Knight-Tint

Oren Muszkal, Emmanuel Levasseur, Luckner Ablard, Stephen Polner
DEPT. OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, UNIVERSITY OF CENTRAL FLORIDA, ORLANDO, FLORIDA, 32816-2450

***Abstract* — Window tinting has been used for several decades for broad applications from commercial shading to privacy tints. Using various existing technologies, this team developed a new system to automatically modify the level of tint of windows. KnightTint is primarily targeted toward a residential application, with the preferences and control of an individual user prioritized. This paper's focus is on the usage of microcontrollers, sensors, and power design to modify window tint levels in a coordinated manner via a smartphone application.**

I. Introduction

KnightTint aims to leverage pre-existing window tint technology with an innovative control system. This system will automatically adjust the tint of the windows in a convenient manner that allows them to reach a desired indoor comfort level while saving money on the HVAC electricity costs associated with cooling their rooms. KnightTint is designed with the multifaceted approach of providing consumers an automatic tinting experience that targets desired indoor light intensity and temperature while also allowing for manual control for any purpose, such as privacy or a desire to have better outdoor visibility. To facilitate this, this tinting system will use a temperature sensor and light sensor to accurately measure the amount of light entering through a window, as well as the temperature of their room. These two sensors will be on or connected to a custom circuit board, which we will design to house an MCU that will communicate with a centralized microcontroller board. This central control unit will oversee the algorithmic and logical work and will then transmit coordinated instructions for controlling the tint percentage of each window. A smartphone application will be designed to configure the system to the user's preferences, manually override the automatic control, and expand system controllability.

II. System Components

KnightTint consists of many purchased components that are uniquely integrated to achieve its goals. This section offers a semi-technical overview of the selection of each of these components.

A. Window Tint

The window technology used is PDLC film. PDLC (Polymer Dispersed Liquid Crystal) film was selected for its ability to switch between transparent and opaque (frosted) states, offering dynamic privacy and light control. This smart glass technology integrates liquid crystals in a polymer matrix, which align to allow light transmission when an electric current is applied, and scatter to block light when off. It enhances energy efficiency by reducing the need for artificial lighting and climate control. PDLC offers nearly instantaneous switching time, a transmissivity range of 40-90%, very low power consumption, and most importantly, high commercial availability at a price point that is realistic for our project, though it varies widely by manufacturer.

Filmbase PDLC was chosen for its realistic price point, availability in a small sample size, and electrical and optical characteristics that were appropriate for this project and in line with the expectations of this technology.

B. Temperature Sensor

The TMP1075 temperature sensor is used for its high accuracy and low power draw while also being easy to implement. This device has a response time of ~100ms and is accurate to ±0.25°C. This sensor utilizes the simple communication method of I2C. It also has a very low cost of entry, at $0.24.

C. Light Sensor

The TI OPT4001 is used for its low cost, advanced functionality, and high range. Its maximum lux value is 117 klux, is capable of I2C communication, with a standard 1.6-3.6V input voltage. It has a cost of $2.12.

D.Battery

The Samsung 30Q 18650 battery was chosen for its high capacity, good voltage characteristics, and appropriate current ratings. This battery's mAh is 3000, which is a good rating for this battery type. Its high nominal voltage of 3.6V reduces the number of batteries needed, and the lithium-ion technology makes it energy dense and rechargeable. The price per battery of $2.95.

E. Linear Regulator

The TPS561201 linear regulator was chosen for its ability to regulate a wide range of voltage input, 4.5 V to 17, down to a consistent 3.3V at high efficiency. The voltage input encompasses the discharge profile of the batteries and the consistent 3.3V output is needed to reliably power the MCU and sensors. The regulator can support an output current of up to 1A, which is comfortably above the current requirements of this

application. Its implementation and design requirements are simple and explicitly laid out in its documentation.

F. Op-Amp

The op-amp of choice is the OPA454. This op-amp was chosen for its 120V output, as 90V are needed for the PDLC tint, wide bandwidth of 6.5 MHz, and high slew rate of 32V/us. This product's cost per unit is $3.90.

