SCR RESEARCH DOCUMENTATION

A Handbook for Research Assistants, Working for Professor Kar on the LESA Center Project

Handbook Version 1.0

March 31, 2020

Contents

Overview

Introduction

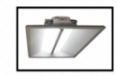
This handbook serves as both documentation and a guide for students working with Professor Kar on the Lighting Enabled Systems & Applications (LESA) Engineering Research Center Smart Conference Room Project. Everything discussed strictly covers my work with Professor Kar and LESA from January 2019 to May 2020, with the objective of passing it down to the next students working on the project. Information regarding any extension of my work, that was not something I directly worked on, can be obtained using the contact information, at the end of this handbook, under *References*.

LESA and the SCR

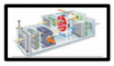
Funded primarily by the National Science Foundation, the LESA Center is dedicated to developing autonomous intelligent systems to address modern challenges in the connected environment. LESA's main research areas are communication, efficient buildings, healthcare, plant science, and cognition. My projects mostly involved the efficient buildings area, in particular, the area of autonomous HVAC and lighting control for smart buildings. My roles mostly involved integrating occupant localization, optimizing system controls, and implementing and evaluating an integrated system for collecting user preferences.

The Smart Conference Room (SCR) serves as a testbed for the research. It is located on the 7th floor of CII at RPI, in room CII 7003. On the following page are a collection of diagrams, that outline the makeup of the SCR, in terms of the key components, their spatial distribution, and the networking backbone of the room.

Key Components



8-color Telelumen LED fixtures (x 10)



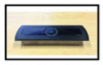
Air Handling Unit (AHU) with 2 peripheral radiators

Motorized Lutron blinds (x 4)

Programmable HVAC controller (x 1)

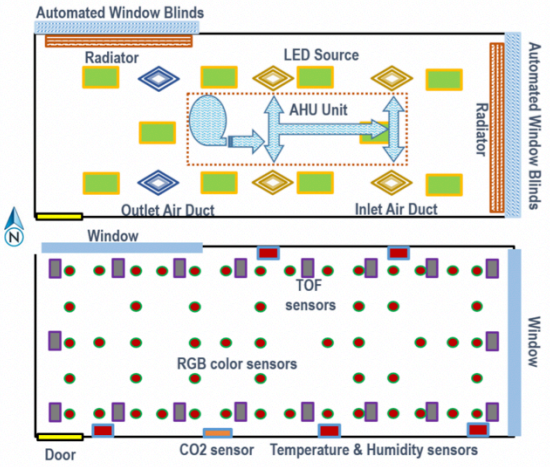
RGBC illumination sensors on a raspberry-pi platform (x 53)

Wireless temperature (x 5), humidity (x 1) and CO_2 sensors (x 1)



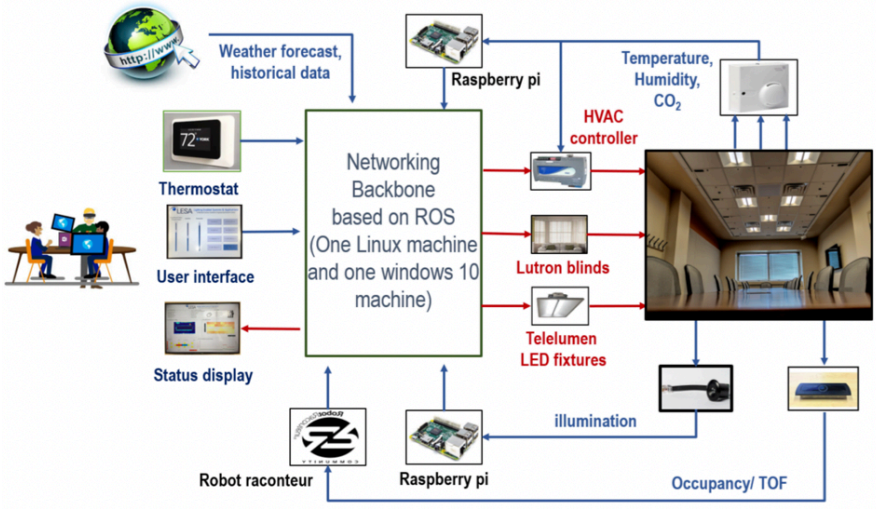
Iris matrix, time-of-flight camera (x 18)

Spatial Distribution of Key Components



The room also includes a Windows 10 Machine, a Linux Machine, an HVAC Server, and a Preference Coordinator Server, all within the SCR server closet. In addition, the conference room table contains six preference clients and the room has a status television, that provides the current status of the room, as well as a tablet that controls many of the room's operations, such as the lighting and window blinds. The next section will discuss these specific components of the SCR, that pertain to this manual of operations.

