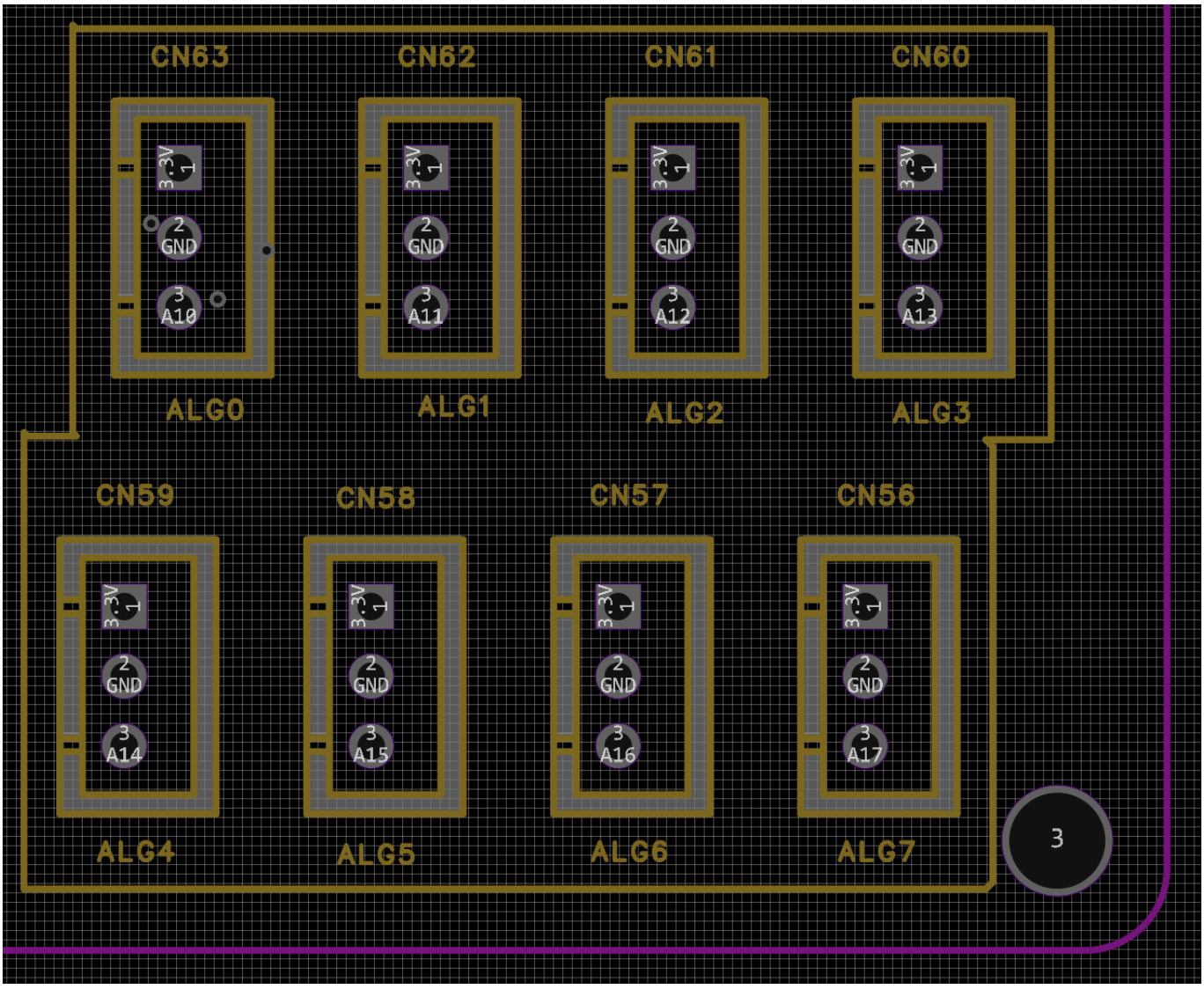
# Analog Block

There are 8 analog connectors, labeled ***ALG0*** through ***ALG7***. Each connector in the picture has pin 1 at the top and pin 3 at the bottom. Each pin's function is:

1. 3.3 volts. Can be used as a voltage reference or to power the sensor.
2. Ground.
3. Analog signal. Must be in the voltage range of 0 to 3.3.




