Autonomous Drink Shooter

ME102B Project Report

Mohamed Mouhab
Ibad Ali
Chester Zelaya
Shayan Moghaddam

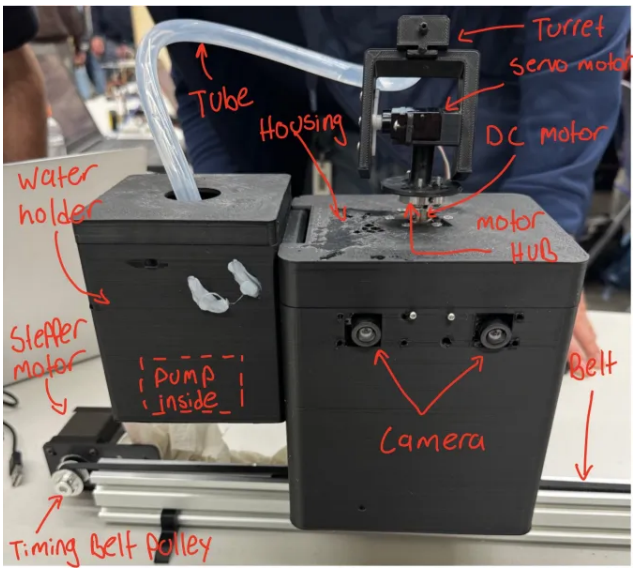
December 14th 2023

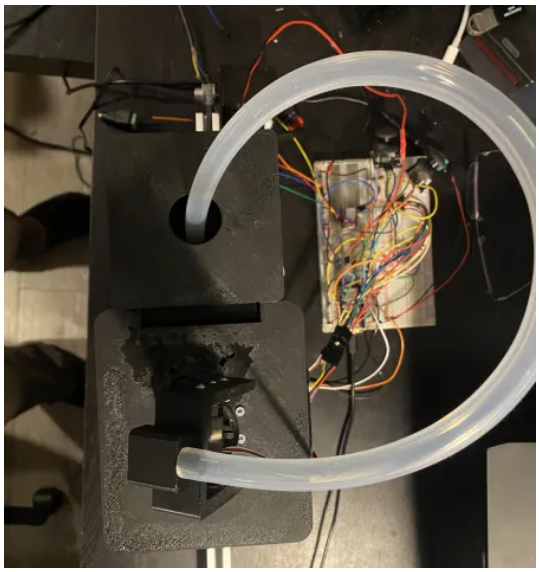
Opportunity

We have observed that environments such as parties and bars often experience high levels of crowding, resulting in extended wait times for service. To address this issue, we have developed a novel and engaging solution for the expedient delivery of beverages in such settings. Introducing the Automatic Drink Shooter (ADS), a groundbreaking system that streamlines the drink serving process. The ADS utilizes advanced facial recognition technology to accurately calibrate and project beverages directly to the recipient. By integrating reverse kinematics, the system incorporates a sophisticated tracking mechanism capable of locking onto and following targets, thereby ensuring precise and efficient drink delivery. This innovative approach not only enhances the customer experience but also optimizes service efficiency in high-density social environments.

High-Level Strategy

In the development of our project, considerable attention was devoted to optimizing the design of the nozzle to achieve the most efficient laminar flow. This process involved thorough experimentation and refinement of various nozzle designs. During the assembly phase, we encountered an unforeseen challenge: the housing of our system was exerting undue stress on the belt mechanism. To alleviate this issue, spacers were introduced between the housing and the gantry plate, a modification not originally accounted for in our CAD drawings. These spacers proved critical in creating sufficient clearance to relieve the belt from stress, thereby enhancing the overall mechanical integrity of the system.





Further, to optimize the performance of our linear actuator, we incorporated eccentric spacers for the bearings. This adjustment was crucial in increasing the frictional force between the rail and the bearings, ensuring smoother and more controlled movement.

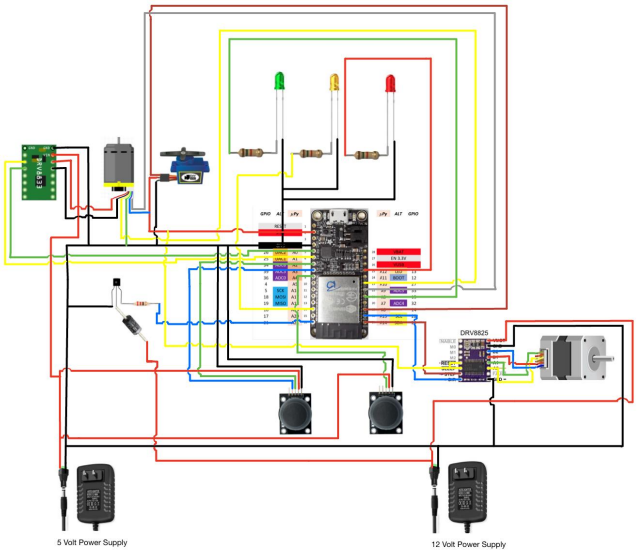
A key feature of our design is the turret system, which boasts three degrees of freedom, enabling it to cover an extensive range of operational angles. The linear actuator is responsible for the translation along the x-axis, providing precise lateral movement. Additionally, the DC motor, integral to the shooter mechanism, allows for complete 360-degree rotation about the y-axis, facilitating angular adjustments. Completing this setup is a servo mounted on the turret, which manipulates the nozzle's orientation about the x-axis for height adjustments. This tripartite system ensures that our turret can achieve optimal shooting ranges with high precision, a testament to the careful engineering and design considerations implemented throughout our project.

Function Critical Decisions

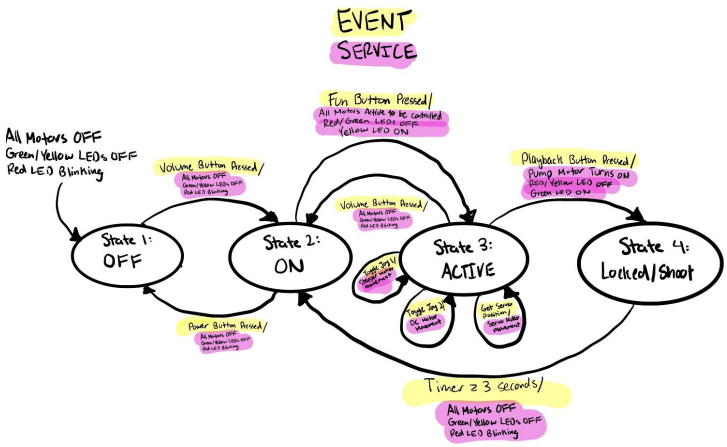
In the development of our linear actuator, we chose to utilize a stepper motor due to its exceptional capability for controlling precise positioning. This decision was grounded in the need for accuracy and reliability in our system. Additionally, for the tilt control mechanism of the turret, we implemented a servo. This choice was informed by the ease with which angular inputs in our code could be reverse-engineered, facilitating effective tracking of individuals.

Regarding the design of the linear actuator, we were presented with two main options: a screw-driven system or a belt-driven system. After careful consideration, we decided on the belt-driven actuator. This decision was influenced by our familiarity with its applications from previous academic coursework. Our experience in class provided us with valuable insights into the practical advantages of belt-driven systems, such as their efficiency and adaptability, making them a more suitable choice for our project requirements.

Circuit Diagram



State Diagram



Function-Critical Decision & Calculations

1) **Preload force Calculation**

$T_1 = T_i + \frac{\tau}{d}$ …… 1

$T_2 = T_i + \frac{\tau}{d}$ ……. 2

$F_{pre} = 2T_i$ …… 3

Equating these by subbing equation 2 into 3 and substituting the torque of our motor…

$T_2 = \frac{F_{pre}}{2} - \frac{\tau}{d} \geq 0$

$F_{pre} \geq 11.3\ lb$

2) **Applied Force on Bearings**

Sum of Forces in the Y direction

$\sum F_y = 4F_{by} - F_H = 0;\ Let\ F\_H\ be\ the\ force\ on\ the\ linear\ actuator\ by\ the\ Housing\ load \sim 6\ lb$

$F_{by} = \frac{6\ lb}{4}$

$F_{by} = 1.5\ lb$

Sum of Forces in the X direction

$\sum F_x = T_1 - T_2 = 0\ ;\ No\ force\ exerted\ by\ bearings\ in\ the\ x\ -\ direction$

Sum of Forces in the Z direction

Forces on bearings are adjustable in the z-direction because we used eccentric spacers to increase friction between the bearing and the V-rail.