Networking Backbone



The Components

Overview

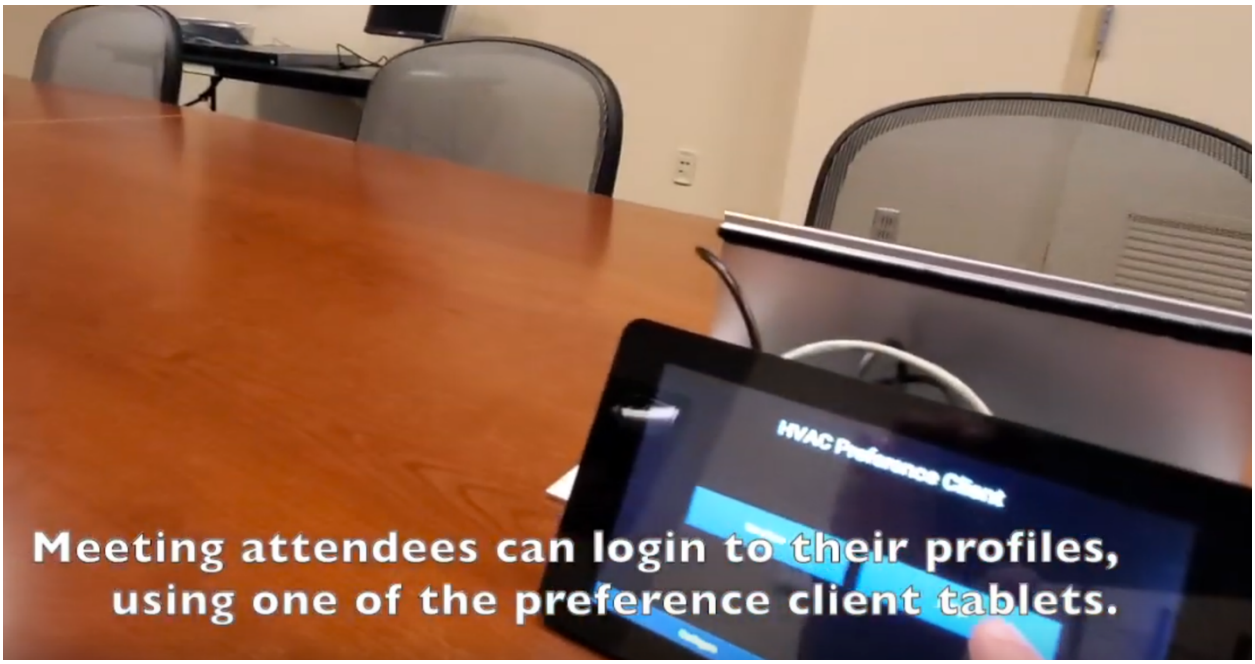
This section will go over the relevant components of the SCR and provide you with enough information to get started on them individually, and as a collective. Note that the process of receiving the login credentials for these components is described under the Credentials section of *References*, not here.

Preference Clients and Preference Coordinator Server

The SCR includes an integrated system for collecting the user preferences of occupants in the room. By providing a way for conference room attendees to share their preferences with the room (such as their preferred temperature and lighting preferences), Sam Atkinson and I have been able take their ideal preferences into consideration. This functionality drives much of the exciting work currently being done in the SCR.

