Internet of Things Information Display with Geofencing

Project proposal

Benjamin Daszkiewicz
Jacob Nading

Advised by Aleksander Malinowski

Abstract

This document provides an in-depth description and solution proposal for the IoT Information management Display (formerly IoT Smart Calendar) project advised by Dr. Aleksander Malinowski of the Electrical and Computer Engineering department at the Caterpillar College of Engineering and Technology at Bradley University. It also offers a look at prior work, a list of materials, a tentative schedule, division of labor, and a look at future development of the Display. The project is now in its second year of development. Benjamin Daszkiewicz and Jacob Nading are continuing the work started by Jason Morris and Cole Linderman in the Fall of 2016.

Contents

Introduction

Description

A common problem arises in modern society in which ease of access has led to a tendency to put things off that are not readily accessible. It is for that purpose that we intend to expand upon the IoT Calendar Display begun last year by Mr. Jason Morris and Mr. Cole Lindeman. Their project brought ease of access to advising instructor Dr. Aleksander Malinowski and students trying to stay in touch via an Internet of Things display. This year, Dr. Mohammad Imtiaz expressed interest in having a display outside of his office as well. As opposed to simply installing a clone of the existing IoT Calendar, we are looking to create a new and improved version of the software that can be ported to the existing hardware outside of Dr. Malinowski's office as well new hardware installed near Dr. Imtiaz's office.

The primary functionality of the display revolves around a Google calendar display pulled from Dr. Malinowski's Google account. It is a place where students can easily check up on his availability and office hours. In addition, the original project design featured a messaging platform with which students could send SMS and e-mail messages to Dr. Malinowski via the device. The professor was also able to upload a series of static images to the program that would display advertisements for upcoming courses.

Our goal is to take the original concept of the IoT Calendar and expand it into an IoT Information Display. The core functionality will remain, and some code will be salvaged and recycled from the original project's GitHub tree. However, the vision for this project is to take the functional and make it run more smoothly, more streamlined, and implement additional features such as GPS geofencing to accent a paging feature that will result in minimal extraneous messages to the professor. We also intend to include a message feed system powered by Twitter's API that will allow the professor to post instant updates and messages that are not convenient to deliver via the calendar.

Scope

The scope of the project, as it stands, is in a set of minimum functionality requirements laid out by Dr. Malinowski. Completion of these requirements by the beginning of May 2018 is the objective of the project and incremental goals will be laid out in subsequent sections of this document. In addition to the minimum functionality requirements, there is a set of features that may be implemented by future groups, or may be implemented should required functionality be completed prior to the May deadline.

Functionality Requirements

The finished IoT information display will include the following:

- Daily calendar data
    - Display will show professor's schedule, synchronized automatically from their Google account
- Short memos or announcements including an urgent announcement feature
    - Professor will be able to easily publish announcements to the display remotely from their personal device
- Advertisements for the courses taught be the calendar owner
    - Display will contain automatic timeout feature that displays relevant course advertisements when its other functions are not in use
    - Device will wake from advertisement timeout when engaged by a user
    - Advertisements will also be available on demand from the home screen

- Paging function based on geofencing
    - A server will receive GPS data from the professor's personal device (cell phone) to detect their location and cross-reference it with geofence points as well as the calendar
    - Paging button will send message to professor's cell phone via SMS if they are within the building and available for meetings or office hours
- Current and forecast weather data
    - Display users will be able to access current and future local weather data and radar maps

Additional Functionality

While it is outside of the scope of the current project, it is worth mentioning that other ideas for the IoT Display have been discussed and requested by Dr. Malinowski and Dr. Imtiaz. Future direction for the project is discussed in detail.

Review of Literature and Prior Work

From the prior project development team, the main focus of the IoT Information Management Display is mostly the same as what it is now. Currently, the subsystems are extended to have more applications as well as including more program application aspects. Based on the previous iteration of the development of the project, the focus entailed a wall-mounted smart calendar, interface with sensors, and communication with the internet. The current iterations include displaying advertisements and calendar information to passing by people, allowing users to leave messages, and access to a professor's announcements. The focus of sensor interface was left somewhat unexpanded on the current iteration of the project. The advancement of this goal previously didn't get as developed and stable as originally intended. Therefore, a few of these features are disabled to allow the better functionality of the device. Communication with the Internet is essential to how it works and is a currently well-developed process in the iteration today.

