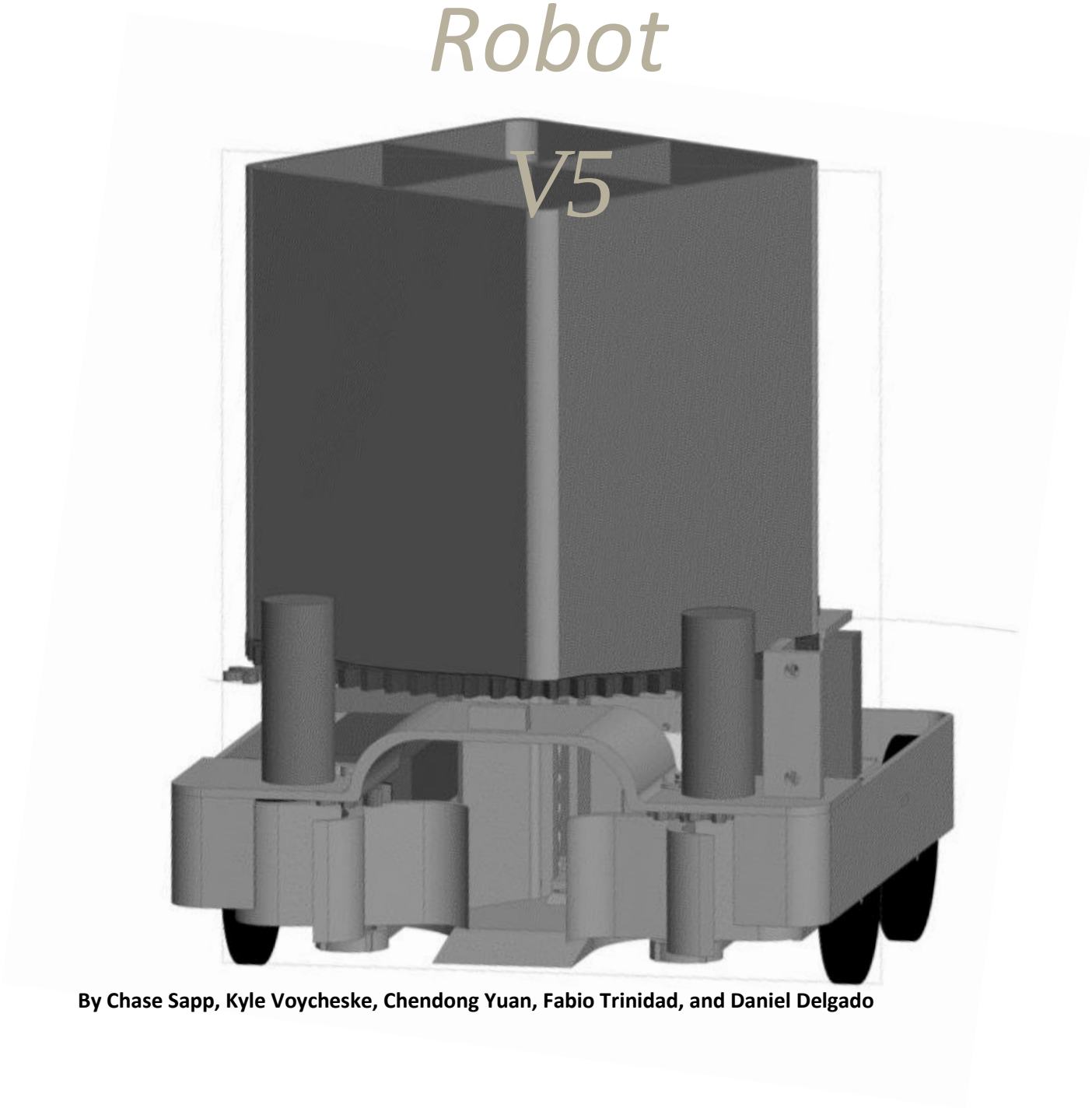
Operational Manual

Southeast Con 2019

Robot

V5



By Chase Sapp, Kyle Voycheske, Chendong Yuan, Fabio Trinidad, and Daniel Delgado

Operation Manual

Assembling the Robot 1

Circuit Layout 6

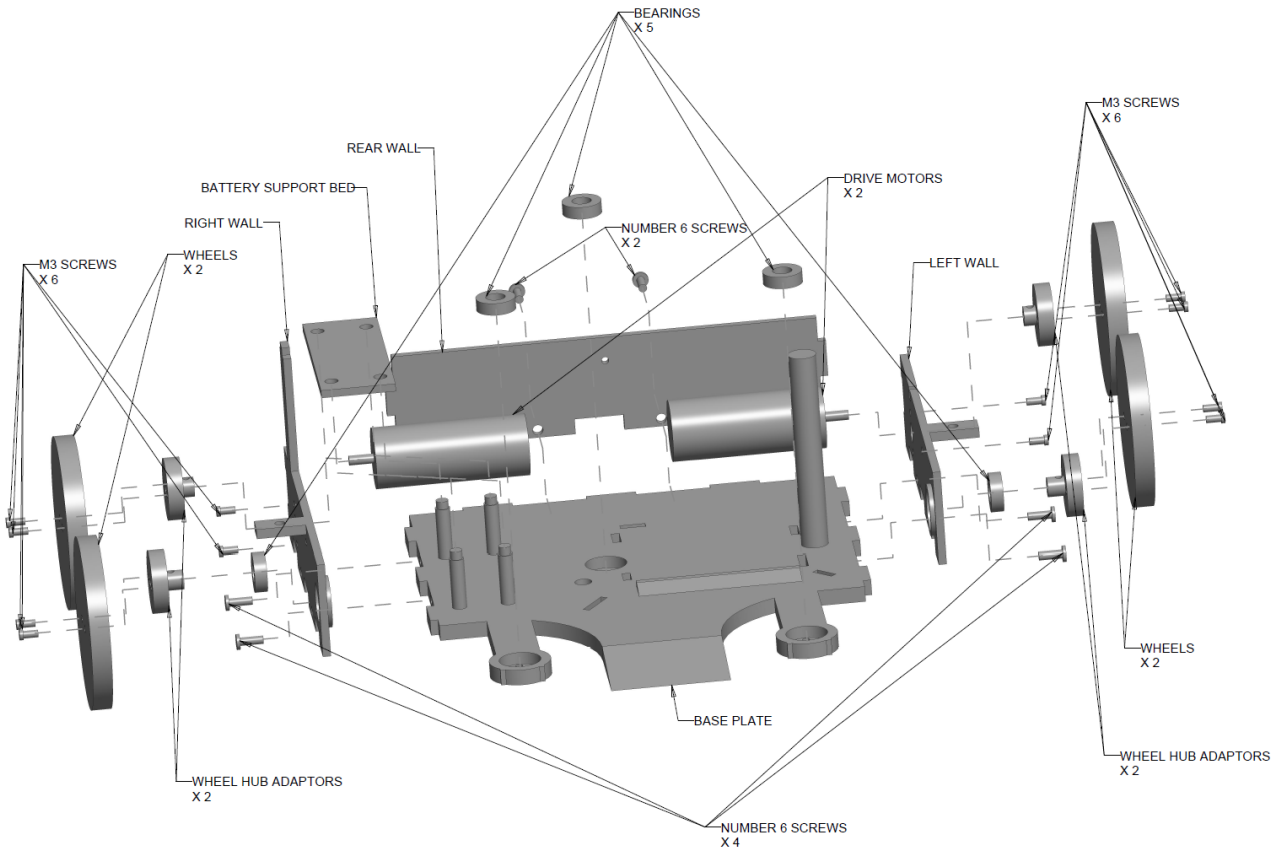
Code Guidance 7

