U Compass

Borui Wang
DEA 6210
Oct. 2024

Introduction

The inspiration for this project came from an image from the class, titled "Trying to Remember," where a woman stands in a doorway, puzzled, asking herself, "Hmm... Now why did I come in here again?" This humorous yet relatable moment resonated deeply with me, as I've often found myself easily distracted, struggling to focus on more than one task at a time. This sparked the idea to create a device that helps people stay on track and avoid forgetfulness, especially in the midst of everyday distractions.

Fig. Here(1957)

As I observed how people operate in their daily lives, I noticed how much we rely on spatial memory—the tendency to associate certain tasks or reminders with physical locations. For example, we often remember to grab an item simply because we walk past a specific spot. This insight became the core of my design approach, as I wanted to build something that can provide interactions as natural to us as possible.

In thinking about a physical form, I was drawn to the ancient Chinese compass, commonly known as the "south pointer." One of the earliest human inventions used to navigate the world, the south pointer symbolized the human desire to find direction and guidance. I saw the

compass as a perfect metaphor for a reminder device, guiding people not through the world, but through all the distractions happening in their daily routines.

Fig. Replication of the ancient Chinese Compass, “South Pointer” or “司南” [1]

[1]: CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=553022

Scenario

Time: A normal summer day in 1999
Location: An apartment in northern Ithaca
Characters:

- **Erika:** 46, a housewife whose husband is at work.
- **Havi:** A playful black cat, constantly in need of food and attention.

It’s a warm summer afternoon in 1999, and Erika has just returned home from a long grocery trip, carrying several heavy bags filled with fresh meat and vegetables. Sweat drips down her face as she kicks the door closed behind her, eager to wash away her tiredness with a quick shower before preparing dinner for herself and her husband.

As she puts the groceries down in the kitchen, she hears the meow of her black cat, Havi, reminding her it's feeding time. But before feeding Havi, she remembers she needs her bath towel, which she keeps in the closet inside her bedroom. However, the bathroom to shower, and the kitchen area that she's standing right now, is in the living area, separated from the closet with doors and turns.

Erika is no stranger to this situation. Distracted by her cat, or a phone call from her friend, or even a fur ball on the floor that needs sweeping, she’s forgotten her towel too many times. She’s been in the bathroom, wet, with nothing but face tissues to dry herself. But today, she has a solution—her trusty new smart home decoration.

Sitting on the table in her living room is a unique device, resembling an ancient Chinese compass, with a spoon-like feature on top. Erika walks over to the device and gently turns the spoon to point toward her bedroom closet. The spoon vibrates softly, signaling that her reminder is set. Now, she can go feed Havi without worrying about forgetting her towel.

Option 1: Dismiss

Once Havi is happily fed, Erika starts heading toward the bathroom, her thoughts now occupied with dinner plans and laundry. She’s already forgotten about the towel. As she walks past the living room table, a soft beeping sound catches her attention. The spoon on the compass-like device is pointing toward the closet. “Ah, the bath towel,” she recalls, realizing she nearly forgot it again. She turns the spoon clockwise, dismissing the reminder, heads to her closet to grab the towel, and then enjoys a relaxing shower without any issues.

Option 2: Snooze

Earlier, when Erika had passed by to pick Havi's favorite toy, the reminder device had also beeped, but she wasn’t ready for her shower yet, so she turned the spoon counterclockwise to snooze the reminder. The device stayed silent until she passed by again, when she was ready for her shower.

Thanks to the smart reminder device with its ancient compass design, Erika avoids another moment of forgetfulness, enjoying a smooth and relaxing afternoon after her busy day.