Preference Clients

To begin with, six "preference clients" are used to collect the ideal temperature and lighting of occupants. Each client is a touchscreen, connected to (and run on) a raspberry pi. Six of them are evenly dispersed on the main conference room table. Users can create a "preference account" on here or login to their accounts, which will allow the Smart Conference Room to know you are there. On these clients, you can also see information about their current HVAC status' and connectivity to the preference coordinator server. All preference clients are the same, with the exception of "Preference Client 1" (they are all labelled with their corresponding numbers). Preference Client 1 has administrator options that can manage things, from the existing accounts of users in the SCR, to updating the location of clients, in terms of SCR coordinates. Despite having additional functionality, because the clients are aware of which one is Preference Client 1, the code on each raspberry pi is identical.



This always-running (boots on startup, unless there is a bug in your code) natural user interface program, on each of these preference clients, uses Python 2.7 as a programming language, Kivy 1.10 as the Python library for NUI development, and Requests 2.18, as a way to send HTTP requests, from the preference client, to the preference coordinator server. All of the source code, including the code for the preference clients, can be found under the Links section of *Resources*. The *screens* directory contains files for each possible "screen" a user may come across, such as the home screen (Home.py), the login screen (Login.py), and the Status screen (Status.py). The server directory allows the preference client to communicate with the preference coordinator server, which is the next SCR component to discuss.

Preference Coordinator Server

In order to handle the preferences of users in the SCR, the preference coordinator server communicates with each preference clients, using Python 2.7 and CherryPy, a minimalist, object-oriented web application framework. This coordinator server can be run under the username *public* by using the command

```
$ python coordinator.py
```

and using tmux to keep the program running, after detaching from your shell. It is important to note that before running the preference coordinator server, one must run the HVAC Server first. If the preference coordinator server is already running in a tmux shell, it can be accessed with the command

```
$ tmux a -t Preference_Coordinator      (or whatever you named your tmux shell)
```

To end the program, once attached, use "Ctrl-c" followed by

```
$ tmux kill-session -t Preference_Coordinator
```

to kill the tmux session. Otherwise, you can create a new tmux session, and run the program with

```
$ tmux new -s Preference_Coordinator
$ python coordinator.py
```

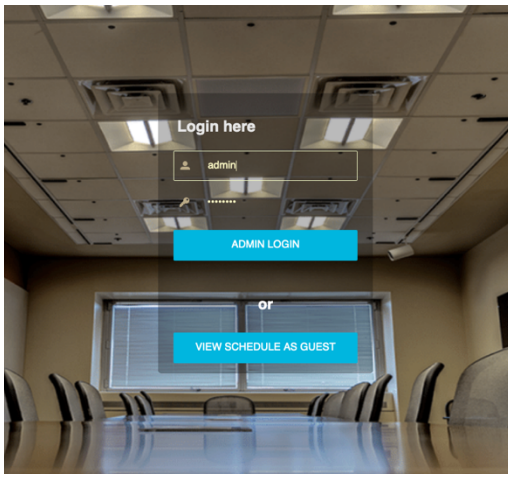
Leave the Python script running by using Ctrl+b, followed by d, to detach from the tmux session. If connected through SSH, you can now end your session with the command

```
$ exit
```

Using an SQLITE database (through queries.py and coordinator.py) CherrPy can do many important tasks, such as adding new users, deleting the accounts of users, retrieving the preferences of users, for the preference clients, and managing the upcoming meetings in the Smart Conference Room. This meeting management functionality is a perfect segue into the SCR Schedule Website.

SCR Schedule Website

Providing a way for the Smart Conference Room to be aware of when meetings occur is important for some of the current research tasks Ahsan and I have worked on. Steven Cano and I created the SCR Schedule Website in the spring and summer of 2019, to provide a way for the conference room to prepare the room temperature of a meeting, based on the preferences of meeting attendees, saved through the database of our preference coordinator server.



The website resembles a simplified Apple or Google Calendar app, that allows a SCR meeting to be scheduled. When scheduling a meeting, the LESA administrative assistant can add the event to the online calendar and list the emails of the people attending. If the email (username of an account previously created on either the preference client or website) matches a user, their temperature preference will be considered. Otherwise, at the moment, an account is created for the unregistered emails, and set to some default temperature and lighting preference. For example, the current default temperature is 70 degrees Fahrenheit.

