The Call of Gordon Cooper PLC

What did we do leading up to the final project to ensure a successful outcome?

1. We brainstormed ideas for our display. We thought about what kind of interactive trigger we can use to start the show and how we can incorporate motion, light, and sound components.

2. We planned the dimensions of our display to fit within the maximum size of 3ft x 3ft x 3ft. This will help ensure that our project stays within the given restrictions.

3. We Considered using the provided programmable logic controller kit to control the different components of our display. We made sure to document our code well, following the SOLID principles in Java to ensure readability and maintainability.

4. To meet the requirement of pausing and stopping the display, we can incorporate buttons or switches that control the flow of the show.

5. We Assembled the different components for our display, making sure to include at least one motion component, one light component, and one sound component.

What are our Components?

Motion Component: We Created a rotating space capsule and moon that moves in a circular path. We used a small motor to achieve this effect.

Light Component: We Added LED lights to simulate stars around our display. We programmed them to twinkle to create a mesmerizing effect.

Sound Component:. We recorded a narration telling the audience what the PLC is about along with the history of the mission. We got the audio onto an MP3 that transmitted the sound to a speaker.

Interactive Trigger: We designed a button that visitors can press to start the display.

What were some of our initial theme ideas?

1. Enchanted Forest: twinkling lights, moving woodland creatures, and soothing nature sounds.

2. Space Adventure: Rotating Earth , blinking stars, and MP3 speaker narrating Gordon Coopers Space mission, project mercury.

3. Underwater Wonderland: shimmering lights, floating sea creatures, and calming ocean sounds.

4. Circus Extravaganza: colorful lights, spinning acrobats, and lively circus music.

5. Winter Wonderland: sparkling lights, snowflakes, and festive music.

6.Mardi Gras: vibrant colors, masks, and lively music. We could create a display that captures the spirit of Mardi Gras with colorful lights, animated masks, and upbeat music.

7. Lunar New Year: we could design a display inspired by the Year of the Tiger. Incorporate elements like red and gold decorations, tiger imagery, and traditional Chinese music.

8.Winter Sports: Display winter sports like skiing, snowboarding, or ice skating. We could create a display that showcases the excitement and thrill of these activities with moving figures, twinkling lights, and a soundtrack of winter sports sounds.

9.Winter Olympics: February often coincides with the Winter Olympics. You could design a display that celebrates the spirit of athleticism and competition with moving sports equipment, flags of different countries, and energetic sports-themed music.

10. Amusement Park, Roller Coaster, Carousel, moving horses, carnival music string lights.

We decided to go with the space adventure theme for our display. We thought it would be super cool to create an immersive experience that takes people on a journey through the final flight of project mercury. We will make a small replica of the Faith 7 spacecraft that orbited the Earth 22 times in 1-1/2 days. We will include Gordon Cooper in our diorama as a narrating puppet that tells us the story of his mission and the experiments he conducted up in space. We will Include Earth and the moon, and twinkling stars. We wanted to add an extra layer of historical significance that ties into our school by paying homage to this incredible milestone in space exploration.

Materials Needed:

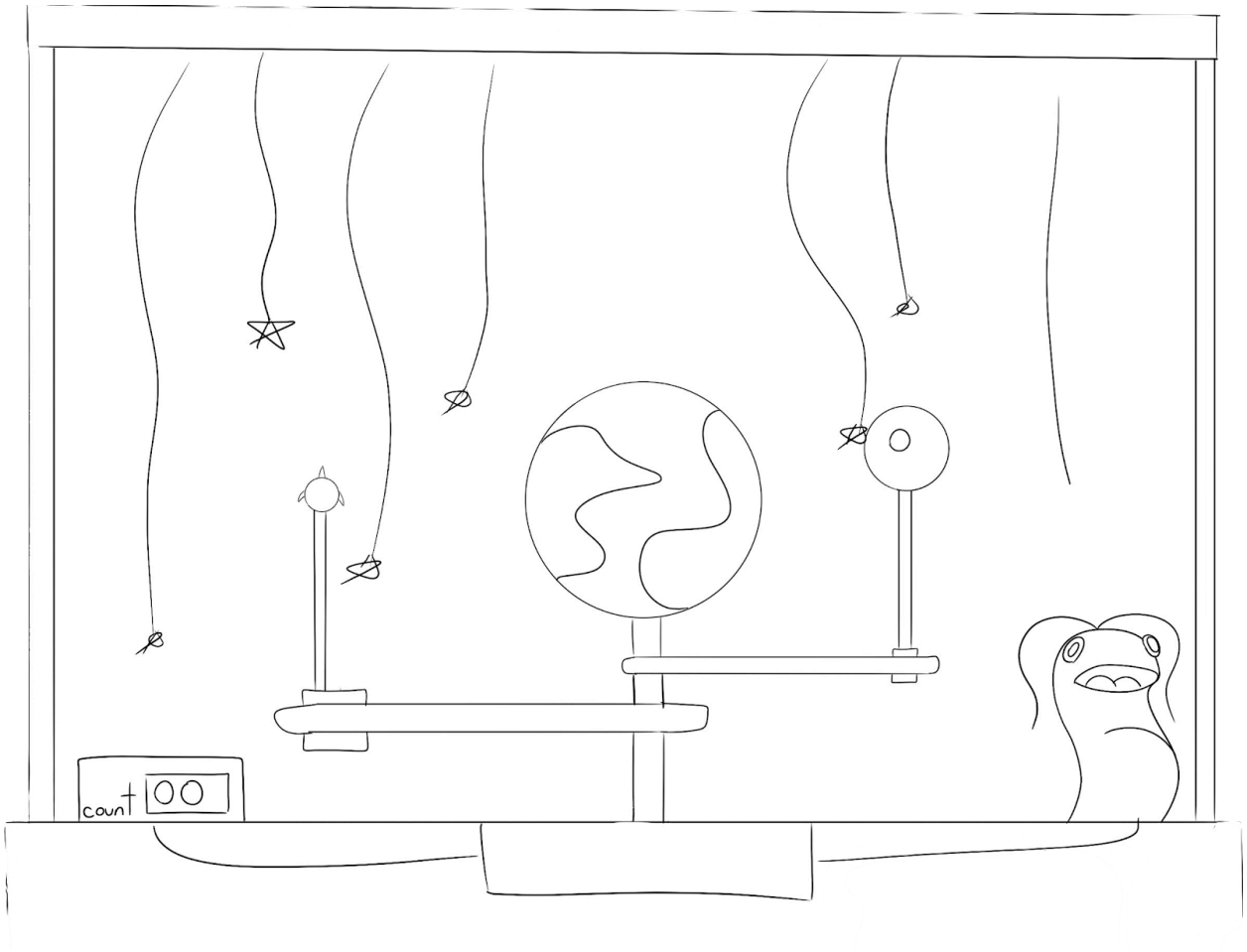
Styrofoam Ball: 2.5”- Amazon
Wood dowels:- Amazon
Paint: Black,Green,White,Red, Blue-Lowes
Air Dry Clay: Amazon
LED Fairy lights: Amazon
Felt:White, Brown,Peach/tan,Red- Amazon
Foil:Amazon
Beach Ball:Amazon
Plywood:¼ inch 31x31
Motor:Direct Current
Batteries:Double A
Speaker:Pin plug/Aux- Amazon
Hot Glue