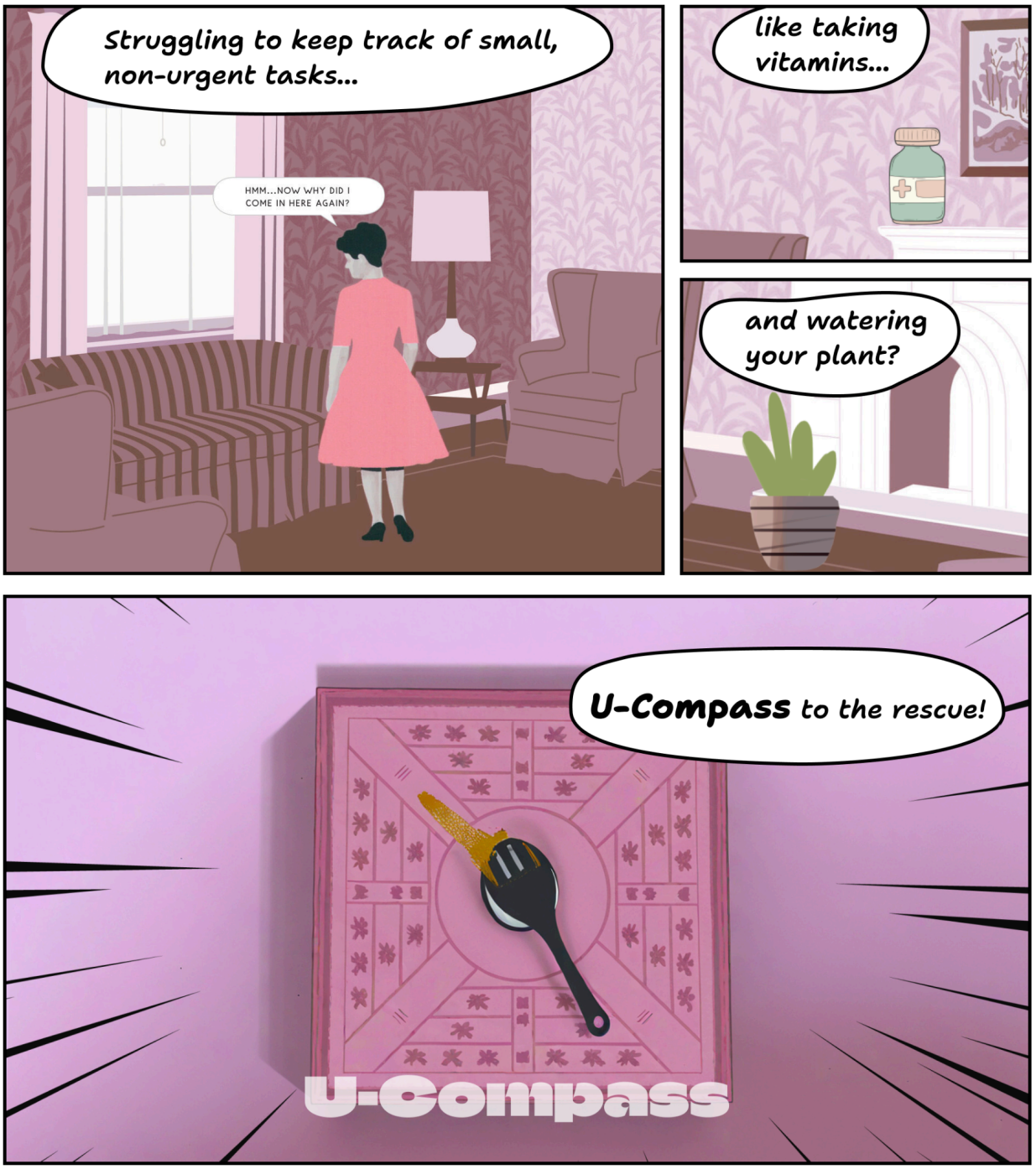

Fig. A comic style illustration I made for the scenario

Operation

This project uses a DMS-MG90-A servo motor with feedback to set and trigger reminders. The servo is controlled by an Arduino, using a light sensor and a buzzer for user feedback. The servo's position is adjusted by the user to set a reminder. If the light sensor detects a threshold light level or if the reminder is triggered, the system vibrates the servo and sounds the buzzer as an alarm. The user can dismiss or snooze the reminder by rotating the servo to specific positions. A built-in calibration routine maps the servo feedback to precise positions, allowing for accurate positioning and feedback.

Key functionalities include:

- **Reminder Setting**: Users rotate the servo to a position where the reminder is set.
- **Alarm Trigger**: When a reminder is active and the light sensor is triggered, the servo vibrates and the buzzer sounds.
- **Snooze/Dismiss**: The user can snooze or dismiss the reminder by rotating the servo to different positions.

Construction

The device is constituted of two parts, an inner housing with all the controlling electronics, and an outer housing with power supply. The inner housing is constructed with corrugated plastic provided from class. It has an opening cut on top with the servo mounted from the inside, as shown in the figure below.

