CSC513 Assignment 1:

Lighthouse Tangible User Interface

Chloë Farr

CSC 513: Designing Creativity Support Tools
Prof. Sowmya Somanath
Computer Science
University of Victoria
May 27, 2025

1 Introduction

2 Aim

The aim of my project is to create a simulation of sunlight based on real weather data. Using an infrared remote as the input, the user will be able to trigger a time lapse of sunlight based on weather data, as well as scrub back and forth in hourly increments. These input behaviours will result in data being outputted through an RGB LED and an LCD display. The LED will change colour and brightness based on time, cloud cover, and position of the sun.

Justifying the Sun Simulator as a Creativity Support Tool

The Sun Simulator is a digital system that allows users to explore how dynamic, time-based lighting conditions affect spatial environments. At its core, the system accepts tangible input from a handheld infrared remote and translates that input into colour values expressed through an RGB LED. These values correspond to a simulated daylight cycle—sunrise to sunset—providing a programmable, pseudo-natural lighting experience.

This system is inherently creativity-focused. It offers designers the ability to externalize, test, and iterate on lighting decisions in real time. By removing the need for complex digital rendering software or unpredictable reliance on natural sunlight, the Sun Simulator provides an accessible and embodied way to experiment with temporal lighting. It bridges data-driven simulation with tactile interaction, allowing users to engage with light as a dynamic design element.

The primary users of this tool are professionals who work at the intersection of space and aesthetics: architects, interior designers, and lighting designers. These individuals frequently create physical models or digital plans, but rarely have access to tools that let them feel the passage of time through light. Sun Simulator allows them to observe how shadow, hue, and brightness affect physical prototypes, to in turn inform choices such as material selection, spatial layout, or placement of structural elements like windows and support columns. It can also serve as a communication tool, helping these experts convey lighting effects to clients or collaborators who may not intuitively grasp abstract renderings or static models.

From a creativity process perspective, the tool supports both experiential and contextual dimensions. It encourages perception and reflection by letting the user see how lighting transforms a space throughout the day. It promotes learning and communication, not just in design ideation but in design justification—answering the question "why this configuration?" with tangible visual feedback.

The tool is also designed with context in mind. Environmental variables such as the orientation of a structure, time of year, and surrounding geography all influence how light behaves in a space. Sun Simulator allows users to approximate and respond to these factors in a controlled and repeatable way during the design process.

The domain of use includes environmental and architectural design, lighting art, set design, and even education. Its design process reflects an inspirationalist approach: it was conceived not from a specific technical requirement, but from a realization about how light affects space and how few tools exist to manipulate that variable meaningfully in physical design practice. The system grew out of this insight, through iterative sketching, observation, and prototyping.

In broader CST theory, creative work emerges from the interplay between individual, domain, and field. The Sun Simulator aligns with this model. It empowers individuals—those with a design-oriented temperament—to engage deeply with the domain of spatial and lighting design, while also serving as a compelling tool to communicate creative ideas to the field—clients, reviewers, or collaborators who evaluate and refine those designs.

In summary, the Sun Simulator is a digital CST that offers tangible, dynamic, and repeatable input-output experiences, supporting creative ideation, evaluation, and communication. By simulating the sun, it sheds light not only on architectural spaces, but on the creative decisions that shape them.

Approach. *What is your approach to designing the CST*: **inspirationalist** The approach I took to designing the CST is similar to the way I hope someone would take when using it. I came to design this project organically, being inspired by spaces around me, and becoming aware of how space and light interact. To foreshadow Project 3, I intend to create a diorama of a house with modular lighting that a lighting designer can play with to develop creative lighting designs. As I went through the process, I had a realization that interior lighting will always be affected by natural light penetrating a building, be it through a window or cracks in a door. That realization, and the fact that light is dynamic.

Break away from the structure to create an aha moment!