G. Microcontroller

The ESP32-WROOM-32E-N4 microcontroller module was selected for its robust performance, versatile connectivity options, and efficient power consumption. This microcontroller features a dual-core processor with a clock speed of up to 240 MHz, integrated Wi-Fi and Bluetooth capabilities, and a wide range of I/O interfaces. It has its own PCB antenna and its own DAC within, negating the need to select external versions of these components. This ESP32 module offers 520KB of SRAM and 4MB of flash memory, making it suitable for complex applications. This device has access to vast libraries of content and example code with low hardware implementation complexity. Its price per unit is approximately $3.50.

The ESP32-WROOM-32 development board was selected to act as a central control unit because it has an ESP32-WROOM-32 chip as well, making coding and Wi-Fi communication between MCUs streamlined and simple. This development board can be easily powered from a common AC to USB converter, which makes its power design very simple.

IV. PCB BOARD DESIGN

This project is implemented on a development board and 4 printed circuit boards. One board houses the MCU, temperature sensor, and battery compartment. Another board is a breakout board for the first one, which contains the linear regulator and its associated components. Another is designed to house the light sensor and its associated components, which can be placed on the outside of the window. The final board contains our high voltage op-amp and outputs to the window tint film. Our PCBs were designed in Altium Designer.

A. MCU PCB

Below are two images for the MCU board. The first image is the 2D PCB layout view on the top layer. In the second image, the 3D model of the PCB can be seen, which gives a more practical view of the PCB physical realization. This is the largest board of the project with dimensions of 5" by 4.5". PCB traces were routed through vias between the top and bottom layers to facilitate optimal routing. The bypass capacitors and pull-up resistors have been placed near their respective components in accordance with the datasheets and hardware design guidelines. The PCB has a battery compartment that houses three 18650 Li-Ion batteries which will be able to power KnightTint's low voltage devices after being regulated down to 3.3V. In each of the corners, a 3.2mm mounting hole has been placed, through which we will mount the PCB using M3 screws. In the top right, the regulator breakout board 3-pin connector can be seen with that board's planned outline above it. Below that is the MCU, located in a central location to facilitate trace routing. Below the MCU are pin connectors for the light sensor and op-amp PCBs, which will be connected with wires. There is another pin connector that will be connected together with a shunt connector for programming the MCU. Finally, the temperature sensor and its associated components are located in the bottom right to help mitigate thermal interference from the MCU, regulator, or batteries.