3) **Max Radial Force in DC motor**

Motor Stall Torque: 18 kg/cm = 1.7658 N

Shaft Radius: 3 mm

Calculation: $F = \frac{T}{r} = \frac{1.7658}{0.003} = 588.6\ N$

4) **Fluid velocity Calculation**

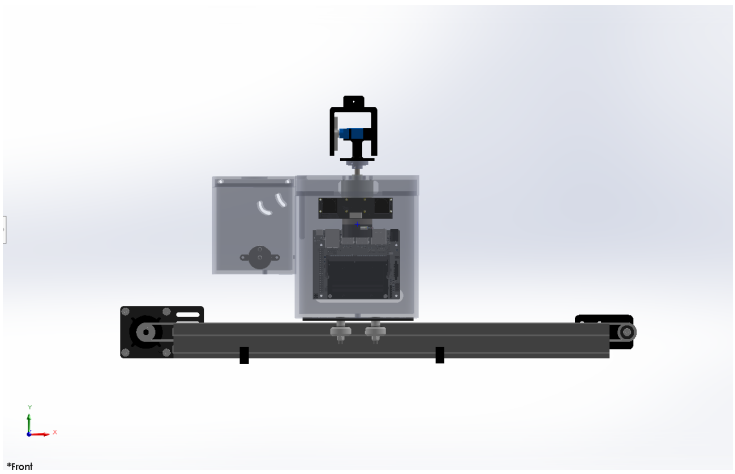
$P_1 + \frac{1}{2}v_1^2\rho + \rho g z_1 = P_2 + \frac{1}{2}v_1^2\rho + \rho g z_2$

Assuming P1 and P2 are equal. v1=5cm/s, z2-z1=0.13m, rho = 1000kg/m^3

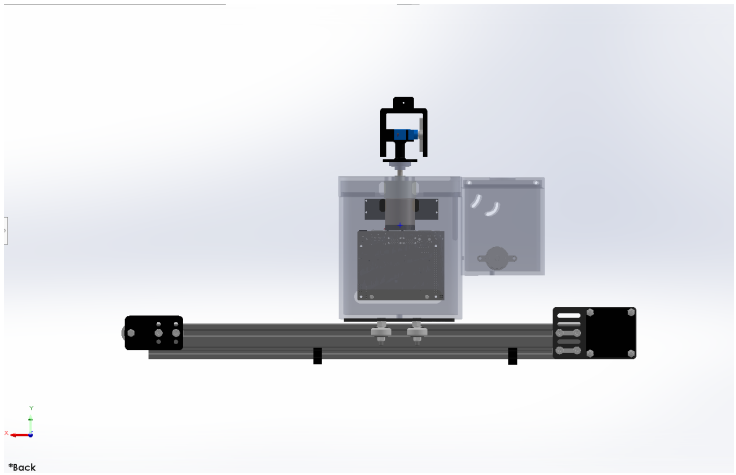
$v_2 = \sqrt{v_1^2 + 2g(z_2 - z_1)} = \sqrt{0.05^2 + 2(9.81)(0.13)} = 1.6\ m/s$