The past development team also did a bit of research on two devices that were similar to the aspects they were looking for in designing the project. They more so felt like an even greater idea for the more updated and advanced project ideas and thoughts that are ongoing today. A device, the DAKboard, is very similar to the eye-appealing aspect that the current project wants to incorporate. It consists of a customizable wall display that can also show pictures, calendar events, and weather. Some of these concepts are very similar to what is wanted today; a customizable interface where a user can decide what they want to be synchronized and displayed in an easy-to-read format. Additionally, in the DAKboard design, everything is done through a web interface, which is the direct design of the IoT Information Management Display currently [1]. There is also another previously researched display, the Raspberry Pi Framed Informational Display, that specifically uses a Raspberry Pi for the whole interface [2]. It displays a Google calendar along with local weather data, which is already incorporated in the current iteration of the project. This display, with a very similar process to the previous completed IoT Calendar, has a bigger reinforcement on displaying everything in a better layout as well as making it look more appealing and organic. There have been multiple similar functionalities and devices designed with very similar ideas in mind. Another example of a design that was an interesting concept was an informational display within a mirror [3]. Because of the excessive, consistent use of a mirror, information can constantly be reminded to a user, making remember certain tasks exponentially easier.

Applicable Standards and Patents

Standards to Consider

Many standards to consider are already met because the project uses entirely COTS (commercial off-the-shelf) parts. Standards that would otherwise have to be considered an internet connection protocols for connection of the IoT.

Patents for Similar Ideas

There are a few ideas and patents for similar ideas. These ideas, with a few much closely related to the IoT Display than others, are relatively related, as the concept is pretty general. Having a device display information is too broad an idea to have any specific patent for it. There are a few more in-depth ideas that go into much more detail about their device and its usage.

- A display device and content display system
  - https://patents.google.com/patent/WO2016061626A1/en?q=smart&q=touchscreen&q=information&q=display&q=raspberry&q=pi
- Raspberry Pi based smart device control apparatus and control method
  - https://patents.google.com/patent/CN106789459A/en?q=smart&q=touchscreen&q=information&q=display&q=raspberry&q=pi
- Smart interactive billboard device
  - https://patents.google.com/patent/US20050021393A1/en?q=smart&q=device&q=display

Subsystem Level Function Requirements

Applied Technology and Solutions

Hardware

Figure 1 shows a connection diagram of the very simple hardware connections necessary for the project. The Raspberry Pi and display will pass data back and forth via HDMI (to display) and USB for touch data (to Raspberry Pi).

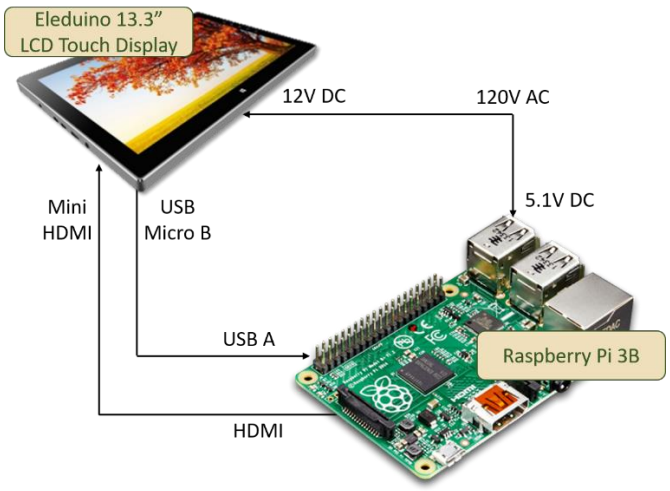

Figure 1: Hardware configuration diagram

Software

In brief, the technologies, languages, and modules that we foresee using are:

- Python scripting language
    - wxPython graphics module
    - Python Time library
    - Python HTML library
- Google Calendar API
- Twitter API
- App for iPhone to send GPS data to a publicly-hosted web server for geofencing
- SMS server

The GUI

The GUI will be the central component of the project. One of the main focuses is to convert the Display software, which is currently HTML and PHP scripts that run in an instance of Chromium, to a graphically rich application written in Python using the wxPython graphics module.

The module is superior for our purposes to Python's native TkInter GUI module due to its flexibility and better handling of image alpha values and .png images. These types of tools are essential for creating an attractive and modern interface.