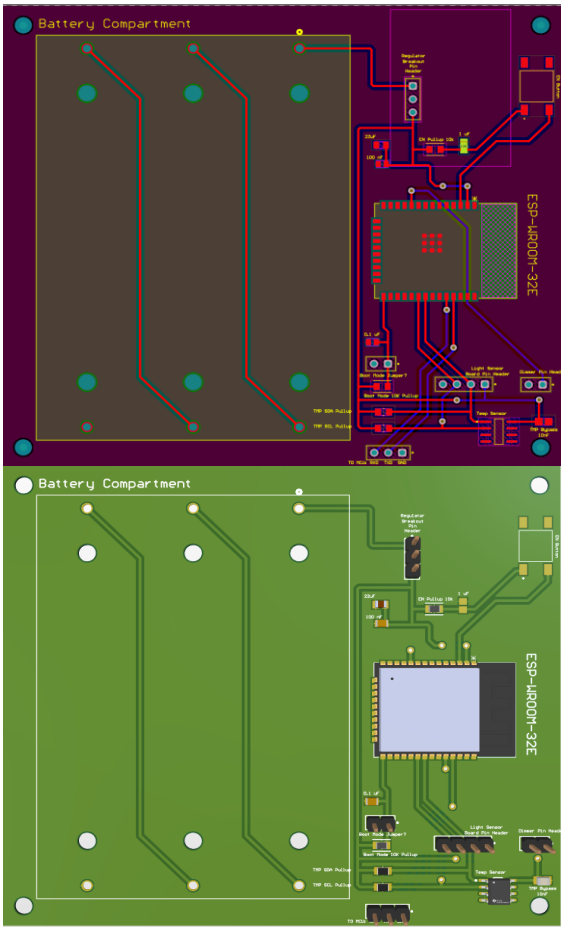

Figure 1- MCU Board

B. Regulator Breakout PCB

Another board that was designed was the regulator breakout board. Because we know that regulators can be fickle components, it was decided that this board would be attached to the main board via a pin connector and thus be a breakout board. This board was needed because the MCU and sensors are designed for 3.3V input. However, the three batteries in series offer a nominal voltage of 10.8V, which is stepped down by the regulator.

We want to input a steady 3.3V for several reasons. Firstly, we want to make sure the voltage stays within the acceptable input range. If there is a spike in voltage for any reason, such as a faulty battery, we want to make sure that the hardware does not get damaged. A steady voltage input also helps minimize the impact of noise and interference on the circuit.

The components of the regulator were selected and placed in strict accordance with the datasheet to prevent the component from malfunctioning.

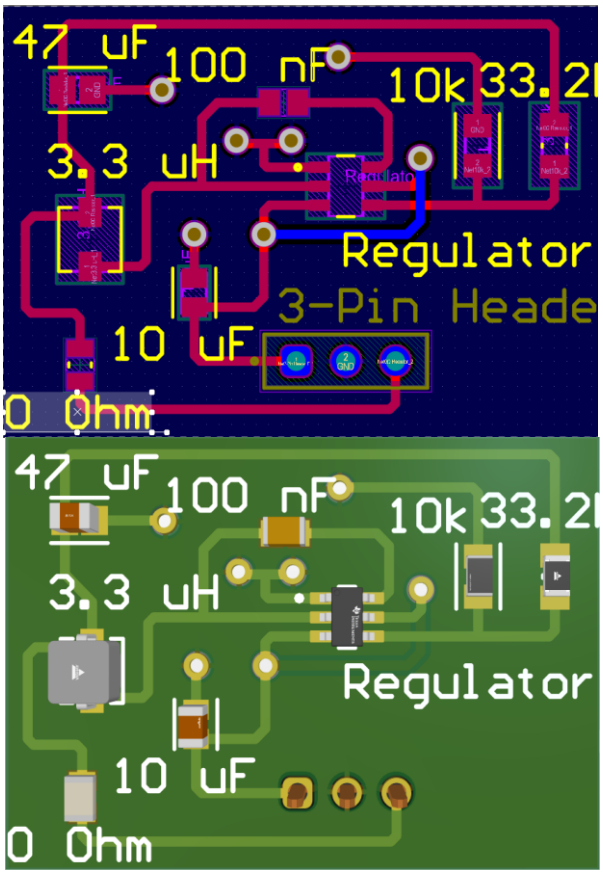

Figure 3- Regulator Breakout

C. Op-Amp PCB

The Op-Amp PCB was another board designed separately from the MCU board over concerns of component burnout. Our PDLC film needs a square wave with a Vpp of 90V, meaning our op-amp must be a high voltage component. These high voltage components can be very prone to burnout, giving a high chance that we would need to redesign and reorder this board. We also would not want the high voltage to destroy any of our MCU board components if a short were to occur. The op-amp on this board has its rail voltage supplied by a pre-purchased booster module, which takes 12V from a 12V power supply and boosts it to 90V. This is then split to +45V and -45V and supplied to the rails via pin headers. High voltage rails are placed at each rail input to reduce noise. The Op-Amp accepts a 0-3V input from the ESP32 -WROOM-32E DAC and acts as a non-inverting amplifier to apply a gain of 31. Since the DAC output from this module cannot be negative, a AA battery is used as a reference voltage, which mathematically allows the op-amp to output -45 to 45 V rather than 0 to 90V.