Brainstorming, sketching, random stimuli, going to new locations, taking a break etc.

situationalist [7] Creative work is social, and it comes from: • Domain – some set of rules and procedures relevant to a domain e.g., computer science, biology • Field – the people who shape the domain e.g., conference reviewers • Individual – the person who generates the ideas

Person. [6] *who is the end-user of the tool?*

Lighting designer - film, stage, residential.

COGNITIVE VIEW: Person — personality, intellect, temperament, physique, traits, habits, attitudes, self-concept, value systems, defence mechanisms, and behaviour

Process. *what would they need to do to achieve that creative outcome?* EXPERIENTIAL VIEW: Process — perception, learning, thinking, and communicating

CONTEXTUAL VIEW: Press — relationship between human beings and their environment

Domain. *what is the domain in which the person is working?*

Creative outcome. *what is the person hoping to create that would be original and effective?*

Four C model. [1]

The Sun Simulator supports both mini-c and pro-c creativity within the Four C Model. For mini-c, the system provides non-professional users such as clients or buyers with a meaningful way to engage with architectural or interior design concepts. By seeing how daylight would move through a space, they may gain personal insights about furniture placement, paint choices, or spatial use that they hadn't previously considered. These discoveries may not be groundbreaking at a professional level but represent genuine creative insight for the individual. At the pro-c level, the tool enables experienced designers to test, refine, and iterate their work using time-based lighting data, allowing for expert-level adjustments to design elements such as material choice, layout, and orientation. By accommodating both intuitive learning and professional-level iteration, Sun Simulator fosters creativity across the spectrum of user experience.

Location. [6] *where is creativity located?*

3 Inspiration

[3] Which research paper did you select? [2].

4 Motivation

This project is motivated by a desire to give lighting designers and architects a way to evaluate their models under realistic daylight conditions. While digital tools like AutoCAD and Lumion help visualize architectural designs, the corresponding physical 3D models rarely portray how those models interact with actual light.

Architectural models are often viewed in uncontrolled lighting environments, limiting designers' ability to assess how their creations interact with natural light throughout the day. This project introduces a programmable light source that simulates the progression of daylight, offering a first step toward physically recreating realistic lighting conditions around a scale model.

5 Design and Methodology

[5] Explain the design and implementation of your TUI

5.1 Extending the Inspiration

how is your system extending the research paper idea?

5.2 Design Rationale

The design choices for this project were made to balance usability, scalability, and system reliability. A key component is the use of an infrared (IR) remote control for input. This decision was motivated by a desire to offer a wide range of user interactions while keeping the interface intuitive. The IR remote comes pre-labeled, making it easier for users to understand and access its functions without requiring a custom housing or an external schematic.

Compared to using multiple physical buttons wired directly to the Arduino, the IR remote offers several advantages: it reduces power consumption, simplifies wiring complexity, and minimizes the risk of hardware failure due to loose connections. It also enables future expansion, as additional commands can be added without requiring more physical inputs on the board. While the IR receiver itself is a single point of failure, it is more robust overall than a collection of manually wired push buttons.

The broader design goal is to eventually integrate this system into a scaled architectural model with modular interior lighting. In real-world environments, interior lighting is influenced by natural light entering through windows or other openings. By simulating daylight as part of the model's lighting conditions, the project allows for a more accurate representation of how a space would behave under varying environmental conditions.

5.3 Schematics and Code

The system runs on an Arduino Uno and uses an RGB LED to represent changes in daylight across a 24-hour cycle. It accepts input from an infrared (IR) remote and outputs status updates on a 16x2 LCD screen. The breadboard layout and pin connections are shown in Figure X (inline image), and additional reference materials including the schematic, code, and full components list can be found in the appendices.

The remote communicates with the system through an IR receiver module, which reads unique IR codes emitted by each button press. These IR codes are hexadecimal values interpreted by the Arduino using the IRRemote library. The script maps these IR codes to commands like turning the system on/off, starting or pausing the time lapse, stopping it entirely, or manually scrubbing forward and backward through each hour of the day.