Operation 8

Troubleshooting 9

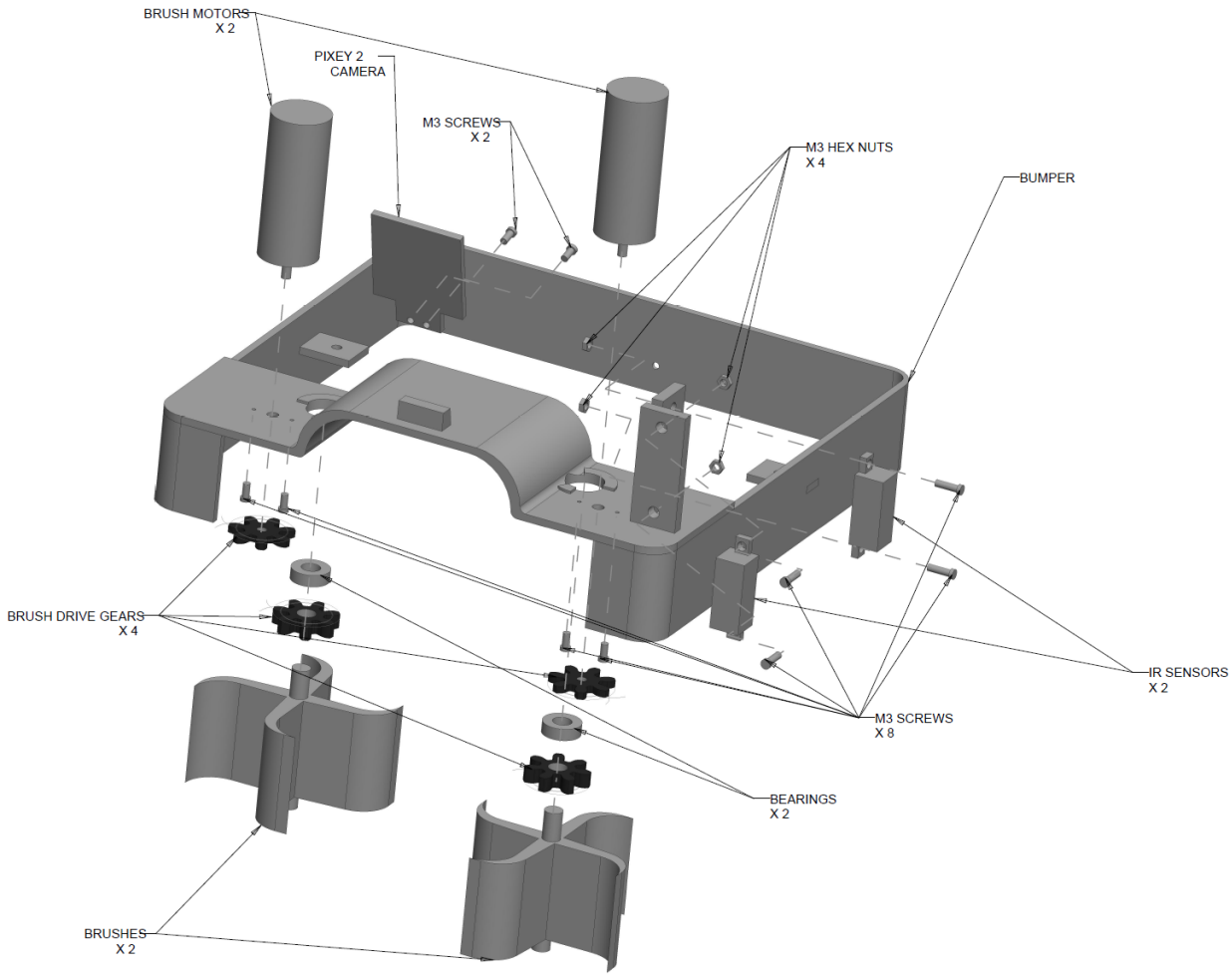
Assembling the Robot

The Southeast Con 2019 Robot has a complex structural design due to the size restrictions of the competition and incorporation the mechanisms required to quickly identify, gather and sort debris. To