**Appendix A: PreLab State Diagrams**

**Avoid Obstacle:**

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| States | Input | Output | Next State |
| Random Direction | N/A, No Obstacle | Direction | Go to Goal |
| Go to Goal | Direction | Power Motor | Obstacle |
| Obstacle | Sensor Data | Yes  No | Obstacle Direction  Random Direction |
| Obstacle Direction | Sensor Data & Obstacle | Front, Left, and Right  Front and Left, Left  Front, Right, Front and Right | Turn 180 Degrees and Go Forward  Go Back and Turn Right  Go Back and Turn Left |
| Turn 180 Degrees and Go Forward | Front, Left, and Right | Power Motor | Obstacle |
| Go Back and Turn Right | Front and Left, Left | Power Motor | Obstacle |
| Go Back and Turn Left | Front, Right, Front and Right | Power Motor | Obstacle |

**Smart Wander:**

Diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| States | Input | Output | Next State |
| Stop Moving | N/A, No Obstacle, All 4 Directions | Stop Motor | Obstacle |
| Obstacle | Sensor Data | Yes  No | Direction  Stop Moving |
| Direction | Sensor Data & Obstacle | Only 3 and less directions  All 4 Directions | Direction Vector  Distance  Stop Moving |
| Direction Vector | Only 3 and less directions | Calculate Obstacle Direction Vector | Go with speed and Opposite Direction |
| Distance | Only 3 and less directions | Calculate Obstacle Distance:  10<d<30  D<10 | Slow Speed  Fast Speed |
| Slow Speed | 10<d<30 | Set slow motor speed | Go with speed and Opposite Direction |
| Fast Speed | D<10 | Set fast motor speed | Go with speed and Opposite Direction |
| Go with speed and Opposite Direction | Obstacle Direction Vector and Speed | Power Motor | Obstacle |