As detailed in the software development lifecycle (SDLC) diagrams, particularly Figure 3, most features of the app will be visible and accessible from the home screen. This contrasts the present design, where users must scroll through pages or navigate a menu to see content. Not all calendar, announcement, or advertisements data will fit on the home screen, however. Abbreviations can be shown, and the full data is accessible with a touch.

Calendar

The calendar was the heart of the original design as indicated by the title of last year project. We look to make some improvements from last year's calendar. Instead of directly displaying the calendar on a page in a web browser, as is the case presently, we will make use of Google Calendar's rich Python API to pull the data itself from the calendar which can then be embedded in the GUI however we choose.

One particular feature that Dr. Malinowski is looking for in the calendar is the ability to use the color coding feature available on Google Calendar's native web platform. For instance, when viewing calendars there, events can be color-coded by each course a professor may teach. When importing the calendar into a custom browser layout as is done presently, the events display in one uniform color and do not keep their color coding. Whether we can pull the event color data from the API or whether we have to color code events manually from event title data, we look to implement a more colorful and useful calendar.

Announcements

In today's world, one of the easiest ways to update your status is through social media. Most people are already accustomed to using such technology and do so anyways. Harnessing the tools given to us by Twitter, we can create a system that allows the professor to post messages and updates quickly and easily that will not only be visible at the IoT Display but will also be accessible through students' personal devices.

Setting up a Twitter account for the professor to use for their classes will keep it separate from their personal account. Then, they can post messages that can be updated to the display periodically using Twitter's feed-pulling functions in the API. Using a tag or keyword system, the messages can be sorted by course so that students do not have to sort through irrelevant data to find announcements relevant to them. If applicable, they can also follow the individual account on their own Twitter accounts to receive the messages to their phones.

Advertisements

Special technology need not be applied to the advertisements. The advertisements will remain as they are now. They are presently static images that can be uploaded to the Raspberry Pi file system and displayed from there by the Python (presently browser-based) program.

Paging with Geofencing

The paging feature will allow students to press a button in the Display application that will alert the professor that they have been requested at their office. The purpose of geofencing and scheduling checks in the paging feature is to ensure that students cannot disrupt the professors in meetings, teaching lectures, or on their personal time by abuse of the paging button.

It will be necessary to take GPS data from the professor's phone in order for this functionality to work. Due to the difficulty in developing small, non-commercial applications for iOS, we will be using an already available application that will pull the data from the phone and host it on a server which we will then be able to access via an API or other such technology. The application must be available for both Android and iOS so as not to limit the Display's flexibility.

The raw GPS coordinates will then be used by the Python application to determine if the professor is in the building. See the geofencing reference under "Additional References" for more information on geofencing. Once the app has determined if the professor is within the geofence, it will check the time and professor's schedule to determine if they are available. Only then will it enable the paging button, which can be pressed to send an SMS message to the professor's phone, informing them that they are requested at their office.

Weather Data

The weather data functionality of the calendar will be kept and improved upon. Modern devices often show the weather at a glance from the home screen, and we will implement similar convenience, integrating the weather data in an interactive way.

Presently, data is being pulled from the National Weather Service via a modifiable URL that will return different data depending on how it is modified. The information is accurate and reliable, and this method will be kept to meet the minimum requirements of the project. However, if given time, we may seek out a cost-free API that will give data in an easier to handle way. Radar images and maps can be displayed via links as well, and we will use the standard Python HTML library to fetch the images and display them.

The changes being implemented to the weather portion of the project are on the home screen. The center of the home screen will be a large icon that represents current weather conditions at that time.

The page background will also be representative. As the day/night cycle changes and the weather changes, the icon will change. For instance, once the sun is set, the “clear weather” icon will change from Sun to Moon and the background from a clear blue sky to a clear starry one. If it begins raining midday, a “partly cloudy” icon and puffy cloud background will change to rain clouds. Current, high, and low temperature data will also be displayed and clicking on the large weather icon is how more detailed weather information and maps will be accessed.

Software Development Life Cycle Diagrams

Data Flow

The Context Data Flow Diagram shows on a high level how data and what kind of data move between modules with the GUI home page main loop being centric to the model. Here, the functions that move data are named and tasks can be broken down by module. A Level 0 Data Flow Diagram would break down each module into submodules and display data stores along with the functions that pass data around the sub-modules.

