P2 - Prototype Development

Overall process

Hardware development:

- To begin, our entire group conducted brainstorming to discuss our prototype goals. Once our goals were established, we decided which electronics were required to build the prototype:

    - Raspberry Pi Model 3
    - Speaker
    - Speaker Amplifier
    - Neopixel Strip (RGB LEDs)
    - Two USB wife cards
    - 12v to 5v 3A usb convertor

- Finally, once all electronics were acquired we wired them all together and performed preliminary tests to ensure they were all working as designed

Software development:

- During our group brainstorming session, we discussed what our software would need to do to meet our prototype functionality goals software flow diagram.
- From the software flow diagram, we wrote out the application control pseudocode on a whiteboard
- Then the pseudo code transcribed into python.
- The team work on the access point and music playing portions of the software as well as working on the calendar syncing and website blocking code

Design decisions

The prototype was primarily designed to remove digital distractions and create a conducive work environment for our target users. It currently supports the following tasks:

a. Automatically sync with Google calendar and find events labeled "work"
b. During time slots labelled as work in the calendar:
    i. Automatically blocks distracting websites for the user
    ii. Plays study music from Pandora to help the user focus on work
    iii. Automatically schedules break for the user to help balance cognitive load
    iv. Notifies the user of work and rest times through notification RGB LEDs that light up blue during work time and green during rest time

Our prototype design was inspired from the concept of context aware computing particularly the automatic contextual reconfiguration (1). Our main goal is to engage user in task (i.e. to begin work or take a break) with minimal interaction with the tool. The design decisions for the prototype were guided by the following:

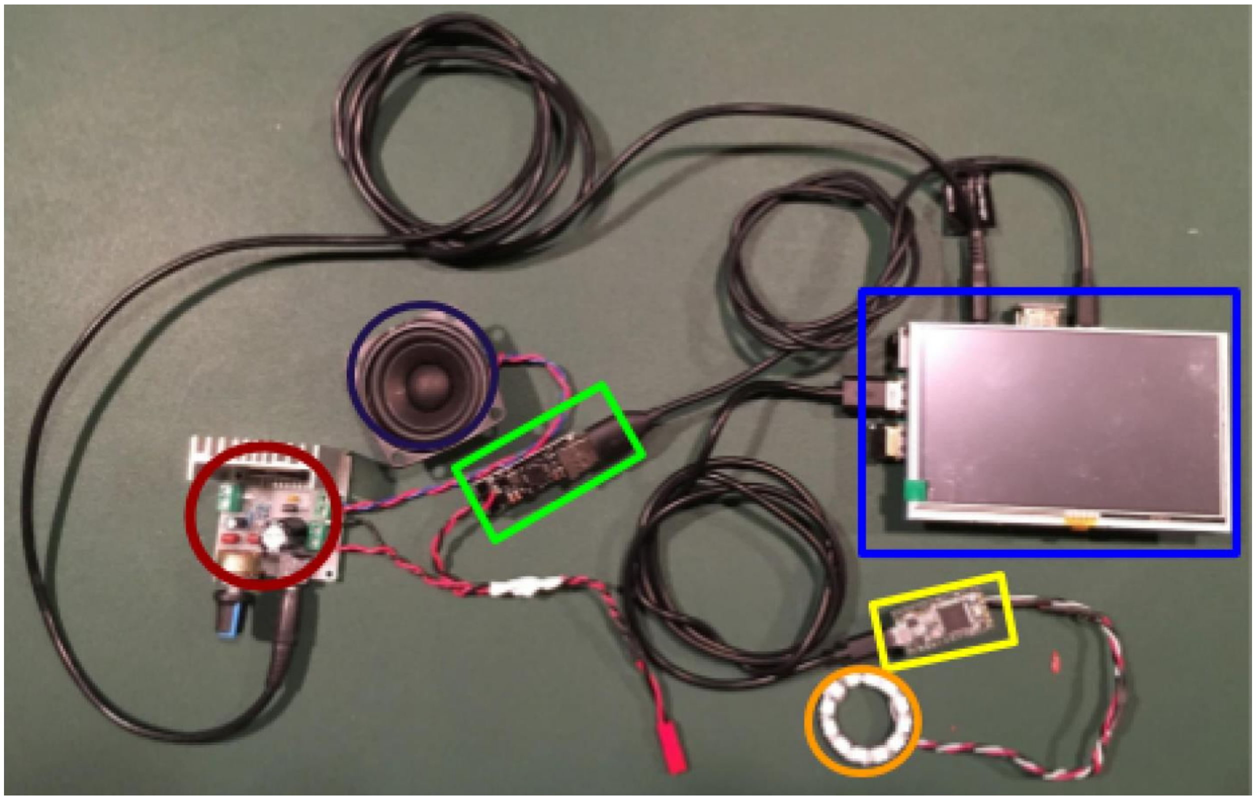
- Context awareness: the tool syncs with the user's calendar automatically. It "sees" when the user is supposed to be working and creates conducive environment for working
- Minimal user interaction: automatically plays music and sets timer for breaks thereby minimizing the cognitive load of user while engaging in task
- Able to connect with multiple digital devices: able to block distracting websites from multiple devices
- Portable: easy enough to be used in more than one location, ideal for predesignated study/work spaces

