		if (today.getHours() <= 12){
			str = str + "AM";
		}
		else{
			str = str + "PM";
		}
	}

	time.text(str);
}
$( document ).ready(function () {
	'use strict'
	var longWait = 2000;	//for not showing seconds
	var shortWait = 250;
	var mytext = {contents: ""};
	readTextFile("http://localhost/config.json",mytext);			//read config file
	var config = JSON.parse(mytext.contents);			//get a JSON array from the raw file
contents
	time(config); //intialize time
	var timeInterval = longWait;
	if(config.general.showSec){
		timeInterval = shortWait;
	}
	var newTimeInterval = timeInterval;
	var timeFunc = setInterval(function () {
		time(config);
	}, timeInterval); //Update time every this many ms
	setInterval(function () {
		mytext = {contents: ""};
		readTextFile("http://localhost/config.json",mytext);			//read config file
	   config = JSON.parse(mytext.contents);			//get a JSON array from the raw file
contents
		if(config.general.showSec){
			newTimeInterval = shortWait;
		}
```

```
                else{
                        newTimeInterval = longWait;
                }
                if(newTimeInterval != timeInterval){        //check if showSec has changed
                        clearInterval(timeFunc);
                        timeInterval = newTimeInterval;
                        var timeFunc = setInterval(function () {
                                time(config);
                        }, timeInterval);
                }
        }, 4000); //check for style changes every this many ms
});
```

wifitest.py

```
# Auth: Lucas Speer
# This file formats the list of wifi networks found by the command in SmartMirror/lib/scr/PIinit.sh
import urllib
wifiFile = open('wifilist.save', 'r+') #open the file containing the list of wifi networks generated in PIinit.sh by
the line "iwlist wpa0 scan | grep ESSID | sudo nano wifilist"
listStr = "\n" #for the duplicate check later on
for line in wifiFile: #for each line in the rough wifilist
        tmp = line
        if "\"\"" not in tmp:
                if "ESSID" in tmp:
                        sep = tmp.split("\"") #create a string array split on each quotation mark (sep[1] is
the string containing just the ssid)
                        if ("\n" + sep[1] + "\n") not in listStr: #check for duplicates
                                listStr += sep[1] + "\n"
wifiFile.close()
wifiFile = open('wifilist.save', 'w') #close and reopen the wifilist file to overwrite the contents with the better
formatted list
wifiFile.write(listStr)
wifiFile.close()
```

saveToSever.sh

```
#!/bin/bash
cd ~/SmartMirror/www
sudo cp rfcomm-server.py wifitest.py config.json index.html main.css time.js test.js /var/www/html/
```

Sensors.py

```
import RPi.GPIO as GPIO
import os
import time
import Adafruit_MCP3008

Master_Off = False

def button_callback(channel):
        if Master_Off:
```

```
                    Master_Off = True
                    os.system("xset -display :0 dpms force on")
          else:
                    Master_Off = False
                    os.system("xest -display :0 dpms force off")

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(16, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.add_event_detect(16,GPIO.RISING,callback=button_callback)

CLK = 18
MISO = 23
MOSI = 24
CS = 25
mcp = Adafruit_MCP3008.MCP3008( clk=CLK, miso=MISO, mosi=MOSI)
os.system("xset -display :0 s blank)

while True:
          value=0
          value=mcp.read_adc(0)
          if Master_Off==False:
                    if value>500:
                              os.system("xset -display :0 dpms force off)
                    if value<500
                              os.system("xset -display :0 dpms force on")
          time.sleep(2)
```

rfcomm-server.py

```
# file: rfcomm-server.py
# Auth: Lucas Speer f
# Based on code by: Albert Huang <albert@csail.mit.edu>
# desc: simple demonstration of a server application that uses RFCOMM sockets
# for use with github.com/lucasspeer/SeniorDesignSmartMirror
# $Id: rfcomm-server.py 518 2007-08-10 07:20:07Z albert $

from bluetooth import *
import os, subprocess
import time

# Set up the bluetooth socket as a server
server_sock=BluetoothSocket( RFCOMM )
server_sock.bind(("",PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]
# Must be the same as the Android app
uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

advertise_service( server_sock, "SmartMirror",
               service_id = uuid,
                service_classes = [ uuid, SERIAL_PORT_CLASS ],
                profiles = [ SERIAL_PORT_PROFILE ],
#                 protocols = [ OBEX_UUID ]
                )


def dataHandler(data):
        setFile = open('config.json', 'w')
        setFile.write(data)                     #Overrite any data with as neccesarry and save
        print(data)
        data = ""
        setFile.close()

def wifiHandler(data):
        wifiArr = data.split("\n"); #split the data on the return key to get ssid and key seperate
        ssid = wifiArr[0]
        key = wifiArr[1]
        print( wifiArr )
        wifiConf = open('/etc/wpa_supplicant/wpa_supplicant.conf', 'w')
#/etc/wpa_supplicant/wpa_supplicant.conf
        strToWrite = "ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev\nupdate_config=1\ncountry=US\n\nnetwork={\n ssid=\"" + ssid + "\"\n     psk=\"" + key +
"\"\n      key_mgmt=WPA-PSK\n}"
        wifiConf.write(strToWrite)
        wifiConf.close()
        command = "sudo reboot"                 #Kill the existing chromium process(that failed due to no
internet)...
        os.system(command)

while True:
        print("Waiting for connection on RFCOMM channel %d" % port)

        client_sock, client_info = server_sock.accept()             #Accept incoming connections
        print("Accepted connection from ", client_info)
        wifiFile = open('wifilist.save', 'r')
        wifi = wifiFile.read()
        try:
                client_sock.send(wifi)
        except IOError:
                pass
        wifiFile.close()
        try:
                while True:
                        data = client_sock.recv(1024)
                        if len(data) == 0: break
                        if data.startswith("{"):
                                dataHandler(data)
                        else:
                                wifiHandler(data)
        except IOError:
                pass
```

update.sh

```
#!/bin/bash
#this file updates the local files

cd /home/pi/
sudo rm -R SmartMirror-1.0/SmartMirror/
sudo mkdir SmartMirror-1.0/SmartMirror/
sudo cp SmartMirror/* -r SmartMirror-1.0/SmartMirror/
cd /home/pi/SmartMirror-1.0/SmartMirror/lib/
sudo cp main.conf /etc/bluetooth/
echo "Copying main.conf into /etc/bluetooth/"
echo "Making /var/run/sdp executable"
sudo cp rc.local machine-info /etc/
echo "Copying machine-info and the rc.local file that doesn't call this script into /etc/"
sudo cp panel /home/pi/.config/lxpanel/LXDE-pi/panels/
echo "Copying the panel file into /home/pi/.config/lxpanel/LXDE-pi/panels/ to remove the confirm BT pair
prompt"
sudo cp dbus-org.bluez.service /etc/systemd/system/
echo "Copying dbus-org.bluez.service into /etc/systemd/system/ to run BT in comptability mode"
sudo cp autostart /home/pi/.config/lxsession/LXDE-pi/
echo "Copying autostart into /home/pi/.config/lxsession/LXDE-pi/ to boot chrome on startup"
```

MainActivity.java