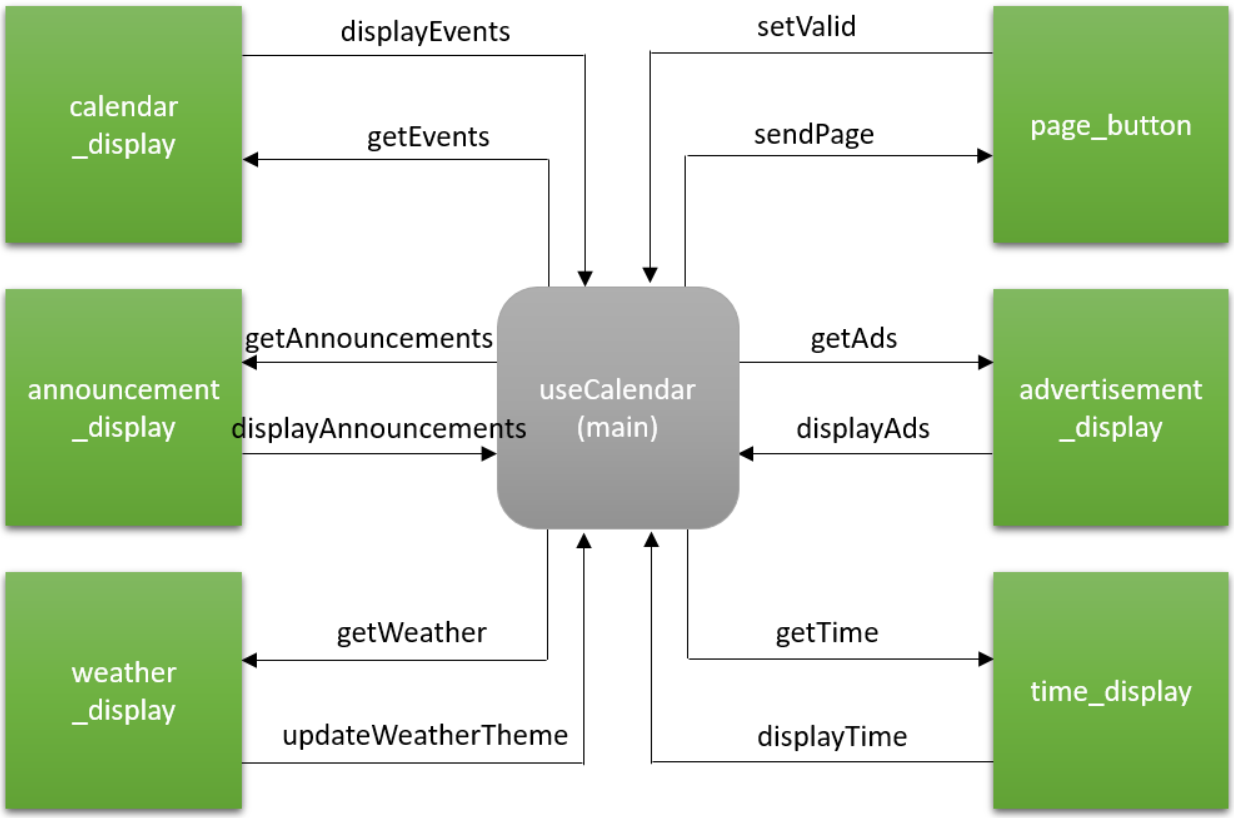

Figure 2: Context DFD for Software Modules

State Transition

Figure 3 shows a high-level state transition diagram (STD) of the software focusing on the minimum functionality requirements. It details how users will navigate from place to place in the software and gives an idea of what features will be accessible from certain areas of the software. The simple STD indicates successfully streamlined software that is user-friendly to navigate.