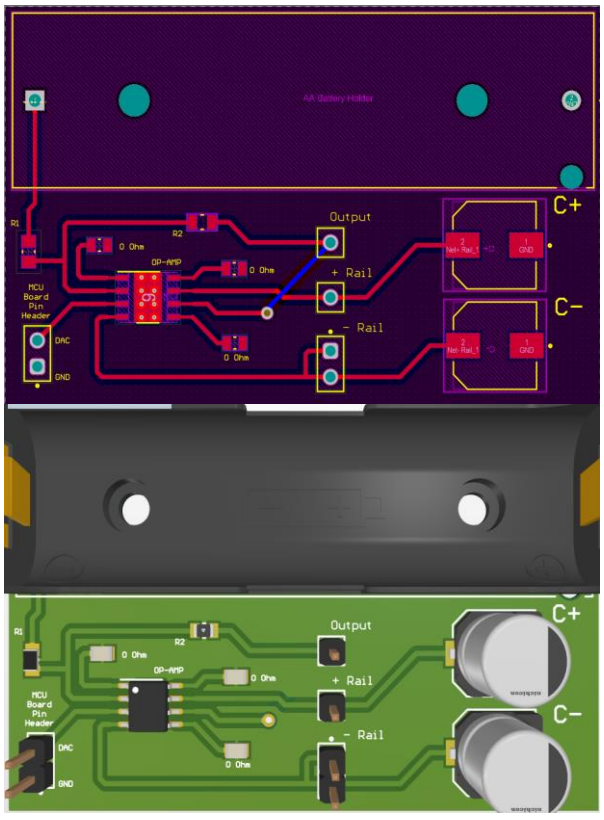

Figure 2- Op-Amp Board

D. Light Sensor PCB

The final board produced was the light sensor board. This board contains only the light sensor and its associated components, such as I2C pullup resistors. The light sensor had to be placed on a separate board because it must be on the exterior of the tint to measure light intensity before the light is filtered by the tint. This PCB is not designed for outdoor ruggedness, which is why a casing made of clear plastic was produced to protect it from the elements.