Fig. DFRobot servo mounted on plastic panel

The buzzer is put inside the plastic housing while the light sensor is placed outside and connected to the Arduino through a grove cable via a cut hold as shown in the lower right of the figure below.

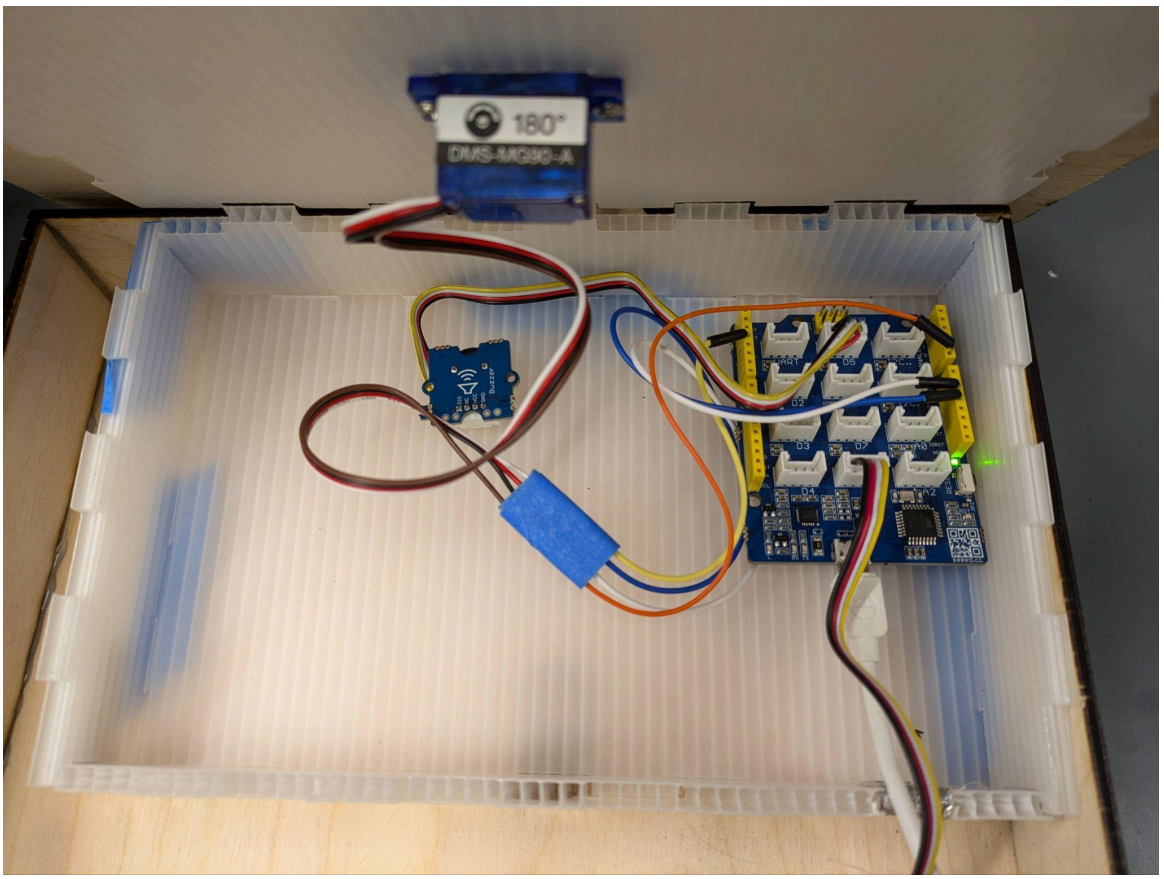

Fig. Internal of the inner housing, with the cutout shown in the lower right corner

The outer housing, on the other hand, is made from plywood and cut with a laser cutter in Rhodes Hall. The top panel is cut and engraved according to the ancient Chinese compass. While the bottom half is designed and made with Casemaker, which provides online case design that can be cut with a laser cutter. The top panel is slightly larger than the bottom half, to make it more similar to the traditional Chinese compass.

On top of the outer housing, a black spoon is placed that serves as the indicator and main interaction point of the design. The spoon mimics the shape of a traditional Chinese soup spoon, the same one that is placed on the ancient Chinese compass. The spoon is designed to have a grove on the bottom to have the servo hub snap in. It is 3D printed in Rhodes Hall. Initially 2 colors of the spoon is tested – gold and black, since the true color of the compass is in between the two, with a dark copper color. In the end black is chosen as it is more aesthetically appealing and closer to the true compass color. Below is a side by side comparison of the two versions.

