Project Title: Waste on Wheels

Group members: Hae Chan Park, Lydia Huang, Noah Shih

1 Project Goal and Functionality

The prototype is a remote-controlled waste bin that contains one trash and one recycling compartment, which will open when you wave your hand near each of them. It can move around a predesigned obstacle course/map of UCSB in a square or circular pattern. It can stop immediately or at specific colors with the click of a button.

The trash and recycling bin uses 2 ultrasonic sensors for motion-sensing, which are paired with 2 servo motors to open lids. The L298N motor driver allows it to move forward, backward and turn, and we programmed it to continuously run in a square route. By sending inputs in the Serial Monitor, you can communicate to the Arduinos on the car through the transmitter and receiver to stop the car whenever you want. The user can also use the Pixy2 Camera to view colors on the ground, and then stop at any of the colors you send to the Arduino through the Serial Monitor.

2 Hardware Components

1. 3 Arduinos (one on car chassis and one with computer) (Arduino website $23 each)
2. nRF24L01 transmitter + receiver (ECE shop $6.39 each) for radio communication
3. 4 wheeled chassis with brushed DC motors (Amazon $20.99)
4. L298N Motor Driver/H Bridge (ECE shop $3.46)
5. 2 servo motors SG90 for opening the trash and recycling lid (from kits)
6. 2 ultrasonic sensors for opening trash and recycling lid (ECE shop $3.38)
7. 2 6AA batteries holders (Amazon $7.99 for both) to power the Arduino and DC motors
8. 1 Pixy2 CMUcam5 Camera (SparkFun $64.50) for color detection
9. 1 foam board (28*40 inch) for trash and recycling container (Target $10.99)

3 Design timeline

Week 6: Bought all the components. Drew schematics for most of the wiring of the parts to the Arduinos. Constructed the circuit for trash lid opening (ultrasonic sensors and servo motors). Soldered capacitors on brushed DC motors.

Week 7: Coded the trash lid opening circuit. Soldered all wires onto DC motors. Assembled the car chassis. Tested basic car movement. Began testing Pixy2 camera.

Week 8: Tested transceiver/receiver and SPI communication between Arduinos. Built the obstacle course. Finished Pixy2 color detection. Finished car movement and turning.

Week 9: Assembled all the hardware together (car chassis, receiver, pixy2, trash can, servo motors, etc.). Finished the code for Pixy 2 camera with the car movement. Prepared for the final presentation and science fair.

Week 10: No lab section

4 Software Design

The software of this project was split between three different files: the transmitter/controller file, the master file, and the slave file. Majority of the functionality of the prototype is located on the slave file while conditionals and transceiver components are found on the other two files.

(Because three Arduinos were used for this project, three files were used. Although long, the whole code is shown in order to explain all the moving parts as there are many. Comments are colored in green.)

Transmitter File

```
//Transmitter Code File
#include <SPI.h>
#include <RF24.h>


//initialize radio pins for transmitter and variables
RF24 radio(7,8);
const byte address[6] = "00001";
```

```
char input[] = "";
String readString;
const rf24_datarate_e dataRate = RF24_250KBPS;


//initialize serial monitor and radio, and make sure radio is working
void setup() {
  Serial.begin(9600);
  delay(50);
  if (radio.begin()) {
    Serial.println("Radio okay");
  } else {
    Serial.println("Radio begin failed");
  }
  radio.setPALevel(RF24_PA_LOW);
  radio.openWritingPipe(address);
  radio.stopListening();
}


void loop() {
    //stores the serial input as a variable and string
    while (Serial.available()) {
        delay(2);  //delay to allow byte to arrive in input buffer
        char input = Serial.read();
        readString += input;
    }
    //if there is a readString, then store it as an integer.
    if (readString.length() >0) {
       int a = readString.toInt();
       Serial.println("2nd part wokring");

       // If the integer value is 1, then send transmit the integer to the reciever through the radio
communication
       if(a == 1){ //1 represents color green
            Serial.println("3rd part working");
            Serial.println(a);
            //radio.openWritingPipe(address);
            radio.write(&a,sizeof(a));

        }
        else if(a == 2){ //2 represents color red
            Serial.println("3rd part working");
            Serial.println(a);
            //radio.openWritingPipe(address);
            radio.write(&a,sizeof(a));
        }
        else if(a == 3){ //3 represents color blue
            Serial.println("3rd part working");
            //radio.openWritingPipe(address);
            radio.write(&a,sizeof(a));
            Serial.println(a);
        }
        else if(a == 4){ //4 represents color yellow
            Serial.println("3rd part working");
            radio.write(&a,sizeof(a));
```

```
            Serial.println(a);
        }
        else if(a == 5){ //makes car start moving
            Serial.println("3rd part working");
            radio.write(&a,sizeof(a));
            Serial.println(a);
        }
        else if(a == 6){ //makes car stop
            Serial.println("3rd part working");
            radio.write(&a,sizeof(a));
            Serial.println(a);
        }
    //reset readString to length of 0
    readString="";
  }
  delay(5);
}
```