Appendix

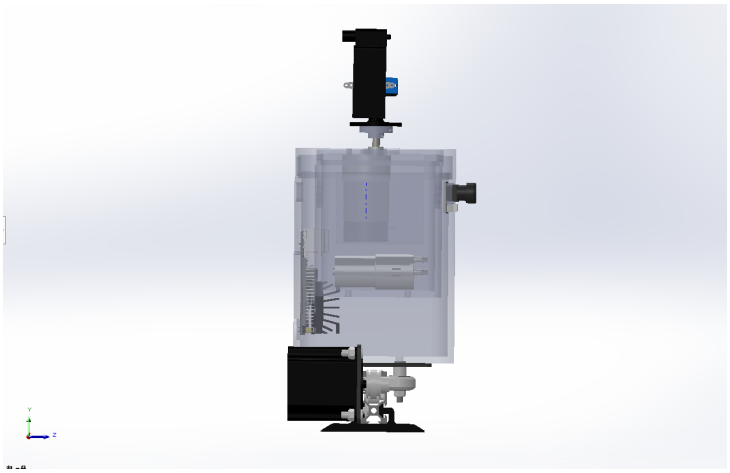
CAD Design



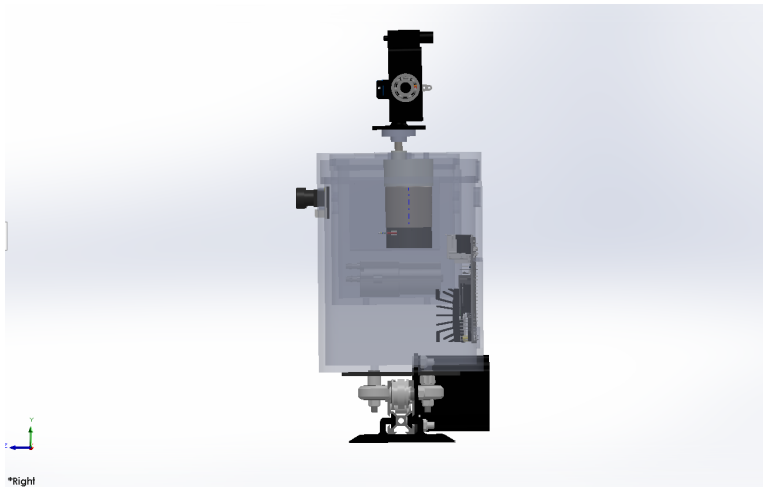
Front View



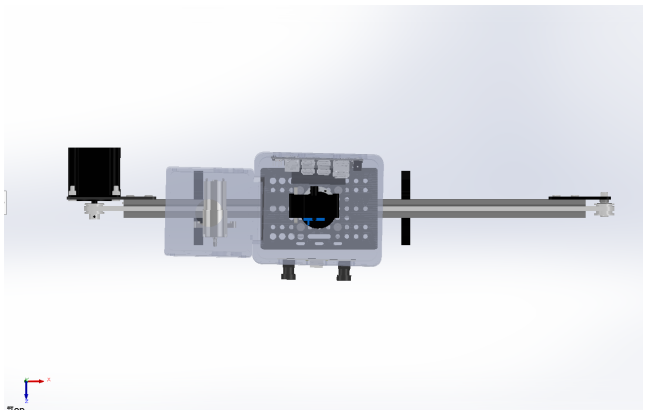
Back View

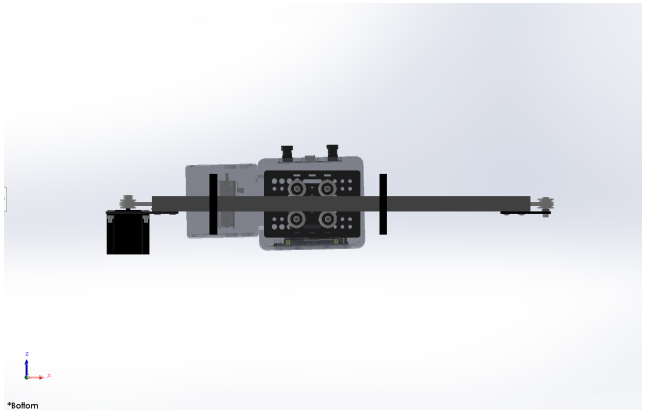


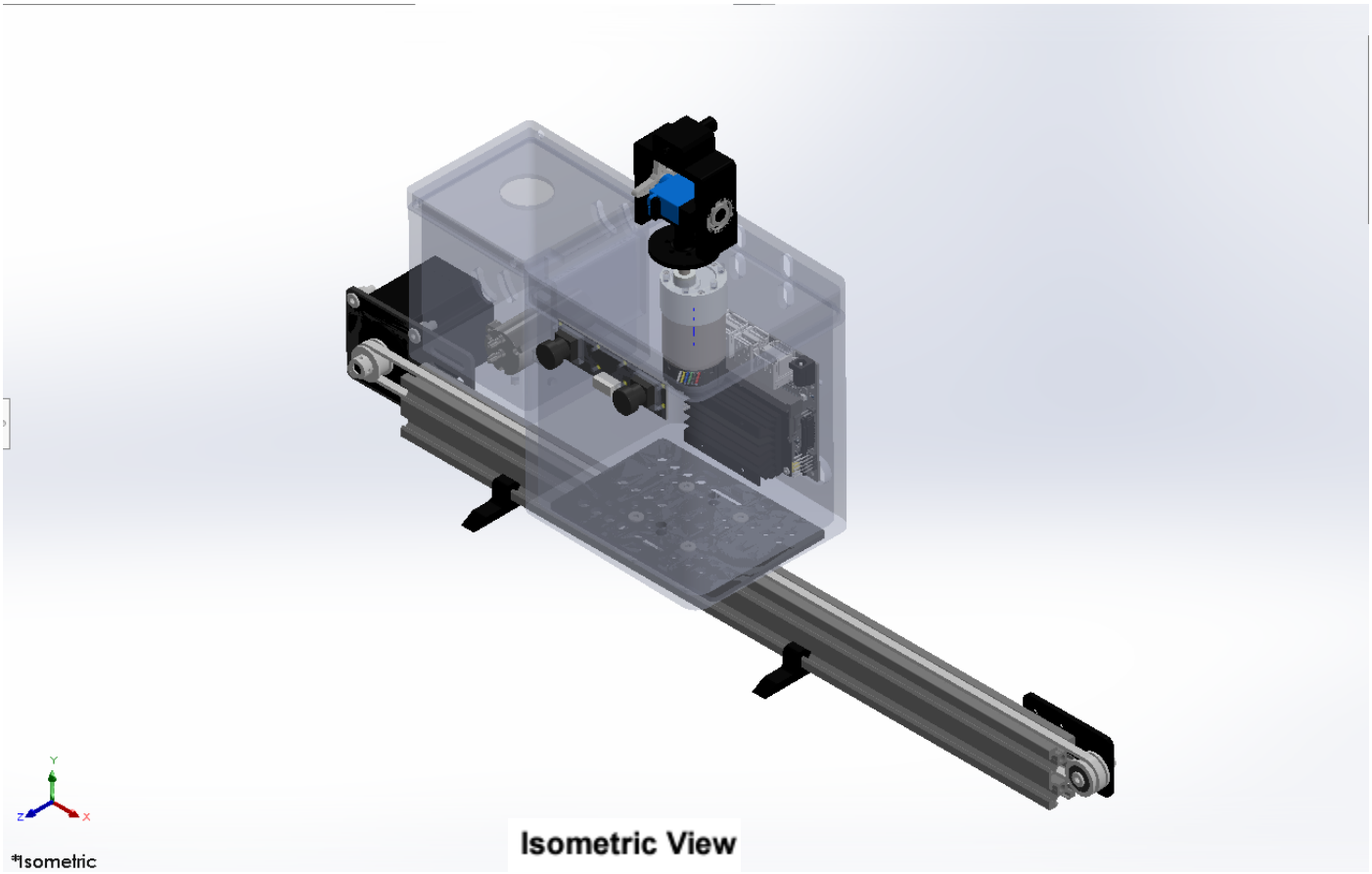
Left View



Right View

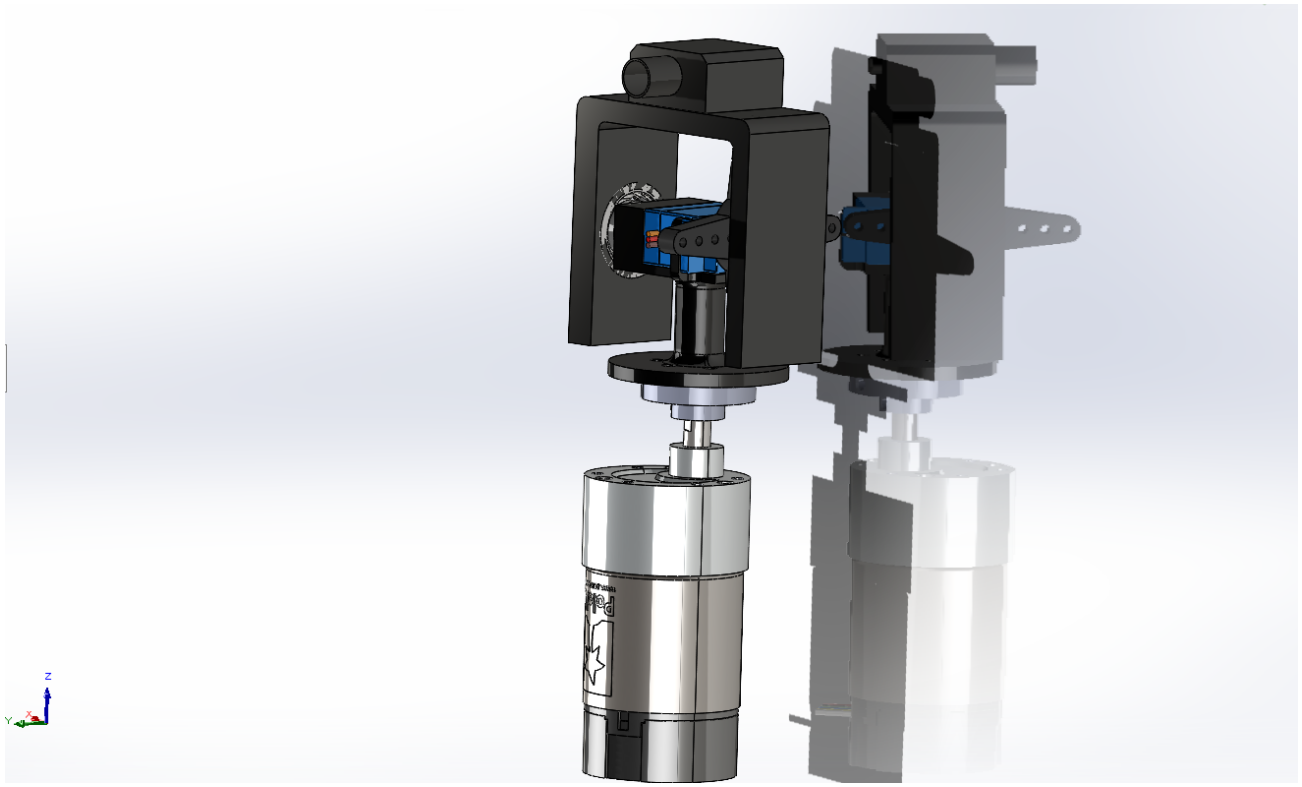


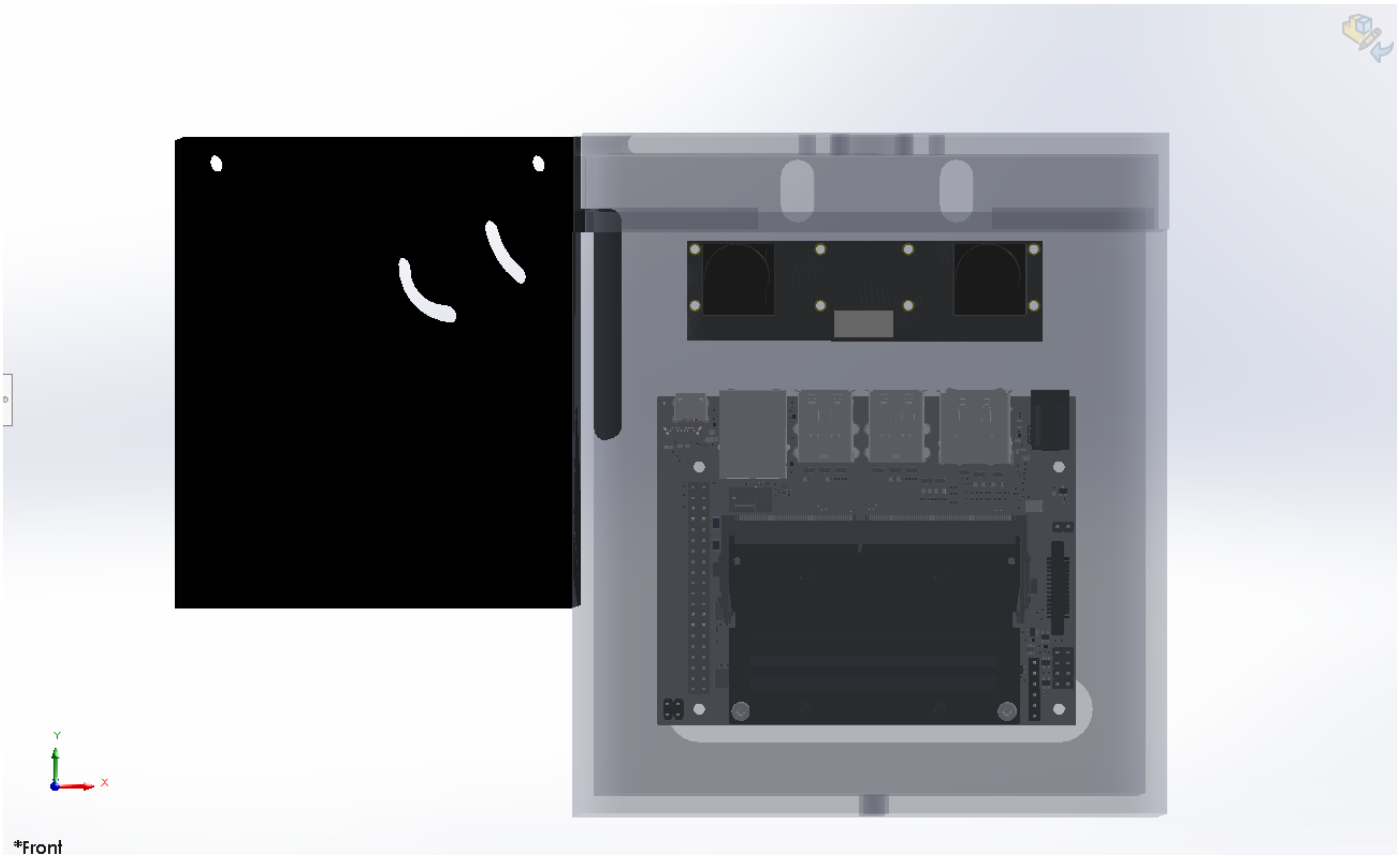




*Isometric

Isometric View




Y
X
*Front

Bill Of Material

Link	Part	Qty	Unit Cost	Cost	
Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store	Low Profile Screws M5 - 8mm	6	$0.99	$5.94	
Aluminum Spacers (10 Pack) - OpenBuilds Part Store	Aluminum Spacer - 6mm	2	$3.39	$6.78	
Aluminum Spacers (10 Pack) - OpenBuilds Part Store	Aluminum Spacer - 3mm	1	$2.49	$2.49	
Double Tee Nut - OpenBuilds Part Store	Double Tee Nuts	3	$0.69	$2.07	
3GT (GT2-3M) Timing Pulley - 20 Tooth - OpenBuilds Part S	GT3 (2mm) Timing Pulley - 20 Tooth	1	$7.99	$7.99	
Eccentric Spacer - OpenBuilds Part Store	Eccentric Spacers - 6mm	2	$1.99	$3.98	
Solid V Wheel Kit (openbuildspartstore.com)	Solid V Wheel Kit	4	$5.19	$20.76	
Smooth Idler Pulley Kit - OpenBuilds Part Store	Smooth Idler Pulley Kit	1	$5.99	$5.99	
Idler Pulley Plate - OpenBuilds Part Store	Idler Pulley Plate	1	$6.99	$6.99	
Motor Mount Plate - NEMA 23 Stepper Motor - OpenBuild	Motor Mount Plate for Nema 23	1	$7.99	$7.99	
Cable Ties (10 Pack) - OpenBuilds Part Store	Cable Ties	4	$0.79	$3.16	
Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store	Low Profile Screws M5 - 15mm	4	$1.19	$4.76	
OpenRail Gantry Plate - OpenBuilds Part Store	V-Slot Gantry Plate (20mm)	1	$6.99	$6.99	
3GT (GT2-3M) Timing Belt - By the Foot - OpenBuilds Part	3GT (GT2-3M) Timing Belt - By the Foot	1	$3.49	$3.49	
V-Slot 20x40 Linear Rail Aluminum Extrusion Profile (open	V-Slot® Linear Rail - 20x40	1	$3.99	$3.99	
Nylon Insert Hex Locknut - M5 (10 Pack) - OpenBuilds Part	Nylon Insert Hex Locknut - M5	4	$0.99	$3.96	
Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store	Low Profile Screws M5 - 25mm	4	$1.39	$5.56	
NEMA 23 Stepper Motor - OpenBuilds Part Store	NEMA 23 Stepper Motor	1	$27.99	$27.99	