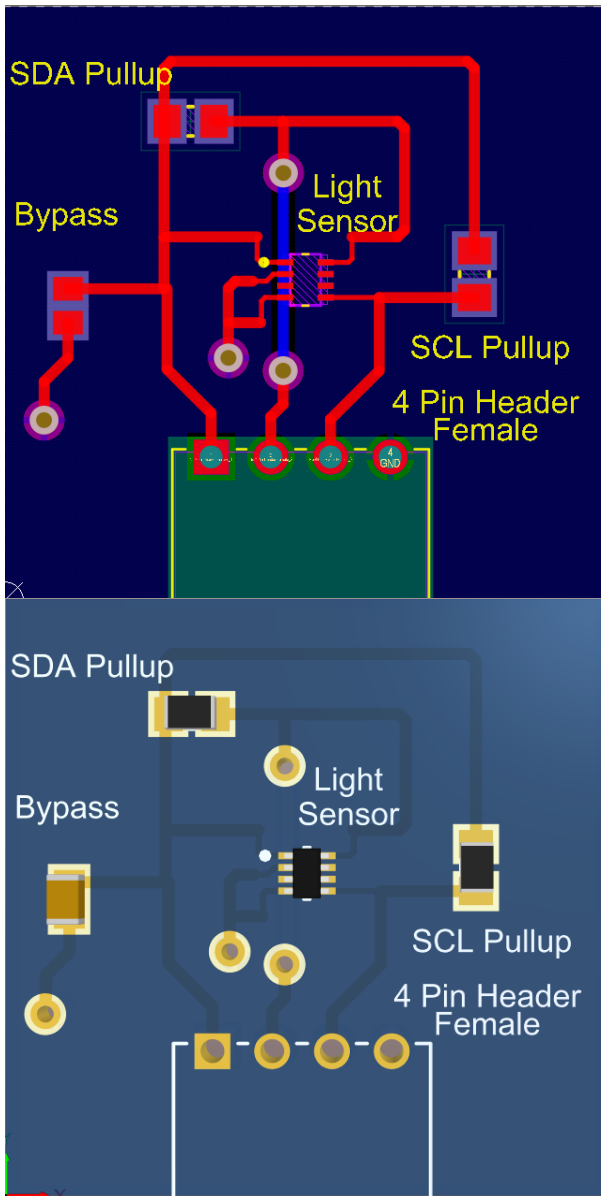

Figure 4- Light Sensor Board

IV. Software Design

A. Embedded Design

This diagram represents the process flow for adjusting the tint level of the tinting system. The process begins with checking the mode that leads to three different types of paths: scheduled, manual, and auto. The system in the path of the schedule will request the schedule from the app and set a time and the desired tint for that specific time. In manual mode, it responds directly to the user's request to change the tint level and sends the signal to adjust the system appropriately. The auto mode involves an automatic algorithm that determines the best tint level based on light level and temperature. Each path converges at two decision points; whether the tint level is set correctly or not. If correct, the system will enter a form of low power mode. If it's not correct in the automatic mode, it adjusts the tinting before going into low-power mode. This diagram shows how the different modes are handled and how the different decisions are made based on the desired outcome. This diagram provides a clear overview of the function of the system and the point of interaction for the user.

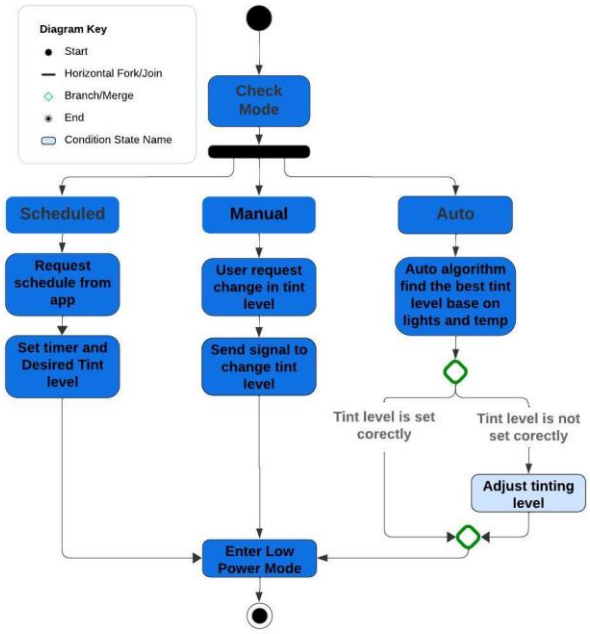

Figure 5- Software Activity Diagram

The light sensor will be connected to our board that is going to be on the window and it will serve to provide the automatic mode function to our controller. As part of the algorithm that will be written the information for the sensor will be taken in and based on the amount of light that is coming in the program will be able to adjust to different levels of tinting. The OPT4001 supports the transmission protocol for standard mode (up to 100 kHz), fast mode (up to 400 kHz), and high-speed mode (up to 2.6 MHz) both the MCU and the eUSCI module must support the speed to be able to run the sensor at that speed. The eUSCI module in our MCU supports only the Standard Mode and the Fast Mode and so we will limit the I2C clock frequency to 400 KHz. After configuring our module then we will configure the system to start talking to the sensor. Once configured, the system will seamlessly communicate with the sensor, allowing for real-time adjustments based on incoming light levels, thus providing efficient and adaptive tinting functionality for enhanced user comfort and energy efficiency. The temperature sensor, unlike the light sensor, will also be attached to the board on the window side and will provide us with information on how much light is currently present at that window. It will also determine if increasing or decreasing the intensity of the tint will affect the current temperature of the room. We will take the information that is coming from the sensor and compare it to what the user has set and change the tinting level based on that. This will be done using an SPI module in the microcontroller, run at a speed matching that of the sensor. The graphic below shows how the module will be connected, and which line will be connected to what.

