WI

What components do I need to blink an LED with an Arduino?

To blink an LED using an Arduino, you will need the following components:

1. Arduino board (e.g. Arduino Uno)
2. LED (light emitting diode)
3. Resistor (to protect the LED from excess current)
4. Breadboard (optional, but recommended for prototyping)
5. Jumper wires (to connect the components to the Arduino)

You will also need a computer with the Arduino software installed, as well as a USB cable to connect the Arduino to the computer.

Here is a brief overview of how to set up the circuit:

1. Connect the positive leg of the LED (the longer leg) to a digital output pin on the Arduino.
2. Connect the negative leg of the LED (the shorter leg) to a resistor.
3. Connect the other end of the resistor to ground on the Arduino.
4. Use the Arduino software to write a sketch (program) that will turn the LED on and off at a desired frequency.
5. Upload the sketch to the Arduino using the USB cable.

Once you have set up the circuit and uploaded the sketch, the LED should start blinking.
Let me know if you have any questions!

WI

I have everything wired up. Can you write the program for me?

Certainly! Here is a simple Arduino sketch that will make an LED connected to digital pin 13 blink on and off at a frequency of 1 Hz (once per second):

scss

```
    delay(1000);
}
```

To use this sketch, connect the LED to digital pin 13 on the Arduino as described in my previous message. Then copy the sketch and paste it into the Arduino software. Click the "Upload" button to upload the sketch to the Arduino. The LED should start blinking once the sketch has been uploaded successfully.

I hope this helps! Let me know if you have any questions.

WI

It worked! thank you

You're welcome! I'm glad to hear that the sketch worked for you. If you have any other questions about using Arduino or if you want to learn more about programming, feel free to ask. I'm here to help!

WI

Okay, do you think you could tell me the steps to building a line following robot using an Arduino?

Certainly! Here are the steps to build a basic line following robot using an Arduino:

1. Gather the necessary components:

Arduino board (e.g. Arduino Uno)

Motor driver (e.g. L293D)

DC motors (2)

Wheels (2)

Chassis

Line following sensor (e.g. QRE1113GR)

Jumper wires

9V battery and battery holder (for powering the motors)

Breadboard (optional, but recommended for prototyping)

2. Assemble the robot chassis and attach the wheels to the motors.

3. Connect the motors to the motor driver according to the motor driver's datasheet.

4. Connect the motor driver to the Arduino using jumper wires. Make sure to connect the motor driver's power and ground pins to the appropriate power and ground pins on the Arduir

5. Connect the line following sensor to the breadboard, and then connect the breadboard to the Arduino using jumper wires.

6. Use the Arduino software to write a sketch that will read the values from the line following sensor and control the motors to make the robot follow a line.

7. Upload the sketch to the Arduino.

8. Power the robot using the 9V battery and verify that the robot is able to follow a line.

I hope this helps! Let me know if you have any questions or if you need further guidance.

WI

I have most of those components! Some I am missing though, do you think I can use the L298N motor driver instead? And maybe a set of 4 AA batteries instead of a 9v?

Yes, you can use the L298N motor driver as a substitute for the L293D. The L298N is a dual full-bridge driver that can be used to drive two DC motors, and it has a similar set of pins and functions as the L293D. To use the L298N with an Arduino, you will need to connect the motor driver's power and ground pins to the appropriate power and ground pins on the Arduino, and then connect the motors and the line following sensor to the motor driver usin

As for the power source, you can use a set of 4 AA batteries to power the motors, but you will need to use a battery holder and a voltage regulator (such as the 7805) to ensure that the voltage supplied to the Arduino is stable and within the recommended range (usually 7-12V). The voltage regulator will also protect the Arduino from overvoltage, which can damage the board.

Here is an example of how you can set up the power supply and the voltage regulator:

1. Connect the positive terminal of the battery holder to the voltage regulator's input pin.
2. Connect the negative terminal of the battery holder to the voltage regulator's ground pin.
3. Connect the voltage regulator's output pin to the VIN pin on the Arduino.
4. Connect the voltage regulator's ground pin to the GND pin on the Arduino.

This should give you a stable power supply for your line following robot. I hope this helps! Let me know if you have any questions.