amazon.com/dp/B0BMVFJTNR?psc=1&ref=ppx_yo2ov_dt	SZMP Water Fountain Pump	1	$17.98	$17.98	
Camera Module for NVIDIA Jetson Nano Board | 8MP Sen	SainSmart IMX219 Camera Module	2	$21.99	$43.98	
amazon.com/NVIDIA-Jetson-Nano-Developer-945-13450-	NVIDIA Jetson Nano Developer Kit	1	$149.00	$149.00	
Pololu - TB67H420FTG Dual/Single Motor Driver Carrier	Dual/Single Motor Driver Carrier	1	11.95	$11.95	
amazon.com/Duracell-Coppertop-Alkaline-AAA-Batteries/	AAA Batteries	1	$6.79	$6.79	
amazon.com/Micro-Servos-Helicopter-Airplane-Controls/	Servo Motors	1	$8.99	$8.99	
amazon.com/Pack-Battery-Holder-Bundle-QTEATAK/dp/B	Battery Holder	1	$6.99	$6.99	
	Brushless DC Motor	1	$0	$0.00	
				$376.56	**Total Cost**

BoM was created in Excel, so the links are pasted in order of the parts below:

Links

Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store

Aluminum Spacers (10 Pack) - OpenBuilds Part Store

Aluminum Spacers (10 Pack) - OpenBuilds Part Store

Double Tee Nut - OpenBuilds Part Store

3GT (GT2-3M) Timing Pulley - 20 Tooth - OpenBuilds Part

Store Eccentric Spacer - OpenBuilds Part Store

Solid V Wheel Kit (openbuildspartstore.com)

Smooth Idler Pulley Kit - OpenBuilds Part Store

Idler Pulley Plate - OpenBuilds Part Store

Motor Mount Plate - NEMA 23 Stepper Motor - OpenBuilds Part

Store Cable Ties (10 Pack) - OpenBuilds Part Store

Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store

OpenRail Gantry Plate - OpenBuilds Part Store

3GT (GT2-3M) Timing Belt - By the Foot - OpenBuilds Part Store V-Slot

20x40 Linear Rail Aluminum Extrusion Profile (openbuildspartstore.com)

Nylon Insert Hex Locknut - M5 (10 Pack) - OpenBuilds Part Store

Low Profile Screws M5 (10 Pack) - OpenBuilds Part Store

NEMA 23 Stepper Motor - OpenBuilds Part Store

amazon.com/dp/B0BMVFJTNR?psc=1&ref=ppx_yo2ov_dt_b_product_details

Camera Module for NVIDIA Jetson Nano Board | 8MP Sensor | 160 Degree

FoV – SainSmart.com

amazon.com/NVIDIA-Jetson-Nano-Developer-945-13450-0000-100/dp/B084DSDDLT?ref_=ast_sto_dp

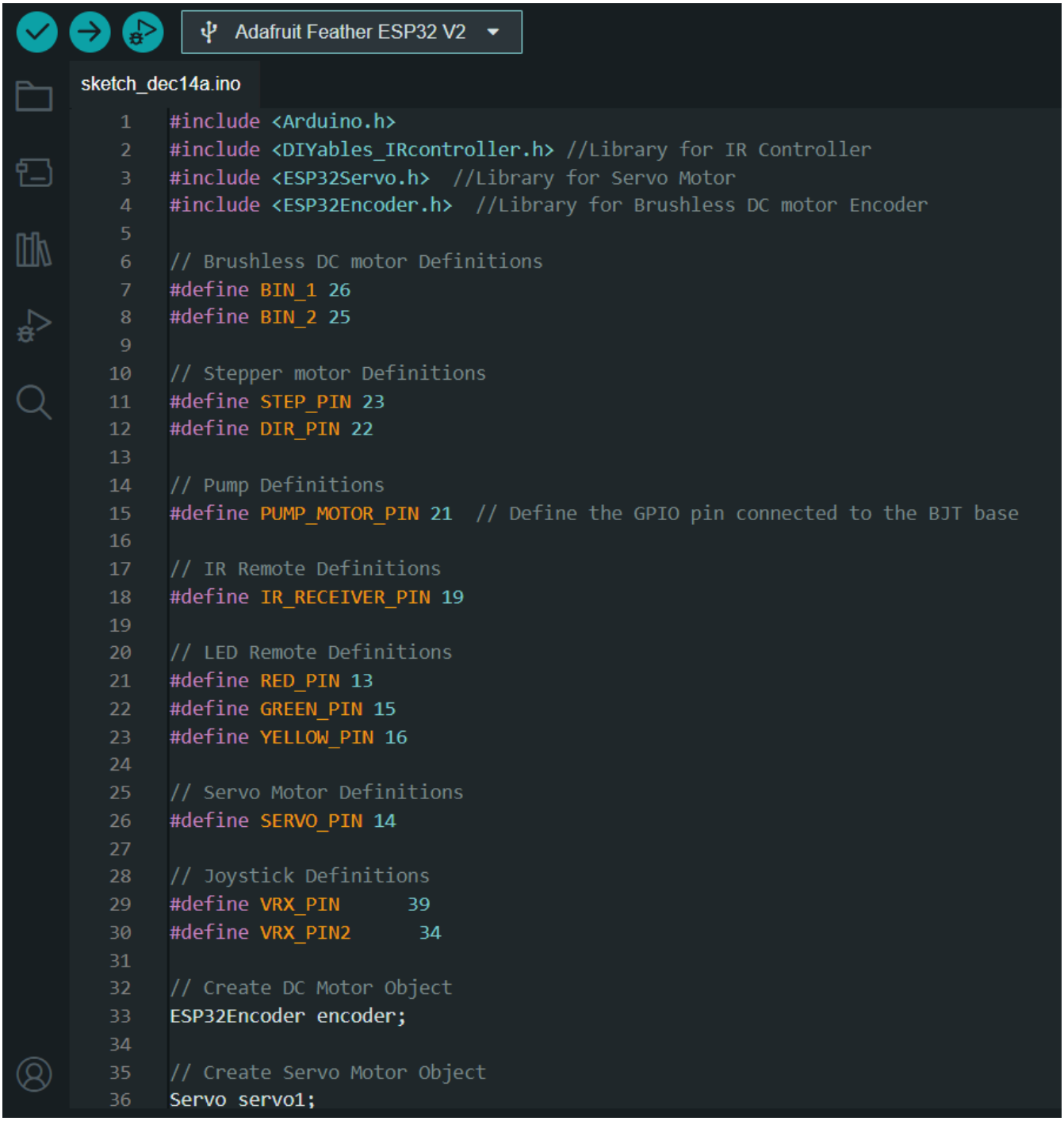
Pololu - TB67H420FTG Dual/Single Motor Driver Carrier