Prototyping process

Hardware:

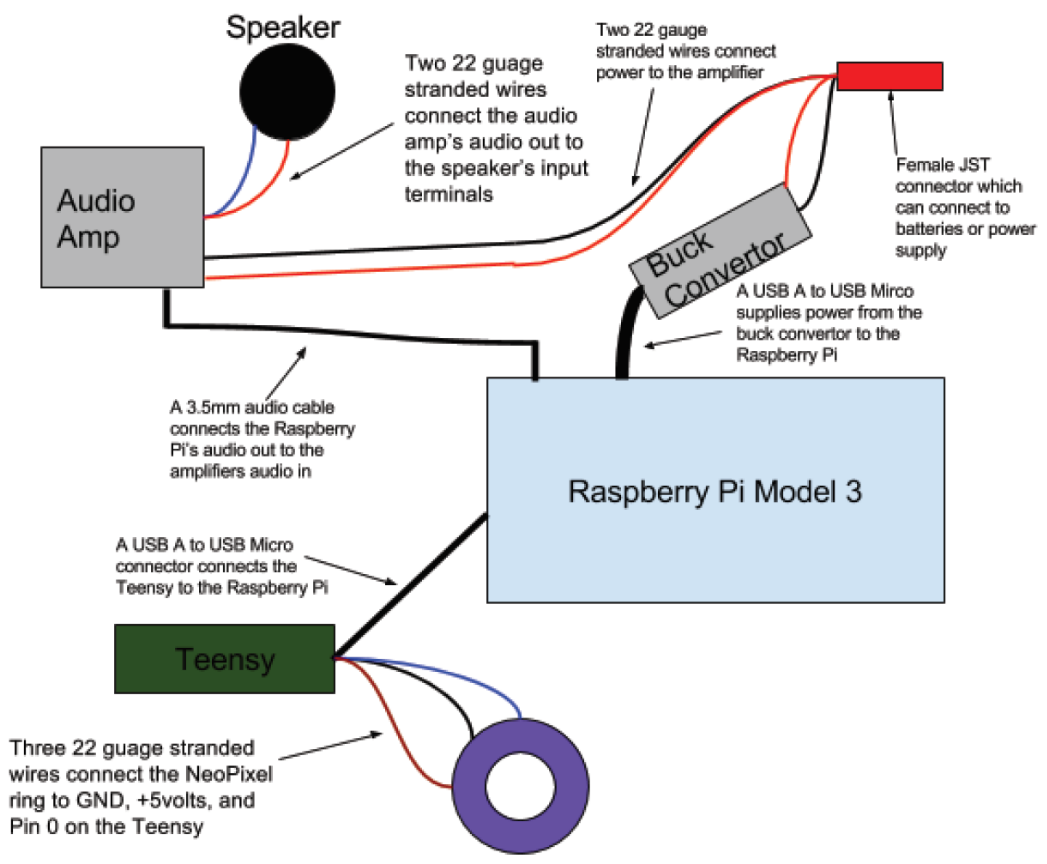
Our prototype consists of a Raspberry Pi Model 3 which is capable of acting as a WiFi router, is able to play music through an external amplifier and speakers, and is able to control RGB LED notification lights. The Raspberry Pi has been programmed to run a Python script on startup that automatically controls all Google calendar API calls and hardware functions.
Since this is prototype version one, all of our hardware components were assembled and tested in as modular a way as possible. For instance, standard USB A to USB micro connectors were used for powering the Raspberry Pi from the buck convertor and for connecting the Teensy microcontroller to the Raspberry Pi. Also, a standard male-to-male 3.5mm audio cable was used to connect the Raspberry Pi's audio output to the audio input of the external amplifier. By utilizing standard connectors, Chad was able to test each component individually to ensure functionality before hooking the complete system together. For connections that were unable to be made with standardized cables, 22-gauge stranded wire was utilized with solder connections on each board.