Time lapse mode simulates the passing of time by gradually fading the LED between hourly colour values. This creates a fluid transition between states and mimics the subtle progression of daylight. In contrast, scrubbing mode lets users jump directly between hours—useful for quickly previewing lighting conditions at a specific time. These lighting states are currently based on placeholder RGB values, but the structure is built to accommodate real-world data in the next phase.

The LCD provides real-time feedback by displaying the system's power state, current hour, and whether the time lapse is active. It's controlled via a potentiometer to adjust contrast and is updated dynamically with each interaction.

Although the codebase is relatively lightweight, it's structured to separate responsibilities for better clarity and future scaling. One header file handles the mapping of IR codes to commands, keeping the main sketch focused on system logic and LED control.

Code is also in Appendix B. See Figure 1 for a photo of the arduino setup. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. Include images/screenshots where appropriate.

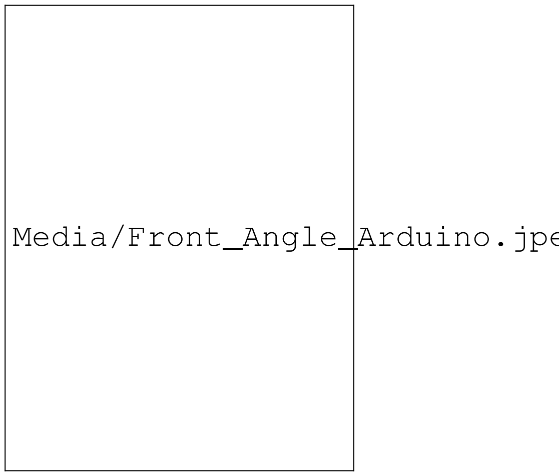


Figure 1: System prototype

6 Next Steps

While the current system effectively simulates a 24-hour light cycle using pre-defined RGB values, the next phase will replace this placeholder logic with real-world data. Specifically, Sun Simulator will incorporate historical environmental data—such as solar angle, cloud cover, and temperature—sourced from public meteorological archives. These inputs will be processed to generate dynamic RGB values that more accurately reflect the colour temperature and intensity of natural light throughout the day.

The lighting simulation will also expand from a single point light source to a spatially distributed lighting system. A motorized pulley mechanism will move the LED along an arched track above the model, enabling simulation of the sun's movement across the sky. This will allow for both directional and time-based lighting cues, giving designers a more immersive and context-rich preview of their designs.

This foundation will ultimately support a full creative lighting sandbox—where time, orientation, and environmental conditions can be programmatically or manually adjusted to explore design decisions in a controlled, data-driven environment.

[4]

7 Conclusion

References

[1] James C Kaufman and Ronald A Beghetto. "Beyond Big and Little: The Four C Model of Creativity". eng. In: *Review of general psychology* 13.1 (2009), pp. 1–12. ISSN: 1089-2680.

[2] Youqin Lin et al. "Daylight simulation algorithm and application based on atmospheric radiation theory". In: *Optics & Laser Technology* 190 (2025), p. 113247. ISSN: 0030-3992. DOI: https://doi.org/10.1016/j.optlastec.2025.113247. URL: https://www.sciencedirect.com/science/article/pii/S0030399225008382.

[3] Júlia Nacsa, Emilia Barakova, and Joep Frens. "Sharing meaning and physical activity through a tangible interactive lighting object". In: *Procedings of the Second Conference on Creativity and Innovation in Design*. DESIRE '11. Eindhoven, Netherlands: Association for Computing Machinery, 2011, pp. 227–232. ISBN: 9781450307543. DOI: 10.1145/2079216.2079249. URL: https://doi.org/10.1145/2079216.2079249.

[4] Magy Seif El-Nasr and Ian Horswill. "Automating Lighting Design for Interactive Entertainment". In: *Comput. Entertain.* 2.2 (Apr. 2004), p. 15. DOI: 10.1145/1008213.1008238. URL: https://doi.org/10.1145/1008213.1008238.