Fig. Comparison of the black and gold spoon

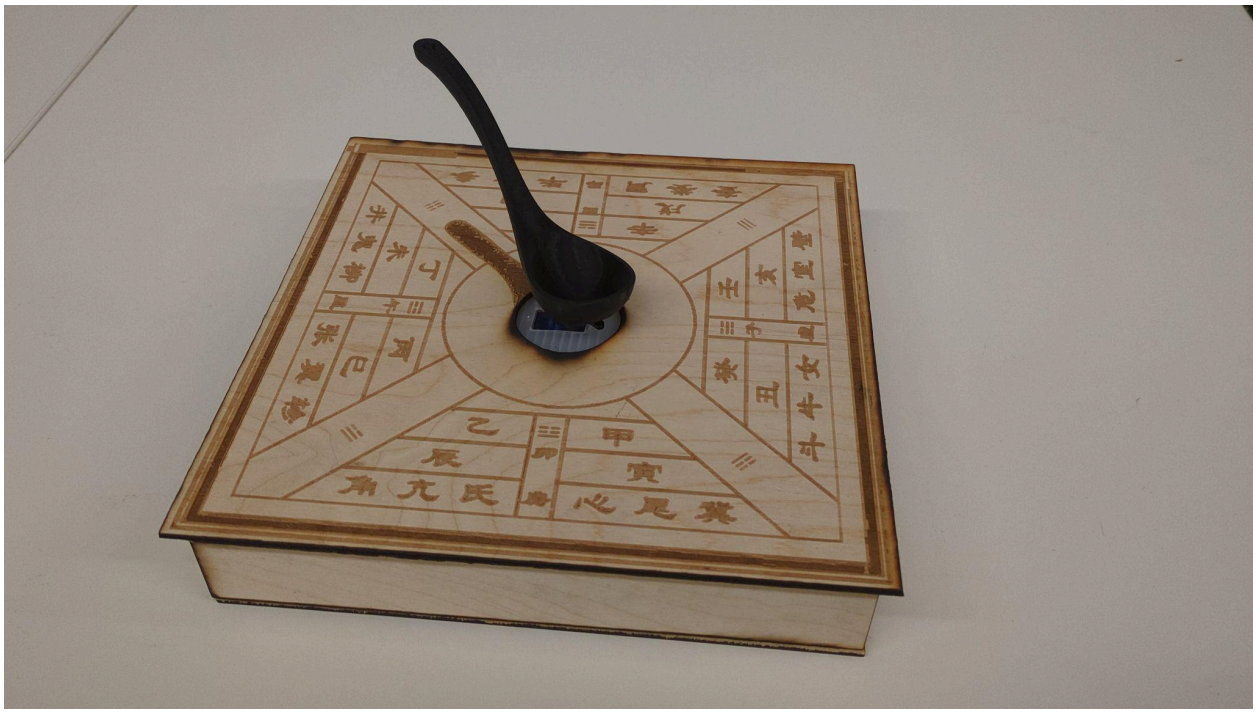

Fig. Isometric view of U Compass

Implementation

List of Electronics

- **DMS-MG90-A Servo Motor**: Provides the rotational mechanism for setting reminders and giving tactile feedback through vibration.
- **Light Sensor (Grove - Light Sensor)**: Detects changes in ambient light to trigger the alarm.
- **Buzzer (Grove - Buzzer)**: Provides audio feedback when the reminder is triggered or dismissed.
- **Seeed Studio Grove Arduino Kit**: Provides modular components and an Arduino board for controlling the system.
- **Arduino Uno**: The microcontroller used to control the servo, light sensor, and buzzer.
- **Wires and Connectors**: To connect the servo, sensors, and buzzer to the Arduino.

Electrical Connections

The components are connected as follows:

- **Servo Control Pin**: Pin 3 on the Arduino is used for sending control signals to the DMS-MG90-A servo.
- **Servo Feedback Pin**: Pin A0 is connected to the analog feedback pin of the DMS-MG90-A to read the servo's current position.
- **Light Sensor Pin**: Pin A6 is connected to the light sensor to read the ambient light levels.
- **Buzzer Pin**: Pin 5 is connected to the buzzer to produce sounds when the alarm or reminder is activated.
- **Power and Ground Connections**: All components are powered through the Arduino’s 5V and ground pins, with the servo motor powered directly from the 5V rail for adequate current supply.