Master File

```
#define SOFTSPI
// Libraries used for the project
// SoftSPI specifically used to fix the issue of having limited BUS' caused by
// the usage of the pixy2 camera and a transceiver module at the same time
#include <DigitalIO.h>
#include <SoftSPI.h>
#include <RF24.h>
#include <Pixy2.h>
#include <SoftwareSerial.h>
#include <Servo.h>

// Define the servo ojects from the servo libraries.
Servo servoleft;
Servo servoright;

// Set the ports for the Arduino master/slave relationship
SoftwareSerial ArduinoSlave(2,3);

//Define the pixy2 object
Pixy2 pixy;

// Define the receiver
RF24 radio(7,8);

// Define the address for the receiever module
byte addresses[][6]={"00001"};

// Global variables set
int currentVal = 0;
int c=100;
unsigned long clockCurrent;
long int clockReal0 = 0;
long int clockReal1 = 0;
const int interval = 250;
```

```
int ledState = LOW;
int ledState1 = LOW;

// All setup code are within the following
void setup() {

  // Setting he pinmodes if the ports used for the servo motor and ultrasonic sensor component of this
code
  pinMode(A0, OUTPUT);
  pinMode(A1, INPUT);
  pinMode(A2, OUTPUT);
  pinMode(A3, INPUT);
  servoleft.attach(10);
  servoright.attach(9);

  // Begin the serial monitor
  Serial.begin(9600);
  Serial.print("Starting...\n");

  // Begin the master/slave component
  ArduinoSlave.begin(9600);

  // Conditionals to ensure that the radio can be used
  if (radio.begin()){
    Serial.println("radio.begin succeeded!");

  }
  else{
    Serial.println("Not able to contact radio module"); // Used to see if the radio module is broken
  }
  radio.setPALevel(RF24_PA_LOW);
  radio.openReadingPipe(0,addresses[0]);
  radio.startListening();

  // Initialize the pixy camera
  pixy.init();

}

// Body of the code and the looped program of the project
void loop() {

  // Define the speed of the motor here
  int speed =100;

  // Set numBloks to the number of colored objects the pixy2 camera identifies
  int numBlocks = pixy.ccc.getBlocks();
  clockCurrent = millis();

  // If a connection to the radio module is made, this code will run a connection between the
transmitter and receiver to get a value, c
  if (radio.available()) {
    Serial.print("received: ");
    radio.read(&c,sizeof(c));
    Serial.println(c);
  } else
```

```
  Serial.println(" send FAILED!");

  // Set value of c to a more explanatory variable
  currentVal = c;

  // The next four conditionals will run through to check whether or not the camera detects the color
green, red, blue, or yellow
  // as well as if the color matches the transmitter input and sends a stopping cmd to the slave code
if it does
  Serial.println(currentVal);
  if (currentVal==1){
    if (numBlocks == 0){
      ArduinoSlave.write(5);
    }
    else if (numBlocks){
      int i;
      for (i = 0 ; i < numBlocks; i++){
        int signature = pixy.ccc.blocks[i].m_signature;
        if (signature == 1){
          ArduinoSlave.write(currentVal);
          Serial.println("First Branch");
          break;
        }
      }
    }
  }
  else if (currentVal ==2){
    if (numBlocks == 0){
      ArduinoSlave.write(5);
    }
    if (numBlocks){
      int i;
      for (i = 0 ; i < numBlocks; i++){
        int signature = pixy.ccc.blocks[i].m_signature;
        if (signature == 2){
          ArduinoSlave.write(currentVal);
          break;
        }
      }
    }
  }
  else if (currentVal == 3){
    if (numBlocks == 0){
      ArduinoSlave.write(5);
    }
    if (numBlocks){
      int i;
      for (i = 0 ; i < numBlocks; i++){
        int signature = pixy.ccc.blocks[i].m_signature;
        if (signature == 3){
          ArduinoSlave.write(currentVal);
          break;
        }
      }
    }
  }
```

```
  else if (currentVal == 4){
    if (numBlocks == 0){
      ArduinoSlave.write(5);
    }
    if (numBlocks){
      int i;
      for (i = 0 ; i < numBlocks; i++){
        int signature = pixy.ccc.blocks[i].m_signature;
        if (signature == 4){
          ArduinoSlave.write(currentVal);
          break;
        }
      }
    }
  }

  // Hard stop conditional just in case something goes wrong
  else if (currentVal == 6){
    ArduinoSlave.write(4);
  }

  // Force continue conditional to begin the movement
  else {

    ArduinoSlave.write(5);
    Serial.println("Else Branch");
    Serial.println("No condition read");
  }

  // Defining variables for ultrasonic sensor and motors
  int durationleft, distanceleft;
  int durationright, distanceright;

  // Conditional for ensuring the program only runs for one second utiizing a clock (trash component)
  if (clockCurrent - clockReal0 >= interval) {
    clockReal0 = clockCurrent;
    if (ledState == LOW) {
      ledState = HIGH;
      digitalWrite(A0, ledState);
    } else {
      ledState = LOW;
      digitalWrite(A0, ledState);
      durationleft = pulseIn(A1, HIGH); // Measure the pulse input in echo pin
      distanceleft = (durationleft/2) / 29.1;
    }
    if (distanceleft <= 50 && distanceleft >= 0) // if distance less than 0.5 meter and more than 0 (0
or less means over range)
      {
      servoleft.write(50);

    } else {
      servoleft.write(160);
    }
  }

  // Same program but for the recycling bin
```

```
  if (clockCurrent - clockReal1 >= interval) {
    clockReal1 = clockCurrent;
    if (ledState1 == LOW) {
      ledState1 = HIGH;
      digitalWrite(A2, ledState1);
    } else {
      ledState1= LOW;
      digitalWrite(A2, ledState1);
      durationright = pulseIn(A3, HIGH); // Measure the pulse input in echo pin
      distanceright = (durationright/2) / 29.1;

    }

    // Shuts the lid of the bins
    if (distanceright <= 50 && distanceright >= 0) // if distance less than 0.5 meter and more than 0
(0 or less means over range)
      {
      servoright.write(50);

    } else {
      servoright.write(160);
    }
  }
}
```