To use the analog block, you would have code similar to this:

```
// Set the analog resolution, in bits for all analog pins.
analogReadResolution(10);

// Make sure the pin is not in output mode.
// In this example, pin 24 corresponds to the port
// for the ALG0 connector.
pinMode(24, INPUT);
```

```
    // Read the analog voltage.
    int16_t value = analogRead(24);
```

For a practical example, consider the TMP36 temperature sensor from AdaFruit. Their discussion gives:

> Temp in °C = [(Vout in mV) - 500] / 10
>
> So for example, if the voltage out is 1V that means that the temperature is ((1000 mV - 500) / 10) = 50 °C

Here is code that would read the sensor from the ***ALG0*** port and convert the result to a floating point value representing tenths of a degree Centigrade.

```
    // Get a reading that will read 0 to 3.3 volts spread
    // over 1024 buckets (10-bits).
    int raw = analogRead(24);

    // Scale and remove the offset voltage.
    float degreesC = ((raw * 3300 / 1024.0) - 0.5) / 10.0;
```

The pin number to be used for the [pinMode] and [analogRead] values are:

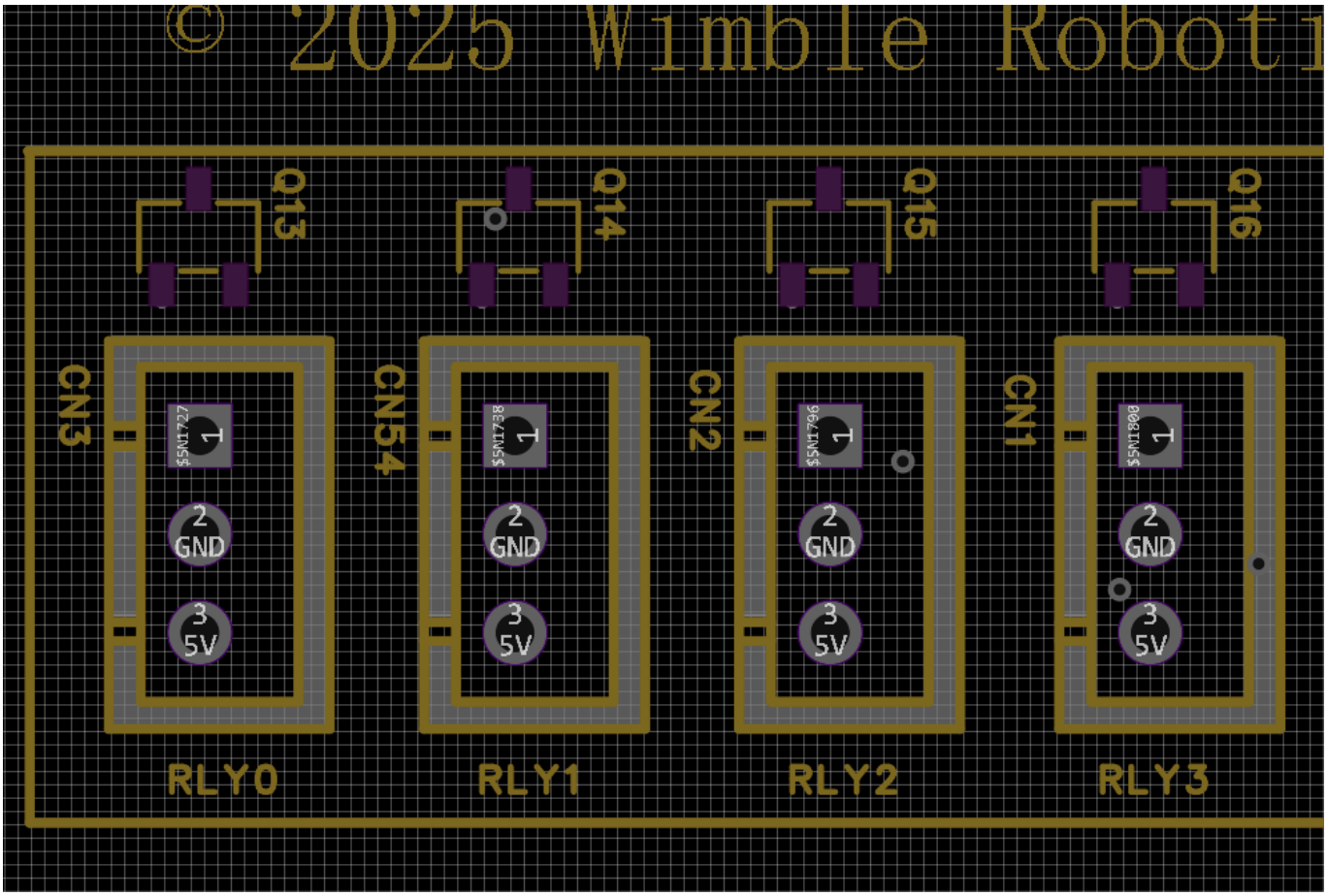
| Connector | Pin Number |
|---|---|
| ALG0 | 24 |
| ALG1 | 25 |
| ALG2 | 26 |
| ALG3 | 27 |
| ALG4 | 38 |
| ALG5 | 39 |
| ALG6 | 40 |
| ALG7 | 41 |

# Relay Block

There are 4 relay connectors, labeled ***RLY0*** through ***RLY3***. The intent is that each connector acts as a switch. The output will either be driven to ground or allowed to float, which means the switch signal should be connected to a pull up resistor which then is connected to the desired 'on' voltage for the switch. That maximum voltage that can be switched is 5 volts.