WI

Thanks for the info! Although I think the L298N already has a 5V regulator on board

---

You're welcome! I'm glad to help.

You are correct that the L298N motor driver has a built-in 5V voltage regulator, which can be used to po s voltage regulator, you ca

1. Connect the positive terminal of the battery holder to the L298N's VCC pin.
2. Connect the negative terminal of the battery holder to the L298N's GND pin.
3. Connect the L298N's 5V pin to the VIN pin on the Arduino.
4. Connect the L298N's GND pin to the GND pin on the Arduino.

This will provide a stable 5V power supply for the Arduino and other 5V devices. Note that you may need to adjust the value of the current limiting resistors (R1 and R2) on the L298N's 5V pin to ensure that the voltage regulator is able to deliver the required current to the devices. You can refer to the L298N's datasheet for more information on how to calculate the resistor values.

I hope this helps! Let me know if you have any questions.

WI

Great! But how do I connect the motors to the motor driver?

To connect the motors to the L298N motor driver, you will need to follow the motor driver's datasheet. The L298N has two channels, each of which can be used to drive a DC motor. Each channel has two input pins (IN1 and IN2) that are used to control the direction of the motor, and an enable pin (EN) that is used to enable or disable the channel.

Here is an examp

1. Connect the positive terminal of the motor to the motor driver's output pin (OUT1 or OUT2).

2. Connect the negative terminal of the motor to the motor driver's ground pin (GND).
3. Connect the motor driver's IN1 (or IN2) pin to a digital output pin on the Arduino.
4. Connect the motor driver's IN2 (or IN1) pin to ground on the Arduino.
5. Connect the motor driver's EN pin to a PWM-enabled digital output pin on the Arduino.

Once you have connected the motors to the motor driver, you can use the Arduino's digital output pins to control the direction and speed of the motors. For example, to make a motor spin in one direction, you can set the corresponding IN1 and IN2 pins as follows:

IN1: HIGH
IN2: LOW

To make the motor spin in the opposite direction, you can set the IN1 and IN2 pins as follows:

IN1: LOW
IN2: HIGH

You can also use the Arduino's PWM (pulse width modulation) functionality to control the speed of the motor by pulse width modulating the EN pin. The duty cycle of the PWM signal (i.e. the ratio of the ON time to the total period) determines the speed of the motor. For example, a duty cycle of 50% will result in the motor spinning at half speed.

I hope this helps! Let me know if you have any questions.

WI

I disagree with bullet point 2 and 4. Wouldn't it make more sense to connect one motor terminal to OUT1 and the other motor terminal to OUT2 instead of the GND? Otherwise the motor will only ever spin in one direction? Similarly don't we can to connect IN2 to a digital output on the Arduino instead of ground so the motor can be given commands to spin in both directions?

I apologize for the mistake in my previous message. You are correct that the negative terminal of the motor should be connected to OUT2 (or OUT1) instead of GND, and that IN2 (or IN1) should be connected to a digital output pin on the Arduino instead of ground.

Here is the correct way to connect the motors to the L298N motor driver:

1. Connect the positive terminal of the motor to the motor driver's output pin (OUT1 or OUT2).
2. Connect the negative terminal of the motor to the motor driver's output pin (OUT2 or OUT1).
3. Connect the motor driver's IN1 (or IN2) pin to a digital output pin on the Arduino.
4. Connect the motor driver's IN2 (or IN1) pin to a digital output pin on the Arduino.
5. Connect the motor driver's EN pin to a PWM-enabled digital output pin on the Arduino.

With this configuration, you can use the Arduino's digital output pins to control the direction and speed of the motors by setting the IN1, IN2, and EN pins as follows:

To make a motor

IN1: HIGH

IN2: LOW

To make a motor spin in the opposite direction:
IN1: LOW
IN2: HIGH
You can also use the Arduino's PWM functionality to control the speed of the motor by pulse width modulating the EN pin. The duty cycle of the PWM signal determines the speed of the motor. For example, a duty cycle of 50% will result in the motor spinning at half speed.

I apologize for any confusion caused by my previous message. I hope this clears things up. Let me know if you have any further questions.