[5] Jeongseok Oh, Seungju Kim, and Seungjun Kim. "LumiMood: A Creativity Support Tool for Designing the Mood of a 3D Scene". In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. Honolulu, HI, USA: Association for Computing Machinery, 2024. ISBN: 9798400703300. DOI: 10.1145/3613904.3642440. URL: https://doi.org/10.1145/3613904.3642440.

[6] Mel Rhodes. "An Analysis of Creativity". eng. In: *Phi Delta Kappan* 42.7 (1961), pp. 305–310. ISSN: 0031-7217.

[7] Ben Shneiderman. *Creativity support tools: accelerating discovery and innovation*. eng. New York, NY, 2007.

A Components List

Name	Quantity	Component
UArduino Uno	1	Arduino Uno R3
U2	1	LCD 16×2 display
RLCD, RLED Red, RLED Green, RLED Blue	4	220 Ω Resistors (for LED current limiting and LCD backlight)
RpotLCD	1	10 kΩ Potentiometer (for LCD contrast)
D1	1	RGB LED (Common Anode)
UDUTTY	1	38 kHz IR Receiver (e.g., VS1838B)
IR Remote	1	Infrared remote control
Breadboard	1	Full-size or half-size breadboard
Jumper Wires	24	Male-to-male jumper wires for circuit connections
USB Cable	1	USB A to B for Arduino power/programming

Table 1: Component list for Sun Simulator (Input/Output)