Slave File

```
// Libraries needed for this code
#include <SoftwareSerial.h>

// Defining the master/slave component
SoftwareSerial ArduinoMaster(2,3);

// Global variables defined here
int cmd;
const int In1pin =5; // for Arduino Uno must be one of the PWM pins : 3, 5, 6, 9, 10 , and 11
const int In2pin =9; // for Arduino Uno must be one of the PWM pins : 3, 5, 6, 9, 10 , and 11
const int In3pin =10; // for Arduino Uno must be one of the PWM pins : 3, 5, 6, 9, 10 , and 11
const int In4pin =11; // for Arduino Uno must be one of the PWM pins : 3, 5, 6, 9, 10 , and 11
const int EnablePin =12;
unsigned long clockCurrent;
long int clockReal0 = clockCurrent + 2950;
long int clockReal1 = clockCurrent + 3250;

// Setup function
void setup(){

  // These are the pinmodes for the motor/chassis movement
  pinMode ( In1pin , OUTPUT );
  pinMode ( In2pin , OUTPUT );
  pinMode ( In3pin , OUTPUT );
  pinMode ( In4pin , OUTPUT );
  pinMode ( EnablePin , OUTPUT );

  // Begin the serial monitor
  Serial.begin(9600);
```

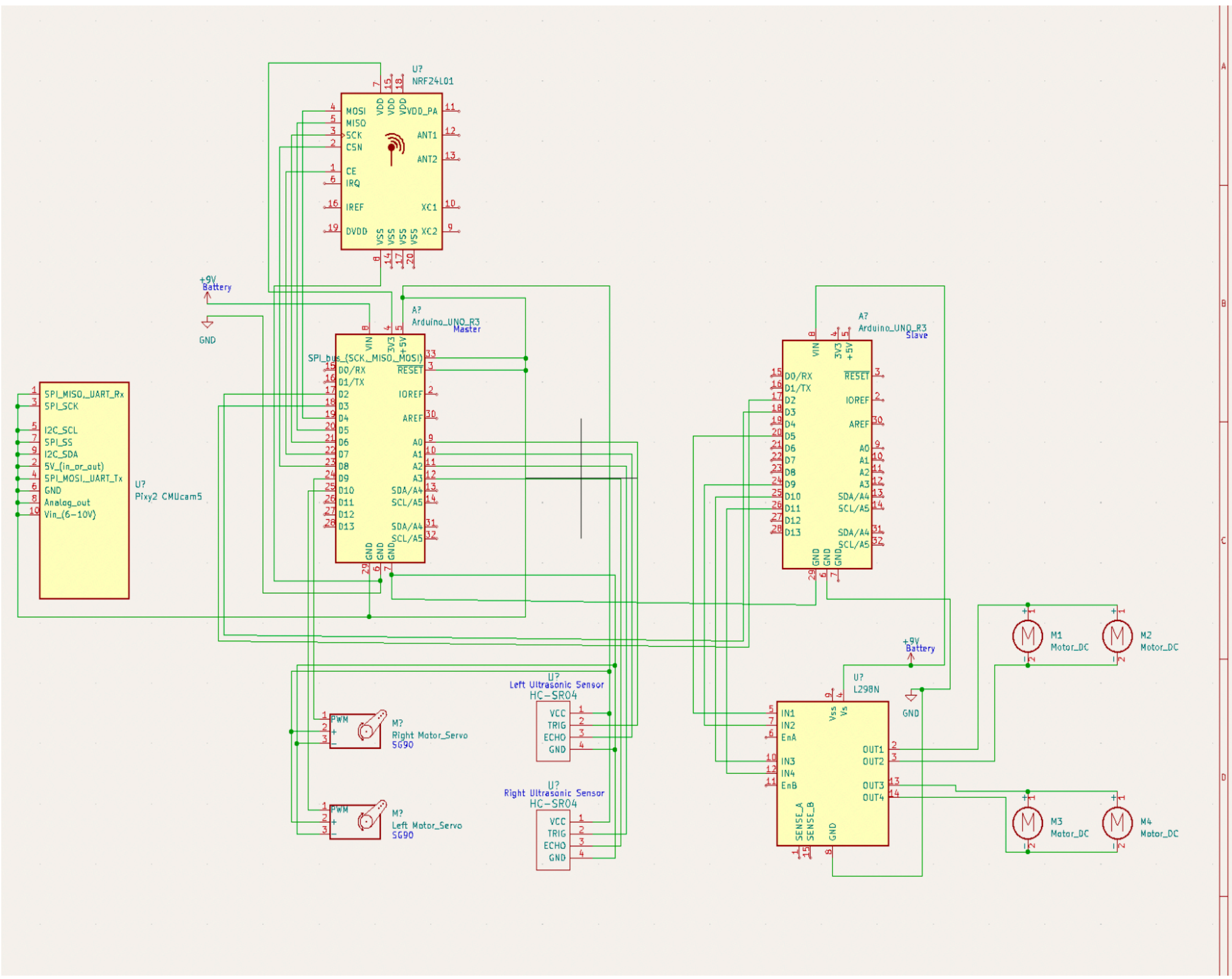
```
  // Begin the master/slave component
  ArduinoMaster.begin(9600);

}

// Loop function
void loop(){

  // Setting speeds for the forward and turn components
  int speed = 500;
  int turnspeed = 250;

  // Start the clock for the movement intervals
  clockCurrent = millis();

  // Serial Monitor readings for testing purposes
  Serial.print("Current: ");
  Serial.println(clockCurrent);
  Serial.print("Real0: ");
  Serial.println(clockReal0);

  // If the master/slave reading is available, the following set of conditionals will run
  if (ArduinoMaster.available()){