```
/*
  * Auth: Lucas Speer
  * repository: github.com/lucasspeer/BTMirror-App
  * for use with github.com/lucasspeer/SmartMirror as part of our (Lucas Speer & Isaac Matzke's) senior
design project
  * The main menu and initial activity for the BTSM (BlueTooth SmartMirror) app
  * This app acts as a settings menu and configuration for the SmartMirror running on a raspberry pi
  * This activity attemps to trigger a backgrounded RFCOMM serial connection over bluetooth
  * Initial commit date: 11/19/17
 */

package sendesign.btmirror;

import android.annotation.SuppressLint;
import android.app.DialogFragment;
import android.app.FragmentManager;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.os.Build;
import android.os.Handler;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Objects;
import java.util.Set;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    public static String MAC;
    public static UUID uuid = null;
    @SuppressWarnings("WeakerAccess")
    public static BluetoothAdapter mBluetoothAdapter;
    private BluetoothDevice BTdevice;
    public static InputStream mmInStream = null;     //Initialize IO streams
    public static OutputStream mmOutStream = null;
    public static Boolean BTFound = false;
    public static String BTStatus; //Paired, notPaired, Connected
    public static String wifiStatus;
    public static String layoutStr = "";
    public static String settingsStr = "";
    private SharedPreferences prefs = null;      //create a shared preference for storing settings
    private SharedPreferences.Editor editor;
    public BluetoothHandler BTHandler;
    public BroadcastReceiver receiver;
    private Resources resources;
    private static String conStatusText[];              //from strings.xml conStatText[] = {"Connection Status :",
"Attempting to Connect", "Successful", "Failed", "\nMac Address: "};
    private TextView statusText;
    public static final Handler handler = new Handler();
    public static String wifiSSID;
    public static String wifiKey;
    private static FragmentManager fragmentManager;
    private Button wifiSetup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /*
            Initial view setup and variable initialization.
            The saved settings/layout are defined here (Because both LayoutConfig.java and Settings.java
need to access them and couldn't if the other activity hasn't been created)
         */
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final Resources res = getResources();
        resources = res;
        prefs = this.getPreferences(Context.MODE_PRIVATE);    //retrieve default preference file for storing
layout as key value pairs {(string) "L1", (int)1}
        editor = prefs.edit();
        conStatusText = resources.getStringArray(R.array.ConStatText);
        statusText = findViewById(R.id.conStatus);

        layoutStr = prefs.getString("layoutStr", res.getString(R.string.defLayout));        //Get saved
settings/Layout
        settingsStr = prefs.getString("settingsStr", res.getString(R.string.defSettings));
        BTStatus = prefs.getString("BTStatus", "notPaired");        //Get last BTStatus
        wifiStatus = prefs.getString("wifiStatus", "notConnected");

        if(mmOutStream == null && BTStatus != "notPaired") BTStatus = "paired";

        final Button layout = findViewById(R.id.layout);     //Layout config button
        layout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, LayoutConfig.class);
                startActivity(intent);
            }
        });

        Button settings = findViewById(R.id.settingsButton);    //Configure Modules Button
        settings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, Settings.class);
                startActivity(intent);
            }
        });

        wifiSetup = findViewById(R.id.setWifiButton);
        wifiSetup.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent wifiIntent = new Intent(MainActivity.this, WifiSetup.class);
                startActivity(wifiIntent);
            }
        });

        TextView deviceList = findViewById(R.id.devList);
        TextView listTitle = findViewById(R.id.devListTitle);
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();      //get bluetooth adapter
        if (!mBluetoothAdapter.isEnabled()) {        //If bluetooth is not enabled, enable it
            mBluetoothAdapter.enable();
        }
        findDevices();
        if (!BTStatus.equals("notPaired")) {
            deviceList.setVisibility(View.INVISIBLE);      //If a SmartMirror is paired, hide the list of paired
devices
            listTitle.setVisibility(View.INVISIBLE);
        }
        uuid = UUID.fromString("94f39d29-7d6d-437d-973b-fba39e49d4ee");      //UUID which must be the
same as on the RaspPi
        if(BTdevice != null){
            BTHandler = new BluetoothHandler(BTdevice);     //create the Handler and and run it
            BTHandler.run();
        }
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                updateStatus();
            }
        }, 4000);
        final Button retry = findViewById(R.id.retryButton);
        retry.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {          //Retry Button
                // if no SmartMirror has been found already,
                findDevices();
                if(mmInStream == null && !BTStatus.equals("notPaired")){
                    BTStatus = "paired";
                }
                if(BTdevice != null){
                    BTHandler = new BluetoothHandler(BTdevice);     //create the Handler and and run it
                    BTHandler.run();
                }
                if (!BTStatus.equals("connected")) {
                    Toast.makeText(getApplicationContext(), R.string.connFailed,
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(getApplicationContext(), R.string.connSucceeded,
Toast.LENGTH_SHORT).show();
                }
                updateStatus();     //and update the status Text

            }
        });
        editor.apply();
        updateStatus();
    }

    @Override
    protected void onResume() {
        super.onResume();
        updateStatus();
    }

    @SuppressLint("SetTextI18n")
    private void findDevices() {
        /*
        findDevices() first gets the list of devices paired, then checks if any is named SmartMirror.
        If one is found the status text is updated/hidden and the device is saved.
        If none is found an error is shown
     */
        Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();   //check if already
paired
        TextView deviceList = findViewById(R.id.devList);
        TextView listTitle = findViewById(R.id.devListTitle);
        String devStr = "";
        int i = 0;
        if(!BTFound) {
            if (pairedDevices.size() > 0) {       // There are paired devices. Get the name and address of each
paired device.
                for (BluetoothDevice device : pairedDevices) {
                    devStr += device.getName() + " - " + device.getAddress() + "\n";
                    i++;
                    String deviceName = device.getName();
                    if (Objects.equals(deviceName, "SmartMirror")) {       //Mirror found among paired devices
                        MAC = device.getAddress();
                        if(mmOutStream == null){
                            BTStatus = "paired";
                        }
                        BTdevice = device;
                        deviceList.setVisibility(View.INVISIBLE);    //If a SmartMirror is found among the paired
devices hide the list of paired devices
                        listTitle.setVisibility(View.INVISIBLE);