WI

Okay great! How can I connect a second motor to the L298N?

To connect a second motor to the L298N motor driver, you can use the other channel on the L298N. The L298N has two channels, each of which can be used to drive a DC motor. Each channel has two input pins (IN3 and IN4) that are used to control the direction of the motor, and an enable pin (EN) that is used to enable or disable the channel.

Here is an example of how you can connect a second DC motor to the L298N:

1. Connect the 's output pin (OUT3 or OUT4).

2. Connect the negative terminal of the second motor to the motor driver's output pin (OUT4 or OUT3).
3. Connect the motor driver's IN3 (or IN4) pin to a digital output pin on the Arduino.
4. Connect the motor driver's IN4 (or IN3) pin to a digital output pin on the Arduino.
5. Connect the motor driver's EN pin to a PWM-enabled digital output pin on the Arduino.

You can use the same approach as described in my previous message to control the direction and speed of the second motor using the Arduino's digital output pins. Just make sure to use the correct input and enable pins for the second motor.

For example, to make the second motor spin in one direction, you can set the corresponding IN3 and IN4 pins as follows:

IN3: HIGH
IN4: LOW
To make the second motor spin in the opposite direction, you can set the IN3 and IN4 pins as follows:

IN3: LOW
IN4: HIGH
You can also use the Arduino's PWM functionality to control the speed of the second motor by pulse width modulating the EN pin. The duty cycle of the PWM signal determines the speed of the motor.

I hope this helps! Let me know if you have any questions.

WI

Thanks! You mentioned I should connect the EN pins to PWM pins on the Arduino. Can you tell me what pins on the arduino uno are PWM ones?

Sure! On the Arduino Uno, the following pins are PWM (pulse width modulation) enabled:

Digital pins 3, 5, 6, 9, 10, and 11
You can use any of these pins to output a PWM signal, which can be used to control the speed of a motor or the brightness of an LED, among other things.

To use a PWM pin on the Arduino Uno, you can use the `analogWrite` function, which takes two arguments: the pin number and the duty cycle (a value between 0 and 255). For example, to set the duty cycle of the PWM signal on pin 9 to 50%, you can use the following code:

```scss
analogWrite(9, 128);  // 50
```


The duty cycle is of 0% (always off) and 255 represents a duty cycle of 100% (always on). Intermediate values represent intermediate duty cycles.

I hope this helps! Let me know if you have any questions.

WI

Thanks! can you confirm I have wired the L298N to the Arduino Uno correctly? I have: ENA -> 5 IN1 -> 6 IN2 -> 7 IN3 -> 8 IN4 -> 9 ENB -> 10 As you instructed ENA and ENB are connected to PWM pins 5 and 10

Based on the wiring you have described, it looks like you have connected the L298N motor driver to the Arduino Uno correctly.

The L298N's ENA and ENB pins should be connected to PWM-enabled digital output pins on the Arduino in order to be able to control the speed of the motors using pulse width modulation. You have connected ENA to pin 5 and ENB to pin 10, which are both PWM-enabled pins on the Arduino Uno, so that's correct.

The L298N's IN1, IN2, IN3, and IN4 pins should be connected to digital output pins on the Arduino, and you have connected them to pins 6, 7, 8, and 9, respectively, which are all digital output pins on the Arduino Uno.

The L298N's VCC und pins on the Arduino, resp to a power source (e.g. a battery or a voltage regulator) that is capable of providing enough current to the motors.

I hope this helps! Let me know if you have any further questions.

WI

Perfect! Finally I have an array of 5 infrared sensors. I am not sure what the part number is but it has 7 pins labelled: GND, 5V, OUT1, OUT2, OUT3, OUT4, OUT5. Considering the fact that some of the pins are already being used to connect the Arduino to the L298N motor driver, how do you suggest I connect the infrared sensor array?

It sounds like you have an infrared sensor array with 5 separate sensors, each of which has an output pin. To connect this sensor array to the Arduino, you can use the following approach:

1. Connect the sensor array's GND pin to a ground pin on the Arduino.
2. Connect the sensor array's 5V pin to a power pin on the Arduino (e.g. the 3.3V or 5V pin).
3. Connect the sensor array's OUT1, OUT2, OUT3, OUT4, and OUT5 pins to digital input pins on the Arduino.