assemble the robot it recommended to follow these steps to ensure all the parts attached correctly so that the robot is structurally sound, and all the necessary mechanisms work correctly.



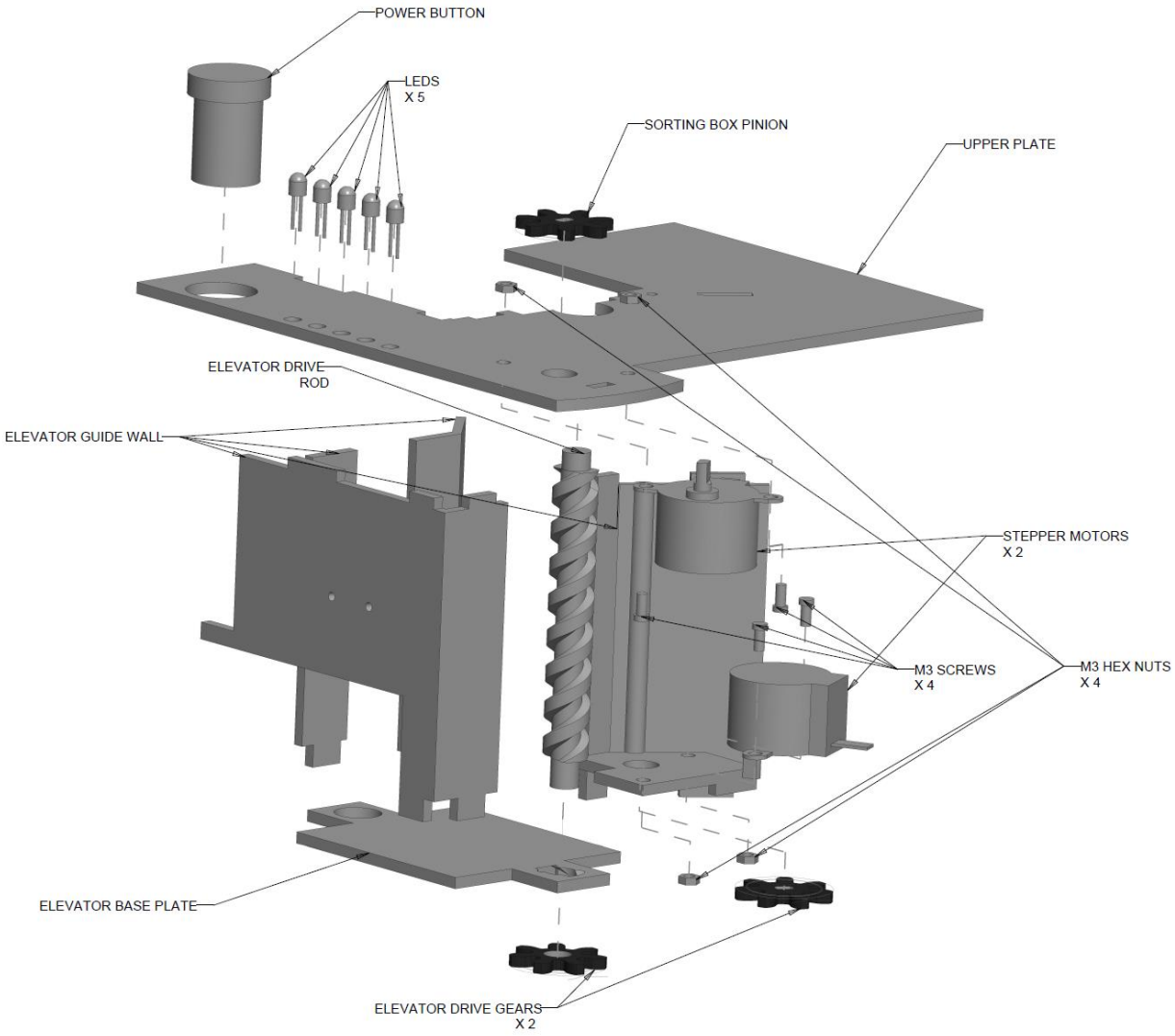
To assemble the robot's drive assembly please follow these steps:

1. Attach the drive motors to the left and right walls using M3 machine screws.
2. Press fit the bearings into the left and right walls.
3. Attach the wheels to the wheel hub adaptors using M3 machine screws.
4. Press fit the wheel hub adaptors into the bearings and onto the drive motors shaft.
5. Press fit three bearings into the base plate.
6. Screw the right, left and rear walls to the base plate using number 6 sheet metal screws.
7. Place batter support bed on top of four support beams near the right wall.



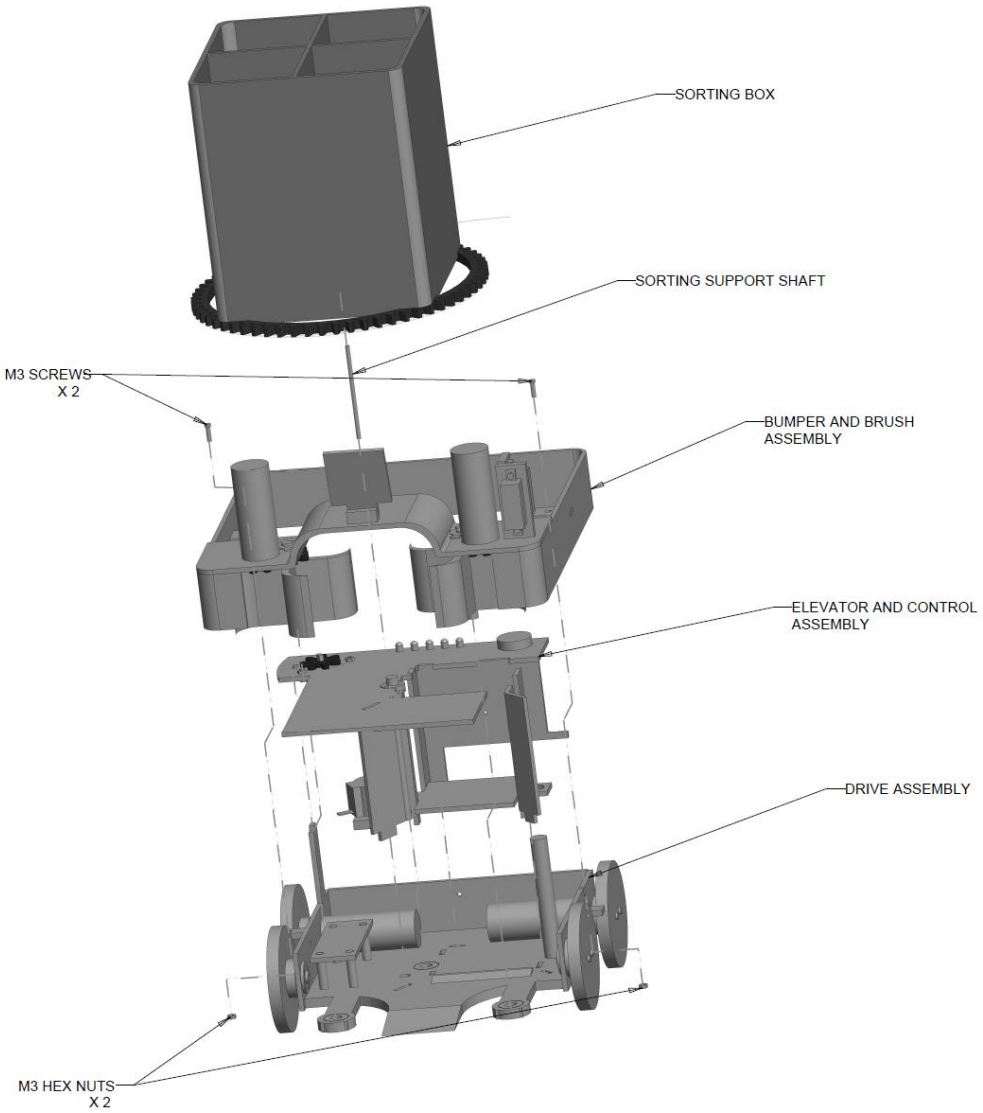
To assemble the robot's bumper and brush assembly please follow these steps:

1. Attach the IR sensors to the bumper using M3 machine screws and hex nuts.
2. Press fit the bearings to the bumper.
3. Attach the brush motors to the assembly using M3 machine screws.
4. Press fit the brush drive gears to the brush motors and brushes.
5. Press fit the brushes into the bearings ensuring the gears mesh properly.
6. Attach the Pixey 2 camera to the bumper using M3 machine screws.