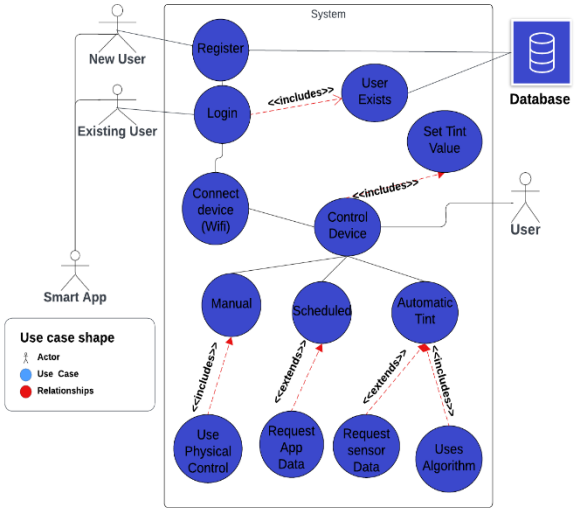

Figure 6- Use Case Diagram

B. Application Design

The login page offers users the option to sign in using their existing account credentials. Illustrated below, it comprises several key components: a username field for inputting a unique identifier, and a password field for authentication. Both username and password details are securely stored in the database. Additionally, a "forgot password" link is provided, allowing users to reset their password if forgotten. Clicking this link directs users to a password reset page, where they can verify their identity via security questions and create a new password. Upon entering valid credentials, the focus shifts to the sign-in button, initiating the authentication process upon clicking. Finally, for new users without an account, a signup link is available, redirecting them to the registration page where they can create an account by providing the necessary details such as username and password.

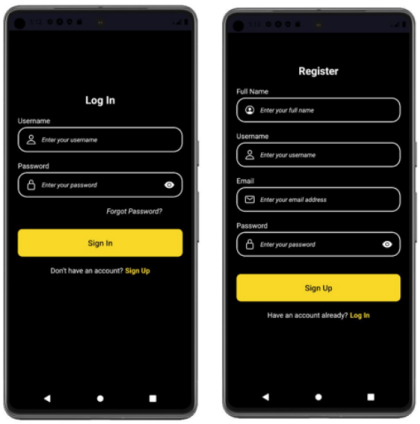

Figure 7- Login Page

The landing page showcases an elegant design with a well-structured navigation system. At the top, the navigation bar features a personalized greeting text that adjusts according to the time of day and displays the user's first name who has logged in. On the left side of this bar, there is a prominently placed sign-out button for easy access. Directly beneath this, the middle navigation bar is positioned, which includes buttons for "Devices" to facilitate quick access to the device management section. Below this bar, the page displays all the window devices connected to the hub, providing a comprehensive view of the networked devices. The bottom navigation bar rounds out the layout with intuitive icons for "Home" and "Sync," allowing users to effortlessly navigate back to the home screen or synchronize their devices.

Figure 8- Landing Page

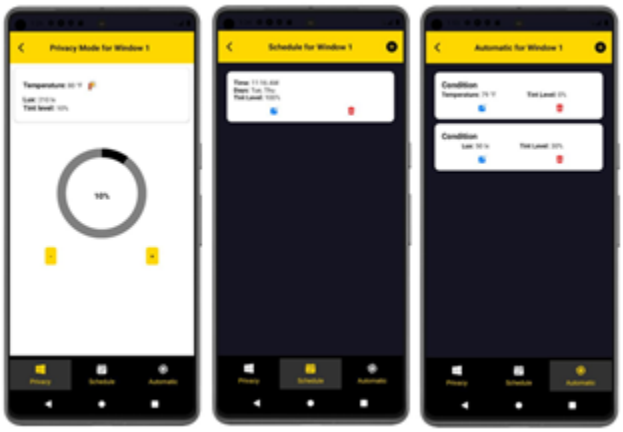

Figure 9- Window Page

The KnightTint system provides an advanced window tint control solution with four distinct modes—Manual, Privacy, Schedule, and Automatic—along with a synchronization option to manage multiple windows seamlessly. This mode is designed for intuitive control, featuring a bottom navigation bar that includes labeled buttons and icons for easy access to Privacy, Schedule, and Automatic modes.