The following is a wiring schematic of U Compass, please notice that the connection between Arduino and the grove kits is not depicted here, as it is not possible to do with the schematic. However, the light sensor grove and buzzer grove are connected to A6 and D5 to the Arduino grove kit, with the grove connector. The connection to the servo motor still requires normal cables to connect.

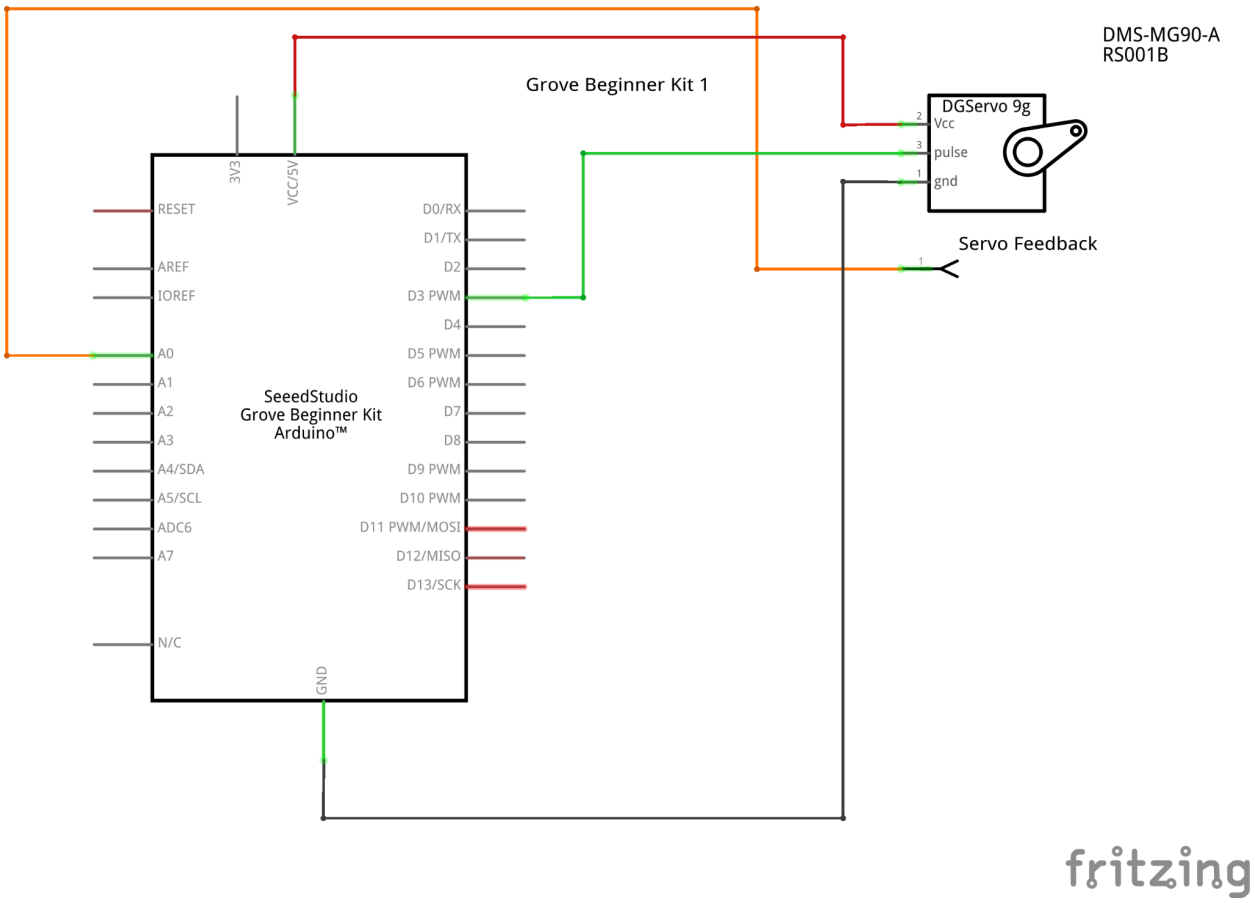

Fig. Wiring Schematic for U Compass

U Compass is powered by a small power bank through the micro-USB port to the Arduino. The power bank and cable is placed between the outer and inner housing toward the back.