                    } else {
                        BTStatus = "notPaired";
                    }
                }
                deviceList.setText(devStr);
            } else {
                deviceList.setText(R.string.devlisterror);  //error - no devices found
            }
        }
        else{
            deviceList.setVisibility(View.INVISIBLE);        //If a SmartMirror is found among the paired devices
hide the list of paired devices
            listTitle.setVisibility(View.INVISIBLE);
        }
    }

    private void updateStatus() {
        String statText;
        switch (BTStatus) {
            case "paired":
                statText = conStatusText[0] + conStatusText[5];        //"Connection Status: Paired, Listening"
                break;
            case "notPaired":
                statText = conStatusText[0] + conStatusText[3];
                break;
            case "connected":
                statText = conStatusText[0] + conStatusText[2];
                wifiSetup.setVisibility(View.VISIBLE);             //When connected, show wifi setup button
                break;
            default:
                statText = "typo";
                break;
        }
        statusText.setText(statText);
    }

    public Boolean isConnectedBT(){
        if(mmOutStream != null){
            BTStatus = "connected";
            return true;
        }
        else{
            return false;
        }
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        editor.putString("layoutStr", layoutStr);
        editor.putString("settingsStr", settingsStr);
        editor.putString("BTStatus", BTStatus);
        editor.putString("wifiStatus", wifiStatus);
        editor.apply();   //When mainActivity is destroyed, save current settings

    }
}
```

Settings.java

```
package sendesign.btmirror;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;

import static sendesign.btmirror.MainActivity.mmOutStream;

/*
    Author: Lucas
    Settings Activity
        shared pref, checkboxes, and dynamic titles are all working
        When the configure button is pushed the setting will be saved and if a bluetooth connection is
established they will be sent
 */
public class Settings extends AppCompatActivity {
    private SharedPreferences prefs = null;   //create a shared preference for storing settings
    private SharedPreferences.Editor editor;
    String modules[];
    public String zipcode;
    public String options[] = {"zipCodeGiven", "UseC", "24Hour", "secondsShown"};   //Keys for Boolean
options
    public Boolean defaults[] = {false, false, false, false};  //Default Settings in order with options[]
    public Boolean currentOptions[] = null;
    public Boolean savedOptions[] = null;
    private CheckBox boxArr[];
    private int optionCount = 4;   //For indexing through Shared Preferences
    private Resources resources = null;
    private int boxCnt = 3; //number of checkboxes

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.default_settings);
        resources = getResources();

        prefs = this.getPreferences(Context.MODE_PRIVATE);  //retrieve default preference file for storing
layout as key value pairs {(string) "L1", (int)1}
        editor = prefs.edit(); //open a SharedPreferences editor
        savedOptions = defaults; //initialize the
        for(int i = 0; i < optionCount; i++){
            savedOptions[i] = prefs.getBoolean(options[i], defaults[i]);
        }
        zipcode = prefs.getString("zipcode", "Enter a 5 Digit zipcode");

        modules = resources.getStringArray(R.array.modules); //get the array containing module name
strings (WARNING: The order of the string array can not be changed, add new modules to the end)
        TextView weatherTitle = findViewById(R.id.weatherTitle);  //initialize all the titles of sections using
strings from strings.xml
        weatherTitle.setText(modules[2]);
        TextView timeTitle = findViewById(R.id.timeTitle);
        timeTitle.setText(modules[3]);
        TextView title = findViewById(R.id.settingsText);
        title.setText(R.string.settingText);
        final TextView currentZip = findViewById(R.id.currentZip);
        currentZip.setText(zipcode);
        currentOptions = savedOptions; //initialize currentOptions[]

        Button back = findViewById(R.id.SettingsBack);  //initialize both buttons and set their listeners
        back.setOnClickListener(new View.OnClickListener() {                              //Back button to
Return to main menu
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Settings.this, MainActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        });

        Button config = findViewById(R.id.settingsConf);
        config.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
```

```
            config();
            currentZip.setText(zipcode);  //Update the text in the zipcode textview after config sets the
zipcode variable
        }
    });

    /*
       below this is the setup for the checkboxes, first initialize an array to hold their boolean values, then
create a listener for each box.

       To add another checkbox first add it to the layout, add name to the end of options[] and give it a
default setting in defaults[], increase boxCnt and option count,
       lastly add a chunk of code below (after the previously added checkboxes but before the for loop)
that follows the same format
     */
    boxArr = new CheckBox[boxCnt];   //Create an array to store each checkBox

    CheckBox celsius = findViewById(R.id.useCelCheck);
    celsius.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  //On state
change
            currentOptions[1] = isChecked;      //if box is now checked set appropriate boolean
        }
    });
    boxArr[0] = celsius;

    CheckBox hourFormat = findViewById(R.id.hourFormat);
    hourFormat.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            currentOptions[2] = isChecked;
        }
    });
    boxArr[1] = hourFormat;

    CheckBox showSeconds = findViewById(R.id.showSecondToggle);
    showSeconds.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            currentOptions[3] = isChecked;
        }
    });
    boxArr[2] = showSeconds;


    /*
    template for checkboxes:

    CheckBox name = findViewById(R.id.nameFromLayout);
    name.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            currentOptions[i] = isChecked;
        }
    });
    boxArr[j] = name;
     */
    for(int i = 0; i < boxCnt; i++){    //Iterates through current options to set the checkboxes (which default
to unchecked) to their appropriate value
        if(currentOptions[i+1]){      //use +1 because zipCodeGiven has no checkbox
            boxArr[i].setChecked(true);
        }
        else{
            boxArr[i].setChecked(false);
        }
    }
    editor.apply();
}