You will need to use jumper wires to connect the sensor array to the Arduino. Depending on the number of available digital input pins on the Arduino and the number of pins being used by the [...] to connect the sensor array and

Once you have connected the sensor array to the Arduino, you can use the Arduino's digital input pins to read the values from the sensors. You can use the Arduino's `digitalRead` function to read the values from the sensors. For example, to read the value from the OUT1 pin, you can use the following code:

```scss
int value = digitalRead(OUT
```

The `digitalRead` function returns a value of `HIGH` (1) or `LOW` (0), depending on the state of the input pin. You can use this value to determine whether the sensor is detecting an object or not.

I hope this helps! Let me know if you have any further questions.

WI

Would it be okay if I connect the infrared sensor to the analog pins instead?

Yes, you can conn rduino instead of the digital inpu to measure the voltage of an external signal and convert it to a digital value using an analog-to-digital converter (ADC).

To connect the infrared sensor to an analog input pin on the Arduino, you can use the following approach:

1. Connect the sensor array's GND pin to a ground pin on the Arduino.
2. Connect the sensor array's 5V pin to a power pin on the Arduino (e.g. the 3.3V or 5V pin).
3. Connect the sensor array's OUT1, OUT2, OUT3, OUT4, and OUT5 pins to analog input pins on the Arduino (e.g. A0, A1, A2, etc.).

You will need to use jumper wires to connect the sensor array to the Arduino. Depending on the number of available analog input pins on the Arduino and the number of pins being used by the L298N motor driver, you may need to use a breadboard to connect the sensor array and the Arduino.

Once you have connected the sensor array to the Arduino, you can use the Arduino's `analog

---

WI

Awesome! everything is wired together! although it doesn't look much like a robot. You mentioned I should use a robot chassis. Can you explain what that is and how I could fabricate one?

A robot chassis is a frame or structure that serves as the base of a robot and provides a platform for the robot's components, such as the motors, wheels, sensors, and microcontroller. The chassis can be made of various materials, such as plastic, metal, or wood, and it can be fabricated using a variety of methods, such as 3D printing, laser cutting, or manual assembly.

There are many ways to fabricate a robot chassis, and the best method for you will depend on your available tools, materials, and skills. Here are a few options you can consider:

3D printing: If you have access to a 3D printer, you can design and print a chassis using a 3D modeling software (such as Fusion 360 or Tinkercad) and a suitable 3D printing filament (such as PLA or PETG). 3D printing allows for a high degree of customization and precision, but it can be time-consuming and require a good understanding of 3D modeling and 3D printing principles.

Laser cutting: If you have access to a laser cutter, you can design and cut a chassis using a vector graphics software (such as Illustrator or Inkscape) and a suitable material (such as acrylic, wood, or cardboard). Laser cutting allows for fast and accurate production of complex shapes, but it requires a high initial investment in a laser cutter and may produce hazardous fumes.

Manual assembly: If you don't have access to advanced fabrication tools, you can still build a simple rob h as cardboard, foam to assemble the chassis, and you can use templates or drawings to guide the construction process.

This method is suitable for beginners and allows for a lot of creativity, but it may be less precise and durable than other methods.
I hope this gives you an idea of the different options you have for fabricating a robot chassis. Let me know if you have any further questions.

WI

I have access to a laser cutter! Vector graphics software like Inkscape generate .svg files, correct? Can you create a sample .svg file for a 25mm square?

Yes, vector graphics software like Inkscape do generate .svg (Scalable Vector Graphics) files, which can be used to create precise, scalable designs that can be cut or etched using a laser cutter.

To create a sample .svg file for a 25mm square, you can use the following steps:

1. Open Inkscape and create a new document.
2. From the toolbar, select the "Rectangle" tool.
3. Click and drag on the canvas to create a rectangle.
4. In the toolbar, set the width and height of the rectangle to 25mm.
5. In the toolbar ... black (or any other color you pre...
6. From the "Fil...

7. In the "Save As" dialog, select the .svg file format and enter a name for the file.
8. Click "Save" to save the .svg file.