To assemble the robot's elevator and control assembly please follow these steps:

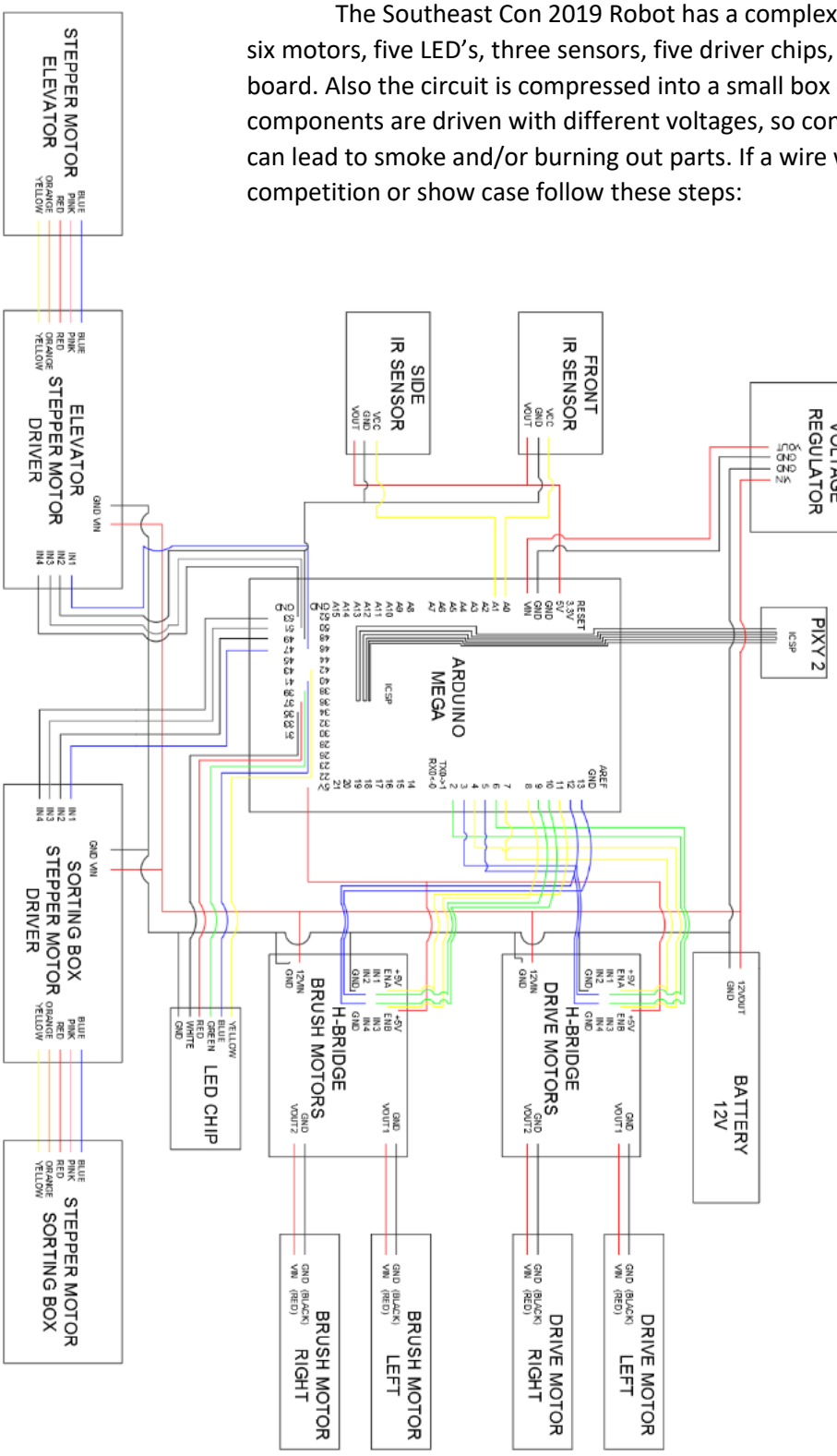
1. Attach the stepper motors to the upper plate and elevator guide wall using M3 machine screws and hex nuts.
2. Press fit the elevator drive gear and the sorting box pinion to the stepper motors.
3. Thread the elevator drive rod through the elevator base plate.
4. Press fit the remaining elevator drive gear to the elevator drive rod.
5. Insert elevator guide wall tabs into the upper plate.
6. Insert the LEDs and power button into the upper plate.



To assemble final robot please follow these steps:

1. Attach the elevator and control assembly to the drive assembly using the tabs on the elevator guide wall.
2. Attach the bumper assembly to the drive assembly using M3 machine screws and hex nuts.
3. Insert the sorting support shaft through the upper plate and into the guide wall.
4. Place the sorting box on top of the sorting support shaft ensuring that the gears mesh properly.