@Override
protected void onResume(){
    super.onResume();
    savedOptions = defaults;
    for(int i = 0; i < optionCount; i++){  //Gets the saved settings from the Shared pref
        savedOptions[i] = prefs.getBoolean(options[i], defaults[i]);
    }
    zipcode = prefs.getString("zipcode", "Zipcode");
    for(int i = 0; i < boxCnt; i++){    //Iterates through current options to set the checkboxes (which default
to unchecked) to their appropriate value
        if(currentOptions[i+1]){
            boxArr[i].setChecked(true);
        }
        else{
            boxArr[i].setChecked(false);
        }
    }
}
private void config(){
    EditText zip = findViewById(R.id.zipcode);
    String ziptmp = zip.getText().toString();     //get text entered under zipcode prompt
    if(ziptmp.length() == 5){                     //if correct length
        zipcode = ziptmp;                         //set zipCode
        currentOptions[0] = true;                 //set zipCodeGiven flag to true
    }
    else if (!currentOptions[0]){         //if no zipCode has been given
        zipcode = "Enter a 5 Digit zipcode";    //Set zipcode to default text
    }
    for(int i = 0; i < optionCount; i++){     //Store current options
        editor.putBoolean(options[i], currentOptions[i]);
        editor.apply();
    }
    if (currentOptions[0]){     //If zipcode has been given, store it
        editor.putString("zipcode", zipcode);
        editor.apply();
    }
    /*
    The string data contains the weather and general setting that come after the layout setting in
config.json on the Rpi
    the format is laid out in strings.xml
     */
    String data = "\n  \"weather\":\n   {\n";
    data += ("    \"useC\": " + savedOptions[1].toString() + ",\n");
    data += ("    \"zipcode\": " + zipcode + "\n   },");
    data += ("\n  \"general\":\n   {\n");
    data += ("    \"military\": " + savedOptions[2].toString() + ",\n");
    data += ("    \"showSec\": " + savedOptions[3].toString());
    data += "\n   }\n}";
    MainActivity.settingsStr = data; //Save the settings in MainActivity's shared prefs
    String strToWrite = MainActivity.layoutStr + data;
    if(MainActivity.BTFound) {
        byte dataByte[] = strToWrite.getBytes();
        try {
            if(mmOutStream != null){  //Prevents a null pointer exception
                mmOutStream.write(dataByte);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        Toast.makeText(this, resources.getText(R.string.applied), Toast.LENGTH_SHORT).show();
//Feedback to user
    }
    else{
        Toast.makeText(this, resources.getText(R.string.savedLocallyStr),
Toast.LENGTH_SHORT).show();
    }
}
}
```

LayoutConfig.java

```
package sendesign.btmirror;
/*
    *Layout config activity
    *Functions:
    * 6 spinners w/ title that contain the list of available modules
    * 2 buttons, back(to main menu), and config(set chosen layout)
    * SharedPreference file to store key value pairs locally
    *
 */
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.OutputStream;
import java.util.Objects;


public class LayoutConfig extends AppCompatActivity implements AdapterView.OnItemSelectedListener {
   private SharedPreferences prefs = null;         //create a shared preference for storing layout
   private boolean isDefault = true;
   final private int defaultLayout[] = {1, 2, 3, 0, 0, 0};      //position in R.arrays.modules, 0 = none
   private int currentLayout[] = null;         //Integer Array to hold current Layout
   private Spinner spinArr[] = null;        //Lets us globally reference each spinner by its location (TL=0,
TR=1, ML=2, MR=3,...)
   private String moduleList[] = null;      //the integer held in the layout arrays correspond to the position in
this array of strings containing the list of modules, populated from strings.xml
   final private String spots[] = {"l1", "r1", "l2", "r2", "l3", "r3"};      //strings for use as keys with
savedPreferences
   private String spotsFull[] = null;
   private int modCnt = 0;        //Number of modules, updated dynamically from Modules.xml
   private long ids[];
   private int firstRunCnt;   //onItemSelected is triggered the first time each spinner is set, this counter
variable works with a loop to counteract that in the onItemSelectedListener
   private String currentText;
   private Resources resources = null;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_layout_config);
      resources = getResources();
      modCnt = resources.getInteger(R.integer.modCount);
      firstRunCnt = 0;
      @SuppressWarnings("UnnecessaryLocalVariable") Spinner spinners[] =     //initialize spinners
(ignore the redundancy warning)
            {findViewById(R.id.LS1), findViewById(R.id.RS1), findViewById(R.id.LS2)
                  , findViewById(R.id.RS2), findViewById(R.id.LS3), findViewById(R.id.RS3)};
      spinArr = spinners;     //Set global spinner array

      prefs = LayoutConfig.this.getPreferences(Context.MODE_PRIVATE);     //retrieve default preference
file for storing layout as key value pairs {(string) "L1", (int)1}
      moduleList = resources.getStringArray(R.array.modules);     //Populate global module string array
moduleList with the values from strings.xml modules[] =  {None, Greeting, Weather, Time, News, Email}

      int savedLayout[] = defaultLayout;
      for (int i = 0; i < savedLayout.length; i++) {
         savedLayout[i] = prefs.getInt(spots[i], defaultLayout[i]);     //create an array from saved key value
pairs where key = spots[i] (a string) and defaultLayout[i] is the value returned when no key is found
      }
      currentLayout = savedLayout;     //set the current layout to our newly retrieved layoutArray
```

```
        spotsFull = resources.getStringArray(R.array.spotNames);
        setCurrentText();

        final Button back = findViewById(R.id.LayoutBack);    //back and config buttons
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(LayoutConfig.this, MainActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        });

        final Button config = findViewById(R.id.config);
        config.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Config();
            }
        });

        long tmpIDarr[] = new long[modCnt];      //Temporary array to get the individual IDs of each spinner
to be used in the onItemSelectedListener
        for (int i = 0; i < spinArr.length; i++) {
            spinArr[i].setOnItemSelectedListener(this);
            tmpIDarr[i] = spinArr[i].getId();
        }
        ids = tmpIDarr;
        setAllSpinners(currentLayout);

    }


    @Override
    protected void onResume() {
        super.onResume();
        firstRunCnt = 0;
        int savedLayout[] = defaultLayout;
        for (int i = 0; i < savedLayout.length; i++) {
            savedLayout[i] = prefs.getInt(spots[i], defaultLayout[i]);    //create an array from saved key value
pairs where key = spots[i] (a string) and defaultLayout[i] is the value returned when no key is found
        }
        currentLayout = savedLayout;
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    @Override
    public void onItemSelected(AdapterView<?> parent, View v, int position, long id) {    //Function On
spinner clicked and item Selected
        String chosenModule = parent.getSelectedItem().toString();      //get String of item selected
        int chosenModInt = 0;
        int newLayout[] = currentLayout;         //Save Layout from before update and set new Layout equal to
it(for size mostly)
```