B SunSimulator Script

```
def hello_world():
    print("Hello,␣world!")

#include <stdio.h>
#include <math.h>
#include <inttypes.h>
#include <IRremote.h> //https://github.com/Arduino-IRremote/Arduino-
   IRremote
#include "IRCodes.h"
#include <LiquidCrystal.h>

struct SunData {
  uint8_t r, g, b;
};

#define RED_PIN 10
#define BLUE_PIN 9
#define GREEN_PIN 8

#define IR_PIN 13

IRrecv irrecv(IR_PIN);
decode_results results;

bool powerOn = false;
bool inTimelapse = true;

float currentHour = 0.0;
unsigned long lastStepTime = 0;
unsigned long interval = 2000; // one hour simulated as 2 seconds
static uint32_t prevCode = 0;

//track duration between button presses
unsigned long lastIRTime = 0;
const unsigned long irCooldown = 300; // 300ms between presses

LiquidCrystal lcd(2,3,A5,A4,A3,A2);
int reply;

SunData dayData[24] = {
  {0, 0, 0},    // 0:00
  {10, 10, 30},   // 1:00
  {20, 10, 40},   // 2:00
  {40, 20, 50},   // 3:00
  {80, 50, 60},   // 4:00
  {120, 80, 70},   // 5:00
  {180, 120, 90}, // 6:00
  {201, 226, 255},// 7:00 overcast sky
  {255, 255, 255},// 8:00 direct sun
  {255, 255, 255},// 9:00
  {255, 255, 255},// 10:00
  {255, 255, 255},// 11:00
  {255, 255, 251},// 12:00 high noon
  {255, 255, 255},// 13:00
  {255, 255, 255},// 14:00
  {255, 255, 255},// 15:00
  {64, 156, 255},// 16:00 clear blue sky
  {255, 160, 100},// 17:00
  {255, 120, 80}, // 18:00
  {180, 80, 60},   // 19:00
  {120, 50, 40},   // 20:00
  {60, 20, 30},    // 21:00
  {20, 10, 20},    // 22:00
  {10, 10, 20}     // 23:00
};

// --- Function Prototypes ---
void newCol(uint8_t r, uint8_t g, uint8_t b);
void print_col(float hour, SunData a);
float scrubHour(int dir);
void timelapseStep();
void handleIR(uint32_t code);

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the IR receiver
  Serial.println("IR␣receiver␣ready");

  //set LED channels
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);

  inTimelapse = false;

  lcd.begin(16, 2);
  print_lcd("Turned␣Off", "POWER:␣turn␣on");
}

//LERP color helper
uint8_t lerp(uint8_t a, uint8_t b, float t) {
  return a + (b - a) * t;
}

void print_lcd(String rowA, String rowB){
  lcd.clear();

  lcd.setCursor(0, 0);        // Column 0, Row 0
  lcd.print(rowA);

  lcd.setCursor(0, 1);        // Column 0, Row 1
  lcd.print(rowB);
}
void timelapseStep() {
  if (millis() - lastStepTime >= 50 && currentHour < 23.99) {
    lastStepTime = millis();

    currentHour += (50.0 / interval); // Advance by fraction of an hour

    int h = floor(currentHour);
    int next = (h + 1) % 24;
    float t = currentHour - h; // How far between hour and next

    SunData a = dayData[h];
    SunData b = dayData[next];
    uint8_t r = lerp(a.r, b.r, t);
    uint8_t g = lerp(a.g, b.g, t);
    uint8_t b_ = lerp(a.b, b.b, t);

    SunData bData = dayData[next];
    newCol(r, g, b_);

    if (currentHour >= 23.99) {
      inTimelapse = false; // auto-stop
      Serial.println("Timelapse␣COMPLETE");
      print_lcd("Day␣has␣ended", "");
      currentHour = 0;
      turnLightOff();
    }
    else{
      print_col(currentHour, bData);
      String hourString = "Lapse␣|␣Hr:␣" + String(currentHour);
      print_lcd(hourString, "STOP␣or␣PAUSE");
    }
  }
}

void print_col(float hour, SunData a){
  Serial.print("Hour:␣");
  Serial.println(hour);
  Serial.println("R:␣" + String(a.r) + "␣G:␣" + String(a.g) + "␣B:␣" +
     String(a.b));
}


float scrubHour(int dir){ // 1 for forward, -1 for backward

  // Cannot go preceed start of day