This will create an .svg file with a 25mm square that you can import into a laser cutting software or directly into the laser cutter. You can use the same steps to create more complex designs by adding and combining different shapes, text, and images.

I hope this helps! Let me know if you have any further questions.

WI

Can you write the .svg file for me?

Sure, here is the .svg file for a 25mm square:

php

```
<svg width="25mm" height="2
  <rect x="0" y="0" width="
</svg>
```


This .svg file cont … d a height of 25mm, and a blac … value of the `fill` attribute, or you can remove the fill color by deleting the `fill` attribute altogether.

To save this .svg file, you can copy and paste the code into a text editor, such as Notepad or TextEdit, and save the file with a .svg extension (e.g. "square.svg"). You can then import the .svg file into a laser cutting software or directly into the laser cutter.

I hope this helps! Let me know if you have any further questions.

WI

I'm going to give you a series of components that I will reference later throughout our conversation The first component will be referred to as the "L298N footprint" this component footprint measures 43mm by 43mm. Within this square footprint are 4 mounting holes, all 4 holes have a 1.5mm radius. There is a mounting hole in each of the 4 corners set in from the sides by 3mm. Can you create an SVG file based on these measurements for me to validate?

Sure, here is an .svg file that represents the L298N footprint you have described:

php

```
  <circle cx="3" cy="40" r=
  <circle cx="40" cy="40" r
</svg>
```

This .svg file contains four circle elements, each representing a mounting hole. The circles have a radius of 1.5mm and are placed in the four corners of the L298N footprint, set in from the sides by 3mm. You can modify the size and position of the circles by changing the values of the `cx`, `cy`, and `r` attributes.

To save this .svg file, you can copy and paste the code into a text editor, such as Notepad or TextEdit, and save the file with a .svg extension (e.g. "L298N_footprint.svg"). You can then import the .svg file into a vector graphics software or a laser cutting software to view or modify the design.

I hope this helps! Let me know if you have any further questions.

WI

This is perfect. Here are a few more components: "Caster Wheel Footprint": measures 31mm by 37mm. Within this rectangular footprint are 4 mounting holes, all 4 holes have a 2mm radius. There is a mounting hole in each of the 4 corners set in from the sides by 4mm. "Infrared Footprint": measures 96.5mm by 20.5mm, let these measurements be the x and y axis, respectively. Wi holes have a 1.5 mm radius. All 4 hole placed along the x axis at the following locations: 7, 24, 72.5, 89.5mm. The next component will be referred to as the

"Arduino footprint", it has dimensions of 68.6mm by 53.4mm. This time the mounting holes have a radius of 1.3mm and are located at coordinates (15.24, 2.54), (15.24, 50.8), (66.04, 17.78), (66.04, 45.72) The final component is "motor bracket footprint" with dimensions 15mm by 5mm with 2 mounting holes, of radius 1.5mm, at coordinates (3.5, 2.5) and (11.5, 2.5) Can you create additional SVG files for each of these components?

Sure, here are the .svg files for the additional components you have described:

Caster Wheel Footprint:

php

```
<svg width="31mm" height="3
  <circle cx="4" cy="4" r="
  <circle cx="27" cy="4" r=
  <circle cx="4" cy="33" r=
  <circle cx="27" cy="33" r
</svg>
```

Infrared Footprin

```php
<svg width="96.5mm" height=
  <circle cx="7" cy="4.25"
  <circle cx="24" cy="4.25"
  <circle cx="72.5" cy="4.2
  <circle cx="89.5" cy="4.2
</svg>
```

Arduino Footprint:

```php
<svg width="68.6mm" height=
  <circle cx="15.24" cy="2.
  <circle cx="15.24" cy="50
  <circle cx="66.04" cy="17
  <circle cx="66.04" cy="45
</svg>
```

Motor Bracket Fo

WI

Looks like you ran out of space. Can you give me the motor bracket footprint again?

Sure, here is the .svg file for the Motor Bracket Footprint:

```php
<svg width="15mm" height="5
  <circle cx="3.5" cy="2.5"
  <circle cx="11.5" cy="2.5
</svg>
```

This .svg file contains two circle elements, each representing a mounting hole. The circles have a radius of 1.5mm and are placed at the coordinates (3.5, 2.5) and (11.5, 2.5). You can modify the size and position of the circles by changing the values of the `cx`, `cy`, and `r` attributes.