```
        int modUpdatingInt = 0;      //Will be set to a value corresponding to which spinner was clicked (Left
top = 0, Right top = 1, Left middle = 2,...)
        int oldMod = 0;
        if (firstRunCnt >= modCnt) {      //if the spinners have been initialized
            for (int i = 0; i < modCnt; i++) {      //sets chosen module integer correspond to the position of the
selected item in moduleList
                if (Objects.equals(chosenModule, moduleList[i])) {    //Compares the string selected to the list of
module names
                    chosenModInt = i;
                }
                if (ids[i] == parent.getId()) {     //Compare the id of the Spinner the was selected with the array of
spinner IDs to determine which spinner was chosen
                    modUpdatingInt = i;
                    oldMod = newLayout[i];
                }
            }
            newLayout[modUpdatingInt] = chosenModInt;    //Update the spot chosen in the temporary layout
array
            for (int k = 0; k < modCnt; k++) {     //Iterate through each spot of the layout(Before the update)
and switch any spot that has the same module as the one just selected
                if (currentLayout[k] == chosenModInt && k != modUpdatingInt && chosenModule !=
moduleList[0]) { //with the one previously in the spinner that was clicked (unless "none" was selected)
                    newLayout[k] = oldMod;
                    String newStringArr[] = getStringForSpinArr(k, newLayout);
                    ArrayAdapter<String> spinnerArrayAdapter = new ArrayAdapter<>(spinArr[k].getContext(),
R.layout.support_simple_spinner_dropdown_item, newStringArr); //Create a new array adapter for the
changed spot

spinnerArrayAdapter.setDropDownViewResource(R.layout.support_simple_spinner_dropdown_item);
                    spinArr[k].setAdapter(spinnerArrayAdapter);
                    firstRunCnt = modCnt;
                }
            }
            currentLayout = newLayout;

        }
        firstRunCnt++;
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }

    /*
    Currently unused, resets current layout to the default. Maybe for a button or whatever.
     */
    private void setDefaultLayout() {                                          //function to reset layout to
defaults (currently unused - todo add a "reset to defaults" button)
        currentLayout = defaultLayout;
    }

    /*
    A function to set the text containing the SAVED text, called during onCreate() and when Config is
clicked
```

```
     */
    private void setCurrentText() {
        TextView currentConf = findViewById(R.id.currentConfig);
        currentText = this.getString(R.string.currentConfig);
        int odd = 0;        //Index that always should = 0 or 1. Used to add spaces after the left side or a new
line after the right side's text
        for (int i = 0; i < modCnt; i++) {
            currentText += (spotsFull[i] + moduleList[currentLayout[i]]);
            if (odd == 0) {
                currentText += "   ";
                odd++;
            } else {
                currentText += "\n";
                odd = 0;
            }
        }
        currentConf.setText(currentText);
    }

    private void setAllSpinners(int layout[]) {
        for (int i = 0; i < modCnt; i++) {
            setSpinner(spinArr[i], layout[i]);
        }
    }

    /*
        setSpinner(...) configures a single spinner(called spinner), chosenModule is the of chosen module
IDs (0-6), spotChanging is an integer to represent the position changing (spots[spotChanging])
     */
    private void setSpinner(final Spinner spinner, final int chosenModule) {
        int sel = chosenModule;
        ArrayAdapter<String> spinnerArrayAdapter = new ArrayAdapter<>(spinner.getContext(),
R.layout.support_simple_spinner_dropdown_item);

spinnerArrayAdapter.setDropDownViewResource(R.layout.support_simple_spinner_dropdown_item);
        for (int i = 0; i < modCnt; i++) {
            if (sel == modCnt) {
                sel = 0;
            }
            spinnerArrayAdapter.add(moduleList[sel]);
            sel++;
        }
        spinner.setAdapter(spinnerArrayAdapter);
    }

    /*
    The purpose of the following function is to create and assign a new array adapter for each spinner
based on the selected spinner
    Called from the onItemSelectedListener
     */
    private String[] getStringForSpinArr(int spinChanging, int layout[]) {
        int mod = layout[spinChanging];
        String resultArr[] = new String[modCnt];
        for (int i = 0; i < modCnt; i++) {
            if (mod == modCnt) {
                mod = 0;
            }
            resultArr[i] = moduleList[mod];
            mod++;
        }
        return resultArr;
    }

    /*
    The Config() function is called from the onClickListener of the Configure button
    Sets the value for the keys defined in spots[], into the default preferences file for this activity see -
https://developer.android.com/reference/android/content/SharedPreferences.html
    Need to add function to send the new layout to the mirror via the bluetooth connection
     */
    private void Config() {
        SharedPreferences.Editor editor = prefs.edit();
        for (int i = 0; i < modCnt; i++) {
            editor.putInt(spots[i], currentLayout[i]);
            editor.apply();
        }
        setCurrentText();    //Updates the containing the saved currentLayout
        /*
        The string data contains the layout settings and it's format is outlined in strings.xml
         */
        String data = "{\n  \"layout\":\n   {\n";
        for(int i = 0; i < currentLayout.length; i++){
            data += ("    \"" + spots[i] + "\": " + "\"" + moduleList[currentLayout[i]] + "\"");
            if(i != currentLayout.length - 1){
                data += ",";
            }
            data += "\n";
        }
        data += "   },";
        MainActivity.layoutStr = data;
        String strToSend = data + MainActivity.settingsStr;
        if(MainActivity.BTFound) {
            byte dataByte[] = strToSend.getBytes();
            OutputStream mOutputStream = MainActivity.mmOutStream;
            try {
                mOutputStream.write(dataByte);
            } catch (IOException e) {
                e.printStackTrace();
            }
            Toast.makeText(this, resources.getText(R.string.applied), Toast.LENGTH_SHORT).show();
        }
        else{
            Toast.makeText(this, resources.getText(R.string.savedLocallyStr),
Toast.LENGTH_SHORT).show();
        }
    }
}
```

WifiSetup.java

```
package sendesign.btmirror;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;

import java.io.IOException;

public class WifiSetup extends AppCompatActivity {

    protected RecyclerView mRecyclerView;
    protected RecyclerView.Adapter mAdapter;
    protected RecyclerView.LayoutManager mLayoutManager;
    public static TextView wifiSelected;
    private String ssid;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.wifisetup);

        String[] wifiList = ConnectedThread.wifiList;
        String cleanSet = "";
        if (wifiList != null) {                        //if a list of networks has been recieved from the pi
            for (int i = 0; i < wifiList.length; i++) {    //loop through the list
                if (wifiList[i].length() > 2) {            //if the string is not empty or a space
                    cleanSet += wifiList[i];               //append it to cleanset
                    if(i != wifiList.length - 1){          //if wifilist[i] isnt the last
                        cleanSet += "\n";                  //append a new line char
                    }
                }
            }
        }
        wifiList = cleanSet.split("\n");       //Split cleanSet on the newline char previously added to get an
array of strings
        ConnectedThread.wifiList = wifiList;         //and set it in ConnectedThread so it can be accessed
universally