  if(dir == 1 && currentHour > 23){
    Serial.println("Cannot␣go␣past␣end␣of␣day");
    print_lcd("End␣of␣day.", "STOP:␣restart");
    turnLightOff();
    return currentHour;
  }

  // Cannot go beyond end of day
  else if(dir == -1 && currentHour <= 0){
    Serial.println("Cannot␣go␣back␣before␣beginning␣of␣day");
    print_lcd("Beginning␣of␣day", "RIGHT:␣sun.");
    turnLightOff();
          return currentHour;
  }

  // Step light back or forward one hour
  else{
    int h = floor(currentHour);
    int next = (h + dir) % 24;
    float t = currentHour - h; // How far between hour and next

    SunData a = dayData[h];
    SunData b = dayData[next];
    uint8_t r = lerp(a.r, b.r, t);
    uint8_t g = lerp(a.g, b.g, t);
    uint8_t b_ = lerp(a.b, b.b, t);

    newCol(r, g, b_);
    int newHr = currentHour + dir;
    print_col(newHr, b);
    String hourString = "Scrub␣|␣Hr:␣" + String(newHr);
    print_lcd(hourString, "LEFT␣or␣STOP");

    return currentHour + dir;
  }
}
```

```
//Easy Turn off
void turnLightOff(){
  newCol(0,0,0);
}

//set color helper
void newCol(uint8_t r, uint8_t g, uint8_t b) {
  float clamp = 0.7;
  analogWrite(RED_PIN, r*clamp);
  analogWrite(GREEN_PIN, g*clamp);
        analogWrite(BLUE_PIN, b*clamp);
}

void loop() {

  if (IrReceiver.decode()) { //if IR receiver gets input from remote
    uint32_t code = IrReceiver.decodedIRData.decodedRawData;

    // Skip NEC repeat code
    if (code == 0xFFFFFFFF) {
      IrReceiver.resume();
      return;
    }

    // if more than 3 milliseconds have passed since repeat input...
       prevents repeat executions on long-press
    unsigned long now = millis();
    if (now - lastIRTime > irCooldown) {
      handleIR(code);
      printIRLabel(code);
      lastIRTime = now;
    }

    IrReceiver.resume();
  }

  //Continue timelapse if it has been started via Start button
  if (powerOn && inTimelapse) {
    timelapseStep();
  }
}


void handleIR(uint32_t code) {
  if (!powerOn && code != IR_POWER) return;

  switch (code) {
    case IR_POWER: // Allow light manipulation, or prevent it (turn the
        system on/off on the user side.)
      powerOn = !powerOn;
      Serial.println(powerOn ? "Power␣ON" : "Power␣OFF");
      powerOn? print_lcd("PLAY:␣timelapse", "RIGHT:␣scrub") : print_lcd
          ("Turned␣Off", "POWER:␣turn␣on");
      turnLightOff();
      break;

    case IR_PLAY: //Start timelapse
      inTimelapse = true;
      lastStepTime = millis();
      Serial.println("Timelapse␣STARTED");
      print_lcd("Timelapse␣begun", "");
      break;

    case IR_STOP: //end timelapse
      if(inTimelapse){
        inTimelapse = false;
        currentHour = 0.0;
        Serial.println("Timelapse␣STOPPED");
        print_lcd("Timelapse␣halted", "");
      }
      currentHour = 0.0;
      Serial.println("Day␣restarted");
      print_lcd("Day␣restarted", "PLAY␣or␣RIGHT");
      turnLightOff();
      break;

    case IR_PAUSE: //pause timelapse (doesn't reset hour of day)
      inTimelapse = false;
      currentHour = floor(currentHour);
      Serial.println("Timelapse␣PAUSED");
      print_lcd("PLAY␣to␣restart", "or␣RIGHT␣or␣STOP");
      break;

    case IR_PURPLE_RIGHT: //pause timelapse and skip forward one hour
      inTimelapse = false;
      if(powerOn) currentHour = scrubHour(1);
      break;

    case IR_PURPLE_LEFT: //pause timelapse and skip backward one hour
      inTimelapse = false;
      if(powerOn) currentHour = scrubHour(-1);
      break;
  }
}
```

