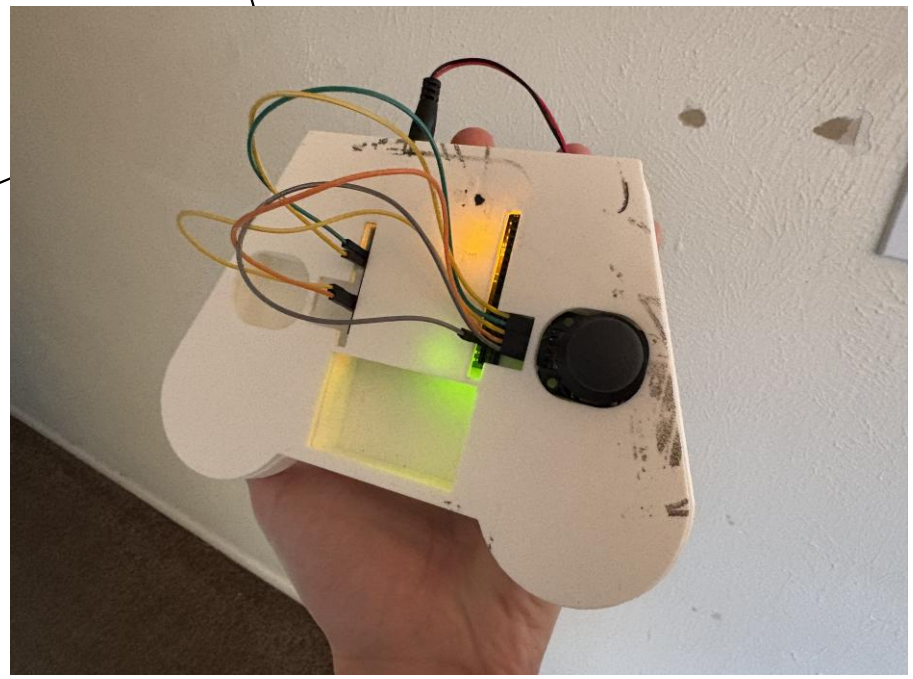
TEAM 95

FINAL DESIGN PRESENTATION

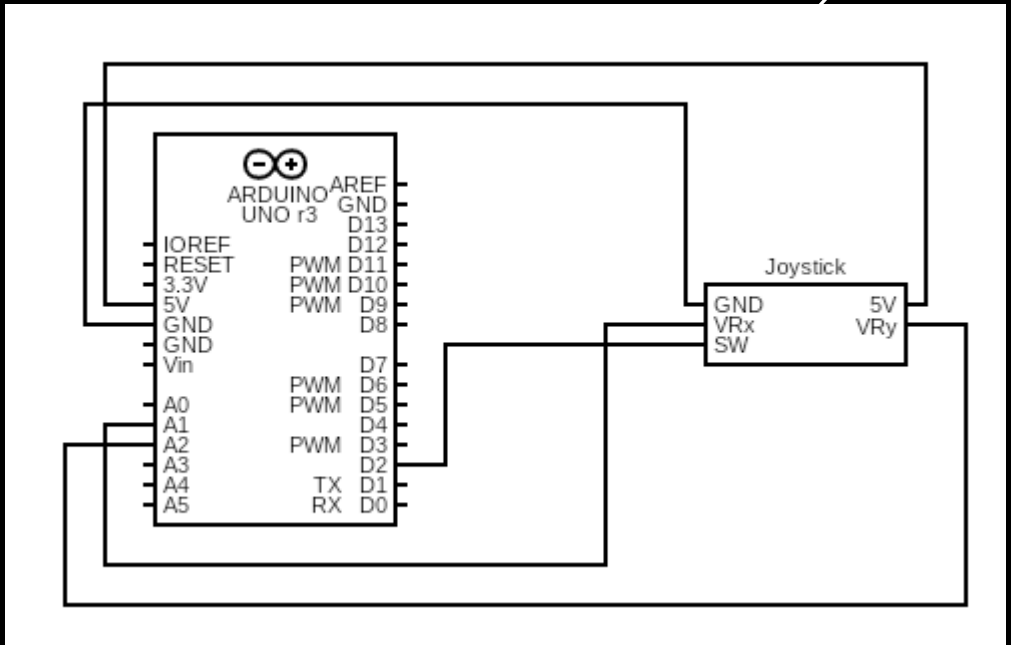
PROBLEM STATEMENT

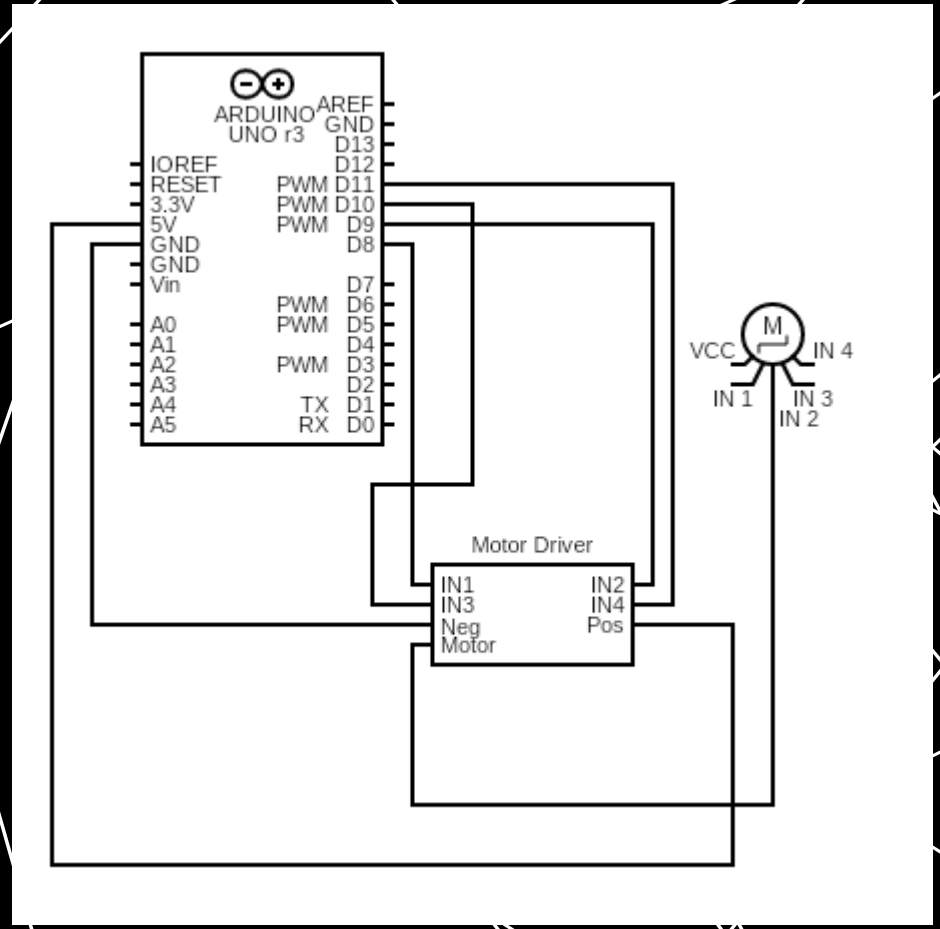
As drones are being considered more for package delivery systems, there is a challenge with being able to reliably pick up packages of varied sizes and shapes while the drone reaches its designation. Options on the market use fixed sized crates/claws or just drop the packages into designated zones. This lack of modularity makes it expensive to buy all the necessary sizes.