**Smart Homing:**

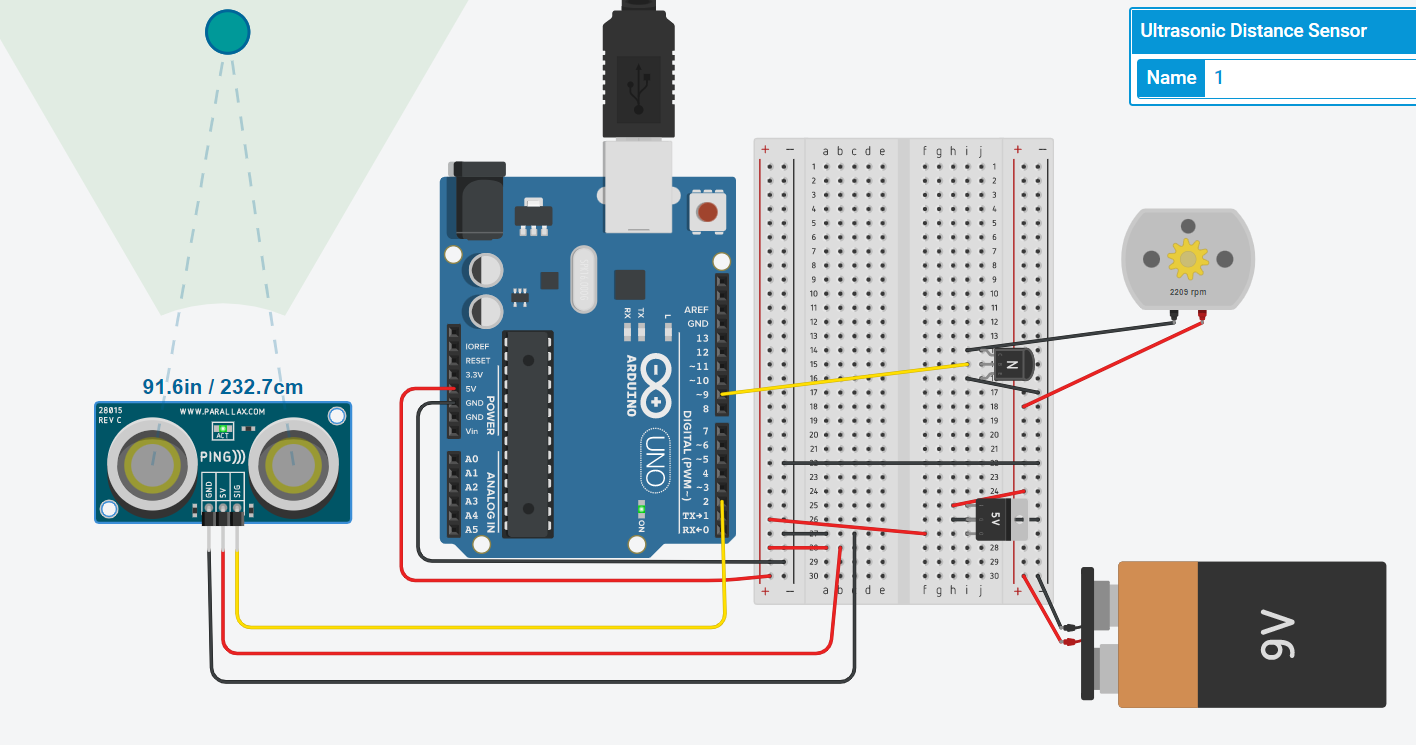
Diagram

Description automatically generated

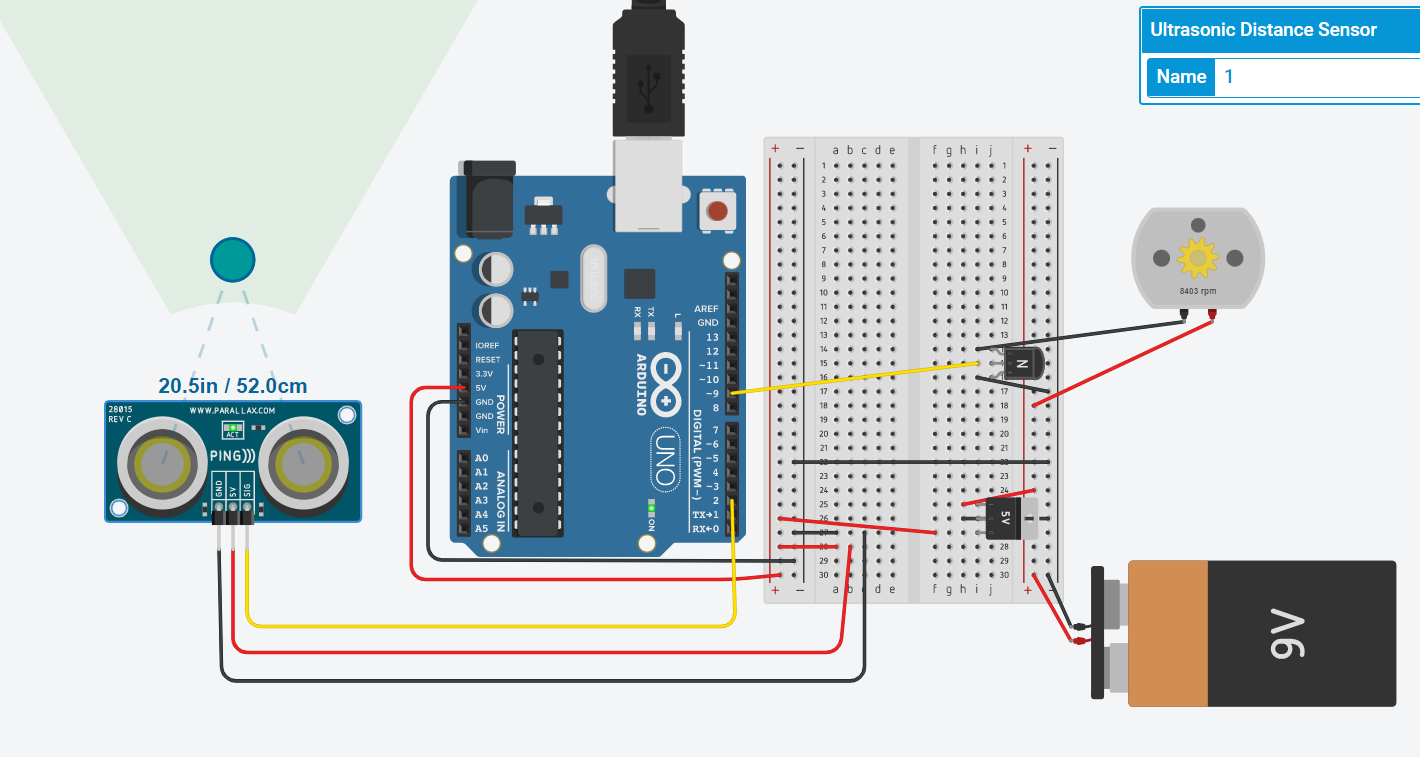
|  |  |  |  |
| --- | --- | --- | --- |
| States | Input | Output | Next State |
| Input Target Position | Target Position, Not Arrived | Save target position  Provide robot relative direction to target | Obstacle |
| Obstacle | Sensor Data | Yes  No | Obstacle Direction  Point to Target Position and Go 1 second |
| Obstacle Direction | Sensor Data & Obstacle | Front, Left, and Right  Front and Left, Left  Front, Right, Front and Right | Turn 180 Degrees and Go Forward  Go Back and Turn Right  Go Back and Turn Left |
| Turn 180 Degrees and Go Forward | Front, Left, and Right | Power Motor | Calculate and Save Direction and Distance to target |
| Go Back and Turn Right | Front and Left, Left | Power Motor | Calculate and Save Direction and Distance to target |
| Go Back and Turn Left | Front, Right, Front and Right | Power Motor | Calculate and Save Direction and Distance to target |
| Point to Target Position and Go | No Obstacle | Power Motor | Calculate and Save Direction and Distance to target |
| Calculate and Save Direction and Distance to target | Moving Direction and Distance | Calculate and Save Direction and Distance for comparison | Arrived |
| Arrived | Robot location and target position | Compare:  Yes  No | Stop  Input Target Position |