    // Reads the input from the master module (this code file is the slave module)
    cmd=ArduinoMaster.read();
    Serial.println(cmd);

    // If the values read from 1-4, indicating a color has been seen by the camera, the trashbot will
stop moving
    if (cmd == 1 || cmd == 2 || cmd == 3 || cmd == 4){
      analogWrite ( In1pin , LOW );
      digitalWrite ( In2pin , LOW );
      digitalWrite ( EnablePin , HIGH );

      digitalWrite ( In3pin , LOW );
      analogWrite ( In4pin , LOW );
      digitalWrite ( EnablePin , HIGH );

      // Reset the clock every time this condition is ran through
      clockReal0 = clockCurrent + 2950;
      clockReal1 = clockCurrent + 3250;

    }

    // If the above conditional is not ran, the following looped square movement will run instead
    else{
      if (clockCurrent <= clockReal0){

        analogWrite ( In1pin , turnspeed );
        digitalWrite ( In2pin , LOW );
        digitalWrite ( EnablePin , HIGH );

        analogWrite ( In3pin , LOW );
        digitalWrite ( In4pin , LOW );
```

```
        digitalWrite ( EnablePin , HIGH );

      }
      else if (clockCurrent <= clockReal1 && clockCurrent >= clockReal0){

        analogWrite ( In1pin , speed );
        digitalWrite ( In2pin , LOW );
        digitalWrite ( EnablePin , HIGH );

        digitalWrite ( In3pin , LOW );
        analogWrite ( In4pin , speed );
        digitalWrite ( EnablePin , HIGH );
        Serial.println("Forward");
      }
      else if (clockCurrent >= clockReal1){
          Serial.println("added");
          clockReal0 +=3250 ;
          clockReal1 +=3250 ;
      }
    }
  }
}
```