Each connector in the picture has pin 1 at the top and pin 3 at the bottom. Each pin's function is:

1. The switched output. The output should include a pull up resistor to the desired voltage, typically 3.3 volts or 5 volts. Don't exceed 5 volts. In my original design, the pin was used to drive a solid state relay.
2. Ground.
3. A 5 volt reference. Can be used to power the switched device and/or be used for that pull up resistor needed for pin 1.




To use the relay block, you would have code similar to this:

```
// Set the pin in output mode.
// Here, 0 corresponds to the pin for RLY0.
pinMode(0, OUTPUT);

// Turn the switch off.
digitalWrite(0, LOW);

// Turn the switch on.
digitalWrite(0, HIGH);
```

The pin number to be used for the [pinMode] and [digitalWrite] values are:

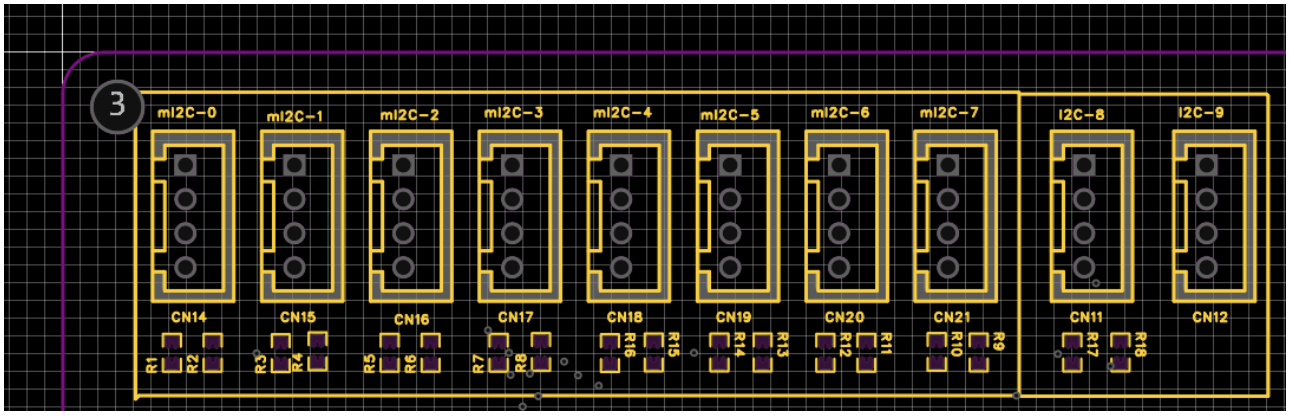
| Connector | Pin Number |
|---|---|
| RLY0 | 0 |
| | |

| | |
|---|---|
| RLY1 | 1 |
| RLY2 | 2 |
| RLY3 | 3 |

# I2C Block

There are 8 multiplexed I2C connectors, labeled ***mI2C-0*** through ***mI2C-7***, and two non-multiplexed I2C connectors, labeled ***I2C-8*** and ***I2C-9***. Each connector in the picture has pin 1 at the top and pin 4 at the bottom. Each pin's function is:

1. 3.3 volts. Can be used to power the sensor.
2. Ground.
3. ***SDC***, the I2C clock.
4. ***SDA***, the I2C data.




To use the multiplexed block, you would have code similar to this:

```
// Enable the multiplexor chip.
pinMode(8, OUTPUT);
digitalWrite(8, HIGH);

// Initialize the Wire library.
Wire.begin();

// Enable one of the multiplexed I2C connectors.
// Replace 'sensor_index' with a value in the
// closed interval [0, 7].
#define I2C_MULTIPLEXER_ADDRESS 0x70
Wire.beginTransmission(I2C_MULTIPLEXER_ADDRESS);
Wire.write(1 << sensor_index);
Wire.endTransmission();

// Small delay to allow multiplexer to switch.
delayMicroseconds(100);

// Continue with I2C code
```