amazon.com/Duracell-Coppertop-Alkaline-AAA-Batteries/dp/B004M7YC8S/ref=sr_1_5?keywords=aaa+batteries+6+pack&qid=1698019386&rdc=1&sr=8-5

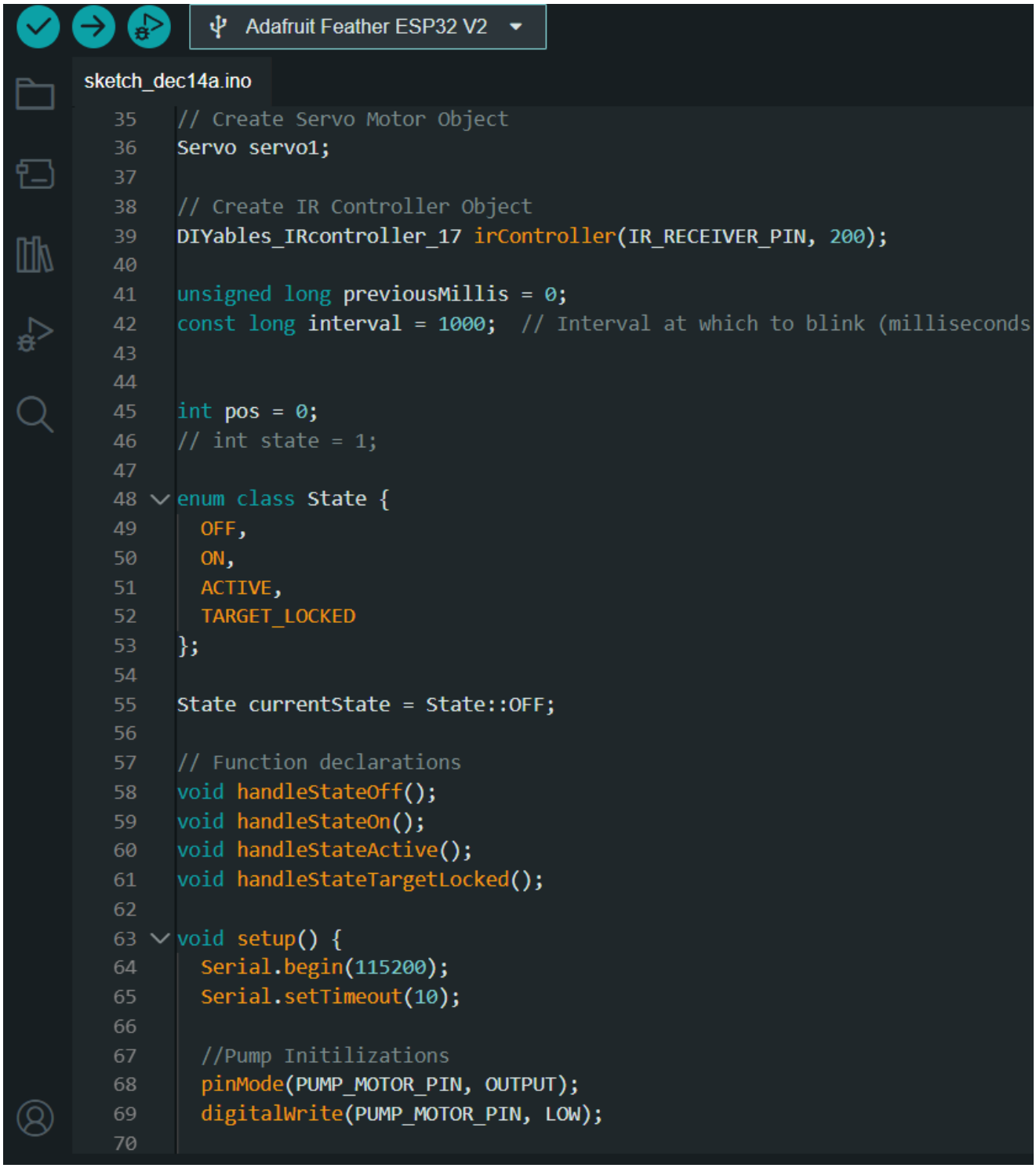
amazon.com/Micro-Servos-Helicopter-Airplane-Controls/dp/B07MLR1498/ref=sr_1_16?crid=XGYPOABGRJK5&keywords=servo+motors&qid=1698019430&sprefix=servo+motor%2Caps%2C172&sr=8-16

amazon.com/Pack-Battery-Holder-Bundle-QTEATAK/dp/B07WY3VMNN/ref=sr_1_11?crid=3BC2MMYXIYNYR&keywords=battery%2Bholder&qid=1698019446&sprefix=battery%2Bhold%2Caps%2C167&sr=8-11&th=1

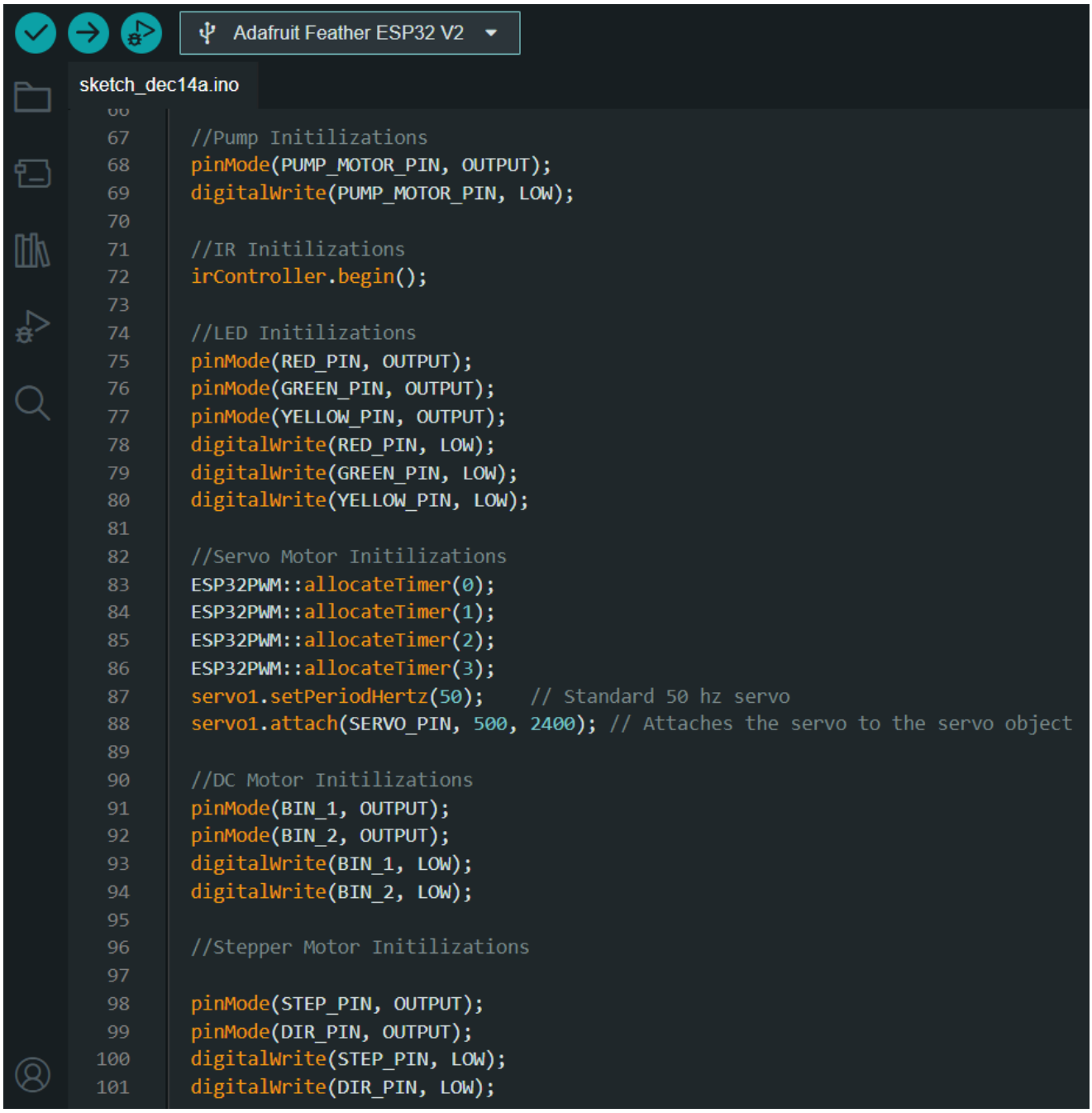
Code:



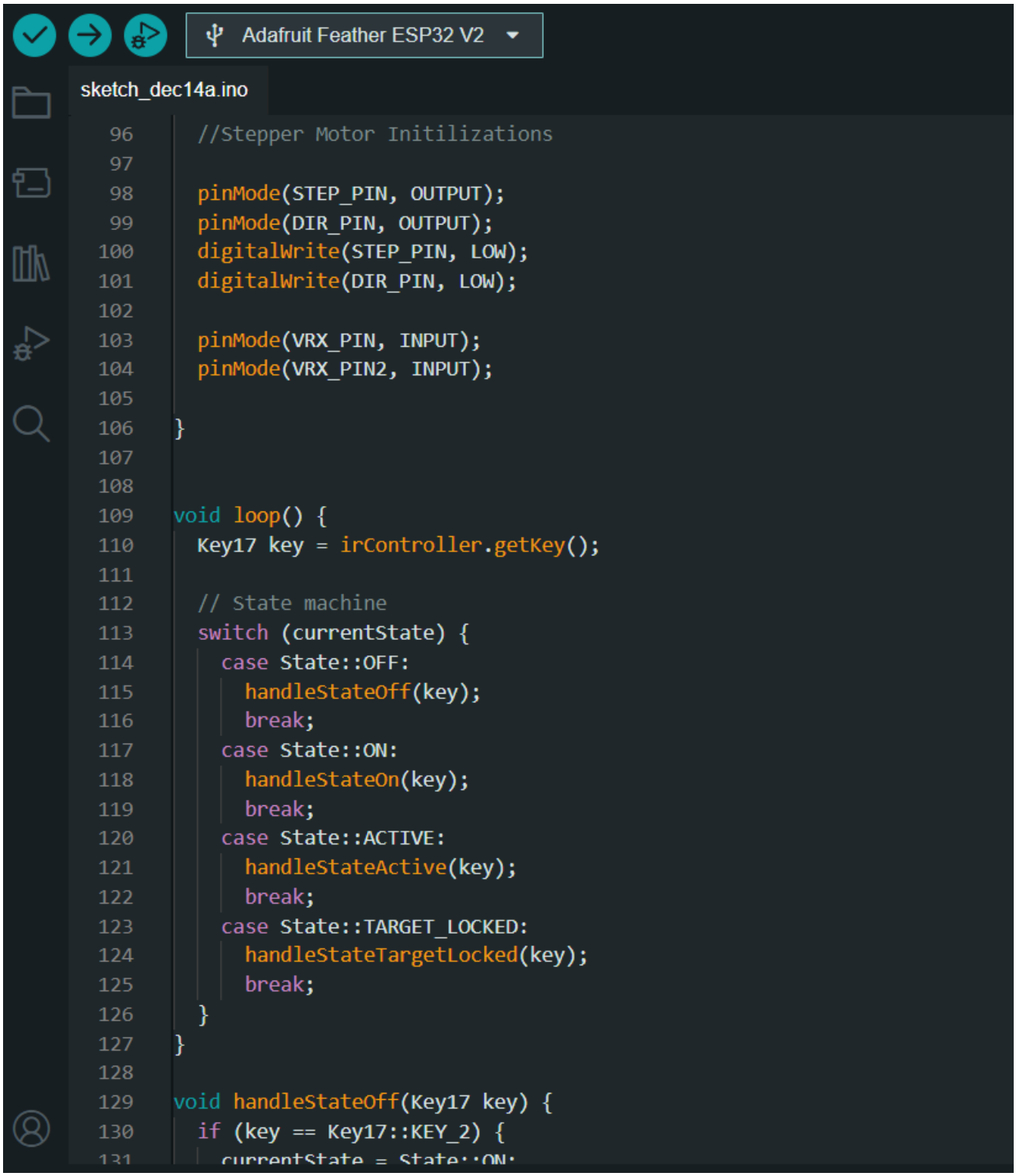

```
#include <Arduino.h>
#include <DIYables_IRcontroller.h> //Library for IR Controller
#include <ESP32Servo.h>  //Library for Servo Motor
#include <ESP32Encoder.h>  //Library for Brushless DC motor Encoder

// Brushless DC motor Definitions
#define BIN_1 26
#define BIN_2 25

// Stepper motor Definitions
#define STEP_PIN 23
#define DIR_PIN 22

// Pump Definitions
#define PUMP_MOTOR_PIN 21  // Define the GPIO pin connected to the BJT base

// IR Remote Definitions
#define IR_RECEIVER_PIN 19

// LED Remote Definitions
#define RED_PIN 13
#define GREEN_PIN 15
#define YELLOW_PIN 16

// Servo Motor Definitions
#define SERVO_PIN 14

// Joystick Definitions
#define VRX_PIN      39
#define VRX_PIN2      34

// Create DC Motor Object
ESP32Encoder encoder;

// Create Servo Motor Object
Servo servo1;
```



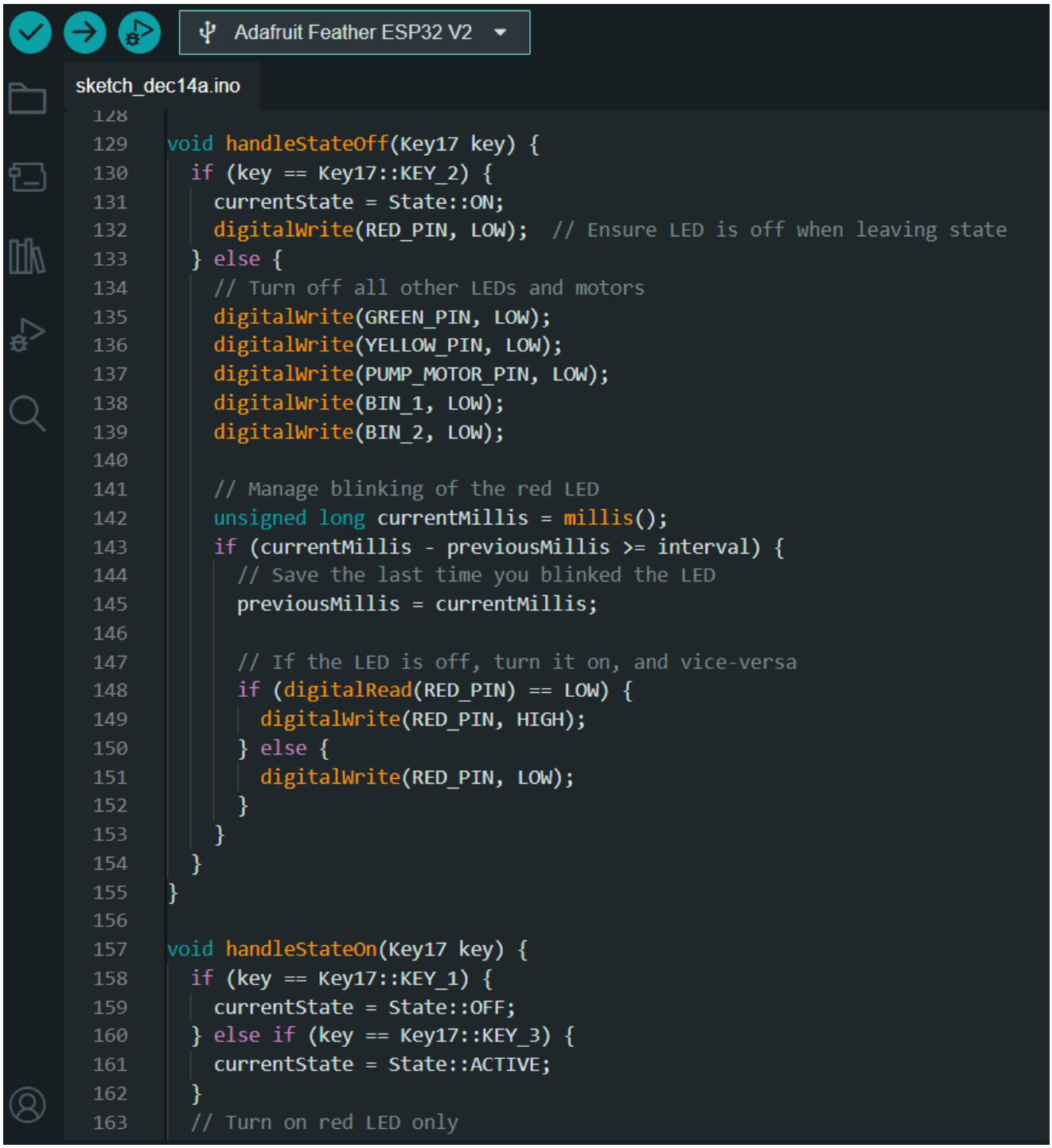

```
35  // Create Servo Motor Object
36  Servo servo1;
37
38  // Create IR Controller Object
39  DIYables_IRcontroller_17 irController(IR_RECEIVER_PIN, 200);
40
41  unsigned long previousMillis = 0;
42  const long interval = 1000;  // Interval at which to blink (milliseconds
43
44
45  int pos = 0;
46  // int state = 1;
47
48  enum class State {
49    OFF,
50    ON,
51    ACTIVE,
52    TARGET_LOCKED
53  };
54
55  State currentState = State::OFF;
56
57  // Function declarations
58  void handleStateOff();
59  void handleStateOn();
60  void handleStateActive();
61  void handleStateTargetLocked();
62
63  void setup() {
64    Serial.begin(115200);
65    Serial.setTimeout(10);
66
67    //Pump Initilizations
68    pinMode(PUMP_MOTOR_PIN, OUTPUT);
69    digitalWrite(PUMP_MOTOR_PIN, LOW);
70
```