Total Material Expenses

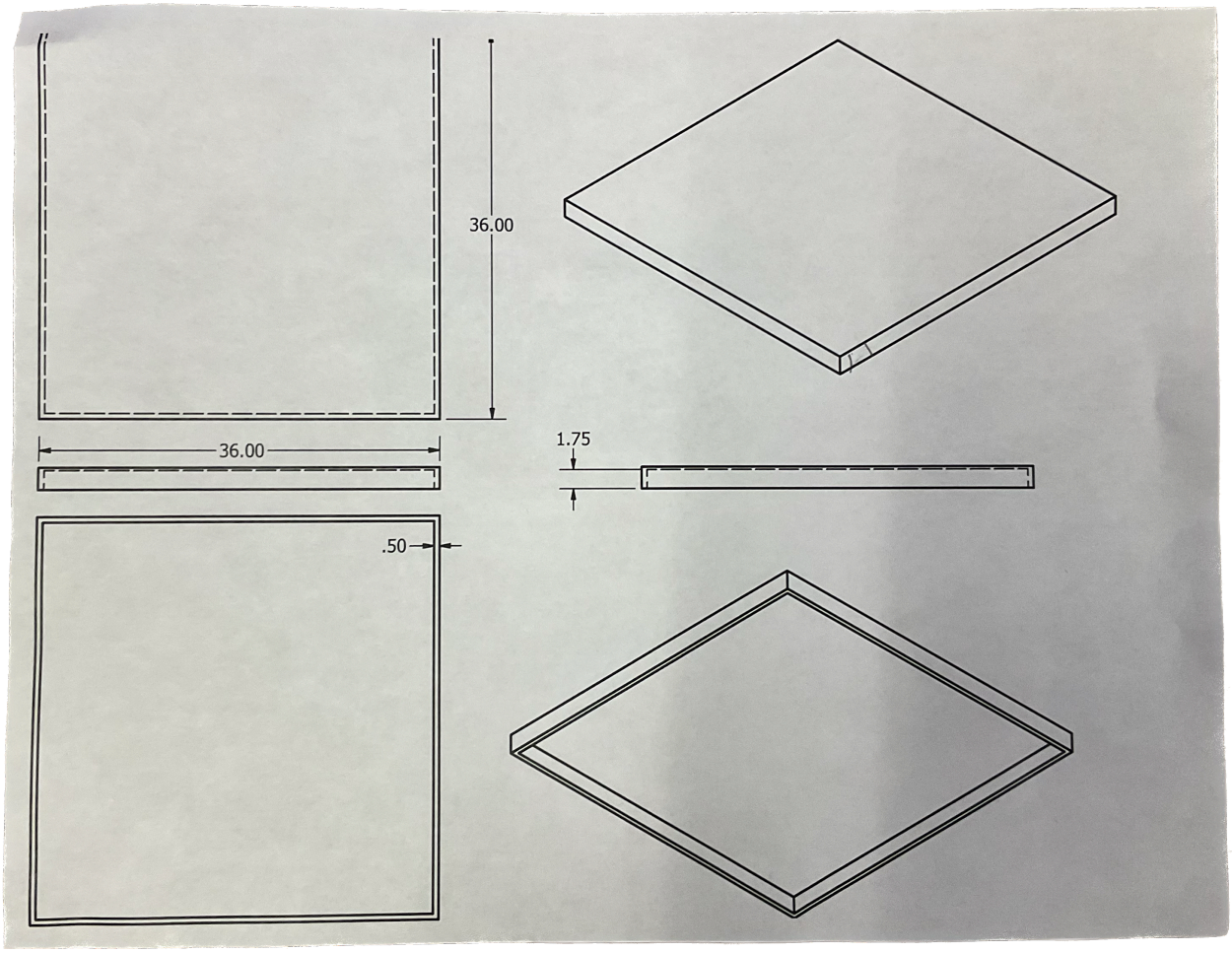
Styrofoam Ball:$11.99

Wood dowels:$5.49
Air Dry Clay:$10.62
LED Fairy Lights:$9.99
Felt:$8.99
Foil:$2.79
Speaker:$13.85
Total
$63.72