To save this .svg f ch as Notepad or TextEdit, and s otprint.svg").

You can then import the .svg file into a vector graphics software or a laser cutting software to view or modify the design.

I hope this helps! Let me know if you have any further questions.

WI

I just realized the above files only include the circles for the mounting holes. There is no rectangles around the border. Can you edit all 4 files so they are enclosed by rectangles at the border?

Sure, here are the revised .svg files for the L298N Footprint, Caster Wheel Footprint, Infrared Footprint, and Arduino Footprint, with added rectangles around the border:

L298N Footprint:

php

```
    <circle cx="40" cy="40" r
  </svg>
```

Caster Wheel Footprint:

```php
  <svg width="31mm" height="3
    <rect x="0" y="0" width="
    <circle cx="4" cy="4" r="
    <circle cx="27" cy="4" r=
    <circle cx="4" cy="33" r=
    <circle cx="27" cy="33" r
  </svg>
```

Infrared Footprint:

```python
```

```
<svg width="96.5mm" height=
```

WI

Can you finish writing the infrared and motor bracket files?

Sure, here are the revised .svg files for the Infrared Footprint and Motor Bracket Footprint, with added rectangles around the border:

Infrared Footprint:

php

```
    <circle cx="89.5" cy="4.2
  </svg>
```

Motor Bracket Footprint:

```php
  <svg width="15mm" height="5
    <rect x="0" y="0" width="
    <circle cx="3.5" cy="2.5"
    <circle cx="11.5" cy="2.5
  </svg>
```

I hope these revised .svg files meet your requirements. Let me know if you have any further questions.

WI

oh and the arduino file too

Certainly, here is the revised .svg file for the Arduino Footprint, with added rectangle around the border:

Arduino Footprint:

php

```
<svg width="68.6mm" height=
  <rect x="0" y="0" width="
  <circle cx="15.24" cy="2.
  <circle cx="15.24" cy="50
  <circle cx="66.04" cy="17
  <circle cx="66.04" cy="45
</svg>
```

I hope this revised .svg file meets your requirements. Let me know if you have any further questions.

WI

Create a new svg d the border). In this file also include the ates of these files to ensure there are no overlaps between any lines or circles

---

Sure, here is an .svg file that includes the L298N Footprint, Caster Wheel Footprint, Infrared Footprint, Arduino Footprint, and Motor Bracket Footprint, with no overlaps between any lines or circles:

php

```
</g>
<g transform="translate(3
  <rect x="0" y="0" widtł
  <circle cx="7" cy="4.25
  <
```

WI

can you continue writing from this line onwards: <circle cx="7" cy="4.25" r="1.5" fill="#000000" />

Sure, here is the rest of the .svg file that includes the L298N Footprint, Caster Wheel Footprint, Infrared Footprint, Arduino Footprint, and Motor Bracket Footprint, with no overlaps between any lines or circles:

php

```
    <rect x="0" y="0" width="
    <circle cx="3.5" cy="2.5"
    <circle cx="11.5" cy="2.5
  </g>
  <g transform="translate(110
    <rect x="0" y="0" width="
    <circle cx="15.24" cy="2.
    <circle cx="15.24" cy="50
    <circle cx="66.04" cy="17
    <circle cx="66.04" cy="45
  </g>
  </svg>
```

This .svg file contains five groups of elements, each representing one of the component footprints. The coordinates of the elements within each group have been transformed using the `transform` attribute to ensure there are no overlaps between any lines or circles. The dimensions of the rectangles enclosing each group have been adjusted to fit the size of the component footprints.

I hope this .svg file meets your requirements. Let me know if you have any further questions.

WI

This is a good first attempt. But the Arduino footprint falls outside the border of the 220x130 bounds. I've also realized that I need two motor bracket footprints (one for each motor). In addition to these changes I also want to rotate the following components by 90 degrees clockwise: caster wheel footprint, infrared footprint, and the arduino uno Can you update all this while ensuring there are no overlaps between footprints?

