Team Leader: Jacob Alspaw
Arbaz Ahmed, Kareem Taleb, Benjamin Young, Tao Li

"Smart" Alarm Clock via Raspberry Pi

Project Description

How often do you find yourself waking up in the morning and repeatedly hitting the snooze button only to be disappointed that you are starting your day much later than originally planned? Individuals in our software development group often fall victim to such a situation, but have trouble managing a routine that gets them out of bed on time. Our group wants to solve the problem of repeatedly hitting the snooze button, but not by removing the snooze button at all. Instead, for our Software Engineering project, our group proposes the development of a smart alarm clock that aims to improve upon the functionality of traditional alarm clocks by integrating new and useful technologies to motivate its users.

The new alarm clock, whose name for the time being is PiClock, is going to increase user productivity by helping them overcome their early morning exhaustion and by providing general information about their upcoming day. The PiClock device will sit nicely on a user's desk or nightstand. The graphical user interface will prominently display the current time and weather info, while user enabled features individually display information on the rest of the screen. There are several key features that this alarm clock will need to offer if we want to make it a success.

Major Features

7" Touchscreen Display - The official touch screen display for the Raspberry Pi measures 7 inches diagonally and makes control of PiClock simple. Because we consider the general use case of the alarm clock to be used through touch as opposed to a mouse, the PiClock graphical user interface will inherit a mobile-minded design.

Piezo Buzzer - A single small form factor device that emits sound at varying frequencies. The buzzer will be used as the device that is responsible for waking the user. It will interface with the Raspberry Pi's GPIO pins and be controlled by the application using the WiringPi library.

Alarm Widget - Arguably the most important feature of the application, the alarm widget sets apart PiClock from any other alarm clock. The alarm widget is persistent in waking up the user, such that a user will need to complete a task before the alarm will stop sounding. Once the task is complete, users can either choose to start their day or continue sleeping by hitting a snooze button. Tasks will include mental challenges that are designed to increase brain activity immediately when users wake, so they will make more informed decisions about returning to sleep. We suspect the use of continuous negative stimuli will motivate the user to start their day.

Clock Widget - The clock widget will be responsible for displaying the current time of day to the user. It will have a dedicated section of screen reality, so the time will always be displayed. Users will also have the option to extend the feature's functionality to periodically display times in cities around the world.

Yahoo Weather Widget - The weather widget will display the area's current weather in a dedicated section of the screen with the help of the Yahoo Weather API. There is possible support for displaying the radar too, but further research is required. Users will also have the option to extend the feature's functionality to periodically display the three day forecast. All weather data will be displayed with the help of personalized weather graphics.

Google Calendar Widget - The calendar widget will display the events upcoming in a user's Google calendar as a table view. Today's scheduled events will be prioritized first and then any remaining screen real-estate will be dedicated to other upcoming events within the next week. Users will have the option to extend the feature's functionality by calculating and displaying the number of days until the next major nationally recognized holiday.

Google News Widget - The news widget will be responsible for querying for today's current events. A selection of news article headlines will be displayed and if possible will include a short synopsis of each article.

Application Settings Controls - Each feature will have its own set of controls to manage the settings to designate how a feature will work and what information the feature will display. For example, the alarm widget allows multiple types of games that need to be completed for the alarm to shut off. Users can select between winning tic-tac-toe against a low-level artificial intelligence, completing a memory game, correctly answering trivia questions, etc. Settings will be remembered every time the application is initialized.

Other misc. features if time permits - The stretch features that will be developed after we have a working prototype will include an inspirational quote widget that queries for randomized quotes to encourage early morning motivation, a system usage statistics widget that displays current information about the user's Raspberry Pi under load, a photo frame widget that displays randomly selected photos, and an automatically adjusting application theme that will reduce strain on the eyes during certain times of day.

Implementation

For the purpose of project management software, our software development team has decided to use Trello for project collaboration and Slack for team communication. The two applications are free, very simple to use, and have support for integrating one application into the other. We will be using a shared Git repository hosted by GitHub for version control because GitHub also has support for both Trello and Slack. Our repository will be setup as a pull-request model where each commit will need to be approved by our project leader before the contribution is added to the codebase. The project leader and team members alike are responsible for submitting clean and efficient code for the review process. Code that is deemed substandard will need to go through a refactor process managed by the submitting member before it can be reviewed again.

The PiClock will be built on a Raspberry Pi 3 using a standard Piezo buzzer and the official Raspberry Pi touchscreen. The Raspberry Pi will run a flavor of Linux called Raspbian Jessie, the version of which should offer long term support. Standard C++ libraries will be used to complete the project along with WiringPi so the application can easily interface with the GPIO pins on the Raspberry Pi board. The GPIO pins are used to send a sine wave to the Piezo buzzer that will act as the alarm sound. The Qt framework will be used to develop the graphical user interface. Specifically, the Qt framework has amazing support for using animation and creating a graphical user interface through a large collection of libraries. The supported Qt IDE called Qt Creator also supports cross-platform compilation between desktop versions of Linux and the Raspberry Pi. However, an agreed upon IDE will not be important at the beginning of the development process because we will have no need for project files while working on isolated features. Nearly each of these isolated features will require the use of a source API for querying purposes; we anticipate querying for weather data, news article metadata, scheduled events on a Google Calendar, trivia questions, and motivational quotes.

Discussion

We believe this project is tough enough for the software engineering class because it will require a deep understanding of operating systems to control the Raspberry Pi and interface it with external hardware (controlling the touchscreen and the Piezo buzzer). The application will require a great deal of concurrent programming because each widget will need to constantly update itself through coordination with third party services and RESTful API calls. These calls will be asynchronous and will need to happen often, so having all widgets run on a single process would significantly decrease the application's performance. There is also a possible need for a database depending on the extent of which we allow users to manage application settings and where we choose to store and retrieve other data like information about national holidays. If we do choose

using a database for our project, it is likely we use SQLite for its easy integration with C++ applications. If we do not use a database for our project, we will likely just use an initialization file formatted as JSON to store application settings.

Our team leader and project founder, Jacob Alspaw, is assuming that each member will need to contribute between 1,500 and 2,000 source lines of code, not considering any testing files, before the project reaches completion. At the beginning, each member will be assigned an isolated feature to develop. Once all of our features are nearing completion, a set of team members will be assigned to work on the user interface. Once the user interface becomes a working horizontal prototype, these isolated features will be added into an overall incremental prototype. From here, we will begin dedicating ourselves to the user interface to maximize the user experience.