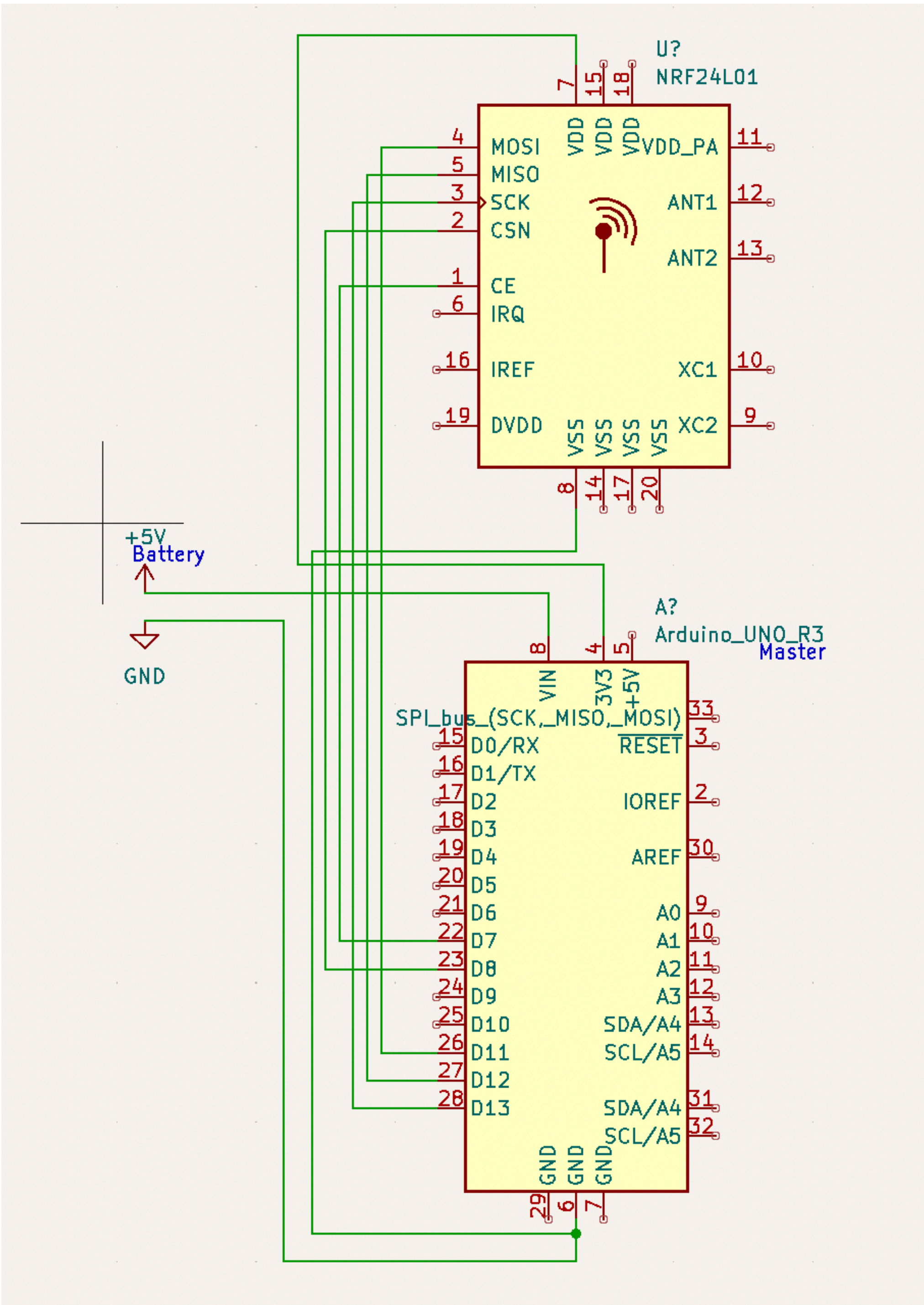
5 Circuit Schematics

The following schematics represent the circuits built for this project, and they were created using KiCAD. There are two schematics as there were two primary modules driving the prototype: the trashbot/receiver end and the controller/transmitting end. The transmitter is used to send numbers to the bot, determining the movement of the bot.