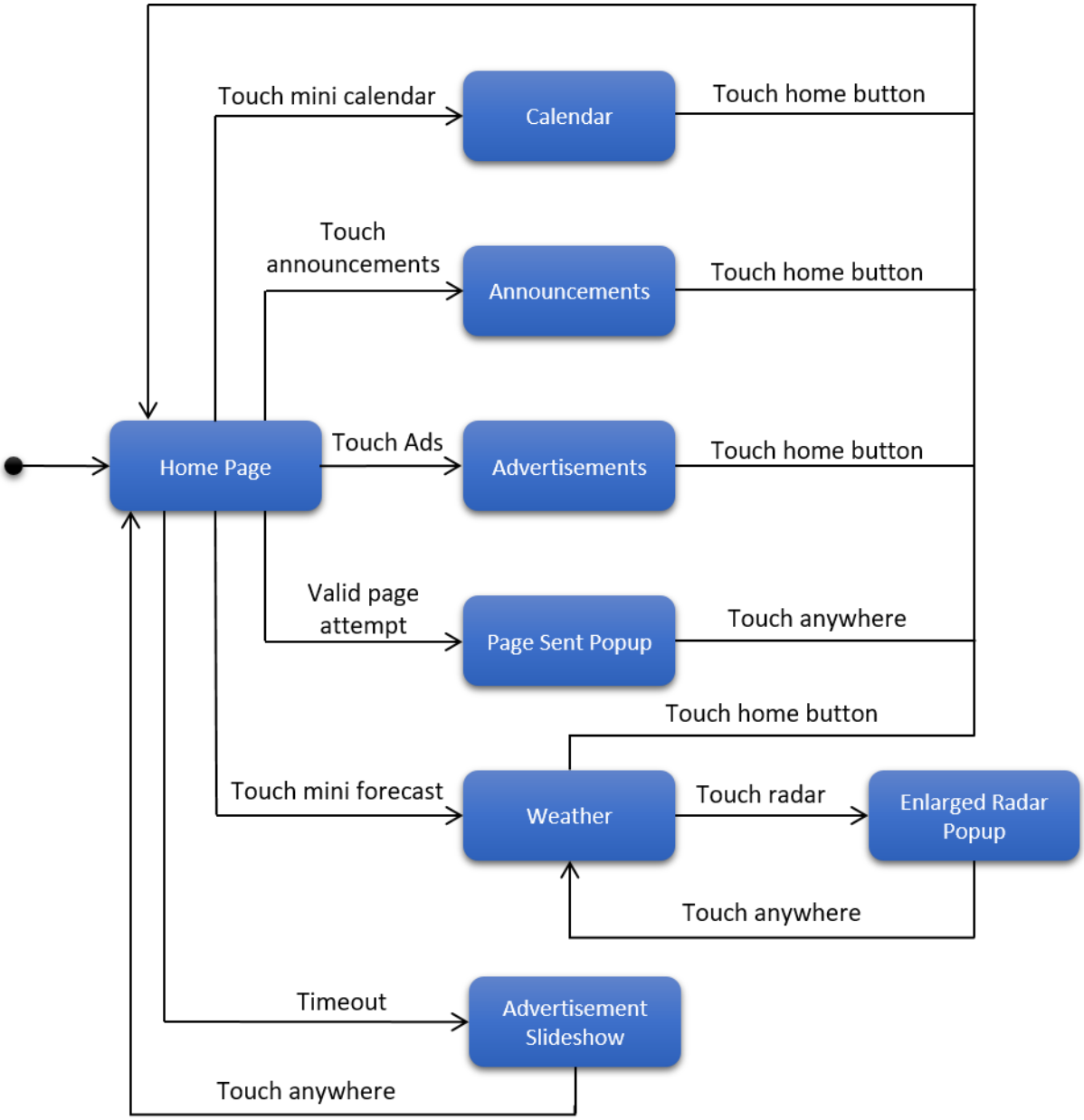

Figure 3: High-level state transition diagram for Display GUI

GUI Hierarchical Design Model

The GUI Hierarchical Design Model breaks down the visible objects on the main page of the Display. Each instance of each element is listed under the general heading. Underneath, attributes of each general element are listed. This diagram helps greatly in the creation of class objects for the GUI environment. Some of these elements exist natively in the wxPython graphics library such as Image, Text, and Button. However, not all native objects are visually customizable. For instance, the Button object has limited customizability on its appearance. In that case, a new object will have to be created that allows its image to be modified. These classes can be inherited from the Image object, which has built-in ability to handle custom object masks and portable network graphic (PNG) alpha values.

