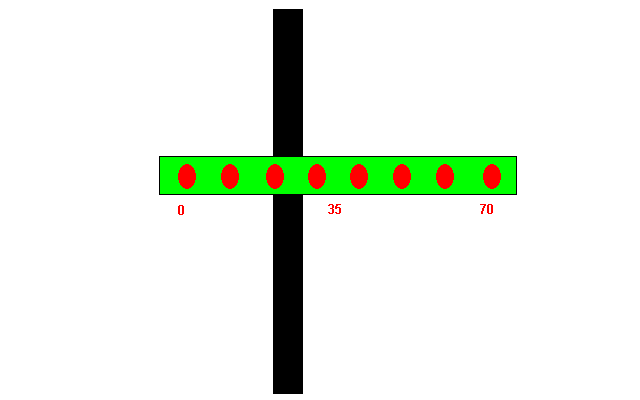
Let’s take a look on how the **LSA08 analog mode works**, and how are we going to interpret the data.



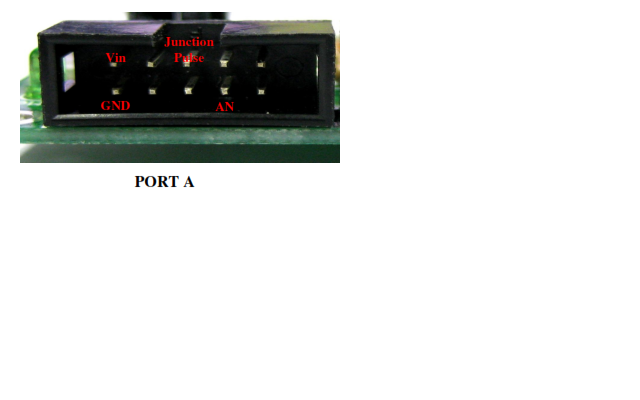
Analog output mode of LSA08 allow users to retrieve line position in the range of 0 – 70 representing the right most sensor to the left most sensor, with 255 as no line detected. This value is converted into voltage signal ranging from 0V to 4.5V indicating 0 – 70, and 5V for 255 no line detected, where the user can get the reading through Analog-Digital Converter (ADC).

The ADC modules of Arduino Uno support 10-bit conversion, so each step size of the result represent 5V/2^10= 4.88mV, meaning that an increase of 4.88mV in the reading will increase the result of ADC conversion by 1.

So, to convert the voltage signal back into what we human can understand, we need to use the formula ((ADC result)/921)\*70 in order to get the actual position of the line. Why divide by 921? Because 921 is the digital representation for 4.5V in 10-bit ADC.

To make the program more beginner friendly, I have separated the line position into 4 categories as below:

* 0 – 18, the line at left hand side, robot needs to move left
* 19 – 52, the line at center, robot move forward
* 53 – 70, the line at right hand side, robot needs to move right
* any value other than 0 – 70, robot stop moving

[[](http://tutorial.cytron.com.my/wp-content/uploads/2015/07/analogporta.bmp)](http://tutorial.cytron.com.my/wp-content/uploads/2015/07/analogporta.bmp)

The connection for analog mode is much more simple than in digital mode, as you only require 4 wires to connect to the LSA08.

Referring to [user manual](http://www.cytron.com.my/p-lsa08?search=lsa08) from Cytron product page, connect LSA08 to Arduino board as shown in the table below:

|  |  |
| --- | --- |
| LSA08 | Arduino |
| Vin | Vin |
| GND | GND |
| AN | analog pin 0 (A0) |
| Junction pulse | digital pin 4 |

\*NOTE: please make sure that you connect to the PORTA on LSA08

Arduino code:

Reference(<https://github.com/CytronTechnologies/Arduino-LSA08/blob/master/LSA08Analog/LSA08Analog.ino> )

|  |
| --- |
| const byte analogPin = 0; // Connect AN output of LSA08 to analog pin 0 |
|  | const byte junctionPulse = 4; // Connect JPULSE of LSA08 to pin 4 |
|  | const byte dir1 = 13; // Connect DIR1 of motor driver to pin 13 |
|  | const byte dir2 = 12; // Connect DIR2 of motor driver to pin 12 |
|  | const byte pwm1 = 11; // Connect PWM1 of motor driver to pin 11 |
|  | const byte pwm2 = 10; // Connect PWM2 of motor driver to pin 10 |
|  | int readVal,positionVal; // Variables to store analog and line position value |
|  | unsigned int junctionCount = 0; // Variable to store junction count value |
|  |  |
|  | void setup() { |
|  | pinMode(junctionPulse,INPUT); |
|  |  |
|  | // Setting pin 10 - 13 as digital output pin |
|  | for(byte i=10;i<=13;i++) { |
|  | pinMode(i,OUTPUT); |
|  | } |
|  |  |
|  | // Setting the initial condition of motors |
|  | // make sure both PWM pins are LOW |
|  | digitalWrite(pwm1,LOW); |
|  | digitalWrite(pwm2,LOW); |
|  |  |
|  | // State of DIR pins are depending on your physical connection |
|  | // if your robot behaves strangely, try changing thses two values |
|  | digitalWrite(dir1,LOW); |
|  | digitalWrite(dir2,LOW); |
|  |  |
|  | } |
|  |  |
|  | void loop() { |
|  | // Checking for junction crossing, if juction detected, |
|  | // keep moving forward |
|  | if(digitalRead(junctionPulse)) { |
|  | while(digitalRead(junctionPulse)) { |
|  | moveForward(); |
|  | } |
|  | // Increment junction count by 1 after the junction |
|  | // You can do whatever you want with the junction count |
|  | junctionCount++; |
|  | } |
|  |  |
|  | readVal = analogRead(analogPin); // Read value from analog pin |
|  |  |
|  | // Convert voltage level into line position value |
|  | positionVal = ((float)readVal/921)\*70; |
|  |  |
|  | // Line at left when 0 - 18, move left |
|  | if(positionVal <= 18) |
|  | moveLeft(); |
|  |  |
|  | // Line at middle when 19 - 52, move forward |
|  | else if(positionVal <= 52) |
|  | moveForward(); |
|  |  |
|  | // Line at right when 53 - 70, move right |
|  | else if(positionVal <= 70) |
|  | moveRight(); |
|  |  |
|  | // If no line is detected, stay at the position |
|  | else |
|  | wait(); |
|  |  |
|  | // Put some delay to avoid the robot jig while making a turn |
|  |  |
|  | } |
|  |  |
|  | // The values work good in my case, you could use other values set |
|  | // to archieve a performance that satisfy you |
|  | void moveLeft() { |
|  | // For robot to move left, right motor has to be faster than left motor |
|  | analogWrite(pwm1,90); |
|  | analogWrite(pwm2,10); |
|  | } |
|  |  |
|  | void moveRight() { |
|  | // For robot to move right, left motor has to be faster than right motor |
|  | analogWrite(pwm1,10); |
|  | analogWrite(pwm2,90); |
|  | } |
|  |  |
|  | void moveForward() { |
|  | // For robot to move forward, both motors have to be same speed |
|  | analogWrite(pwm1,70); |
|  | analogWrite(pwm2,70); |
|  | } |
|  |  |
|  | void wait() { |
|  | // Function to makes the robot stay |
|  | analogWrite(pwm1,0); |
|  | analogWrite(pwm2,0); |
|  | } |