PHYSICAL PROTOTYPE



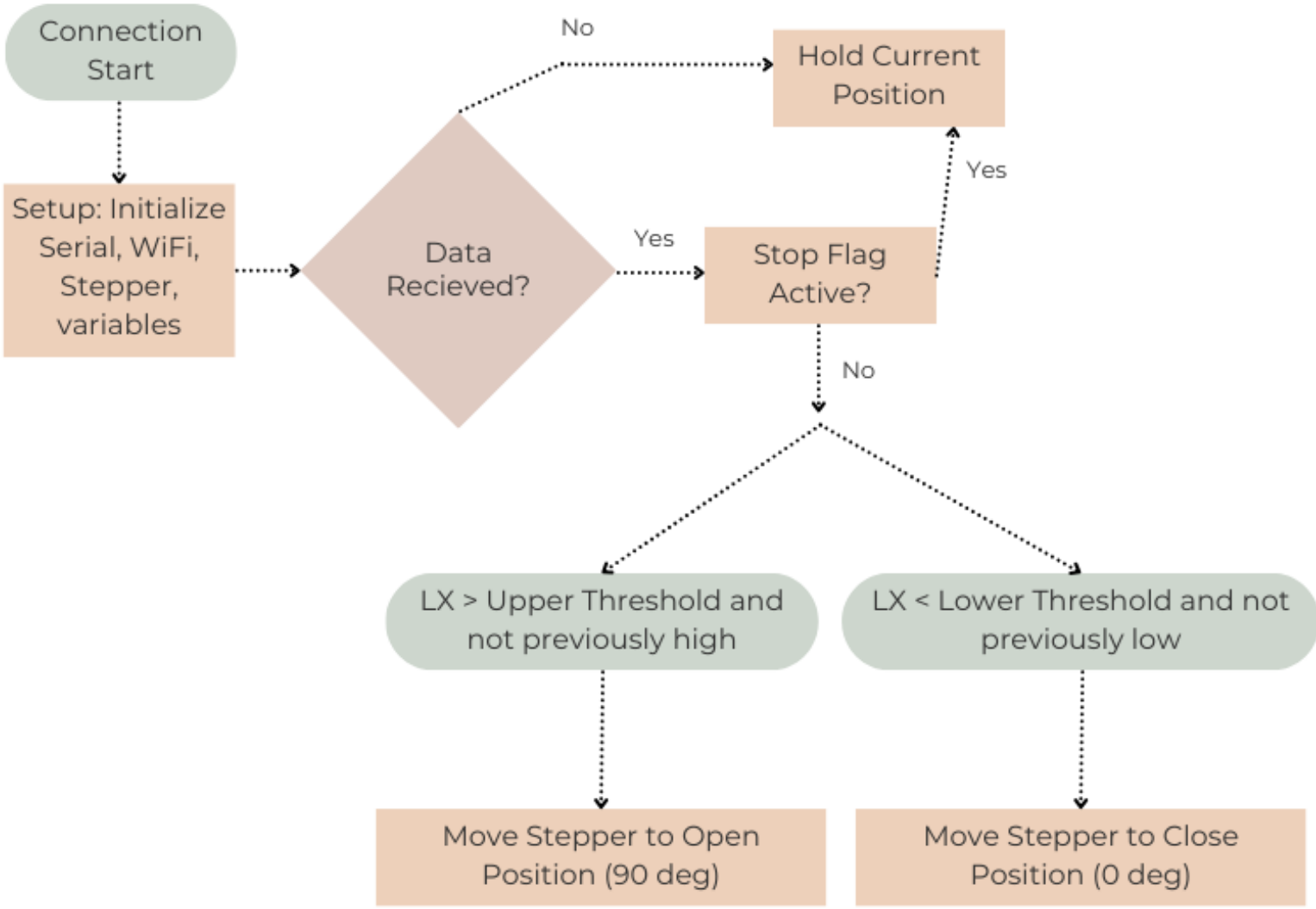
CIRCUIT DIAGRAMS





TEAM 95

LOGIC FLOWCHART



CODE

Transmitter

```
#include "WifiPort2.h"
#include "Stepper.h"

// Data we send
struct DataPacket {
  int LX;   // Left joystick X
  int LY;   // Left joystick Y
  int RX;   // Right joystick X
  int RY;   // Right joystick Y
  int Stop; // Stop button
} data;

WifiPort<DataPacket> WifiSerial;

// Pin setup
int LXpin = A0;
int LYpin = A1;
int RXpin = A2;
int RYpin = A3;

void setup() {
  Serial.begin(115200);
  WifiSerial.begin("Group95", "Orediggas", WifiPortType::Transmitter);
}
```

```
// Read joysticks
data.LX = analogRead(LXpin);
data.LY = analogRead(LYpin);
data.RX = analogRead(RXpin);
data.RY = analogRead(RYpin);

// Send data
WifiSerial.sendData(data);
```