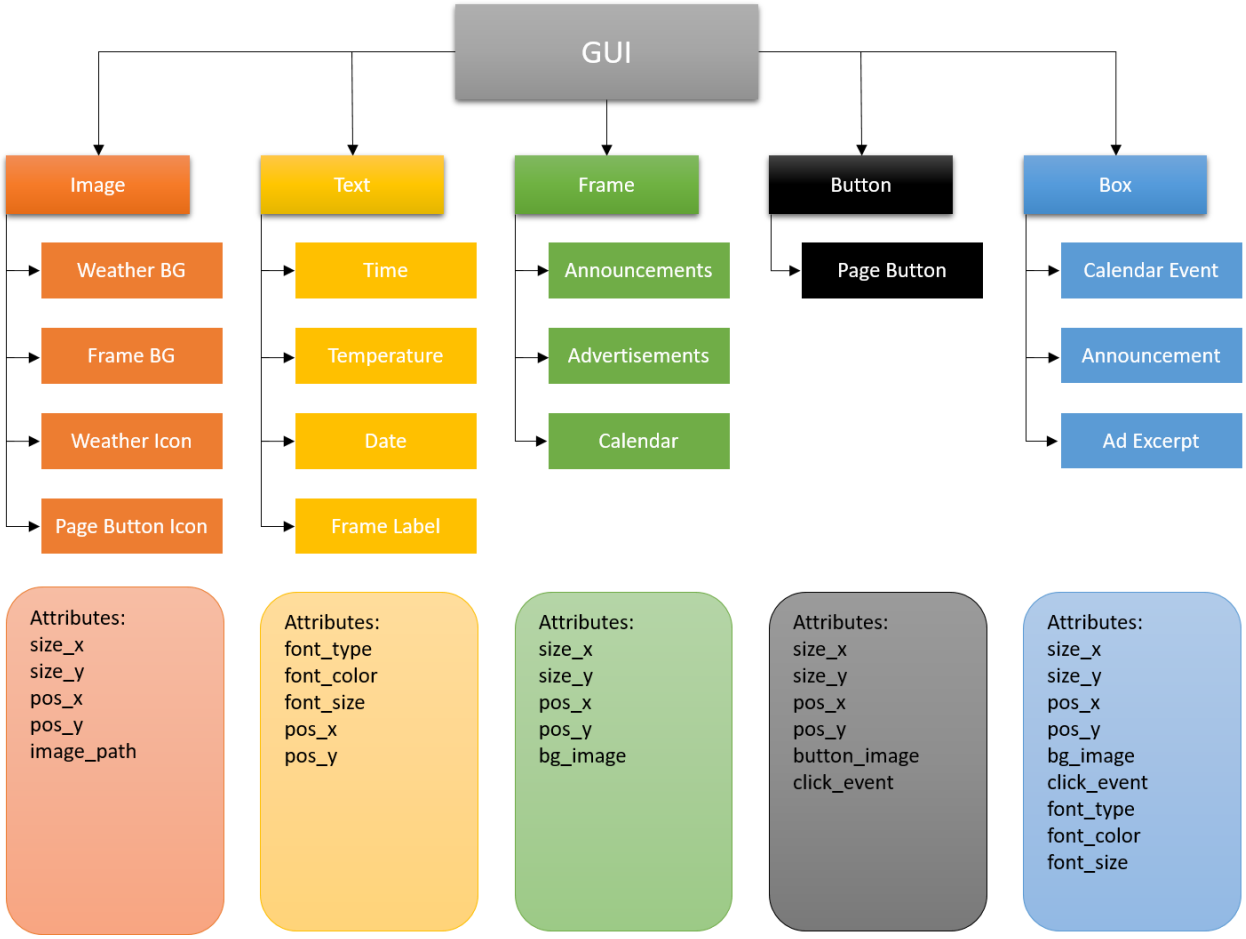

Figure 4: Hierarchical design model for GUI home page

Engineering Efforts Completed to Date

The engineering efforts to accomplish the project started with the initial designing process. This included theorizing new ideas and additions to add to the current calendar, along with improving its functionality. A few sketches were drawn up to help improve the visual appeal as well as the readability. It was ultimately decided to use Microsoft OneDrive and Google Drive to help share and document any data or information so it can easily be edited. Both GitHub and Trello are being used to help manage the structure and designing process. A request for a side project presented itself, which required a bit of work that would later build itself into the IoT Informational Management Display. Furthermore, the

construction of an HTML format to display professors' names and pictures was the goal of the side project. There would be a monitor or screen mounted on a wall, and that screen would display the current electrical engineering staff. The screen would allow for touch capabilities, so if a student wanted to find more information on a professor, they could simply touch the professor's face, and that would pull up more information and his or her schedule. This is further helpful in aid of new students or visitors who want to see a professor but do not know where to look or how to contact them.