Although meetings can only be scheduled on the website by logging in as an administrator, anyone can view the availability of the conference room, by viewing the calendar as a guest. From here, one can contact LESA to reserve the room, based on the information available on the website. The website currently runs on the SCR Linux machine, under the username *arunas*, and uses PHP, CSS, SQL, MySQL, XAMPP, and phpMyAdmin. XAMPP is located at /opt/lampp and one can learn about the commands by running

```
$ ./xampp help
```

in the lampp directory. Meetings scheduled on this site are then sent to the preference coordinator server, and the desired meeting room temperature ends up being the average of all the users' temperature preferences.

HVAC Server

The HVAC Server allows us to give the room HVAC commands from the preference coordinator server. Although you will most likely not need to change any of the code, you will need to run the server, fix any problems with the server, and update my HVAC Library, if there are ever any changes with how HVAC commands are handled.

Once logged onto the HVAC Server, you can run it, using the command

```
$ sudo python3 HVAC_server_LED_Toufiq_extended.py
```

Unlike the previously mentioned Python programs, the HVAC server uses Python 3. Similarly to the preference coordinator server, tmux is used here, in order to keep the program running, after disconnecting via SSH. Follow the same process under the *Preference Coordinator* section, except:

To create a new HVAC Server tmux session

```
$ tmux new -s HVAC_Server
$ sudo python3 HVAC_server_LED_Toufiq_extended.py
```

Detach with Ctrl+b, followed by d. To attach to an active session

```
$ tmux a -t HVAC_Server
```

To quit the program and end the tmux session

```
Ctrl-c
$ tmux kill-session -t HVAC_Server
```

Common Problems

When the HVAC Server is not following any specific commands, it should hand control back to the Building Management System (BMS). For example, if you give the HVAC Server some instructions, such as HVACLib commands, you want to give control back to the BMS, once you no longer need these commands (such as room temperature when you leave the SCR). ALWAYS give control

back the BMS and close the connection between your client and the server, once finished! To do this, use the HVAC.reset() and HVAC.close() commands at the end of your scripts to avoid any issues. You may run into an issue if you terminate your program when it is running, before you run these commands at the end. To solve this problem easily, you can run the Python script called emergencyreset.py, within the same network as the HVAC Server.

$ python emergencyreset.py

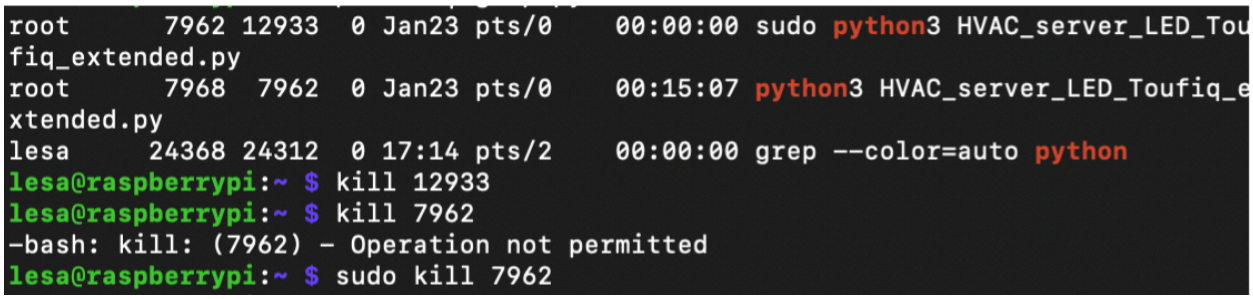
Another common issue with the HVAC server can occur, after running the HVAC_server_LED_Toufiq_extended.py file. The following error message

Fatal error: Server initialization failed
Unexpected error: [Errno 98] Address is already in use

can be solved with the command

$ ps -fA | grep python

From here, look for the relevant running processes, and end that process using the process number and the "kill" command.