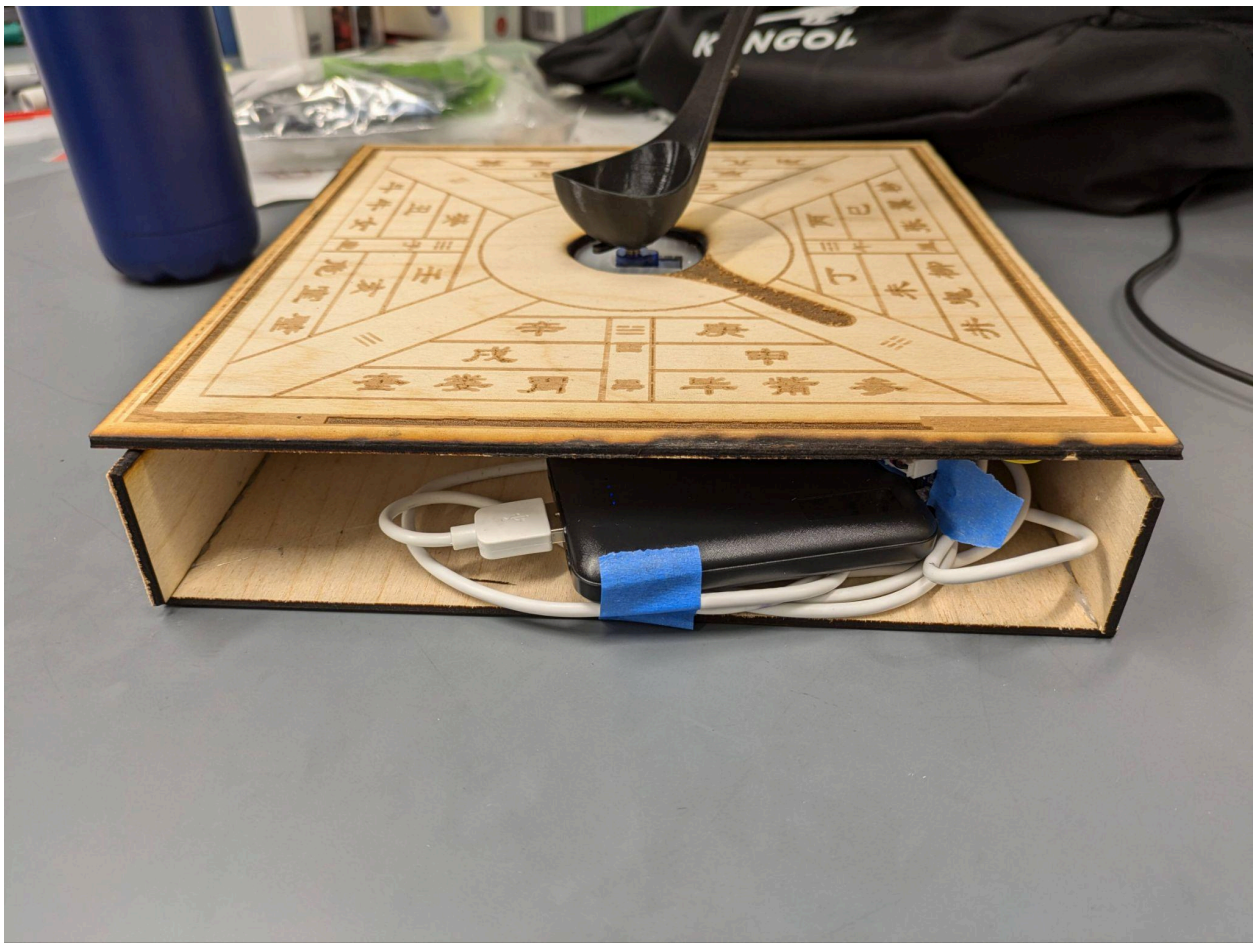

Fig. Back view of the U Compass, the power bank and micro-USB cable connecting to the Arduino

Results

The final design successfully achieved both the intended functionality and aesthetic expectations. The servo motor responds to movements, providing timely reminders when triggered by light sensor input. The servo motor also remains unpowered and is able to be turned by the user when it is not required to move. The vibrating feedback and buzzer sound are activated based on user interactions at the correct time. Some components, like the servo placement and servo power mechanism, underwent several iterations to enhance responsiveness. While the current servo performs adequately, future improvements could include a 360 degree servo or a stepper motor for smoother motion and a wider range of feedback. Overall, U-Compass met its goal of providing spatial reminders.