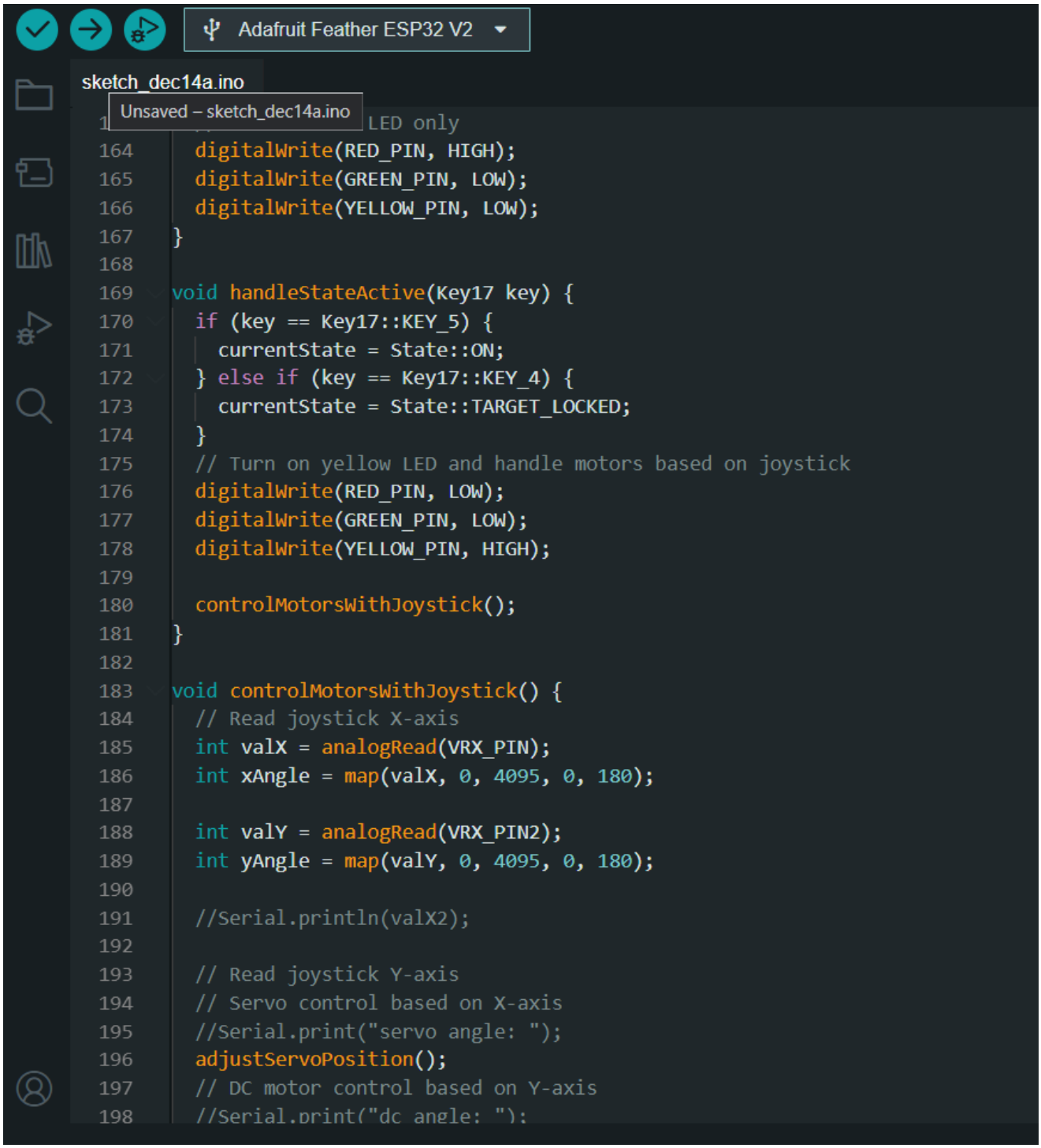

```
67    //Pump Initilizations
68    pinMode(PUMP_MOTOR_PIN, OUTPUT);
69    digitalWrite(PUMP_MOTOR_PIN, LOW);
70
71    //IR Initilizations
72    irController.begin();
73
74    //LED Initilizations
75    pinMode(RED_PIN, OUTPUT);
76    pinMode(GREEN_PIN, OUTPUT);
77    pinMode(YELLOW_PIN, OUTPUT);
78    digitalWrite(RED_PIN, LOW);
79    digitalWrite(GREEN_PIN, LOW);
80    digitalWrite(YELLOW_PIN, LOW);
81
82    //Servo Motor Initilizations
83    ESP32PWM::allocateTimer(0);
84    ESP32PWM::allocateTimer(1);
85    ESP32PWM::allocateTimer(2);
86    ESP32PWM::allocateTimer(3);
87    servo1.setPeriodHertz(50);    // Standard 50 hz servo
88    servo1.attach(SERVO_PIN, 500, 2400); // Attaches the servo to the servo object
89
90    //DC Motor Initilizations
91    pinMode(BIN_1, OUTPUT);
92    pinMode(BIN_2, OUTPUT);
93    digitalWrite(BIN_1, LOW);
94    digitalWrite(BIN_2, LOW);
95
96    //Stepper Motor Initilizations
97
98    pinMode(STEP_PIN, OUTPUT);
99    pinMode(DIR_PIN, OUTPUT);
100   digitalWrite(STEP_PIN, LOW);
101   digitalWrite(DIR_PIN, LOW);
```