Lastly, remotely connecting outside of the SCR Wi-fi network involves using an SSH port, different than the typical port 22. Since the public IP connects to the Linux Machine by default, you must explicitly declare that you would like to SSH to port 26

$ ssh lesa@128.113.122.178 -p 26

ToF Tagging Server

An important element of our research requires the ability to track the occupants in the SCR. The last components to discuss are the Time of Flight (ToF) sensors. Once users walk into the SCR, they are immediately tracked by ToF sensors. Much of the foundational Time of Flight code comes from the work of Tianna-Kaye (TK) Woodstock, and is a great person to reach out to, if you run into any ToF problems. Not only are ToF Sensors more affordable and less privacy-invasive than traditional tracking systems (Patent US9363859B2 from Lighting Enabled Systems & Applications Engineering Research Center), they can be far more accurate, by measuring the distances and positions of occupants through lighting.

These ToF sensors are relevant to us, as when users login to the preference clients, we tag that user, in order to map that user account to the relevant ToF array in the SCR. The following video shows this tagging in action:

https://www.youtube.com/watch?v=f6uynk3P3Do

Considering that this current tracking method will change drastically in the summer of 2020 (see the Uncertainties section of *Current Tasks*), all that is important to know from this guide, until the relevant changes are made, are that the ToF tagging coordinator server can be ran on the SCR Linux machine, under *Arunas*, at /home/arunas/catkin_ws/src/scr_control/scripts/zack/tof_tracking (named coordinator.py), along with toftracking.py, within the same directory.

Bringing it All Together

After going over every component individually, I wanted to briefly explain an example of everything being brought together. Imagine a faculty member, who wishes to schedule a meeting in the Smart Conference Room. They will first check the website for any availability, and then send the LESA administrative assistant an email, in which the faculty member will request a date and time, and list the meeting attendees. The administrative assistant will then mark the meeting on the Smart Conference Room Schedule Website. The website sends the information to the preference coordinator server, so it can set the temperature of the SCR to the average temperature preferences of all the

attendees, before the meeting begins. To control the temperature, the preference coordinator server relies on HVACLib, a library that will send HVAC commands to the HVAC Server, in order to control things, such as the temperature of the room.

When the meeting begins, attendees can log into the preference clients. If an unplanned occupant attends the meeting, who was never on the original SCR Schedule Website attendee list (and does not already have an account), they can either create an account on a preference client during the meeting, or login to one of them as a guest. Once an attendee logs into a preference client, the ToF Tracking server maps the closest user to the relevant tablet, using the Euclidean distance formula. If that specific occupant then moves around the Smart Conference Room, their Time of Flight coordinates will be used to reoptimize the temperature of areas throughout the room. When the meeting ends, the SCR hands back control to the Building Management System. Here is a video that goes over this example:

https://www.youtube.com/watch?v=q9ZZscRmoVI

Current Tasks

The relevance of this section greatly depends on when this manual is being read. A version (1.0) was mentioned on the cover page, as this manual will likely be updated in the near future. While most of this manual should stay relevant for a while, this section mentions some tasks and assignments that may become obsolete within few months from now (now being March 31, 2020).

Big Picture Tasks

There are two related "big picture" tasks, that are currently of the highest priority. That being said, these tasks are on the verge of being finished. The first task is a pre-heating/pre-cooling algorithm, based on much of our current research. In short, when a meeting is scheduled, how long must we wait to pre-heat and pre-cool the room, to a desired room temperature, as late as possible. The more accurately this time can be calculated, the later we can begin pre-heating and pre-cooling the room. As a result, when a meeting is not in session (and we are not heating/cooling the room for the meeting), the room can be set at a low-energy temperature, in order to save energy expenses (financially and environmentally) in the SCR. This work began with Zaid Tariq's research involving the experimental evaluation of data-driven predictive indoor thermal management. From here, Ahsan and I (much of the credit going to Ahsan) focused our research on smart temperature control in cohabited workspaces, through improved personalization and energy efficiency.

The second task extends on our goal of using our algorithm, to reach a given temperature in an indoor area. Using the ToF Tagging Server, as occupants move around in the Smart Conference Room, the algorithm must make fine adjustments, that continuously satisfy the different temperatures of occupants, based on their locations, to the best of our ability. An interesting way to imagine this concept is visioning each conference room attendee with an individual, invisible, personal bubble of temperature comfort. With respect to energy efficiency, and what is realistically possible in an indoor space, these subtle,

always-updating changes in temperature preferences, with respect to locations, are always being satisfied, as best as possible.