Hardware Overview:



From left to right, our hardware components are as follows: 12-volt external audio amplifier (circled in red), 2.5” speaker (circled in purple), 12 volts to 5v USB buck convertor (green rectangle), 12 LED NeoPixel ring (circled in orange), Teensy 3.2 microcontroller (yellow rectangle), Raspberry Pi Model 3 (shown here with a 5” TFT touchscreen used for debugging, in blue rectangle).

Hardware Wiring Diagram:


Speaker
Two 22 guage stranded wires connect the audio amp's audio out to the speaker's input terminals
Two 22 gauge stranded wires connect power to the amplifier
Female JST connector which can connect to batteries or power supply
Audio Amp
Buck Convertor
A USB A to USB Mirco supplies power from the buck convertor to the Raspberry Pi
A 3.5mm audio cable connects the Raspberry Pi's audio out to the amplifiers audio in
Raspberry Pi Model 3
A USB A to USB Micro connector connects the Teensy to the Raspberry Pi
Teensy
Three 22 guage stranded wires connect the NeoPixel ring to GND, +5volts, and Pin 0 on the Teensy

Software engineering:

With the exception of the Arduino code which runs on the Teensy microcontroller to control the NeoPixel LEDs, our entire application is written in Python. Our python script is executable and is called upon system startup as a Chrono task. The Python application utilizes the Google Calendar API to check when a user has scheduled "Work" events, and during those times turns the notification LEDs blue through serial commands to the Teensy microcontroller and plays music from Pandora using an open source utility called PianoBar. In order to host an access point from the Raspberry Pi and block websites through that access point, Udhcpd and Hostapd are utilized.

Google Calendar:

The users google calendar is the key starting point of our system. The smart speaker python application code has been developed to operate all functionalities of the system like playing/ stopping the music, changing light colors and blocking/unblocking distracting websites according to work and consequently rest times calculated from events found from the user's calendar. Based on what have been implemented:

1. the system is setup to sync with Google calendar for a specific period of time; for instance, 5 days.
2. All work events will be detected for such period of time.
3. By using APScheduler library, all detected work events are scheduling to be executed at the right moment. For every 60-minute work event, we consider 45-minute work and 15 -minute rest. If the remaining time for a work event is less than 15 minutes, the work function won't be called any more. If the work time remaining in event after any rest time is less than 45 min, whenever it is done, the rest function announces that work time is over.

Blocking/Unblocking Distracting Websites:

Squid proxy is used to filter internet traffic from most common protocols including HTTP and HTTPS. Fereshteh has implemented and tested squid on Windows to block distracting websites. We have begun Squid integration into the Raspberry Pi prototype but have encountered several configuration bugs that require fixing. In our next iteration, Squid will be fully configured and tested in Linux for Raspberry Pi. Websites to be filtered are set in the squid.conf file. To control which websites are filtered during work versus rest times, two separate squid.conf files for "work" and "rest" functionalities were created; each config file has its own filtering rules. During work times, squid is pointed to the work config file and the rest config file during rest or non-work times.

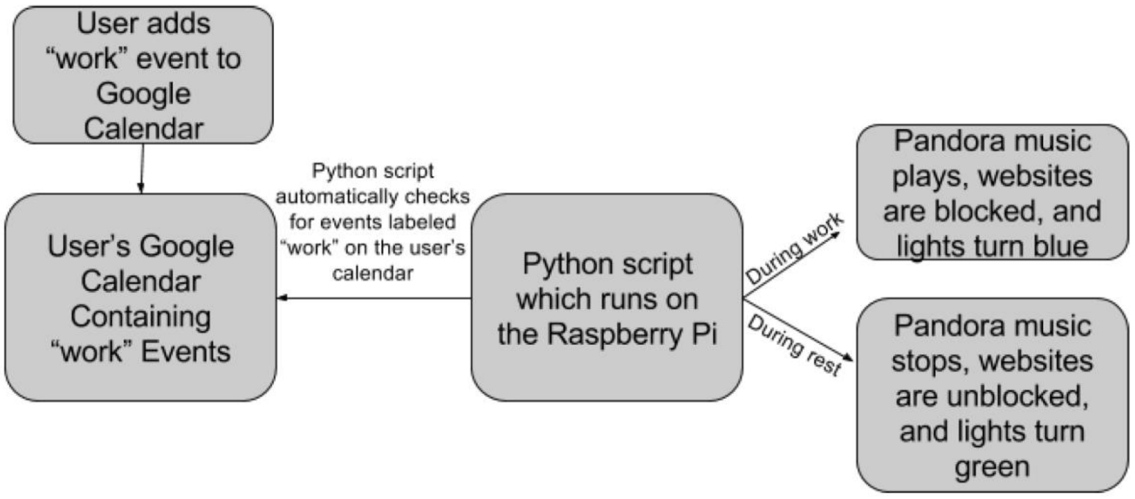
Libraries and APIs used:

- Udhcpd
  A dhcp client used for hosting WiFi on the Raspberry Pi.
  https://udhcp.busybox.net/
- Hostapd
  A utility that uses a USB WiFi dongle as an access point host and runs an authentication server for wireless connections.
  https://w1.fi/hostapd/
- Pianobar
  Pianobar is a console based client for playing Pandora internet radio stations.
  https://github.com/PromyLOPh/pianobar
- Adafruit NeoPixel Library
  An arduino library made for interfacing and controlling NeoPixel LEDs.
  https://github.com/adafruit/Adafruit_NeoPixel
- Google Calendar API
  An API which allows access to Google Calendar user accounts, and user scheduled events.
  https://developers.google.com/google-apps/calendar/quickstart/python
- APScheduler
  A Python library to schedule Python code to be executed later, either just once or periodically.
  https://pypi.python.org/pypi/APScheduler

- Squid
  A web proxy application to filter and cache web traffic.
  https://www.brennan.id.au/11-Squid_Web_Proxy.html

User Manual:

Software Storyboard


User adds "work" event to Google Calendar
User's Google Calendar Containing "work" Events
Python script automatically checks for events labeled "work" on the user's calendar
Python script which runs on the Raspberry Pi
During work
During rest
Pandora music plays, websites are blocked, and lights turn blue
Pandora music stops, websites are unblocked, and lights turn green

Our prototype has been designed to be a context aware, interface-less device with a very minimal setup requirement for the end user. On the device, a Python script runs at startup to automatically check the user's Google calendar and activate the device during work times. In its current iteration state, the prototype user only needs to interact with prototype to customize default settings in the three following parts:

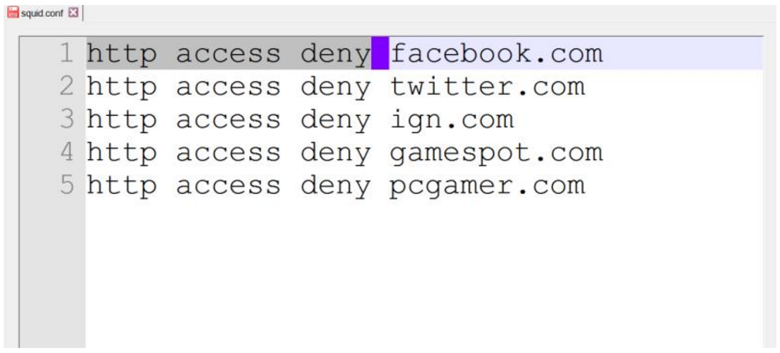
Setting work events in calendar:

In order to have our application automatically find "work" events, the user must schedule events titled "work" in their Google calendar. There is no additional effort required by the user in order to sync his or her calendar with our developed prototype since all of the calendar event detecting, event scheduling, and sending notifications for work and rest times (which controls the music playing, light display and blocking/unblocking websites), are implemented at the backend and works with no need to any interaction with user.

Providing the list of distraction websites as a blacklist:

Since digital distractions are unique to each user, every person who uses the smart speaker device will want to configure their own distraction blacklist. To customize which websites the Squid proxy blocks, the user

must add the website to the squid.conf file. The squid config file for work is located at ~/work/squid.conf on the Raspberry Pi. To add a new website to the blacklist (for instance www.reddit.com), add: “http_access deny reddit.com” to the squid config file.
The format of writing the blacklist is shown in the following image.



```
http access deny facebook.com
http access deny twitter.com
http access deny ign.com
http access deny gamespot.com
http access deny pcgamer.com
```

Configuring which Pandora Station Plays Automatically:

Currently, in prototype version v1.0, all pandora configuration settings must be configured manually in the pianobar application configuration file. In order to change the account information or the default pandora station, open ~/.config/pianobar/config in a text editor and configure the settings to your desired preferences. For instance, to specify your pandora account information insert your username and password in the User variables as shown:

```
#User
user = enter_username_here
password = enter_password_here
```

To configure which pandora station plays on start, enter the station’s ID number into the autostart station variable, for example:

```
autostart_station = 3557476821133827106
```

After you have made any and all desired changes, save the config file and restart the system.