**Appendix B: Tinkercad Simulation**

**Shy Kid**



*The object is 91.6 inches away from the sensor. The motor is spinning at 2200 RPM*



*The object is 20.5 inches away from the sensor. The motor is spinning at 8400 rpm.*

**Shy Kid TinkerCad Code:**

int distanceThreshold = 0;

int cm = 0;

int inches = 0;

void setup()

{

pinMode(9, OUTPUT);

}

long readUltrasonicDistance(int triggerPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(triggerPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(triggerPin, HIGH);

}

void setmotorspeed(int speed){

analogWrite(9,speed);

}

void loop()

{

// set threshold distance to activate LEDs

distanceThreshold = 300;

// measure the ping time in cm

cm = 0.01723 \* readUltrasonicDistance(2);

// convert to inches by dividing by 2.54

inches = (cm / 2.54);

Serial.print(cm);

Serial.print("cm, ");

Serial.print(inches);

Serial.println("in");

if (cm > distanceThreshold) {

setmotorspeed(0);

}

if (cm <= distanceThreshold && cm > distanceThreshold - 100) {

setmotorspeed((300-cm)/2);

}

if (cm <= distanceThreshold - 100 && cm > distanceThreshold - 250) {

setmotorspeed((300-cm)/2);

}

if (cm <= distanceThreshold - 250 && cm > distanceThreshold - 350) {

setmotorspeed(255);

}

if (cm <= distanceThreshold - 300) {

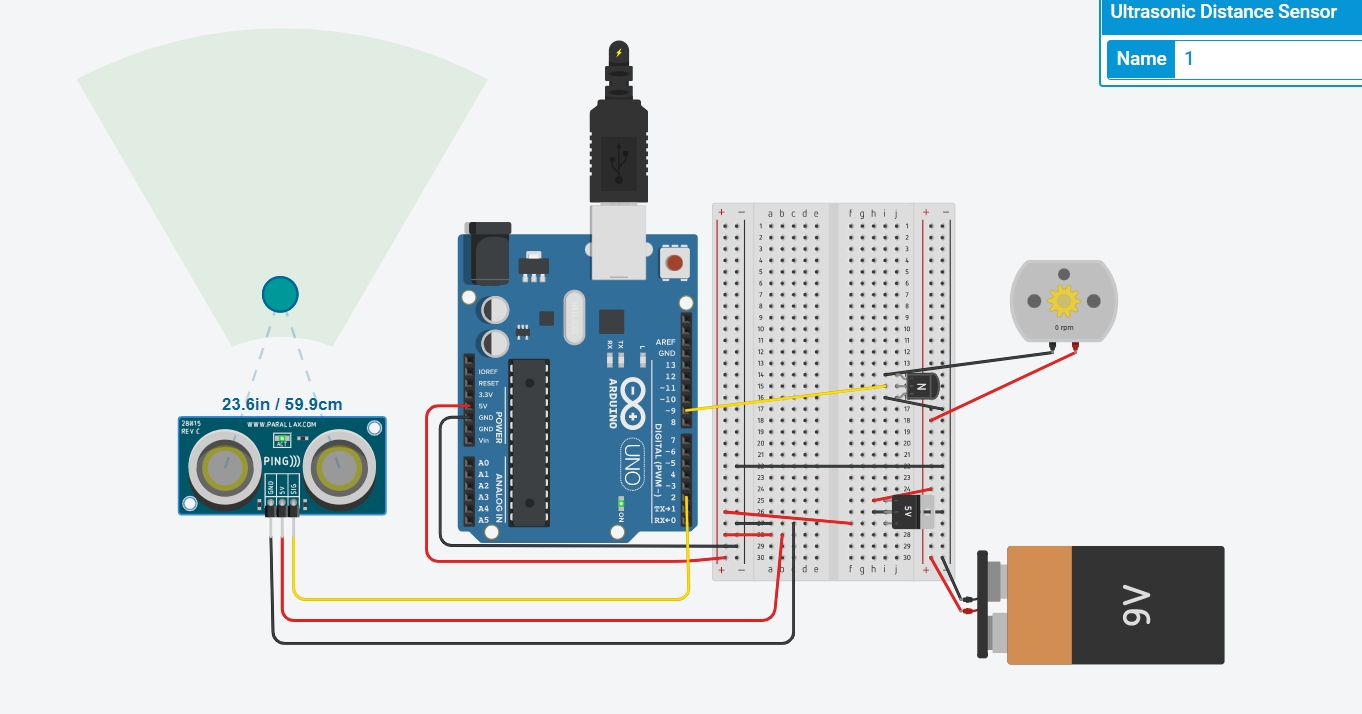
setmotorspeed(255);