Minor Tasks

These tasks should be both quick and easy to complete, as soon as the research center reopens (this manual was written during COVID-19 campus closures). The first task is to finish the implementation of the status of temperature sensors, and thermostat readings, on the preference client tablets. The location of these statuses can be seen under the status tab of each preference client, and the user menu, once logged in. The second task, which would likely take the longest out of all the minor tasks, is the implementation of a priority queue that contains the upcoming meetings in the Smart Conference Room. The priority queue must be able to remove a meeting object, if removed from the website, and will be used preheat or precool meetings. The highest priority in the queue would be the closest approaching meeting scheduled, based on the current date and time. This implementation would also include removing the threading code related to add_meeting on coordinator.py, on the preference coordinator server. Finally, perhaps the quickest task is a bug that can be fixed in 30 minutes. When a user logs into a preference client, the entry field should clear after reaching the user menu. Otherwise, when the next user decides to visit the login screen, the previous username (email) still remains under the username field.

Uncertainties

Currently, much the Smart Conference Room runs on Robot Operating System (ROS); however, it was decided in February 2020 that the research center would be transferring over to MQTT. There are two things, relevant to this handbook, that this change will impact. The first thing that will be impacted, and will arguably be the most time-consuming task to fix, is the recreation of the Time of Flight Tracking and the ToF Tagging Server/Program. For this reason, I did not go into full detail about the Time of Flight tracking and tagging of occupants. The other task also provides some uncertainty, as the switch has not occurred yet. Because MQTT has not been fully implemented in the Smart Conference Room, the current method of sending HVAC commands will change. Fortunately, this task will most likely only involve rewriting the HVAC Library files (such as HVACLib.py).

References

Credentials

Contribute to the GitHub Repositories

To contribute to the GitHub repositories email me at zacknawrocki@gmail.com, with a GitHub username, so I can make you a contributor of all the relevant repositories, as well as get you in contact with the people who can add you to the LESA organization on GitHub, which contains additional repositories, much of which are private, but may be useful for your research.

Getting Access to Relevant Passwords

Much of the work mentioned in this handbook require login credentials, in which you must know information, such as usernames, passwords, networks, accounts, router info, remote access, and IP addresses. Because it would not be the best idea to provide these confidential credentials, all in one place, in this handbook, I have created a text file on the preference coordinator server, under the user *public*, located under NewStudent/credentials.txt. The IP address of this computer is 192.168.0.2.

When visiting the LESA Center, reach out to Arunas Tuzikas, a research engineer at LESA, and he can share with you the password to this machine (or I can, if this handbook is being read before November 2020). Once you can SSH into the server with the password, you will notice that the text file is password protected. Reuse the same password that you used to connect to the Preference Coordinator Server, to gain access to the text file. Once you have permanent access to these passwords, are confident you will not lose access to/forget them, and can pass this information down to future students upon leaving, it is best to delete this text file. Alternatively, you can reach out to me, and I can provide you with the credentials directly.

Links

GitHub Repositories
HVAC and Lighting Preference System -- This repository includes the code for the preference clients and the preference coordinator server.
https://github.com/zacknawrocki/HVAC-and-Lighting-Preference-System

SCR Schedule Website
https://github.com/zacknawrocki/HVACWebsite

Miscellaneous SCR Projects (Private, Email Me for Access) – This repository includes work, such ToF code (based on ROS) and useful scripts, such as emergencyreset.py.
https://github.com/zacknawrocki/Miscellaneous-SCR-Projects

LESA Website
https://lesa.rpi.edu/

SCR Schedule Website (Must Be on RPI Network or VPN to Access)
Please email me (zacknawrocki@gmail.com) for the link.

Contact

SCR Help: Arunas Tuzikas, TUZIKA@rpi.edu
ToF Help: TK Woodstock, woodst@rpi.edu

Feel free to reach out to me if you run into any issues, are confused by anything above, or need help with anything.

Email: zacknawrocki@gmail.com
Meeting Availability: Although I will be working full-time this fall (2020), I will be on RPI-campus, working remotely, so I can finish a final undergraduate course. For this reason, if necessary, I should have the availability to meet if help is needed, or if there are any questions that arise.