        Button cancel = findViewById(R.id.wifiCancelbutton);    //Set cancel button to return to the main
menu
```

```
        cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(WifiSetup.this, MainActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        });

        wifiSelected = findViewById(R.id.currentWifiSelection);     //Get the TextView for the current selection
        mRecyclerView = findViewById(R.id.wifiList);                //Get the RecyclerView for the the list of
SSIDs
        mLayoutManager = new LinearLayoutManager(WifiSetup.this);   //Required for the RecyclerView
        mRecyclerView.setLayoutManager(mLayoutManager);             //       "
        mAdapter = new MyAdapter(wifiList);            //Get the adapter for String[] -> RecyclerView as defined
in MyAdapter
        mRecyclerView.setAdapter(mAdapter);

        final EditText psw = findViewById(R.id.password);       //Set the EditText for the password
        psw.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                psw.bringToFront();
            }
        });

        final Button config = findViewById(R.id.wifiConfButton);    //Set the config button
        config.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String password = psw.getText().toString();         //get the entered password ( "" is acceptable)
                ssid = ConnectedThread.ssid;                        //get the selected SSID
                String toSend = (ssid + "\n" + password);           //the string takes the format
"SSID\nPASSWORD", it is split into an array in rfcomm-server.py on the RPi
                try {
                    MainActivity.mmOutStream.write(toSend.getBytes());      //Send the selection
                } catch (IOException e) {
                    e.printStackTrace();
                }
                MainActivity.BTStatus = "paired";
                Intent intent = new Intent(WifiSetup.this, MainActivity.class);     //return to the main menu
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        });
    }
}
```

BluetoothHandler.java

```
package sendesign.btmirror;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;

import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

import java.io.IOException;
import java.util.UUID;

/**
 * Created by Lucas on 11/19/17.
 * handles the bluetooth connection process. Once connected it hands the Bluetooth Socket to
ConnectedThread, which handles the data transfer
 */

public class BluetoothHandler extends Thread{

    private static final String TAG = "MY_APP_DEBUG_TAG";
    private static final int PERIOD = 5000;
    final private BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    private static BluetoothSocket mmSocket;
    private static BluetoothDevice BTdevice = null;
    private ConnectedThread BTthread;
    private Backgrounder task = null;

    BluetoothHandler(BluetoothDevice device) {
        task = new Backgrounder();
        task.doInBackground();
        BTdevice = device;
    }

    public void run() {
        mBluetoothAdapter.cancelDiscovery();    //Cancel discovery because it otherwise slows down the
connection.
        try {
            if(mmSocket != null){
                mmSocket.connect();   //Connect to the remote device through the socket. This call blocks until
it succeeds or throws an exception

            }
        } catch (IOException connectException) {
            try {
                mmSocket.close();    //Unable to connect; close the socket and return.
                MainActivity.BTStatus = "paired";
            } catch (IOException closeException) {
                Log.e(TAG, "Could not close the client socket", closeException);
            }
            return;
        }
        BTthread = new ConnectedThread(mmSocket);     //Create a new thread to handle the connection.
        BTthread.start();
    }
    void cancel() {
        try {
            mmSocket.close();    // Closes the client socket and causes the thread to finish.
        } catch (IOException e) {
            Log.e(TAG, "Could not close the client socket", e);
        }
        BTthread.cancel();
        if (task != null) {
            task.cancel(false);
        }
    }
    static class Backgrounder extends AsyncTask<Void, Void, Void> {
        /*
            this function creates a background task for the bluetooth connection to keep the UI responsive to
activate call:
                task = new bluetoothTask();
                task.execute();

            make sure to have task.cancel(false); in the cancel() function
         */

        @Override
        protected Void doInBackground(Void... voids) {
            BluetoothSocket tmp = null;   // Use a temporary object that is later assigned to mmSocket
because mmSocket is final.
            UUID uuid = MainActivity.uuid;
            try {
                if(BTdevice != null){
                    tmp = BTdevice.createRfcommSocketToServiceRecord(uuid);    //Get a BluetoothSocket to
connect with the given BluetoothDevice. uuid must match rfcomm-server.py on the Rpi
                }
            } catch (IOException e) {
                Log.e(TAG, "Socket's create() method failed", e);
            }
            mmSocket = tmp;

            return null;
        }
    }
}
```

<u>ConnectedThread.java</u>

```
package sendesign.btmirror;

import android.app.DialogFragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

import static sendesign.btmirror.MainActivity.wifiSSID;

/**
 * Created by pook on 1/2/18.
 * This thread handles the BT socket after a connection is established
 */

public class ConnectedThread extends Thread {
    private static final String TAG = "MY_APP_DEBUG_TAG";
    private Handler mHandler; // handler that gets info from Bluetooth service

    // Defines several constants used when transmitting messages between the
    // service and the UI.
    private interface MessageConstants {
        int MESSAGE_READ = 0;
        int MESSAGE_WRITE = 1;
        int MESSAGE_TOAST = 2;
    }
    // ... (Add other message types here as needed.)
    private final BluetoothSocket mmSocket;
    private byte[] mmBuffer; // mmBuffer store for the stream
    public Boolean isConnected;
    public static String wifiList[];
    public static String ssid;
    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the input and output streams; using temp objects because
        // member streams are final.
        if(socket != null) {
            try {
                tmpIn = socket.getInputStream();

            } catch (IOException e) {
                Log.e(TAG, "Error occurred when creating input stream", e);
            }
            try {
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
                Log.e(TAG, "Error occurred when creating output stream", e);
            }
            MainActivity.mmInStream = tmpIn;
            MainActivity.mmOutStream = tmpOut;
            MainActivity.BTFound = true;
            isConnected = true;
            MainActivity.BTStatus = "connected";       //Update the status for the main menu text
        }
        else{
            MainActivity.BTFound = false;
            isConnected = false;
            MainActivity.BTStatus = "paired";    //Update the status for the main menu text
            cancel();
        }
    }

    public void run() {
        Backgrounder task = new Backgrounder();
        task.doInBackground();