CODE

Receiver

```
#include "WifiPort2.h"
#include <Stepper.h>

// transmitter data
struct DataPacket {
  int LX;
  int LY;
  int RX;
  int RY;
  int Stop;
} data;

WifiPort<DataPacket> WifiSerial;

// Stepper setup
const int StepsPerRevolution = 2048; // number of steps for one full revolution
Stepper myStepper(StepsPerRevolution, 8, 10, 9, 11);

// Angular position tracking (in degrees)
int NewAngle     = 0;
int CurrentAngle = 0;

// Target positions (in degrees)
const int HOME_ANGLE = 0;    // closed home position
const int OPEN_ANGLE = 90;   // angle you want when joystick pushed open
```

```
// Joystick thresholds for one-axis control (LX)
const int CENTER_VALUE   = 512;
const int DEADZONE       = 50;              // +/- range around center
const int UPPER_THRESH   = CENTER_VALUE + DEADZONE; // > this = "open side"
const int LOWER_THRESH   = CENTER_VALUE - DEADZONE; // < this = "close side"

// Latest joystick reading
int joyX = CENTER_VALUE;

// Edge detection (we only move when crossing into a zone)
bool wasHighPrev = false;  // previously in "open" zone
bool wasLowPrev  = false;  // previously in "close" zone

// Safety timeout (if we want to use it later)
unsigned long lastDataTime = 0;
const unsigned long TIMEOUT = 500; // ms

bool hasReceivedData = false;

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("=== Receiver Starting ===");

  WifiSerial.begin("Group95", "Orediggas", WifiPortType::Receiver);
```

CODE

Receiver Continued

```
  myStepper.setSpeed(10); // RPM, adjust as needed

  // Initialize data to safe center values
  data.LX = CENTER_VALUE;
  data.LY = CENTER_VALUE;
  data.RX = CENTER_VALUE;
  data.RY = CENTER_VALUE;
  data.Stop = 0;

  joyX = CENTER_VALUE;
}


void loop() {
  // transmitter reciever
  if (WifiSerial.checkForData()) {
    data = WifiSerial.getData();
    lastDataTime = millis();

    if (!hasReceivedData) {
      hasReceivedData = true;
      Serial.println("*** FIRST DATA RECEIVED ***");
    }
```

```
  // Debug print
  Serial.print("RX Data -> LX:"); Serial.print(data.LX);
  Serial.print(" LY:");  Serial.print(data.LY);
  Serial.print(" RX:");  Serial.print(data.RX);
  Serial.print(" RY:");  Serial.print(data.RY);
  Serial.print(" Stop:"); Serial.print(data.Stop);

  // Use only LX for open/close control
  joyX = data.LX;

  if (data.Stop == 1) {
    Serial.println(" -> STOP");
  } else if (joyX > UPPER_THRESH) {
    Serial.println(" -> JOYSTICK OPEN SIDE");
  } else if (joyX < LOWER_THRESH) {
    Serial.println(" -> JOYSTICK CLOSE SIDE");
  } else {
    Serial.println(" -> JOYSTICK CENTER");
  }
}

// If STOP is pressed, don't move the stepper this loop
if (data.Stop == 1) {
  return; // just listen for new data
}
```

```
  // Stepper Control

  // Current zone based on joystick X
  bool isHigh = (joyX > UPPER_THRESH); // joystick pushed to "open" side
  bool isLow  = (joyX < LOWER_THRESH); // joystick pushed to "close" side
  // if it's between LOWER and UPPER, it's in the deadzone/center

  // Rising edge into HIGH zone: go to OPEN_ANGLE
  if (isHigh && !wasHighPrev) {
    NewAngle = OPEN_ANGLE;                          // target angle
    CurrentAngle = stepperGoTo(NewAngle, CurrentAngle);
    Serial.println("Command: GO TO OPEN_ANGLE");
  }

  // Rising edge into LOW zone: go back to HOME_ANGLE (0)
  if (isLow && !wasLowPrev) {
    NewAngle = HOME_ANGLE;                          // back to home (0°)
    CurrentAngle = stepperGoTo(NewAngle, CurrentAngle);
    Serial.println("Command: GO TO HOME_ANGLE");
  }

  // Update previous zone states for next loop
  wasHighPrev = isHigh;
  wasLowPrev  = isLow;
}
```