Need to double check Google Docu and make changes!!!

Implementation Details of Subsystems

* 1. Infrared (IR) Senor Algorithm for keeping mBot not hitting the walls

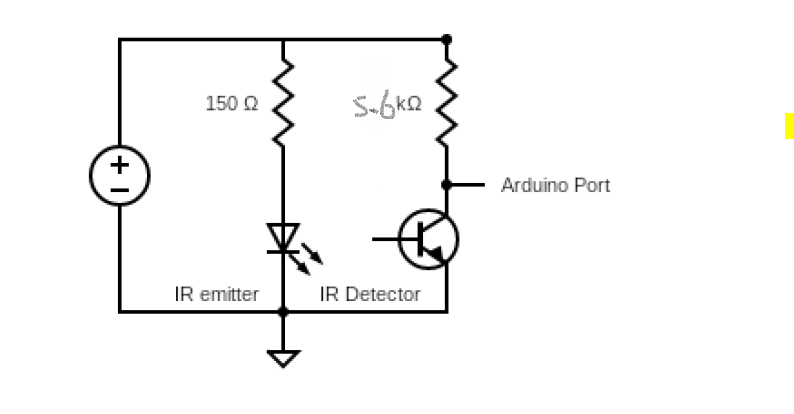


Figure 1.1.1 IR Circuit Diagram

The position of the mBot with respect to the left and right walls of the maze can be determined by readings obtained from two IR sensors attached to the left and right of the mBot respectively. After setting the left and right IR sensors to pin A3 and A2 of the Arduino, we then proceed to calibrate the ideal position of the mBot from the walls of the maze.

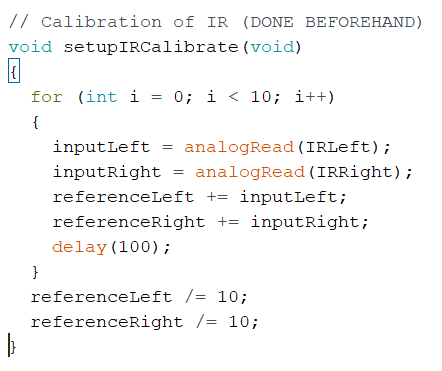


Figure 1.1.2 Code for IR Calibration

Based on our calibration, the left IR and right IR readings are 651 and 653 respectively when the mBot is placed in the ideal position. These two values are used as the reference point. We then proceed to determine the IR reading when a movement correction is needed – the mBot is either too close to the left or to the right of the walls. After multiple trial and error, we have determined the movement correction reference point, whenever the IR reading is less than the movement correction reference point, the mBot will adjust its motor speed and move back to its ideal position.

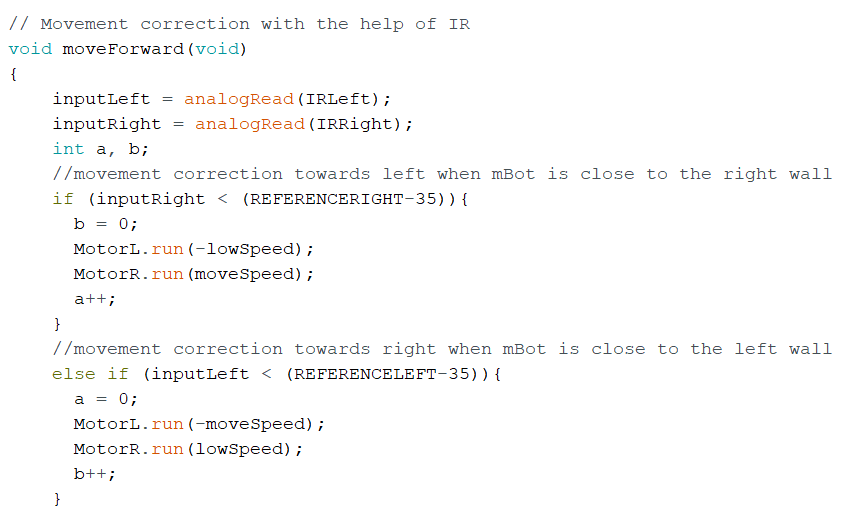


Figure 1.1.3 Code for mBot Movement Correction

As demonstrated in Figure 1.1.3, when the right IR reading is less than the movement correction reference point, which is (REFERENCERIGHT – 35), such that the mBot is closer to the right wall. The speed of the left motor will decrease to low speed while the right motor will continue to function at normal speed. Thus, driving away the mBot away from the right wall.

If both the left and right IR readings are within the movement correction reference point, the mBot will move straight as both motors will run at normal speed but of opposite direction, as shown in Figure 1.1.4.