Throughout the first half of the project design, a lot of conceptualization, like the specifics of how the device will function, was being theorized. The problem description was put together to try to describe the conditions for the Display accurately. Next, the parts and deliverables were focused on. The parts were a debacle, specifically, the screen size and which model to purchase. When it comes to touchscreen devices, it is hard to find the specific set that was needed. Touchscreen displays are generally sold as additional computer monitors or as replacement parts for handled devices. This would entail a computer stand, and a more sizable screen was necessary. When it comes to smaller displays, there is a great selection of smaller screens for interfacing with microcontrollers, that are too small for our purpose. Ideally, the display would need to be bigger than a phone screen and smaller than a desktop monitor. Finding an ideal display with a good review or the other specifications needed was a challenge.

Parts List

The parts for the IoT Information display are all consumer off-the-shelf (COTS). Hardware subsystems come preassembled and only require interconnecting cables as peripheral parts. The parts are presented in Table 1.

Table 1: Parts list

Qty	Item Description	Source	Price/Unit	Price
1	Eleduino 13.3" 1080P IPS Capacitive Touch Display (sky black)	Amazon.com	$189.00	$189.00
2	CanaKit Raspberry Pi 3 Kit	Amazon.com	$49.99	$99.98
2	SanDisk Ultra 8GB Class 10 UHS-I MicroSDHC	Amazon.com	$9.99	$19.98
1	KabelDirekt (15 feet) Mini HDMI to HDMI Cable	Amazon.com	$9.99	$9.99
1	Micro USB Cable, 3 Pack 10 ft Braided High Speed USB 2.0 A Male to Micro B	Amazon.com	$10.99	$10.99
			Subtotal:	$329.94

Firstly, multiple Raspberry Pis will be necessary for the software development process. As soon as a working prototype is established it will be displayed using one Pi while the other Pi is reserved for continued code development. Updates can then be pushed periodically to the Pi that is running the monitor.

Many of the COTS components of the project are sold as kits thus eliminating the need to buy power supply cables, cases, or adapters. The Elduino display comes with necessary power adapter and cable, although the cable may need to be extended pending where the display will ultimately be installed. The Pis will be housed in a case included in the CanaKit, which also contains the power supply for the device.

Two MicroSD cards are necessary, one for each Raspberry Pi. UHS-I cards are capable of 10Mbps data transfer speeds. While the Pi effectively tops out around 22Mbps for reads from the SD card slot, the display application does not require particularly fast data-fetching or writing , and therefore, the Class 1 SD card will be sufficient. While it is undetermined how large the final application file will be, the mounted Raspbian Stretch image

The screen and Pi must be linked together to display and response to touch. The display will be passed with the HDMI to Micro HDMI cable, with the Micro HDMI needed on the monitor-end. Touch data will be returned from the display screen which is done via a micro USB to USB A cable, with the monitor needing the micro USB connector.

Deliverables and Schedule

Deliverables to Date

At this point in the project, the primary deliverable is the Problem Statement and Functional Description Document submitted October 17, 2017. Parts request submissions, while not a graded deliverable, are to be ordered with enough time such that parts will arrive before the Spring 2018 semester at the latest. We will begin work on the proposal presentation once the document draft is submitted. The proposal presentation is November 30 and includes the document and presentation. Progress has also begun on the project website as well, which is due to be prepared by December 7.

Schedule and Future Deadlines

Table 2 lays out the rest of the Fall 2017 semester and the Spring 2018 semester with highlights on deliverable due dates, as well as team goals for meeting functionality requirements.
Note: Deliverable dates are subject to change by prerogative of ECE Department

Table 2: Project schedule of deadlines

Date	Item Due/Requirement Met
Fall 2017	
11/16/17	Proposal presentation draft
11/30/17	Project proposal and presentation
12/7/17	Website with proposal presentation and report
	Non-functional, rough layout prototype for display written in Python
	Majority of graphical project aspects created
Spring 2018	
2/15/18	Midpoint progress report
2/16/18	Working calendar and announcements prototype (API work) with home screen and functional weather icon

2/23/18	Added advertisements and display mounted
3/9/18	Student Expo registration deadline
3/23/18	Weather/radar screen (final minimum requirements implemented)
3/29/18	Final report draft
3/30/18	Student Expo abstract
4/5/2018	Project poster
4/10/2018	Student Expo poster setup
4/12/2018	Expo poster judging
4/13/2018	Award ceremony
4/27/2018	IAB poster session
4/28/2018	Project conference
5/1/2018	All deliverables completed and uploaded to website

Division of Labor

Much of the labor will be a collaborative effort between both team members. However, each team member will have particular portions of the project that they are responsible for, meaning they will make sure that deadlines are met for that portion of the project. Here, particular portions of the project are assigned to the two group members.