    }

    // Call this method from the main activity to shut down the connection.
    public void cancel() {
        try {
            if(mmSocket != null){
                mmSocket.close();
            }
        } catch (IOException e) {
            Log.e(TAG, "Could not close the connect socket", e);
        }
    }
    static class Backgrounder extends AsyncTask<Void, Void, Void> {
        /*
            this function creates a background task for the bluetooth connection to keep the UI responsive to
activate call:
                task = new bluetoothTask();
                task.execute();

            make sure to have task.cancel(false); in the cancel() function
         */
        private byte[] mmBuffer; // mmBuffer store for the stream
        private InputStream mmInStream = MainActivity.mmInStream;
        private OutputStream mmOutStream = MainActivity.mmOutStream;
        private final Handler mhandler = MainActivity.handler;
        @Override
        protected Void doInBackground(Void... voids) {
            mmBuffer = new byte[1024];
            int numBytes = 0; // bytes returned from read()
            String tmp = "";
            // Keep listening to the InputStream until an exception occurs.
            while(MainActivity.mmInStream != null){
                try {
                  // Read from the InputStream.
                  numBytes = mmInStream.read(mmBuffer);
                  char finalByteArr[] = new char[numBytes];
                  for(int i = 0; i < numBytes; i++){
                        finalByteArr[i] = ((char) mmBuffer[i]);
                        tmp += finalByteArr[i];
                  }
                  wifiList = tmp.split("\n");
                  MainActivity.wifiStatus = "notConnected";
                } catch (IOException e) {
                  Log.d(TAG, "Input stream was disconnected", e);
                  break;
                }
            }
            return null;
        }
    }


}
```

MyAdapter.java

```
package sendesign.btmirror;

import android.app.DialogFragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

import static sendesign.btmirror.MainActivity.wifiSSID;

/**
 * Created by pook on 1/2/18.
 * This thread handles the BT socket after a connection is established
 */

public class ConnectedThread extends Thread {
    private static final String TAG = "MY_APP_DEBUG_TAG";
    private Handler mHandler; // handler that gets info from Bluetooth service

    // Defines several constants used when transmitting messages between the
    // service and the UI.
    private interface MessageConstants {
        int MESSAGE_READ = 0;
        int MESSAGE_WRITE = 1;
        int MESSAGE_TOAST = 2;
    }
    // ... (Add other message types here as needed.)
    private final BluetoothSocket mmSocket;
    private byte[] mmBuffer; // mmBuffer store for the stream
    public Boolean isConnected;
    public static String wifiList[];
    public static String ssid;
    public ConnectedThread(BluetoothSocket socket) {
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the input and output streams; using temp objects because
        // member streams are final.
        if(socket != null) {
            try {
                tmpIn = socket.getInputStream();

            } catch (IOException e) {
                Log.e(TAG, "Error occurred when creating input stream", e);
            }
            try {
                tmpOut = socket.getOutputStream();
            } catch (IOException e) {
                Log.e(TAG, "Error occurred when creating output stream", e);
            }
            MainActivity.mmInStream = tmpIn;
            MainActivity.mmOutStream = tmpOut;
            MainActivity.BTFound = true;
            isConnected = true;
            MainActivity.BTStatus = "connected";        //Update the status for the main menu text
        }
        else{
            MainActivity.BTFound = false;
            isConnected = false;
            MainActivity.BTStatus = "paired";    //Update the status for the main menu text
            cancel();
        }
    }

    public void run() {
        Backgrounder task = new Backgrounder();
        task.doInBackground();

    }

    // Call this method from the main activity to shut down the connection.
    public void cancel() {
        try {
            if(mmSocket != null){
                mmSocket.close();
            }
        } catch (IOException e) {
            Log.e(TAG, "Could not close the connect socket", e);
        }
    }
    static class Backgrounder extends AsyncTask<Void, Void, Void> {
        /*
            this function creates a background task for the bluetooth connection to keep the UI responsive to
activate call:
                task = new bluetoothTask();
                task.execute();

            make sure to have task.cancel(false); in the cancel() function
         */
        private byte[] mmBuffer; // mmBuffer store for the stream
        private InputStream mmInStream = MainActivity.mmInStream;
        private OutputStream mmOutStream = MainActivity.mmOutStream;
        private final Handler mhandler = MainActivity.handler;
        @Override
        protected Void doInBackground(Void... voids) {
            mmBuffer = new byte[1024];
            int numBytes = 0; // bytes returned from read()
            String tmp = "";
            // Keep listening to the InputStream until an exception occurs.
            while(MainActivity.mmInStream != null){
                try {
                  // Read from the InputStream.
                  numBytes = mmInStream.read(mmBuffer);
                  char finalByteArr[] = new char[numBytes];
                  for(int i = 0; i < numBytes; i++){
                      finalByteArr[i] = ((char) mmBuffer[i]);
                      tmp += finalByteArr[i];
                  }
                  wifiList = tmp.split("\n");
                  MainActivity.wifiStatus = "notConnected";
                } catch (IOException e) {
                    Log.d(TAG, "Input stream was disconnected", e);
                    break;
                }
            }
            return null;
        }
    }


}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="sendesign.btmirror.MainActivity">

<LinearLayout
   android:id="@+id/linearLayout6"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_marginEnd="8dp"
   android:layout_marginStart="8dp"
   android:layout_marginTop="8dp"
   android:orientation="vertical"
   app:layout_constraintEnd_toEndOf="parent"
   app:layout_constraintStart_toStartOf="parent"
   app:layout_constraintTop_toTopOf="parent">

   <TextView
      android:id="@+id/Title"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/main_menu"
      android:textAlignment="center"
      android:textAppearance="@style/TextAppearance.AppCompat.Display1" />

   <Space
      android:layout_width="match_parent"
      android:layout_height="50dp" />

   <Button
      android:id="@+id/layout"
      style="@style/Widget.AppCompat.Button.Colored"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:paddingTop="10dp"
      android:text="@string/edit_layout"
      android:textAlignment="center" />

   <Button
      android:id="@+id/settingsButton"
      style="@style/Widget.AppCompat.Button.Colored"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/settingText" />

   <Button
      android:id="@+id/retryButton"
      style="@style/Widget.AppCompat.Button.Colored"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="@string/retry" />

   <Button
      android:id="@+id/setWifiButton"
      style="@style/Widget.AppCompat.Button"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/wifiButton"
            android:visibility="gone" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout10"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:orientation="vertical"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout6">

        <TextView
            android:id="@+id/conStatus"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/devListTitle"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@string/deviceList"
            android:textAlignment="center"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/linearLayout6" />

        <TextView
            android:id="@+id/devList"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textAlignment="center"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/devListTitle" />

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

default_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="sendesign.btmirror.Settings">

<android.support.constraint.ConstraintLayout
    android:id="@+id/linearLayout8"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:keyboardNavigationCluster="true"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:ignore="UnusedAttribute">