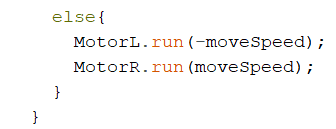
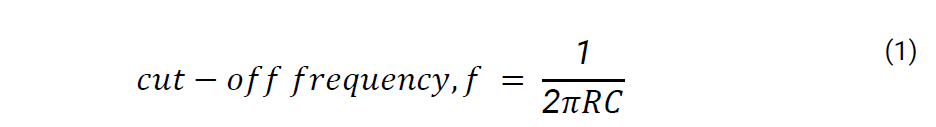


Figure 1.1.4 Code to make mBot in a straight line

* 1. Audio Processing Circuit Design

A close up of a piece of paper

Description automatically generated

The circuit consists of several parts, namely band-pass filter, high-pass filter and microphone. The microphone is connected to the 5V power supply via a 2.2KΩ resistor, to ensure that its operating voltage is of its optimum of 2.5V. The purpose of having a band-pass filter and high-pass filter is to accurately capture the sounds of the required frequency. For the 100-300 Hz signal, a band-pass filter is needed whereas a high-pass filter is required for the 3000 Hz signal. To configure the circuit, appropriate resistance (R) and capacitance (C) values were calculated using the Equation 1 as follow: 

By calculating the respective RC values, we then proceed to choose the appropriate resistor and capacitor. The resistors with the approximate resistance values were used, and the actual cut-off frequencies were calculated to ensure that they were of the appropriate values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Filter type | Cut-off Frequency/Hz | RC Value /ΩF | Resistance Value Used/KΩ | Capacitance Value Used/pF | Calculated Cut-off Frequency/Hz |
| Band-Pass | 100 (High-Pass) | 0.0015915 | 470 | 3300 | 102.61 |
| 300 (Low-Pass) | 0.00053052 | 62 | 8200 | 313.05 |
| High-Pass | 3000 | 0.000053052 | 82 | 680 | 2854.29 |

Figure 1.2.1 Table of chosen RC values for accurate calculation

The processed signals from the filters were then passed into non-inverting op-amp for amplification, with Ri value of 10kΩ and a Rf value of 880kΩ for high-pass filter; with Ri value of 10kΩ and Rf value of 220KΩ for band-pass filter, which amplifies the output voltage to a suitable value which is significant enough for processing and analysis. The output from the op-amp was then connected to an envelope detector with a resistor-capacitor network, to produce an envelope of the signal to be read by the Arduino.

* 1. Audio Processing Circuit Algorithm

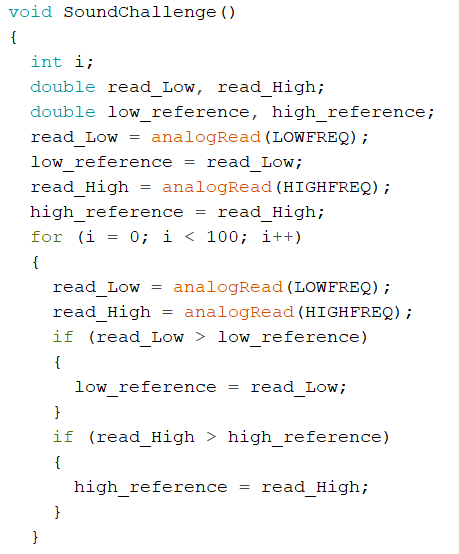


Figure 1.3.1 Code to determine maximum value for low\_reference and high\_reference

100 readings are taken for both low and high frequency challenge, and the maximum reading of each challenge is being stored as the low\_reference and high\_reference respectively. The low\_reference and high\_reference values are used to determine the calibration for the sound challenge.

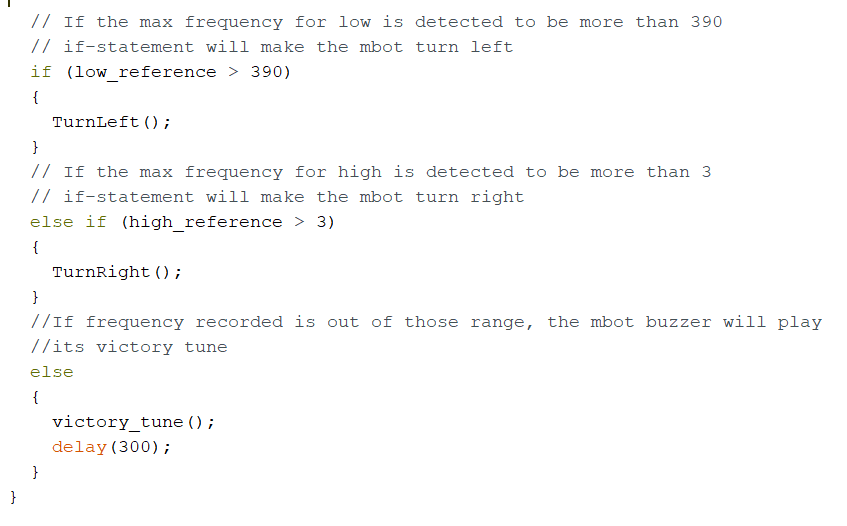


Figure 1.3.2 Code for mBot action after detecting black color board on top

If the low\_reference is larger than 390 and the high\_reference is larger than 3, the mBot will turn left and turn right respectively. Should neither the above conditions are met, the mBot will determine that it is the end of the maze and play the victory tune as there is no sound challenge detected and the color of the board on top of the mBot is black in color.