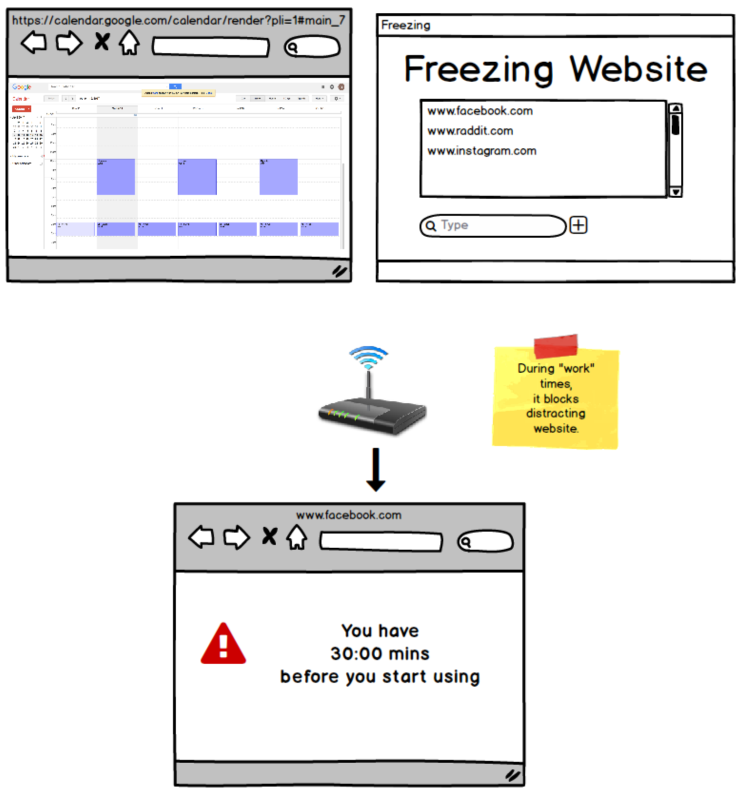
Low-fidelity prototypes

Idea 1:

i. Description:

This tool is a “smart router” that, when turned on, automatically syncs with the user’s calendar and recognizes time slots designated as “work time”. The user can also populate a list of websites, that the router will block during the designated “work time”.

ii. Wireframe:


https://calendar.google.com/calendar/render?pli=1#main_7
Freezing
Freezing Website
www.facebook.com
www.raddit.com
www.instagram.com
Type
During "work" times, it blocks distracting website.
www.facebook.com
You have 30:00 mins before you start using

iii. Storyboard:

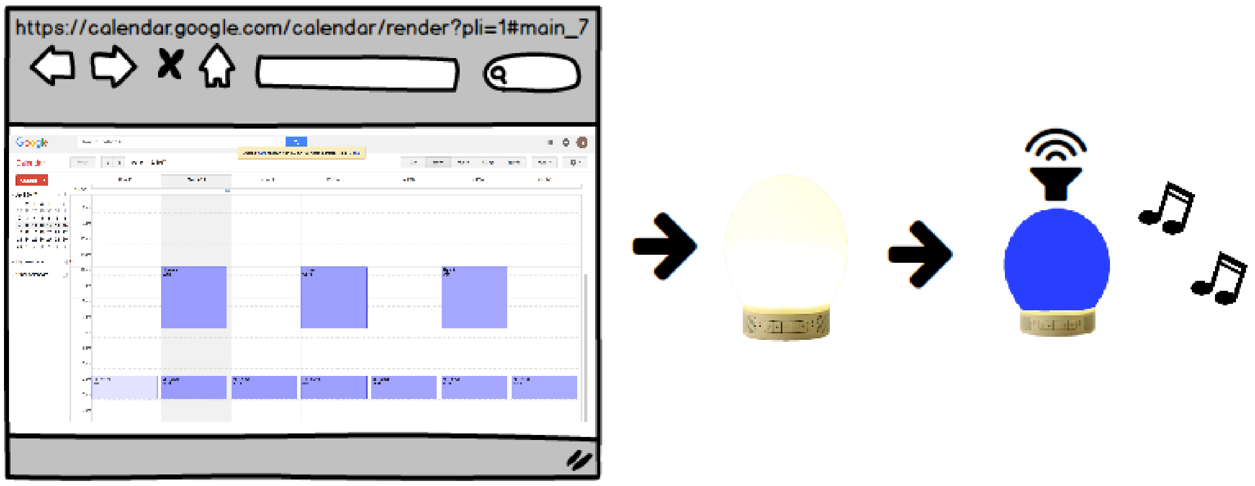
a. User turns on the portable smart router at workplace (home, office, lab etc)
b. Smart router syncs with user's calendar
c. If time slot is designated as "work time", pre-defined websites will be blocked from all digital devices connected to the smart router