Circuit Layout



The Southeast Con 2019 Robot has a complex circuit design. This is due to having six motors, five LED's, three sensors, five driver chips, 12V battery, and an Arduino Mega on board. Also the circuit is compressed into a small box of about 9in by 9in by 4in. Lastly the components are driven with different voltages, so connecting wire in the wrong location can lead to smoke and/or burning out parts. If a wire would disconnect during the competition or show case follow these steps:

1. Locate each end of the wire.
2. Write down the component that it was connected to. If this side does not have a component connected, then check nearby devices while referencing this circuit layout if any pins are missing a wire. Write down that component.
3. Write down the pin number on the Arduino Mega where the other side of the wire was connected. If this side is not connected to a pin, then check nearby pins while referencing this circuit layout if any pins are missing a wire. Write down that that pin number.
4. Highlight the route on the Circuit Layout where it should be connected.
5. Follow the highlighted route and connect the pins to the proper location.

Code Guidance

During the competition or when making adjustments to the robot a test is necessary for any components. In this section will provide test code for the Arduino IDE. It should be noted to **TURN OFF THE ROBOT BY THE ON ON/OFF SWITCH BEFORE CONNECTING THE ROBOT TO A COMPUTER.** This avoids overloading the robot with too much voltage and/or current.

Brush Motor Test

Brush Motor test code is design to test the front brush motors of the Southeast Con 2019 Robot. These motor are connected at the front of the robot on the bumper. The supply and ground is connected to the *H-Bridge Brush Motor*, with a voltage that can range from zero to twelve volts. To control the speed of the brushes, simply set **brush_speed** from zero to one hundred percent. One hundred percent is the maximum speed the motor can go before damaging the motor and/or wired connected to the motor. It is recommended not to go above forty percent to avoid damage to the brush gear, bearings, bumper, and/or brushes. The **brush_direction** affects the direction in which the brushes are spinning. Setting it equal to two will cause the brushes to spin inward. Setting it equal to one will cause the brushes to spin outward. Setting it equal to zero will cause the brushes to stop. Only change the global variables. To view the code, please go to the end of the document.

Drive Motor Test

Drive Motor test code is design to test the drive motors of the Southeast Con 2019 Robot. These motor are connected to the side walls that are attached to base plate one. The supply and ground is connected to the *H-Bridge Drive Motor*, with a voltage that can range from zero to twelve volts. To control the speed of the wheels, simply set **driver_speed** from zero to one hundred percent. One hundred percent is the maximum speed the motor can go before damaging the motor and/or wired connected to the motor. It is recommended not to go below fifty percent, or the robot will not move. The **driver_direction** affects the direction in which the wheels are spinning. Setting it equal to two will cause the wheels to go forward. Setting it equal to one will cause the wheels to spin backward. Setting it equal to zero will cause the wheels to stop. Only change the global variables. To view the code, please go to the end of the document.

IR Sensor Test

IR sensor test code is design to test the drive motors of the Southeast Con 2019 Robot. These IR Sensors are connected to the bumper located near the left brush motor. The supply and ground is connected to the *Arduino Mega*, with a voltage that can range from zero to five volts. This code has two variables that return the distance from in front of the IR sensors. **SIR** or side IR sensor is located ninety degrees away from the front facing of the robot. This sensor is used to locate how far the sensor mass is away from the robot. **FIR** or front IR sensor is facing in the same direction of the opening of the robot. This sensor is used to tell the distance between the walls of the field and any opposing robots. Simply upload the code and open the serial monitor found in the Arduino IDE to see what the IR sensors are reading. To view the code, please go to the end of the document.

LED Color Test

LED Color Test code is design to test the LED's on the Southeast Con 2019 Robot. These LED's are connected to the bumper located near the rear by the ON/OFF switch. The supply and ground is connected to the *LED Chip*, with a voltage that can range from zero to five volts. This code sends on and off signals to each LED to cause them to flash. upload the code and observe the LED light flash. To view the code, please go to the end of the document.

Elevator/Sorting Box Test

Elevator/Sorting Box Test code is design to test the drive motors of the Southeast Con 2019 Robot. These mechanisms are connected to the center of the robot. The stepper motors are connected to their respective drivers, with a voltage that can range from zero to twelve volts. Both of these modules are tested with the same code. The code causes each module to travel at the same speed. To change the direction of the elevator or sorting box change the global variable **rotationController**. The **rotationController** is set to "true", then the stepper motors turn clockwise. If the **rotationController** is set to "false", then the stepper motors turn counter clockwise. To view the code, please go to the end of the document.

Operation

To operate the Southeast Con 2019 Robot, is mostly just setup. Upload the main driving code to the robot by USB connection. Make sure to not have the robot on when uploading the code. Once the code has been uploaded, place the robot in one of the color coded home corners of the playing field. Make sure to place the robot with the ON/OFF switch facing one wall and the brush motor side with no IR sensors facing the other. Then turn the robot on by hitting the ON/OFF switch. Lastly, the switch LED will light up, informing you that the robot is on and about to sort field of debris.

Troubleshooting

The Southeast con2019 Robotics is a tightly enclosed complex system that has a mixture of mechanical and electrical components. During competition or showcasing the robot errors are bound to occur. This section will cover how to recognize and resolve these errors.