Certainly, here is an updated .svg file that includes the L298N Footprint, two Motor Bracket Footprints, Caster Wheel Footprint, Infrared Footprint, and Arduino Footprint, with no overlaps between any lines or circles:

php

```
    <rect x="0" y="0" widtl
    <circle cx="2.5" cy="3.
    <circle cx="2.5" cy="11
  </g>
  <g transform="translate(1
    <rect x="0" y="0" widtl
    <circle cx="2.5" cy="3.
    <circle cx="2.5" cy="11
  </g>
  <g transform="translate(2
    <rect x="0" y="0" widtl
    <circle cx="4" cy="4" r
    <circle cx="33" cy="4"
    <circle cx="4
```

WI

2 / 2Can you continue from this line? <circle cx="33" cy="4" r="2" fill="#000000" />

Certainly, here is Footprint, two

Motor Bracket Fo Arduino

Footprint, with no overlaps between any lines or circles:

php

```
    <circle cx="4" cy="33" r=
    <circle cx="33" cy="33" r
  </g>
  <g transform="translate(10(
    <rect x="0" y="0" width="
    <circle cx="4.25" cy="7"
    <circle cx="4.25" cy="24"
    <circle cx="4.25" cy="72.
    <circle cx="4.25" cy="89.
  </g>
  <g transform="translate(18(
    <rect x="0" y="0" width="
    <circle cx="2.54" cy="15.
    <circle cx="50.8" cy="15.
    <circle cx="17.78" cy="6(
    <circle cx="45.72" cy="6(
```

WI

Can you please continue from this line? <circle cx="17.78" cy="66.04" r="1.3" fill="#000000" />

Certainly, here is the rest of the updated .svg file that includes the L298N Footprint, two Motor Bracket Footprints, Caster Wheel Footprint, Infrared Footprint, and Arduino Footprint, with no overlaps between any lines or circles:

```bash
3" fill="#000000" />
</g>
</svg>
```

This .svg file contains five groups of elements, each representing one of the component footprints. The coordinates of the elements within each group have been transformed using the `transform` attribute to ensure there are no overlaps between any lines or circles. The dimensions of the rectangles enclosing each group have been adjusted to fit the size of the component footprints.

I hope this updated .svg file meets your requirements. Let me know if you have any further questions.

WI

2 / 2I made some transformation changes on my own, see the file below. The last thing I would like is to change the border rectangle of 220 by 130mm to have curved fillet edges of radius 30mm. Can you make this change? <svg width="220mm" height="130mm" viewBox="0 0 220 130" version="1.1" xmlns="http://www.w3.org/2000/svg"> <rect x="0" y="0" width="220" height="130" fill="#FFFFFF" /> <g transform="translate(100, 1)"> <rect x="0" y="0" width="43" height="43" fill="#FFFFFF" /> <circle cx="3" cy="3" r="1.5" fill="#000000" /> <circle cx="40" cy="3" r="1.5" fill="#000000" /> <circle cx="3" cy="40" r="1.5" fill="#000000" /> <circle cx="40" cy="40" r="1.5" fill="#000000" /> </g> <g transform="translate(58, 18)"> <rect x="0" y="0" width="15" height="5" fill="#FFFFFF" /> <circle cx="3.5" cy="2.5" r="1.5" fill="#000000" /> <circle cx="11.5" cy="2.5" r="1.5" fill="#000000" /> </g> <g transform="translate(58, 107)"> <rect x="0" y="0" width="15" height="5" fill="#FFFFFF" /> <circle cx="3.5" cy="2.5" r="1.5" fill="#000000" /> <circle cx="11.5" cy="2.5" r="1.5" fill="#000000" /> </g> <g transform="translate(133, 45.9)"> <rect x="0" y="0" width="37" height="31" fill="#FFFFFF" /> <circle cx="4" cy="4" r="2" fill="#000000" /> <circle cx="4" cy="27" r="2" fill="#000000" /> <circle cx="33" cy="4" r="2" fill="#000000" /> <circle cx="33" cy="27" r="2" fill="#000000" /> </g> <g transform="translate(205.5, 16.75) scale(-1, 1)"> <rect x="0" y="0" width="20.5" height="96.5" fill="#FFFFFF" /> <circle cx="4.25" cy="7" r="1.5" fill="#000000" /> <circle cx="4.25" cy="24" r="1.5" fill="#000000" /> <circle cx="4.25" cy="72.5" r="1.5" fill="#000000" /> <circle cx="4.25" cy="89.5" r="1.5" fill="#000000" /> </g> <g transform="rotate(180) scale(-1, 1) translate(76, -122.6)"> <rect x="0" y="0" width="53.4" height="68.6" fill="#FFFFFF" /> <circle cx="2.54" cy="15.24" r="1.3" fill="#000000" /> <circle cx="50.8" cy="15.24" r="1.3" fill="#000000" /> <circle cx="17.78" cy="66.04" r="1.3" fill="#000000" /> <circle cx="45.72" cy="66.04" r="1.3" fill="#000000" /