Nading

Due to his experience in Photoshop, Nading will handle graphic design challenges. As an engineering project, this is a unique one in that it will require the design of many graphic components before then can be laid out in the GUI. Open license stock images may also be used for the project and will be cited in a "Credits" section of the display.

In addition to providing graphics to integrate into the interface, Nading will also work on modules that will request and process data via API for announcements and geofencing data for the paging sections of the Display. The modules will process the information and pass it to Daszkiewicz's GUI modules for display.

Daszkiewicz

Coding the GUI will be Daszkiewicz's primary responsibility. Because the graphical element of the Display is a large part of the project, it is important to have one team member with a solid understanding of graphics programming in Python. Neither team member came in with a lot of experience in this area. Therefore it would be most effective for a single team member to become proficient in this area.

Daszkiewicz will also integrate Google API data into the calendar section as well as the weather data. The GUI programming will lend itself as the most efficient anchor to bring all of the separate Display software modules together. The separate modules will request and process necessary GPS and text data. They will then be bound together as objects and returned from those functions. Designing how these modules interact with each other will also fall under Daszkiewicz's responsibility due to his experience with software design and engineering.

Discussion and Future Direction

In an ever-growing technological world, with so much focus on media, news and social events that are pushed onto devices and displays for easily readable information, it only logically follows that new systems and devices are put into place, especially in a learning environment with students' idealistic futures in mind. Ideally, the IoT Display is something that can be expanded upon and used in the future heavily. The discussion of the device and its potential was only briefly theorized. The device was desired to have an art appeal with a simplistic modern design in mind. With the baseline creation and consistency of such a thing, the usefulness grows as more ideas and process could be implemented for the device. It is only being developed for a few electrical engineering professors in the department, but it could, however, lead itself to finding a perfect place in the job environment to help coworkers. A manager in a company could have such a device in front of their office to help update any co-workers on information and scheduling that may be pertinent to them.

Conclusion

The IoT Information Display solves the problem of schedulues becoming outdated before their events come to pass. Synchronizing these ever changing schedules with students can cause professors undue frustration. The Display lends a place where the professor's schedule will always be up to date and will synchronize in real time. If off campus, the Google Calendar will still be viewable by traditional means. It is the best of both worlds. Twitter announcement functionality will allow for real-time announcements to be displayed in such a way that they can also be viewed from a student's personal device while they may not be around the display.

Minimal hardware is required, in that only a Raspberry Pi is needed to run the touch display. Some minimal connection is needed between the two, but most of the project will be software based. The system will run as a Python application to make data handling as efficient as possible and avoid the need for hosting a web server or using Bash scripting to pass variable between various php and HTML destinations.

While minimum functionality requirements of a calendar, announcements, paging, weather, and geofencing have been defined for the timeline that our team will work on the project, future goals have also been defined so that the Display can grow to be more useful in the future. Sensors for an increased user experience or text messaging may be added in years to come to continue to expand the IoT Information Display.

References

[1] M. Archambault, "DAKboard is a Customizable Wall Display for Photos, Calendar Events, and Weather," PetaPixel, 19 August 2015. [Online]. Available: https://petapixel.com.

[2] kmccb, "Raspberry Pi Framed Informational Display - Google Calendar, Weather, and More..," 7 April 2016. [Online]. Available: https://imgur.com/gallery/z94Vr.

[3] B. Eagan, "Smart Mirror (with Optional Alexa)," hackster.io, 18 Aprit 2017. [Online]. Available: https://www.hackster.io.

Additional Readings

Creating and Monitoring Geofences
<https://developer.android.com/training/location/geofencing.html>

Geofencing Tutorial with Core Location
<https://www.raywenderlich.com/136165/core-location-geofencing-tutorial>

A Simple Geo Fencing Using Polygon Method
<https://www.codeproject.com/Articles/62482/A-Simple-Geo-Fencing-Using-Polygon-Method>