In Privacy mode, users can quickly increase the tint level to 100% with a single click, ensuring maximum privacy, or decrease it to 0% for a completely clear window. This mode is particularly useful for scenarios requiring immediate changes in tint for privacy or visibility.

Schedule mode provides users with the flexibility to set specific tint levels based on the time of day and day of the week. This mode supports multiple schedules, which are stored securely in the cloud using MongoDB and managed with the CRON dependency. Users can program the system to adjust the tint at precise times, ensuring that their windows are always at the desired tint level throughout the day. This feature is ideal for automating the tint adjustments in response to routine activities or changes in natural light.

Automatic mode enhances the functionality of KnightTint by allowing users to set conditions based on environmental factors such as temperature and light intensity (lux). This mode ensures that the tint level adapts dynamically to changing conditions, optimizing comfort and energy efficiency.

V. Software Detail

A. Embedded Design

To explain the embedded software aspect of this project, we need to understand how everything will come together. The project will consist of two separate windows that will be connected via wifi using the ESPNow protocol. Through this connection, the boards will send and receive data to each other in the form of a struct, as illustrated in the diagram below. The main board, which will receive the data, will process it and send it to the app's backend so that it can be displayed to the user on the home screen.

```
typedef struct board_message {
    int WindowNumber;
    float LUXVal;
    float tempVal;
    uint8_t tintLevel;
} board_message;
```

Figure 10 - Window Information Struct

The software for the window boards needed to be further broken down. Both the light and temperature sensors are connected to the same I2C line. We encountered an issue where the communication between the master and the slave devices clashed, so we ordered it in a way to ensure that we were not requesting information from any of the sensors at the same time, which prevented the devices from clashing. Both sensors must be initialized to control how often and the precision of the data we receive. The temperature sensor had a library that we could use to simplify the coding process, but for the light sensor, we had to start from scratch.

```
void OnDataSent(const uint8_t *mac_addr,
esp_now_send_status_t status) {}
void OnDataRecv(const esp_now_recv_info *mac, const
uint8_t *incomingData, int len) {}
void setup() {}
void loop() {}
float getLUX() {}
float celsiusToFahrenheit(float celsius) {}
```

Figure 11- Sensor Input

The hub sensor will be configured to receive data from both the app and the other two windows in the system. The diagram below illustrates the various functions of the hub board and how they facilitate a smooth process. We encountered an issue with the dual Wi-Fi mode, which required us to ensure that we could both receive and send out information to the boards if necessary.

```
int32_t getWiFiChannel(const char *ssid) {}
void formatMacAddress(const uint8_t *macAddr, char
*buffer, int maxLength){}
void OnDataRecv(const esp_now_recv_info *mac, const
uint8_t *incomingData, int len){}
void OnDataSent(const uint8_t *macAddr,
esp_now_send_status_t status) {}
void broadcast(const board_message& message){}
void setup(){}
void loop() {}
void webSocketEvent(byte num, WStype_t type, uint8_t *
payload, size_t length){}
```

Figure 12- Data Transmission

B. Application Backend

Backend Endpoints

User Registration Endpoint (POST /register): Handles user registration by creating a new user account with parameters: name, email, username, and password.

User Login Endpoint (POST /login): Authenticates users by verifying the username and password.

Check Email Existence Endpoint (POST /checkEmail): Checks if an email is already registered.
Password Reset Endpoint (POST /resetPassword): Manages password reset requests with parameters: email and newPassword.

Endpoint Details

User Registration Endpoint: Checks for existing users, hashes the password, creates a user document, and returns status codes 201 (success), 400 (username/email exists), or 500 (server error).

User Login Endpoint: Verifies username and password, generates a JWT, and returns status codes 201 (success), 400 (invalid credentials), or 500 (server error).

Check Email Existence Endpoint: Confirms email existence, returning status codes 200 (exists) or 500 (server error).