Rough Draft Sketch

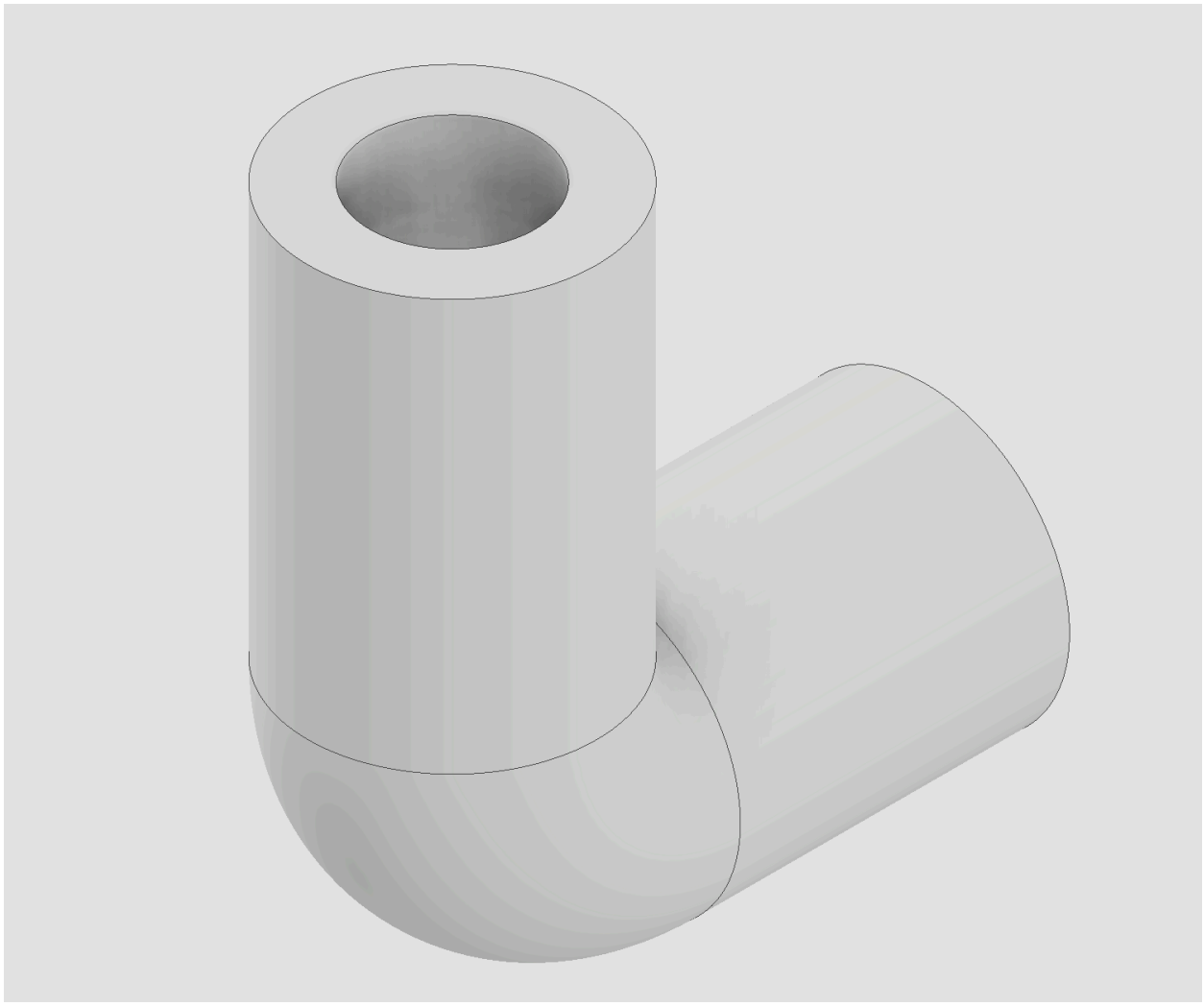


A.Plywood Base-Laser Cut



B. Ship Capsule-3D printed

C.Dowel Rod Elbow- 3D Printed



D.Motor Gear- 3D Printed

Logical Operation and Physical Construction:
Our code was also originally meant to have 2 motors but some complications with our driver made us switch to 1.
First we made the lights and got that code working then we moved on to the motors and finally made the button to pause the code.

The many different ideas come together to give a great example of space and an incredible and informative show.

If we had more time we would have made a speaker and run more bug fixes in our button code.

Robot_Car: this is a file that we call upon in our main code. One of our team members had written it previously so we made some tweaks and reused the code