Certainly, here is the updated .svg file with a border rectangle of 220 by 130mm and curved fillet edges of radius 30mm:

php

```
</g>
<g transform="translate(1
```

WI

can you continue from this line: <g transform="translate(133, 45.9)">

Certainly, here is the rest of the updated .svg file with a border rectangle of 220 by 130mm and curved fillet edges of radius 30mm:

php

```
      <circle cx="4.25" cy="2
      <circle cx="4.25" cy="7
      <circle cx="4.25" cy="8
    </g>
    <g transform="rotate(180)
      <rect x="0" y="0" width
      <circle cx="2.54" cy="1
      <circle cx="50.8" cy="1
      <circle cx="17.78" cy="
      <circle cx="45.72" cy="
    </g>
  </svg>
```

I hope this updated .svg file meets your requirements. Let me know if you have any further questions.

WI

Based on the electronics and pin allocations, discussed earlier, please write an arduino program with 5 functions: drive forward, reverse, left, right, and stop. Include a variable for storing the PWM drive speed. This program should loop through the 5 different drive functions with a 2 second delay be

Certainly! Here is an Arduino program that includes 5 functions: `driveForward()`, `reverse()`, `left()`, `right()`, and `stop()`, as well as a variable for storing the PWM drive speed:

```scss
```

```
  // Set the left motor to
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  // Set the right motor tc
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  // Set the drive speed fc
  analogWrite(ENA, DRIVE_SF
  analogWrite(ENB, DRIVE_SF
}

void reverse() {
  // Set the left motor to
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  // Set the right motor tc
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  // Set the drive speed fc
  analogWrite(ENA, DRIVE_SF
  analogWrite(ENB, DRIVE_SF
}

void left(
```

WI

2 / 2Please continue on from this line until you finish the script: void left() {

Certainly! Here is the rest of the Arduino program:

scss

```
    digitalWrite(IN2, LOW);
    // Set the right motor tc
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    // Set the drive speed fc
    analogWrite(ENA, DRIVE_SF
    analogWrite(ENB, DRIVE_SF
  }

  void stop() {
    // Set both motors to stc
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    // Set the drive speed fc
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
  }
```

This program should loop through the 5 different drive functions with a 2 second delay between each. Th hich can be adjusted as need

I hope this program meets your needs. Let me know if you have any further questions or if you need any additional assistance.

WI

7 / 7Using the same steering functions, modify the last program to make a line following robot that uses binary (ON/OFF) infrared sensors. There are 2 infrared sensors on the right side of the robot (pins A0 and A1) and 2 on the left side (pins A3 and A4). The robot should drive with the goal of staying over a black line, turning whenever one (or more) of the sensors are detected to have crossed over the line

---

Sure! Here is an example of an Arduino program that uses binary infrared sensors to make a line following robot:

```scss
```

```
// Read the values of the
int left1 = digitalRead(I
int left2 = digitalRead(I
int right1 = digitalRead(
int right2 = digitalRead(

// Check if the robot is
if (left1 == HIGH && left
  driveForward();
} else {
  // Check if the left si
  if (left1 == LOW || lef
```

WI

continue please

Sure! Here is the continuation of the Arduino program:

scss

```
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  // Set the drive speed fc
  analogWrite(ENA, 0);
  analogWrite(ENB, 0);
}
```