Idea 2:

Description:

This tool is also a different version of the "smart router" that automatically syncs with the user's calendar during time slots designated as "work-time". Unlike the first solution, this tool will sync with user's study/work playlist and speakers and automatically start playing music to create conducive environment for working.

Wireframe:



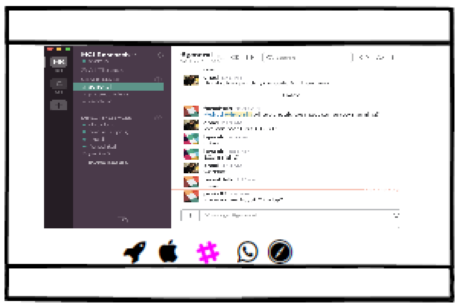
iii. Storyboard:

a. User turns on the portable smart router at workplace (home, office, lab etc)
b. Smart router syncs with user's calendar and speaker will light up to indicate the user about work-time
c. If time slot is designated as "work time", study music will start playing

Idea 3:

This tool aims to utilize the concept of "positive peer pressure". It is a smart light that, when turned on, will notify a group of pre-defined study buddies that the user is now working. It will also automatically open the group's slack channel to encourage collaborative work.

Wireframe:



iii. Storyboard:

a. User turns on the smart light to notify group about his/her working process
b. Group members, upon being notified, will be encouraged to work on the “group project” or any other group task
c. User is directly taken to the slack channel for the group to facilitate conversations

Sprint Feedback:

Idea 1:

Feedback: How is the tool different from currently available time-management software?
Review: In addition to blocking distracting website, the tool:

a. Will work on all connected digital devices, thereby avoiding distractions from multiple sources
b. Syncs with google calendar and ical, thereby removing need to set up schedule separately for the tool

Idea 2:

The idea of syncing the smart router with speakers during work-time was well-liked by viewers of the sprint feedback. We decided to maintain this feature in our prototype.

Idea 3:

The overall concept of the positive peer-pressure was appreciated but the obvious downside of notifying other team members- creating digital distractions in the form of alerts and notifications, led us to drop this feature from the prototype.

User Feedback

We presented the three wireframes with two users:

User 1:

The user liked the idea of being able to sync to the calendar and block distracting websites. The user prefers working at different spots at the campus, therefore wasn’t comfortable with the idea of carrying an additional device. She prefers listening to music while working, and found the automatic sync feature to be really useful as it would minimize her interaction with other apps during work-time. The user also prefers to work on her own schedule, she did not prefer to have any features related to the positive peer pressure.

User 2:

The user prefers to work at a pre-designated spot (home office, lab desk). Therefore, he liked the idea of having a tool that could be used at the desk. Because he works at the lab quite frequently, he also really liked the “lighting when working” feature as it will enable his co-workers to see that he is working and minimize external distractions. The user is focusing on his own research and doesn’t see much use for the positive peer pressure aspect of the tool.

Team Review:

Based on the feedback, we decided to make the device as small and sturdy as possible to make it easily portable. We also recognized that the tool would be most helpful for people who tend to map out their activities in calendars but aren’t able to accomplish the task on-time due to distractions. We decided to retain the sync with calendar feature as it reduced the user’s task load was valued both by the sprint activity viewers and our target users. Furthermore, the device seemed more apt for people who have pre-designated working spaces (home, office, lab) where the device can be used to create a conducive working environment (light to indicate working time and automatic sync with study music) without causing too much distraction to people nearby. We decided to drop the positive peer pressure feature as it seemed counterproductive to our goal of reducing digital distractions.