The video to U Compass can be accessed here: https://youtu.be/nn1wCwkXmvs

Future Work

Currently, the servo motor used in this project is a DMS-MG90-A, which has a limited range of motion restricted to 180 degrees. For future iterations, a 360-degree motor or stepper motor could be integrated to allow for continuous rotation and more precise control over the reminder mechanism. A stepper motor, in particular, would provide finer increments of movement, allowing for smoother user interactions and expanded functionality, such as setting more accurate reminders through rotation.

Additionally, the current reminder system uses a simple buzzer to generate audio feedback. While functional, the buzzing sound may not be ideal for all users or environments. Future improvements could involve replacing the buzzer with a speaker to deliver more customizable and pleasant sound clips, potentially with varied tones for a better user experience.

These enhancements would improve both the versatility of the motion system and the quality of the auditory feedback, making the system more user-friendly and effective.

Code

```
#include <Servo.h>  // Library for controlling the servo
#include <Wire.h>

Servo myServo;                    // Create a Servo object
const int lightSensorPin = A6;    // Pin for the light sensor
const int buzzerPin = 5;          // Pin for the buzzer
const int servoPin = 3;           // Pin for the servo control signal
const int servoFeedbackPin = A0;  // Pin for reading the analog
feedback from the servo

int lightThreshold = 6;               // Threshold for light sensor
(adjust based on environment)
int prevServoPos = 0;                 // Store the previous position of
the servo
int currentServoPos = 0;              // Store the current position of
the servo
int reminderServoPos = -1;            // Store the position of the servo
when the user set the reminder
bool snooze = false;                  // Flag to snooze the reminder
unsigned long lastMovementTime = 0;   // Time of last movement
bool moving = false;                  // Flag to indicate if the servo
is vibrating
bool reminderSet = false;             // Flag to indicate if the
reminder is active
bool alarming = false;                // Flag to check if the reminder
is currently triggered
int idleServoPos = 80;                // When there is nothing happening

int minFeedbackValue = 0;     // To store the minimum feedback value
int maxFeedbackValue = 1023;  // To store the maximum feedback value

void setup() {
 pinMode(lightSensorPin, INPUT);
 pinMode(buzzerPin, OUTPUT);
 digitalWrite(buzzerPin, LOW);       // Ensure buzzer is off at start
 pinMode(servoFeedbackPin, INPUT);  // Feedback pin from the servo
 Serial.begin(9600);
 calibrateServo();
}

void loop() {
```

```
// Read the current position of the servo from the feedback pin
currentServoPos = readServoPosition();
// Update previous servo position
Serial.println(String(prevServoPos) + ", " + String(currentServoPos));

// Reset and dismiss if servo is in invalid position
if (currentServoPos < 0 && !moving) {
  // Dismissed
  tone(buzzerPin, 50);  // Play a sound on the buzzer
  delay(250);
  noTone(buzzerPin);
  delay(250);
  tone(buzzerPin, 50);  // Play a sound on the buzzer
  delay(250);
  noTone(buzzerPin);
  delay(250);
  dismissReminder();
}

// Detect if the user is moving the servo
if (abs(currentServoPos - prevServoPos) > 5 && !moving) {
  lastMovementTime = millis();  // Update the last movement time
  moving = true;                // Reset the vibrating flag
}

// Check if the user stopped moving the servo for 3 seconds
if (millis() - lastMovementTime > 3000 && moving) {
  vibrateServo(currentServoPos);  // Provide vibration feedback
  moving = false;                 // Set vibrating flag
  prevServoPos = currentServoPos;
  reminderSet = true;
  reminderServoPos = currentServoPos;
}

if (alarming) {
  // Check if the user turns the servo to stop the reminder
  if (prevServoPos - currentServoPos > 10) {  // Dismissed
    tone(buzzerPin, 50);                      // Play a sound on the
buzzer
    delay(250);
    noTone(buzzerPin);
    delay(250);
```

```