Chassis + pixy2 camera + servo motors & ultrasonic sensors + master&slave arduino + receiver

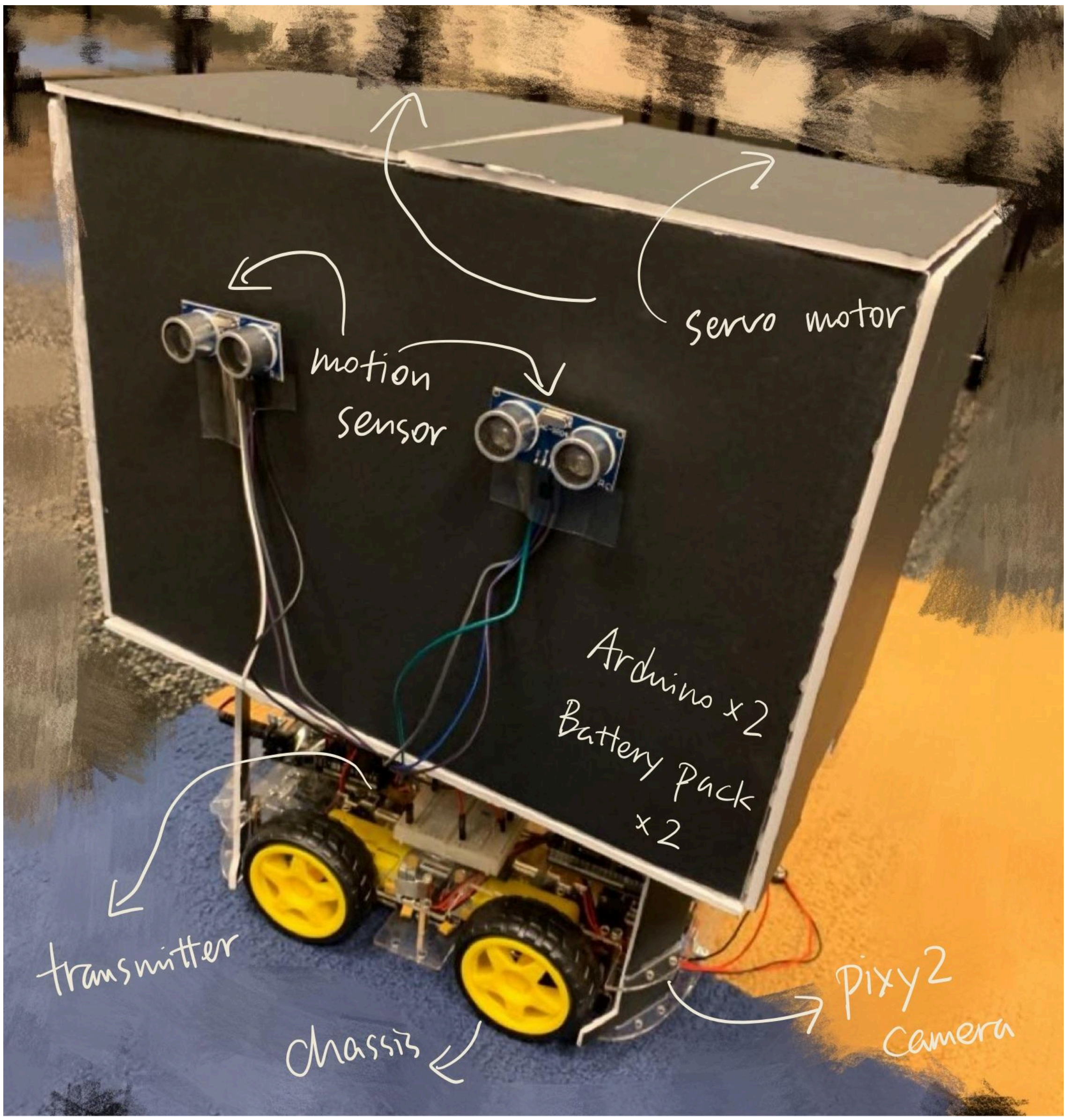


Transmitter module connected to computer

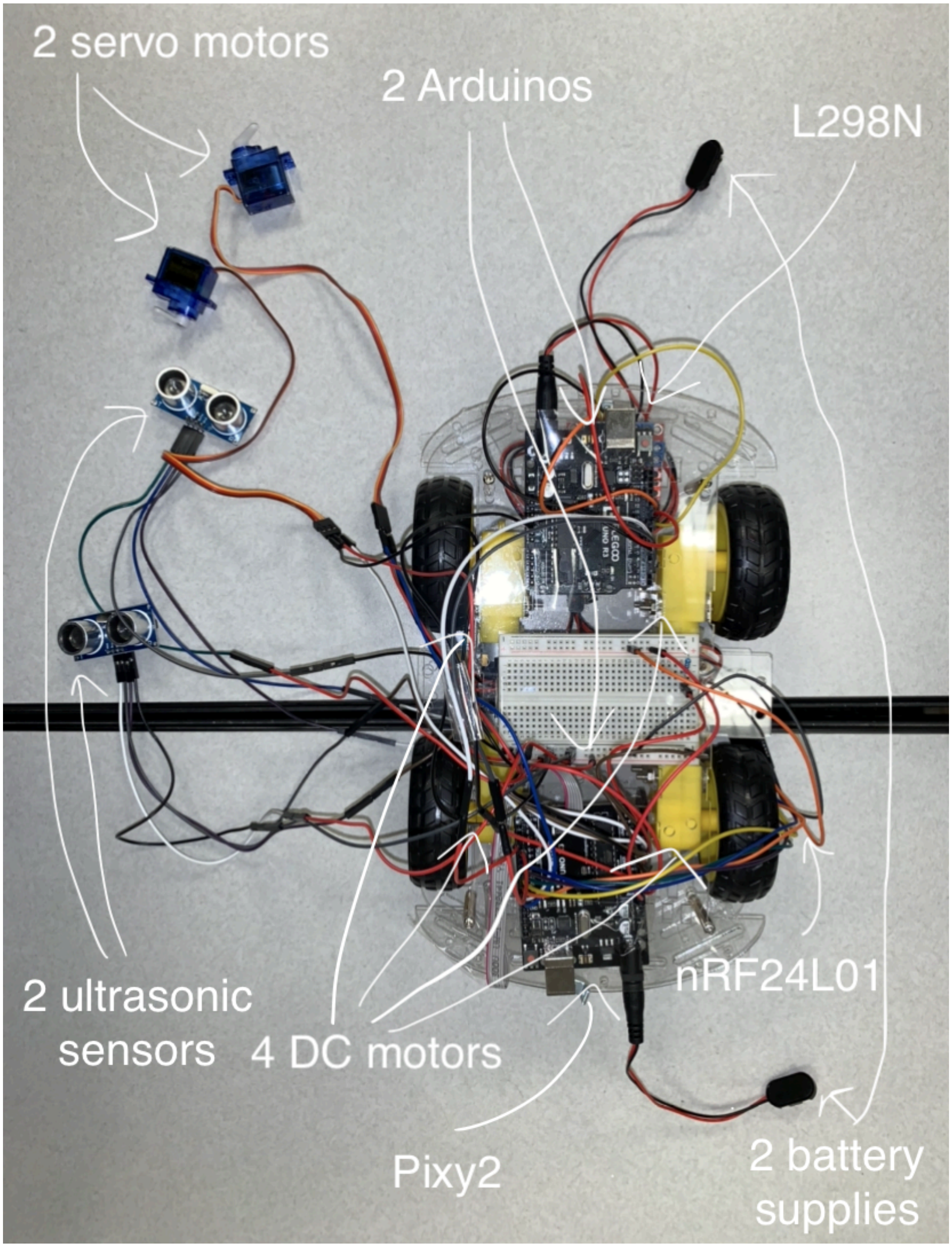


6 Circuit Prototype

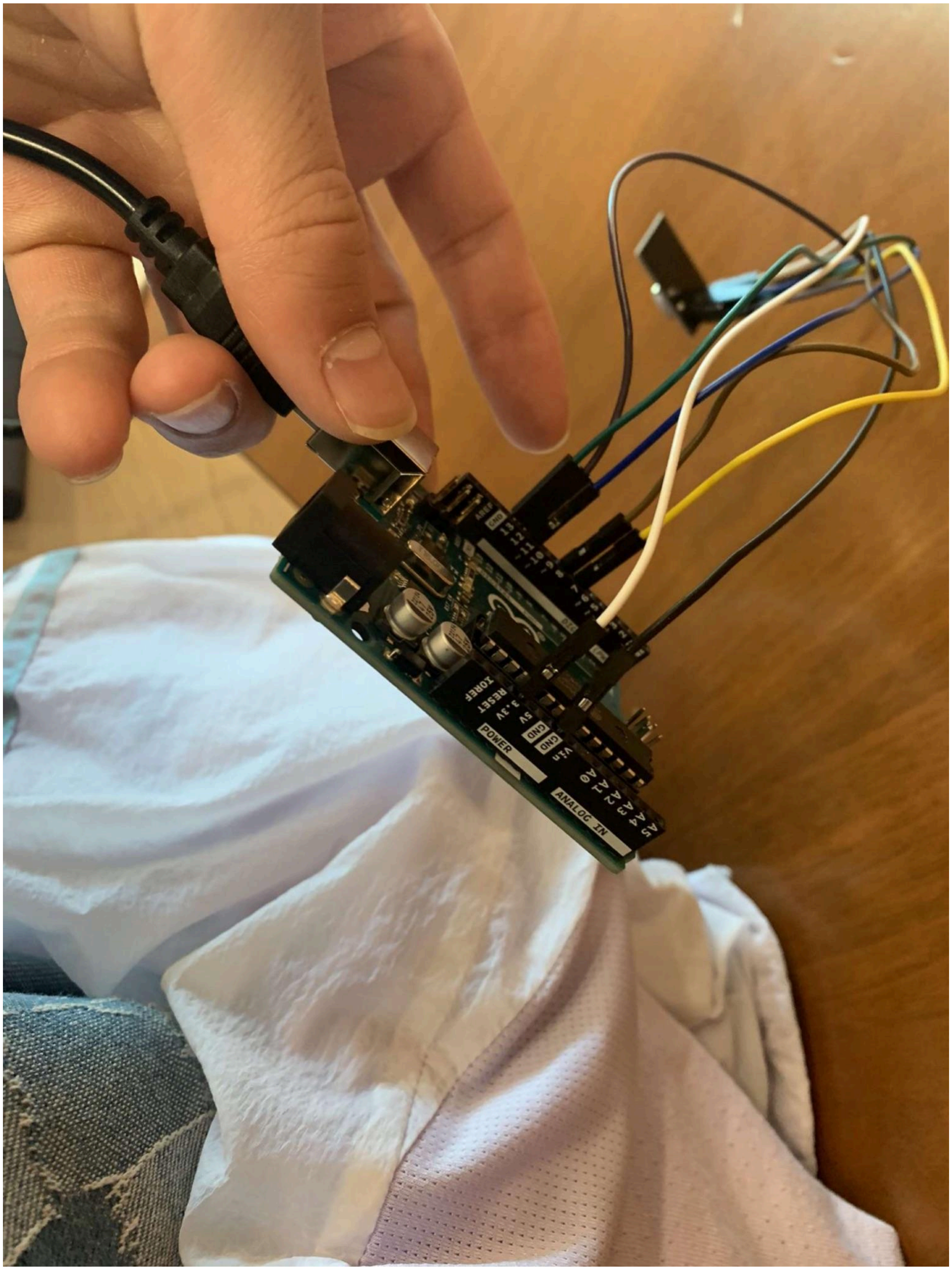
The following three images show the finished prototype. Three Arduino microcontrollers were utilized in the circuit prototype of this project, two of which were bound together using a master/slave library (located on the chassis). The other Arduino just has a radio transmitter wired to it.



The image above labels all the working components of the trashbot as well as a holistic view of the project.



The image above is the car without the trash can on top. Note: The Pixy2 camera, L298N motor driver, and nRF24L01 are not visible as they are underneath.