Main: This is our main code that runs all of our lights and motors. It was meant to have a sound system but we ran out of time before implementing it.

```
from machine import Pin
from machine import PWM
import utime

'''
Class to represent our robot car
'''
class RobotCar:
    MAX_DUTY_CYCLE = 65535
    MIN_DUTY_CYCLE = 0
    def __init__(self, motor_pins, frequency=20000):
        self.left_motor_pin1 = PWM(Pin(motor_pins[0], mode=Pin.OUT))
        self.left_motor_pin2 = PWM(Pin(motor_pins[1], mode=Pin.OUT))
        self.right_motor_pin1 = PWM(Pin(motor_pins[2], mode=Pin.OUT))
        self.right_motor_pin2 = PWM(Pin(motor_pins[3], mode=Pin.OUT))
        # set PWM frequency
        self.left_motor_pin1.freq(frequency)
        self.left_motor_pin2.freq(frequency)
        self.right_motor_pin1.freq(frequency)
        self.right_motor_pin2.freq(frequency)

        self.current_speed = RobotCar.MAX_DUTY_CYCLE
```

```
    def move_forward(self):
        self.left_motor_pin1.duty_u16(self.current_speed)
        self.left_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

        self.right_motor_pin1.duty_u16(self.current_speed)
        self.right_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

    def move_backward(self):
        self.left_motor_pin1.duty_u16(RobotCar.MIN_DUTY_CYCLE)
        self.left_motor_pin2.duty_u16(self.current_speed)

        self.right_motor_pin1.duty_u16(RobotCar.MIN_DUTY_CYCLE)
        self.right_motor_pin2.duty_u16(self.current_speed)

    def turn_left(self):
        self.left_motor_pin1.duty_u16(self.current_speed)
        self.left_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

        self.right_motor_pin1.duty_u16(RobotCar.MAX_DUTY_CYCLE)
        self.right_motor_pin2.duty_u16(RobotCar.MAX_DUTY_CYCLE)

    def turn_right(self):
        self.left_motor_pin1.duty_u16(RobotCar.MAX_DUTY_CYCLE)
        self.left_motor_pin2.duty_u16(RobotCar.MAX_DUTY_CYCLE)

        self.right_motor_pin1.duty_u16(self.current_speed)
        self.right_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

    def stop(self):
        self.left_motor_pin1.duty_u16(RobotCar.MIN_DUTY_CYCLE)
        self.left_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

        self.right_motor_pin1.duty_u16(RobotCar.MIN_DUTY_CYCLE)
        self.right_motor_pin2.duty_u16(RobotCar.MIN_DUTY_CYCLE)

    ''' Map duty cycle values from 0-100 to duty cycle 40000-65535 '''
    def __map_range(self, x, in_min, in_max, out_min, out_max):
```

```
        return (x - in_min) * (out_max - out_min) // (in_max - in_min) +
out_min

    ''' new_speed is a value from 0% - 100% '''
    def change_speed(self, new_speed):
        new_duty_cycle = self.__map_range(new_speed, 0, 100, 40000,
65535)
        self.current_speed = new_duty_cycle


    def deinit(self):
        """deinit PWM Pins"""
        print("Deinitializing PWM Pins")
        self.stop()
        utime.sleep(0.1)
        self.left_motor_pin1.deinit()
        self.left_motor_pin2.deinit()
        self.right_motor_pin1.deinit()
        self.right_motor_pin2.deinit()



from robot_car import RobotCar
import utime
from machine import Pin

# make the lights and assign them pins
YellowLED=Pin(15,Pin.OUT)
RedLED=Pin(14,Pin.OUT)
BlackLED=Pin(13,Pin.OUT)
WhiteLED=Pin(12,Pin.OUT)
GreenLED=Pin(11,Pin.OUT)
PurpleLED=Pin(10,Pin.OUT)

# Pico W GPIO Pin
LEFT_MOTOR_PIN_1 = 16
LEFT_MOTOR_PIN_2 = 17
RIGHT_MOTOR_PIN_1 = 18
```

```
RIGHT_MOTOR_PIN_2 = 19

# define the pins used for motor
motor_pins = [LEFT_MOTOR_PIN_1, LEFT_MOTOR_PIN_2, RIGHT_MOTOR_PIN_1,
RIGHT_MOTOR_PIN_2]

# Create an instance of our robot car
robot_car = RobotCar(motor_pins, 20000)

# Flag to track if the car is paused
car_paused = False

# make a function
def toggle_pause():
    global car_paused
    car_paused = not car_paused

if __name__ == '__main__':
    try:

        print("start")
        for i in range(285):
            if button.value() == 1:
                toggle_pause()
                # Wait for button release to avoid multiple toggles
                while button.value() == 1:
                    utime.sleep(0.3)
            # make it so when its not paused it moves
            if not car_paused:
            robot_car.change_speed(100);
            robot_car.move_forward()
            utime.sleep(.006)

            # turn on the lights and turn off the others
            YellowLED.value(1)
            RedLED.value(0)
            BlackLED.value(1)
            WhiteLED.value(0)
```

```
            GreenLED.value(1)
            PurpleLED.value(0)
            utime.sleep(.497)

            # turn off the lights and turn on the others
            YellowLED.value(0)
            RedLED.value(1)
            BlackLED.value(0)
            WhiteLED.value(1)
            GreenLED.value(0)
            PurpleLED.value(1)
            utime.sleep(.497)



        robot_car.deinit()

    except KeyboardInterrupt:
        robot_car.deinit()
```