    <android.support.constraint.ConstraintLayout
        android:id="@+id/linearLayout"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:keyboardNavigationCluster="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/linearLayout7"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="UnusedAttribute">

        <Button
            android:id="@+id/SettingsBack"
            style="@style/Widget.AppCompat.Button.Colored"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp"
            android:text="@string/cancel"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent" />

    </android.support.constraint.ConstraintLayout>

    <android.support.constraint.ConstraintLayout
        android:id="@+id/linearLayout7"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/linearLayout"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/settingsConf"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:text="@string/confText"
        android:textAlignment="center"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
    </android.support.constraint.ConstraintLayout>
</android.support.constraint.ConstraintLayout>

<LinearLayout
    android:id="@+id/linearLayout4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/settingsText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Headline"
        android:textSize="30sp" />

    <TextView
        android:id="@+id/timeTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginStart="8dp"
        android:textAppearance="@style/TextAppearance.AppCompat.Headline"
        android:textSize="24sp" />

    <CheckBox
        android:id="@+id/hourFormat"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginStart="16dp"
        android:checked="false"
        android:text="@string/time24hrOption" />

    <CheckBox
        android:id="@+id/showSecondToggle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp"
    android:checked="false"
    android:text="@string/secondsOption" />

<TextView
    android:id="@+id/weatherTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginStart="8dp"
    android:textAppearance="@style/TextAppearance.AppCompat.Headline"
    android:textSize="24sp" />

<TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp"
    android:text="@string/zip" />

<TextView
    android:id="@+id/currentZip"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="24dp"
    android:layout_marginStart="24dp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp"
    android:text="@string/enterZip" />

<EditText
    android:id="@+id/zipcode"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp"
    android:ems="10"
    android:inputType="textPersonName"
    android:selectAllOnFocus="false" />

<CheckBox
    android:id="@+id/useCelCheck"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginStart="16dp"
    android:checked="false"
    android:text="@string/useC" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

<u>activity_layout_config.xml</u>

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sendesign.btmirror.LayoutConfig">

    <TextView
        android:id="@+id/layoutText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="@string/edit_layout"
        android:textAlignment="center"
        android:textAppearance="@style/TextAppearance.AppCompat.Headline"
        android:textSize="30sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/linearLayout5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        app:layout_constraintTop_toTopOf="parent"
        android:baselineAligned="false">

        <LinearLayout
            android:id="@+id/linearLayout3"
            android:layout_width="@dimen/halfwidth"
            android:layout_height="wrap_content"
            android:layout_weight="0"
            android:orientation="vertical"
            app:layout_constraintBottom_toTopOf="@+id/linearLayout2"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">

            <TextView
                android:id="@+id/L1"
                android:layout_width="190dp"
                android:layout_height="wrap_content"
                android:paddingTop="50dp"
        android:text="Left Top"
        android:textAlignment="center"
        tools:ignore="HardcodedText" />

    <Spinner
        android:id="@+id/LS1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:clipChildren="false"
        android:entries="@array/modules" />

    <TextView
        android:id="@+id/L2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="20dp"
        android:text="Left Middle"
        android:textAlignment="center"
        tools:ignore="HardcodedText" />

    <Spinner
        android:id="@+id/LS2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/modules"/>

    <TextView
        android:id="@+id/L3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="20dp"
        android:text="Left Bottom"
        android:textAlignment="center"
        tools:ignore="HardcodedText" />

    <Spinner
        android:id="@+id/LS3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:entries="@array/modules"
        android:spinnerMode="dropdown" />

</LinearLayout>

<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="@dimen/halfwidth"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/linearLayout3"
    app:layout_constraintTop_toTopOf="parent">
```

```
        <TextView
            android:id="@+id/R1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingTop="50dp"
            android:text="Right Top"
            android:textAlignment="center"
            tools:ignore="HardcodedText" />

        <Spinner
            android:id="@+id/RS1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/modules" />

        <TextView
            android:id="@+id/R2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingTop="20dp"
            android:text="Right Middle"
            android:textAlignment="center"
            tools:ignore="HardcodedText" />

        <Spinner
            android:id="@+id/RS2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/modules"/>

        <TextView
            android:id="@+id/R3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingTop="20dp"
            android:text="Right Bottom"
            android:textAlignment="center"
            tools:ignore="HardcodedText" />

        <Spinner
            android:id="@+id/RS3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:entries="@array/modules"/>

    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="52dp"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginRight="8dp"
    android:baselineAligned="false">

    <android.support.constraint.ConstraintLayout
        android:id="@+id/ConstraintLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0"
        tools:ignore="InefficientWeight">

        <Button
            android:id="@+id/LayoutBack"
            style="@style/Widget.AppCompat.Button.Colored"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_weight="0"
            android:text="Back"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/config"
            app:layout_constraintStart_toStartOf="parent"
            tools:ignore="HardcodedText,InefficientWeight,NestedWeights" />

        <Button
            android:id="@+id/config"
            style="@style/Widget.AppCompat.Button.Colored"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:layout_marginStart="8dp"
            android:layout_weight="0"
            android:text="@string/confText"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/LayoutBack"
            tools:ignore="InefficientWeight,NestedWeights" />
    </android.support.constraint.ConstraintLayout>

</LinearLayout>

<TextView
    android:id="@+id/currentConfig"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:textAlignment="center"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout5" />

</android.support.constraint.ConstraintLayout>
```

item_simple_itemview.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp">

    <TextView
        android:id="@+id/clciked_text"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="8sp"
        android:background="@color/colorAccent"
        android:text="wifiOption"
        android:textAppearance="@android:style/TextAppearance.Material.Medium"
        android:textSize="30sp"
        android:textStyle="bold"
        android:visibility="invisible" />

    <TextView
        android:id="@+id/simple_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8sp"
        android:text="wifiOption"
        android:textAppearance="@android:style/TextAppearance.Material.Medium"
        android:textSize="24sp" />
</FrameLayout>
```

AndroidMainfiest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="sendesign.btmirror">


    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-feature android:name="android.hardware.wifi" />


    <application
```

```
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:fullBackupContent="@xml/backup_descriptor"
        tools:ignore="GoogleAppIndexingWarning">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".LayoutConfig"
            android:label="LayoutConfig"
            android:windowSoftInputMode="adjustPan"/>
        <activity android:name=".Settings"
            android:windowSoftInputMode="stateHidden|adjustResize"/>
        <activity android:name=".WifiSetup"
            android:windowSoftInputMode="stateHidden|adjustPan"/>
            />
    </application>

</manifest>
```