The image above shows the transceiver with the Arduino that would be connected to the computer.

7 Prototype Tests

Pre-assembly Tests

1. After soldering capacitors onto the motors of the chassis, a small test was done to ensure that the motors would work. The test was successful.
2. Before adding the servo and ultrasonic sensor components onto the prototype chassis, a test was done to ensure that the servos would turn flawlessly when the ultrasonic sensor recorded a certain distance value. The test was successful.
3. The Pixy2 camera was then tested and proved to be the most difficult to test due to the accuracy of the color reading on the camera itself and also its integration into the current circuitry. This caused issues with the transceiver modules. However, the test was successful.
4. The final test was the transceiver test with Pixy2 camera, which proved to be the most challenging test, however, was resolved by utilizing the SoftSPI library. The test was successful.

Final Tests

1. Tests for turning were done numerous times in order to ensure that the robot turned as close to 90 degrees as possible. Ideally, an accelerometer could have been used to ease the process, however, the issue was overall solved.
2. Tests for the speed of the prototype were also done. The rationale was that the camera had a deficiency where if the speed of the prototype was too fast, it was incapable of recording the colors fast enough. Adjustments were made through these tests.
3. Reliability tests were also done, however, much of these tests were done while also doing other tests such as tests for speed and turning. Small issues such as wires disconnecting from the breadboard and motors becoming weaker as more tests were done were observed, however, the functionality of the prototype was overall in ideal conditions.