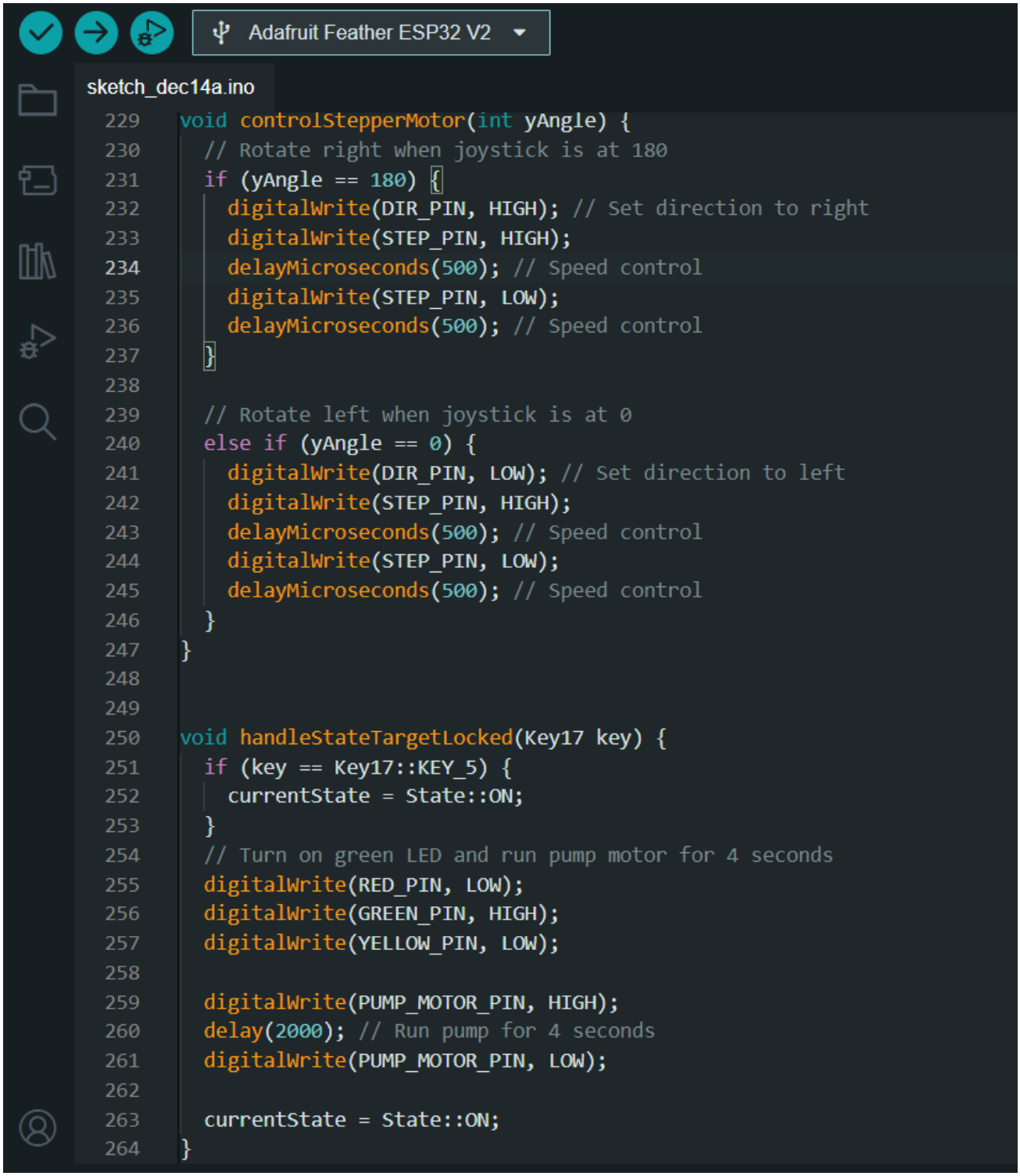

```
  //Stepper Motor Initilizations

  pinMode(STEP_PIN, OUTPUT);
  pinMode(DIR_PIN, OUTPUT);
  digitalWrite(STEP_PIN, LOW);
  digitalWrite(DIR_PIN, LOW);

  pinMode(VRX_PIN, INPUT);
  pinMode(VRX_PIN2, INPUT);

}


void loop() {
  Key17 key = irController.getKey();

  // State machine
  switch (currentState) {
    case State::OFF:
      handleStateOff(key);
      break;
    case State::ON:
      handleStateOn(key);
      break;
    case State::ACTIVE:
      handleStateActive(key);
      break;
    case State::TARGET_LOCKED:
      handleStateTargetLocked(key);
      break;
  }
}

void handleStateOff(Key17 key) {
  if (key == Key17::KEY_2) {
    currentState = State::ON;
```




```
128
129  void handleStateOff(Key17 key) {
130    if (key == Key17::KEY_2) {
131      currentState = State::ON;
132      digitalWrite(RED_PIN, LOW);  // Ensure LED is off when leaving state
133    } else {
134      // Turn off all other LEDs and motors
135      digitalWrite(GREEN_PIN, LOW);
136      digitalWrite(YELLOW_PIN, LOW);
137      digitalWrite(PUMP_MOTOR_PIN, LOW);
138      digitalWrite(BIN_1, LOW);
139      digitalWrite(BIN_2, LOW);
140
141      // Manage blinking of the red LED
142      unsigned long currentMillis = millis();
143      if (currentMillis - previousMillis >= interval) {
144        // Save the last time you blinked the LED
145        previousMillis = currentMillis;
146
147        // If the LED is off, turn it on, and vice-versa
148        if (digitalRead(RED_PIN) == LOW) {
149          digitalWrite(RED_PIN, HIGH);
150        } else {
151          digitalWrite(RED_PIN, LOW);
152        }
153      }
154    }
155  }
156
157  void handleStateOn(Key17 key) {
158    if (key == Key17::KEY_1) {
159      currentState = State::OFF;
160    } else if (key == Key17::KEY_3) {
161      currentState = State::ACTIVE;
162    }
163    // Turn on red LED only
```


Adafruit Feather ESP32 V2
sketch_dec14a.ino
Unsaved – sketch_dec14a.ino
LED only
digitalWrite(RED_PIN, HIGH);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(YELLOW_PIN, LOW);
}
void handleStateActive(Key17 key) {
if (key == Key17::KEY_5) {
currentState = State::ON;
} else if (key == Key17::KEY_4) {
currentState = State::TARGET_LOCKED;
}
// Turn on yellow LED and handle motors based on joystick
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(YELLOW_PIN, HIGH);
controlMotorsWithJoystick();
}
void controlMotorsWithJoystick() {
// Read joystick X-axis
int valX = analogRead(VRX_PIN);
int xAngle = map(valX, 0, 4095, 0, 180);
int valY = analogRead(VRX_PIN2);
int yAngle = map(valY, 0, 4095, 0, 180);
//Serial.println(valX2);
// Read joystick Y-axis
// Servo control based on X-axis
//Serial.print("servo angle: ");
adjustServoPosition();
// DC motor control based on Y-axis
//Serial.print("dc angle: ");


```
  controlDCMotor(xAngle);
  controlStepperMotor(yAngle);
}

void adjustServoPosition() {
  if (Serial.available() > 0) {
    long var1 = Serial.parseInt();
    Serial.println(var1);
    servo1.write(var1);
  }
}


void controlDCMotor(int xAngle) {
  if (xAngle == 0) {
    digitalWrite(BIN_1, HIGH);
    digitalWrite(BIN_2, LOW);
    //Serial.println("DC motor rotating in one direction");
  } else if (xAngle == 180) {
    digitalWrite(BIN_1, LOW);
    digitalWrite(BIN_2, HIGH);
    //Serial.println("DC motor rotating in the other direction");
  } else {
    digitalWrite(BIN_1, LOW);
    digitalWrite(BIN_2, LOW);
    //Serial.println("DC motor stopped");
  }
}

void controlStepperMotor(int yAngle) {
  // Rotate right when joystick is at 180
  if (yAngle == 180) {
    digitalWrite(DIR_PIN, HIGH); // Set direction to right
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(500); // Speed control
    digitalWrite(STEP_PIN, LOW);
```




```
229  void controlStepperMotor(int yAngle) {
230    // Rotate right when joystick is at 180
231    if (yAngle == 180) {
232      digitalWrite(DIR_PIN, HIGH); // Set direction to right
233      digitalWrite(STEP_PIN, HIGH);
234      delayMicroseconds(500); // Speed control
235      digitalWrite(STEP_PIN, LOW);
236      delayMicroseconds(500); // Speed control
237    }
238
239    // Rotate left when joystick is at 0
240    else if (yAngle == 0) {
241      digitalWrite(DIR_PIN, LOW); // Set direction to left
242      digitalWrite(STEP_PIN, HIGH);
243      delayMicroseconds(500); // Speed control
244      digitalWrite(STEP_PIN, LOW);
245      delayMicroseconds(500); // Speed control
246    }
247  }
248
249
250  void handleStateTargetLocked(Key17 key) {
251    if (key == Key17::KEY_5) {
252      currentState = State::ON;
253    }
254    // Turn on green LED and run pump motor for 4 seconds
255    digitalWrite(RED_PIN, LOW);
256    digitalWrite(GREEN_PIN, HIGH);
257    digitalWrite(YELLOW_PIN, LOW);
258
259    digitalWrite(PUMP_MOTOR_PIN, HIGH);
260    delay(2000); // Run pump for 4 seconds
261    digitalWrite(PUMP_MOTOR_PIN, LOW);
262
263    currentState = State::ON;
264  }
```