}

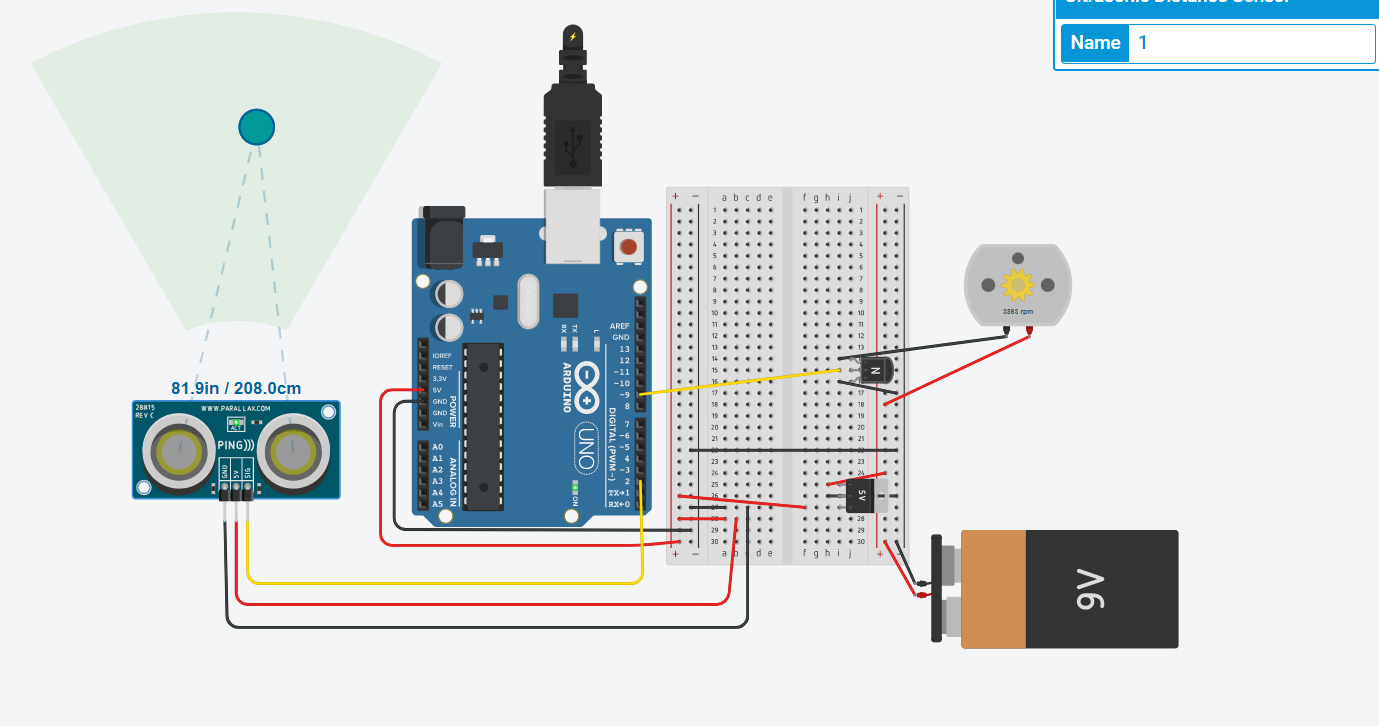
delay(100);