Password Reset Endpoint: Validates email, hashes new password, updates user password, and returns status codes 200 (success), 400 (email not found), or 500 (server error).

Frontend Implementation

Sign In Function: Confirms inputs, sends a POST request to /auth/login, processes the response, handles errors, and redirects to the home page upon success.

Sign Up Function: Validates inputs, sends a POST request to /auth/register, processes the response, handles errors, and redirects to the sign-in page upon success.

Reset Password Function: Validates passwords, verifies email via /auth/checkEmail, resets password via /auth/resetPassword, processes the response, handles errors, and redirects to the sign-in page upon success.

Components and Libraries: The application utilizes various React Native components:

i. expo-router and react-native-safe-area-context for navigation and layout.
ii. expo-status-bar for managing status bar appearance.
iii. react-native-vector-icons for iconography.
iv. react-native-progress-circle for visualizing tint level adjustments.

WebSocket Integration: WebSocket (WebSocket from ‘ws://esp-32-ip:WebScoketServerport/’) enables bidirectional communication, ensuring real-time updates of window tint levels based on user inputs. WebSocket technology facilitates instant data transmission between the mobile app and an ESP32 microcontroller, which controls physical window tinting mechanisms.

State Management and Navigation: State management employs React's 'useState' hook to manage dynamic data such as window tint levels ('windowData') and active navigation ('focusedItem'). Navigation is handled via 'useRouter' for seamless transitions between application screens.

Data Handling: WebSocket 'onmessage' event updates 'windowData' upon receiving JSON-formatted data from the ESP32 microcontroller, including temperature, light intensity (Lux), and current tint level.

User Interface: The user interface features:

i. Real-time display of temperature, Lux, and tint level ('ProgressCircle').
ii. Intuitive controls ('TouchableOpacity' buttons) for adjusting tint levels ('increaseTint', 'decreaseTint').

VI. Conclusion

This paper has gone through a comprehensive journey of the design and development process for our window tinting system, KnightTint. The idea for this tinting system related to our lives in the sunshine state of Florida and helped solve a common issue of energy costs related to indoor air conditioning, as well as built off the smart-home movement that has been occurring over the last few decades. Our group has successfully created a prototype that accomplishes our goals of user comfort, satisfaction, and convenience by responding to dynamic stimuli. This was all accomplished while maintaining advanced user control and customization. This is a product with real-world useful application, and we believe that with more design iterations, this could become a new staple product in the smart home revolution.

VII. ACKNOWLEDGEMENT

The Authors would like to acknowledge the assistance and support of Dr. Wasfy Mikhael, Dr. Mike Borowczak, and Dr. Zakhia Abichar.

The Authors would like to give special acknowledgement to our two advisors, Dr. Chan and Dr.Weeks.

Dr. Chan was instrumental in allowing our concept and giving structured guidance while the team navigated through obstacles and issues.

Dr. Weeks was a guiding hand during the latter stages of the project and helped with all hardware issues that occurred during the prototyping stages as well as demonstrative tests and designs.

VIII. BIOGRAPHY

Oren Muszkal is a 27-year-old Electrical Engineering student at UCF. Upon graduation, Oren will continue to pursue a graduate degree at UCF. His research will be based on Micro-Electro-Mechanical Systems. His primary work will be in Quantum computing applications.

Stephen Polner is 21-year-old Electrical Engineering student at UCF. Upon graduation, Stephen will work in engineering consulting for GHD in Downtown Orlando. He plans to gain industry experience before pursuing a Master's Degree in Nuclear Engineering.

Luckner Ablard is an ambitious undergraduate student pursuing a degree in Computer Engineering at UCF. He plans to work towards his master's degree and looks forward to a future in the engineering field.

Emmanuel Levasseur is a 22-year-old Computer Engineering student at UCF. Upon graduation, Emmanuel will work for Deloitte US as a Solution Analyst. To further his studies, Emmanuel plans to pursue a master's degree in business management.