      tone(buzzerPin, 50);  // Play a sound on the buzzer
      delay(250);
      noTone(buzzerPin);
      delay(250);
      tone(buzzerPin, 50);  // Play a sound on the buzzer
      delay(750);
      noTone(buzzerPin);
      dismissReminder();
      Serial.println("Dismissed");
      delay(2000);

    } else if (currentServoPos - prevServoPos > 10) {  // Snoozed
      snoozeReminder();
      reminderSet = true;
      alarming = false;
      moving = false;
      Serial.println("Snoozed");
      delay(1000);
      currentServoPos = readServoPosition();
      prevServoPos = currentServoPos;
    }
  }
  if (reminderSet) {
    // // Read the light sensor value
    int lightLevel = analogRead(lightSensorPin);
    Serial.println("Light read: " + String(lightLevel));

    // Check if the light sensor is covered
    if (lightLevel < lightThreshold || alarming) {
      vibrateServo(reminderServoPos);  // Shake the servo as a reminder
      playReminder();
      prevServoPos = reminderServoPos;
      alarming = true;
      Serial.println("Alarming!");
      delay(1500);
    }
  }
  delay(500);
}

void calibrateServo() {
  Serial.println("Starting automatic servo calibration...");
```

```
 // Attach the servo
 myServo.attach(servoPin);

 // Move servo to the minimum position (0 degrees)
 Serial.println("Moving servo to 0 degrees...");
 myServo.write(0);
 delay(1000);                                    // Give time for the
servo to move to the position
 minFeedbackValue = analogRead(servoFeedbackPin);  // Read and store
the minimum feedback value
 Serial.print("Min Feedback Value: ");
 Serial.println(minFeedbackValue);

 // Move servo to the maximum position (180 degrees)
 Serial.println("Moving servo to 180 degrees...");
 myServo.write(180);
 delay(1000);                                    // Give time for the
servo to move to the position
 maxFeedbackValue = analogRead(servoFeedbackPin);  // Read and store
the maximum feedback value
 Serial.print("Max Feedback Value: ");
 Serial.println(maxFeedbackValue);
 myServo.write(idleServoPos);
 delay(500);
 // Detach the servo to save power
 myServo.detach();
 delay(500);
 prevServoPos = readServoPosition();
 currentServoPos = prevServoPos;

 Serial.println("Calibration complete!");
}

void resetServo() {
 myServo.attach(servoPin);  // Attach the servo before vibrating
 myServo.write(0);
 delay(800);
 myServo.detach();  // Detach the servo after vibrating
}

void moveServo(int servoPos) {
 myServo.attach(servoPin);  // Attach the servo before vibrating
 myServo.write(servoPos);
```

```
 delay(200);
 myServo.detach();  // Detach the servo after vibrating
}

// Function to read the servo position from the feedback pin
int readServoPosition() {
 int feedbackValue = analogRead(servoFeedbackPin);
// Read the analog feedback from the servo
 int position = map(feedbackValue, minFeedbackValue, maxFeedbackValue,
0, 180);  // Map it to the servo position (0-180 degrees)
 Serial.println("Pos: " + String(position));
 return position;
}

void vibrateServo(int servoPos) {
 myServo.attach(servoPin);      // Attach the servo before vibrating
 for (int i = 0; i < 2; i++) {  // Vibrate the servo back and forth
   myServo.write(servoPos + 5);
   delay(200);
   myServo.write(servoPos - 5);
   delay(200);
 }
 myServo.write(servoPos);
 delay(500);
 myServo.detach();  // Detach the servo after vibrating
}

void playReminder() {
 tone(buzzerPin, 200);  // Play a sound on the buzzer
 delay(500);
 noTone(buzzerPin);
 delay(500);
}

void stopReminder() {
 noTone(buzzerPin);  // Stop the buzzer sound
}

void snoozeReminder() {
 snooze = true;          // Set the snooze flag
 tone(buzzerPin, 150);  // Play a sound on the buzzer
 delay(250);
 noTone(buzzerPin);
```

```
 delay(250);
 tone(buzzerPin, 150);  // Play a sound on the buzzer
 delay(250);
 noTone(buzzerPin);
 moveServo(180);
 delay(500);
 moveServo(reminderServoPos);
 delay(500);
}

void dismissReminder() {
 stopReminder();
 reminderSet = false;
 alarming = false;
 resetServo();
 moveServo(idleServoPos);
 prevServoPos = idleServoPos;
 currentServoPos = idleServoPos;
 reminderServoPos = -1;
 lastMovementTime = millis();
 moving = false;
 Serial.println("Dismissed");
}
```