}

**Aggressive Kid**

**

*The object is 23.6 inches away from the sensor. The motor is spinning at 0 rpm.*



*The object is 81.9 inches away from the sensor. The motor is spinning at 3800 rpm.*

**Aggressive Kid TinkerCad Code:**

int distanceThreshold = 0;

int cm = 0;

int inches = 0;

void setup()

{

pinMode(9, OUTPUT);

}

long readUltrasonicDistance(int triggerPin)

{

pinMode(triggerPin, OUTPUT); // Clear the trigger

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(triggerPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds

return pulseIn(triggerPin, HIGH);

}

void setmotorspeed(int speed){

analogWrite(9,speed);

}

void loop()

{

// set threshold distance to activate LEDs

distanceThreshold = 350;

// measure the ping time in cm

cm = 0.01723 \* readUltrasonicDistance(2);

// convert to inches by dividing by 2.54

inches = (cm / 2.54);

Serial.print(cm);

Serial.print("cm, ");

Serial.print(inches);

Serial.println("in");

if (cm > distanceThreshold) {

setmotorspeed(255);

}

if (cm <= distanceThreshold && cm > distanceThreshold - 100) {

setmotorspeed((cm-100)/2);

}

if (cm <= distanceThreshold - 100 && cm > distanceThreshold - 250) {

setmotorspeed((cm-100)/2);

}

if (cm <= distanceThreshold - 250 && cm > distanceThreshold - 350) {

setmotorspeed(0);

}

if (cm <= distanceThreshold - 350) {

setmotorspeed(0);

}

delay(100);

}

**Appendix C: IR Sensor Calibration**

Reference: <https://acroname.com/articles/linearizing-sharp-ranger-data>

|  |  |  |
| --- | --- | --- |
|  |  |  |

Using the equation above, the distance is converted to Voltage. The k value is experimentally found to maximum the R-squared value to three decimal points using the guess and check method.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  |  |  |

The calibration equations above use the linearized data to convert Voltage, or Analog Output, to a Distance.

**Left and Right IR Sensor Calibration:**

The *m’* value is 2500 and the *b’* value is -32.25 for the Left and Right IR sensors. The resulting calibration equation is

|  |  |  |
| --- | --- | --- |
| Distance (IN) | Analog Output (Left,Right) | Linearized Voltage (k=1.8) |
| ~~1~~ | ~~593~~ | *Too Close* |
| 2 | 670 | 0.263158 |
| 3 | 560 | 0.208333 |
| 4 | 449 | 0.172414 |
| 5 | 388 | 0.147059 |
| 6 | 343 | 0.128205 |
| 7 | 308 | 0.113636 |
| 8 | 280 | 0.102041 |
| 9 | 255 | 0.092593 |
| 10 | 240 | 0.084746 |
| 11 | 223 | 0.078125 |
| 12 | 207 | 0.072464 |
| 13 | 196 | 0.067568 |
| 14 | 187 | 0.063291 |
| 15 | 180 | 0.059524 |
| 16 | 170 | 0.05618 |
| 17 | 164 | 0.053191 |
| 18 | 157 | 0.050505 |
| 19 | 153 | 0.048077 |
| 20 | 144 | 0.045872 |

**Front and Back IR Sensor Calibration:**

The *m’* value is 2000 and the *b’* value is 50 for the Left and Right IR sensors. The resulting calibration equation is

|  |  |  |
| --- | --- | --- |
| Distance (IN) | Analog Output (Left,Right) | Linearized Voltage (k=1.5) |
| 1 | 667 | 0.4 |
| 2 | 512 | 0.285714 |
| 3 | 368 | 0.222222 |
| 4 | 272 | 0.181818 |
| 5 | 225 | 0.153846 |
| 6 | 190 | 0.133333 |
| 7 | 165 | 0.117647 |
| 8 | 138 | 0.105263 |
| 9 | 119 | 0.095238 |
| 10 | 107 | 0.086957 |
| 11 | 96 | 0.08 |
| 12 | 87 | 0.074074 |
| 13 | 79 | 0.068966 |
| 14 | 73 | 0.064516 |
| 15 | 68 | 0.060606 |
| 16 | 60 | 0.057143 |
| 17 | 57 | 0.054054 |
| 18 | 56 | 0.051282 |
| 19 | 55 | 0.04878 |
| 20 | 54 | 0.046512 |