Brushes

- Brushes are not rotating: If this error occurs then there are two possible issues that are occurring with the robot. The first issue is that the brush gear has fallen off the brush DC motor. To resolve this issue first, check the original gear for any damage. If there is no damage replace the gear back onto the brush motor while still grinding with the gear above the brushes. If the gear is broken, replace the part by 3-D printing it. If this does not resolve the issue then check if there are any issues with the H-bridge or brush motor. To test this upload the *Brush Motor Test* to the robot and observe what occurs. If the brushes do not rotate still, check if any debris is obstructing the rotation. Lastly, if there is no debris obstructing the brushes then the motors are burned out and need to be replaced.
- Brushes are spinning in wrong direction: If the brushes are spinning in the wrong direction then there are one possible errors that could be occurring. The issue can occur because the supply and ground were improperly connected to the motor from the *Brush Motor H-bridge*. To resolve this issue flip the wires connected to the motor.

Wheels

- Wheels are spinning freely: This error occurs when the internal cavity of the hub attached to the motor is stripped. To resolve this issue, replace the hub.
- Wheels are not spinning: This error occurs when the motor is not receiving power, the robot has hit a wall and/or the wheels have become stuck. To test if the motor is burned out remove the wheel and hub from the motor and upload the *Driver Motor Test*. If the motor's rotor is not rotating then the motor is stalled out and needs to be replaced. If the wheel has something clogged in the wheel, remove the debris

IR Sensors

- No values being returned: This error occurs when the wires connected wrong or fallen out. Compare the connection to the Circuit Layout and replace the wires as needed.
- Wrong Values being Returned: This error occurs when data yellow wires is not connect properly. Compare to the connection to the Circuit Layout and replace as needed.

Pixy 2

- The Pixy 2 is not returning values: This error occurs when the connection to the Arduino Mega is not connected properly. A sign of this error can be by a small LED below the camera of the Pixy 2 producing a red light. Rotate the pin connection 180 degrees, and the Pixy 2 will produce a multi-color lights.

Brush Motor Test

```
const int IN5=8;// wheel driver:   8
const int IN6=13;// wheel driver:  13
const int ENC=9;// wheel driver:   9
const int IN7=11;// wheel driver:  11
const int IN8=12;// wheel driver:  12
const int END=10;// wheel driver:  10

int brush_speed = 0; // 0% to 100%
int brush_direction = 0; // 2 = pull in  || 1 = pull out || 0 = stop

void setup()
{pinMode(IN5,OUTPUT);
pinMode(IN6,OUTPUT);
pinMode(ENC,OUTPUT);
pinMode(IN7,OUTPUT);
pinMode(IN8,OUTPUT);
pinMode(END,OUTPUT);}

void loop()
{
 brush_driver(brush_speed,brush_direction);
delay(100);

}

void brush_driver (int brush_speed,bool brush_direction)
{
  int Speed = brush_speed*2;
  if(brush_direction == 2)
```

```
  {
    Motor1_Forward(Speed);
    Motor2_Backward(Speed);
  }
  if(brush_direction == 1)
  {
    Motor1_Backward(Speed);
    Motor2_Forward(Speed);
  } else
  {
    Motor1_Brake();
    Motor2_Brake();
  }
}



void Motor1_Forward (int Speed)
{digitalWrite(IN5,HIGH);
digitalWrite(IN6,LOW);
analogWrite(ENC,Speed);
}
void Motor1_Backward (int Speed)
{
  digitalWrite(IN5,LOW);
  digitalWrite(IN6,HIGH);
  analogWrite(ENC,Speed);
}
void Motor1_Brake ()
{
  digitalWrite(IN5,LOW);
```

```
  digitalWrite(IN6,LOW);
}
void Motor2_Forward (int Speed)
{
  digitalWrite(IN7,HIGH);
  digitalWrite(IN8,LOW);
  analogWrite(END,Speed);
}
void Motor2_Backward (int Speed)
{
  digitalWrite(IN7,LOW);
  digitalWrite(IN8,HIGH);
  analogWrite(END,Speed);
}
void Motor2_Brake()
{
  digitalWrite(IN7,LOW);
  digitalWrite(IN8,LOW);
}
```

Drive Motor Test

```
const int IN1=2;
const int IN2=3;
const int ENA=7;
const int IN3=6;
const int IN4=5;
const int ENB=4;
#define PI 3.14159265358979323846264338327 95

const int driver_speed =50;// 40; 0% to 100%
```

```
const int driver_direction = 2; // 2 = forward || 1 = backward || 0 = stop
const int angle = 90;//Range from 0 to 90 degrees || 45 is straight || Greater then 45 is right || less then 45 is left


void setup()
{pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
pinMode(ENA,OUTPUT);
pinMode(IN4,OUTPUT);
pinMode(IN3,OUTPUT);
pinMode(ENB,OUTPUT);

Serial.begin(9600);
}

void loop()
{

 driver(driver_speed,driver_direction,angle);
 delay(100);
}

void driver (int driver_speed,int driver_direction, int angle)
{
 int Speed;

 if (driver_speed <= 100)
   Speed = driver_speed*2;
 else
   Speed = 200;
```

```
 double rad=(angle*PI)/180;
 int Speed_left = Speed*cos(rad);
 int Speed_right = Speed*sin(rad);


 if(driver_direction == 2)
 {
  Motor1_Forward(Speed_left);
  Motor2_Backward(Speed_right);
 }
 else if(driver_direction == 1)
 {
  Motor1_Backward(Speed_left);
  Motor2_Forward(Speed_right);
 } else
 {
  Motor1_Brake();
  Motor2_Brake();
 }
}


void Motor1_Forward (int Speed)
{digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);
analogWrite(ENA,Speed);
}
void Motor1_Backward (int Speed)
{
```

```
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  analogWrite(ENA,Speed);
}
void Motor1_Brake ()
{
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
}
void Motor2_Forward (int Speed)
{
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,Speed);
}
void Motor2_Backward (int Speed)
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENB,Speed);
}
void Motor2_Brake()
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}
```