Listing 1: SunSimulator.ino

C IR Reference Codes script

Infrared codes for the Onn Universal 4-Device Remote.

```
/**
How to use:
Add the following to the top of your .ino script:
'''
#include <IRremote.h> //https://github.com/Arduino-IRremote/Arduino-
   IRremote
#include "IRCodes.h"
'''
*/
#pragma once


#define IR_POWER        0xFD020707
#define IR_YELLOW       0xEA150707
#define IR_BLUE 0xE9160707
#define IR_RED  0x936C0707
#define IR_GREEN 0xEB140707
#define IR_REWIND       0xBA450707
#define IR_FAST_FORWARD 0xB7480707
#define IR_PLAY 0xB8470707
#define IR_PAUSE        0xB54A0707
#define IR_MENU 0xE51A0707
#define IR_STOP 0xB9460707
#define IR_RECORD       0xB6490707
#define IR_HOME 0x86790707
#define IR_GUIDE        0xB04F0707
#define IR_PURPLE_LEFT  0x9A650707
#define IR_PURPLE_RIGHT 0x9D620707
#define IR_PURPLE_UP    0x9F600707
#define IR_PURPLE_DOWN  0x9E610707
#define IR_THIN_LEFT    0xA7580707
#define IR_INFO 0xE01F0707
#define IR_VOLUME_UP    0xF8070707
#define IR_VOLUME_DOWN  0xF40B0707
#define IR_CHANNEL_UP   0xED120707
#define IR_CHANNEL_DOWN 0xEF100707
#define IR_P_CH 0xEC130707
#define IR_MUTE 0xF00F0707
#define IR_1    0xFB040707
#define IR_2    0xFA050707
#define IR_3    0xF9060707
#define IR_4    0xF7080707
#define IR_5    0xF6090707
#define IR_6    0xF50A0707
```