IR Sensor Test

```
#define side_IR A0
#define front_IR A1
void setup() {
```

```
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  Serial.begin(9600);
}

void loop() {
  int SIR = analogRead(A0);// SIDE IR SENSOR VALUE
  int FIR = analogRead(A1);// FRONT IR SENSOR VALUE

  Serial.print("SIDE IR: ");
  Serial.print(SIR);

  Serial.print("\n");

  Serial.print("FRONT IR: ");
  Serial.print(FIR);



  Serial.print("\n");
  delay(1000);



}
```

LED Color Test

```
const int IN1=2;
const int IN2=3;
const int ENA=7;
const int IN3=6;
const int IN4=5;
```

```
const int ENB=4;
#define PI 3.14159265358979323846264338327 95


const int driver_speed =50;// 40; 0% to 100%
const int driver_direction = 2; // 2 = forward || 1 = backward || 0 = stop
const int angle = 90;//Range from 0 to 90 degrees || 45 is straight || Greater then 45 is right || less then 45 is left


void setup()
{pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
pinMode(ENA,OUTPUT);
pinMode(IN4,OUTPUT);
pinMode(IN3,OUTPUT);
pinMode(ENB,OUTPUT);

Serial.begin(9600);
}

void loop()
{

  driver(driver_speed,driver_direction,angle);
  delay(100);
}

void driver (int driver_speed,int driver_direction, int angle)
{
  int Speed;
```

```
 if (driver_speed <= 100)
   Speed = driver_speed*2;
 else
   Speed = 200;

 double rad=(angle*PI)/180;
 int Speed_left = Speed*cos(rad);
 int Speed_right = Speed*sin(rad);


 if(driver_direction == 2)
 {
   Motor1_Forward(Speed_left);
   Motor2_Backward(Speed_right);
 }
 else if(driver_direction == 1)
 {
   Motor1_Backward(Speed_left);
   Motor2_Forward(Speed_right);
 } else
 {
   Motor1_Brake();
   Motor2_Brake();
 }
}


void Motor1_Forward (int Speed)
{digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);
```

```
analogWrite(ENA,Speed);
}
void Motor1_Backward (int Speed)
{
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  analogWrite(ENA,Speed);
}
void Motor1_Brake ()
{
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
}
void Motor2_Forward (int Speed)
{
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  analogWrite(ENB,Speed);
}
void Motor2_Backward (int Speed)
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  analogWrite(ENB,Speed);
}
void Motor2_Brake()
{
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
}
```

Elevator/Sorting Box Test

```
int bluePin = 47;
int pinkPin = 49;
int yellowPin =51;
int orangePin = 53;

int currentStep = 0;
bool rotationController = false;
void setup() {
  Serial.begin(9600);

  pinMode(bluePin, OUTPUT);
  pinMode(pinkPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(orangePin,OUTPUT);

  digitalWrite(bluePin, LOW);
  digitalWrite(pinkPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(orangePin, LOW);
}

void loop() {

  //Comment out the Serial prints to speed things up
  //Serial.print("Step: ");
  //Serial.println(currentStep);
if (rotationController == true)
{
  switch(currentStep){
```

```
    case 0:
      digitalWrite(bluePin, HIGH);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    case 1:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, HIGH);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    case 2:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, HIGH);
      digitalWrite(orangePin, LOW);
      break;
    case 3:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, HIGH);
      break;
  }
}
else
{
    switch(currentStep){
    case 0:
```

```
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, HIGH);
      break;
    case 1:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, HIGH);
      digitalWrite(orangePin, LOW);
      break;
    case 2:
      digitalWrite(bluePin, LOW);
      digitalWrite(pinkPin, HIGH);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    case 3:
      digitalWrite(bluePin, HIGH);
      digitalWrite(pinkPin, LOW);
      digitalWrite(yellowPin, LOW);
      digitalWrite(orangePin, LOW);
      break;
    }
}
  currentStep = (++currentStep < 4) ? currentStep : 0;

  //2000 microseconds, or 2 milliseconds seems to be
  //about the shortest delay that is usable.  Anything
  //lower and the motor starts to freeze.
```

```
  //delayMicroseconds(2250);
  delay(2);
}
```