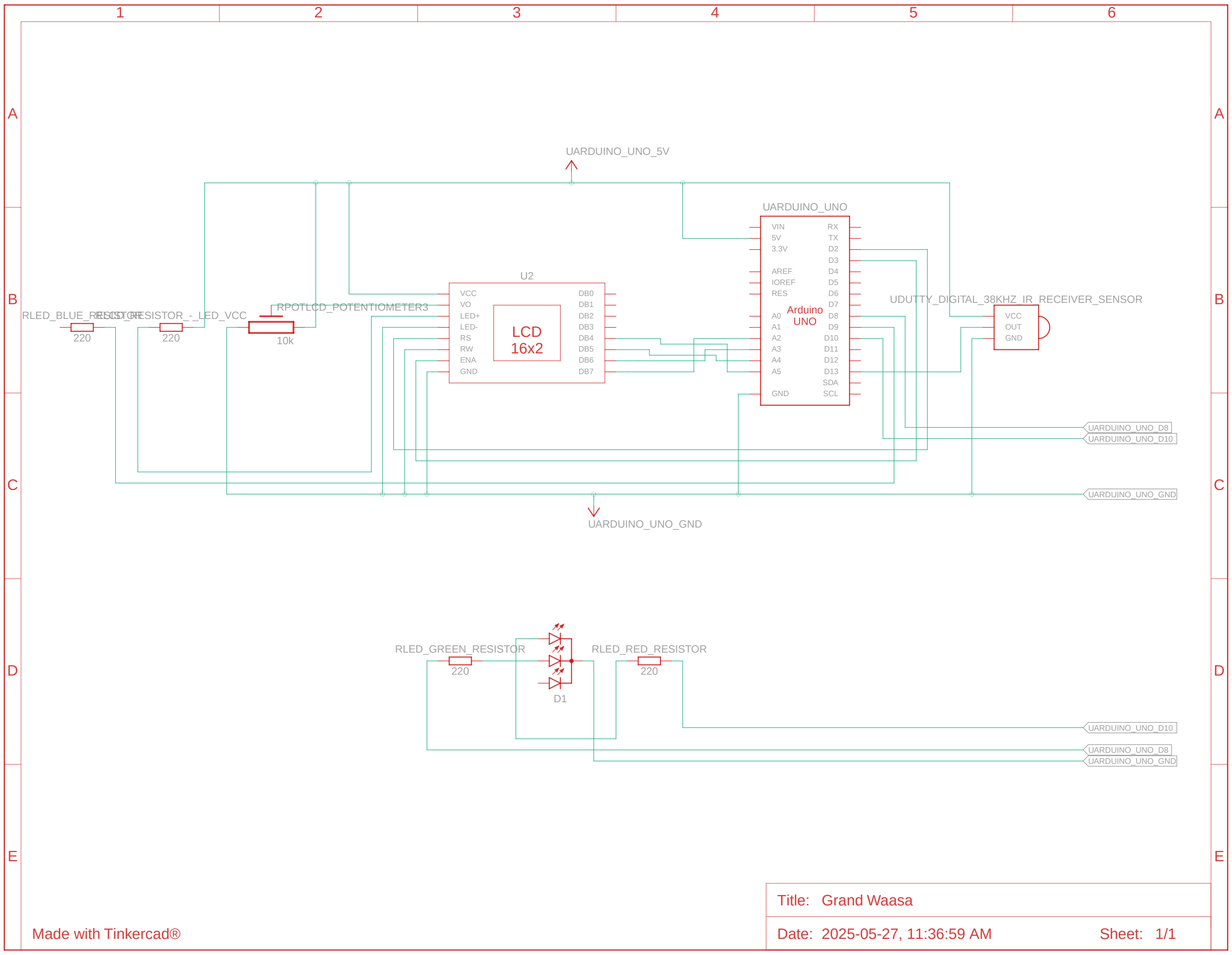
```
#define IR_7    0xF30C0707
#define IR_8    0xF20D0707
#define IR_9    0xF10E0707
#define IR_BOTTOM_DOT   0xDC230707
#define IR_0    0xEE110707
#define IR_ENTER        0x97680707


void printIRLabel(uint32_t code) {
  switch(code){
    case IR_POWER: Serial.println("Power␣button"); break;
    case IR_YELLOW: Serial.println("Yellow␣button"); break;
    case IR_BLUE: Serial.println("Blue␣button"); break;
    case IR_GREEN: Serial.println("Green␣button"); break;
    case IR_REWIND: Serial.println("Rewind"); break;
    case IR_FAST_FORWARD: Serial.println("Fast␣Forward"); break;
    case IR_PLAY: Serial.println("PLAY"); break;
    case IR_PAUSE: Serial.println("PAUSE"); break;
    case IR_MENU: Serial.println("IR_MENU"); break;
    case IR_STOP: Serial.println("IR_STOP"); break;
    case IR_RECORD: Serial.println("IR_RECORD"); break;
    case IR_HOME: Serial.println("IR_HOME"); break;
    case IR_GUIDE: Serial.println("IR_GUIDE"); break;
    case IR_PURPLE_LEFT: Serial.println("IR_PURPLE_LEFT"); break;
    case IR_PURPLE_RIGHT: Serial.println("IR_PURPLE_RIGHT"); break;
    case IR_PURPLE_UP: Serial.println("IR_PURPLE_UP"); break;
    case IR_PURPLE_DOWN: Serial.println("IR_PURPLE_DOWN"); break;
    case IR_THIN_LEFT: Serial.println("IR_THIN_LEFT"); break;
    case IR_INFO: Serial.println("IR_INFO"); break;
    case IR_VOLUME_UP: Serial.println("IR_VOLUME_UP"); break;
    case IR_VOLUME_DOWN: Serial.println("IR_VOLUME_DOWN"); break;
    case IR_CHANNEL_UP: Serial.println("IR_CHANNEL_UP"); break;
    case IR_CHANNEL_DOWN: Serial.println("IR_CHANNEL_DOWN"); break;
    case IR_P_CH: Serial.println("IR_P_CH"); break;
    case IR_MUTE: Serial.println("IR_MUTE"); break;
    case IR_BOTTOM_DOT: Serial.println("IR_BOTTOM_DOT"); break;
    case IR_0: Serial.println("IR_0"); break;
    case IR_ENTER: Serial.println("IR_ENTER"); break;
    case IR_1: Serial.println("#1"); break;
    case IR_2: Serial.println("#2"); break;
    case IR_3: Serial.println("#3"); break;
    case IR_4: Serial.println("#4"); break;
    case IR_5: Serial.println("#5"); break;
    case IR_6: Serial.println("#6"); break;
    case IR_7: Serial.println("#7"); break;
    case IR_8: Serial.println("#8"); break;
    default:
      Serial.print("Unknown␣IR␣code:␣0x");
      Serial.println(code, HEX);
      break;
  }
}
```

Listing 2: IRCodes.h.

Sunlight Simulator Schematic

The following document shows the schematic view of the input and output system for the Sun Simulator.


1
2
3
4
5
6
A
B
C
D
E
UARDUINO_UNO_5V
UARDUINO_UNO
VIN
5V
3.3V
AREF
IOREF
RES
A0
A1
A2
A3
A4
A5
GND
RX
TX
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
SDA
SCL
Arduino UNO
U2
VCC
VO
LED+
LED-
RS
RW
ENA
GND
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
LCD 16x2
RLED_BLUE_RESISTOR
RLCD_RESISTOR_-_LED_VCC
RPOTLCD_POTENTIOMETER3
220
220
10k
UDUTTY_DIGITAL_38KHZ_IR_RECEIVER_SENSOR
VCC
OUT
GND
UARDUINO_UNO_D8
UARDUINO_UNO_D10
UARDUINO_UNO_GND
UARDUINO_UNO_GND
RLED_GREEN_RESISTOR
220
D1
RLED_RED_RESISTOR
220
UARDUINO_UNO_D10
UARDUINO_UNO_D8
UARDUINO_UNO_GND
Title: Grand Waasa
Date: 2025-05-27, 11:36:59 AM
Sheet: 1/1
Made with Tinkercad®