CG1111 mBot Project

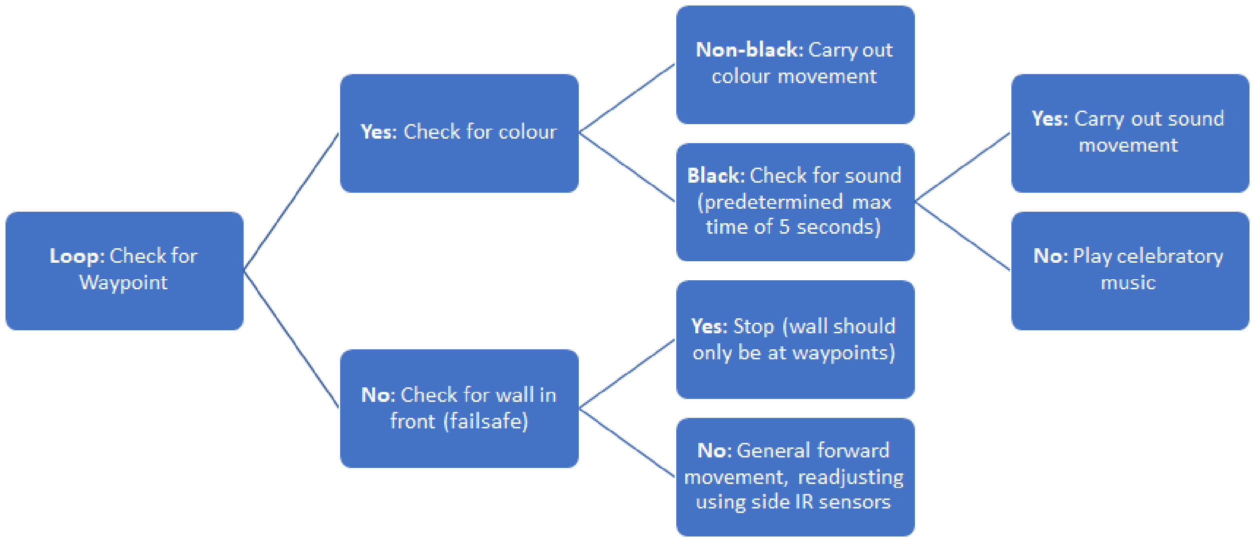
Team members:	Wira Azmoon () Wang Zihao () Walter Kong () Wang Shuyi () Brendan Wan ()
Studio Group Number:	4
Section Number:	3
Team Number:	B

Overall Algorithm

After calibration, the code will begin looping to check for waypoints and other obstructions as the mBot starts moving forward. First, the mBot will check if there are any black strips on the ground to determine if there is a waypoint challenge. If there is, the bot will stop, check if there is a colour based challenge, and perform the appropriate movement. If the colour detected is black, meaning there could be a sound-based challenge, then the bot will check whether the audio detected is low or high frequency, and then turn left or right, respectively. If there is no sound challenge detected, then the bot will determine that it has reached the finish line and play celebratory music.

Otherwise, if there is no black strip detected, the mBot will continue to move forward. Meanwhile, the side Infrared (IR) and ultrasonic sensors checks if the mBot is getting too close to a wall. If the mBot gets too close to the wall, it will make adjustments to its path accordingly to avoid collision.

The following flowchart shows the general flow of the mBot decisions.


Loop: Check for Waypoint
Yes: Check for colour
No: Check for wall in front (failsafe)
Non-black: Carry out colour movement
Black: Check for sound (predetermined max time of 5 seconds)
Yes: Stop (wall should only be at waypoints)
No: General forward movement, readjusting using side IR sensors
Yes: Carry out sound movement
No: Play celebratory music

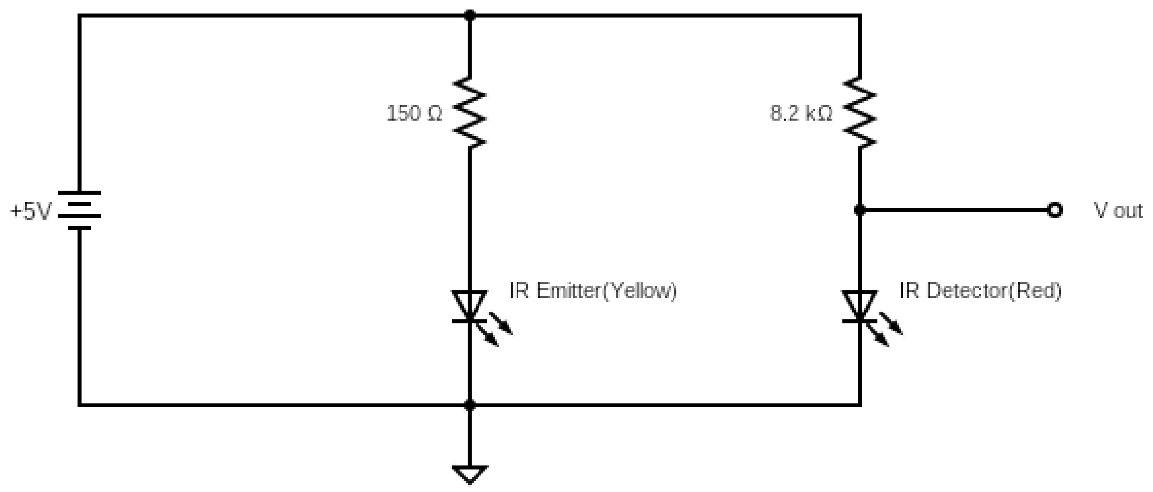
Subsystems

General Movement

The program uses the MeDCMotor object to move by setting motors on both sides to a constant speed. Turning left or right is accomplished by setting the motors to opposite speeds for a period of time which we determined through experimentation.
We used a variety of functions to control the movement of the mBot, such as forward(), turnRight(), turnLeft() and uTurn(). They are used to complete the waypoint challenges, as well as to move through the maze. The function stopMove() is also called in all the general movement functions for reliability.

For forward movement, different functions such as moveForward(), and forwardGrid() were used. forwardGrid() will only be used in turning the mBot for the blue and purple waypoint challenges. This is so the mBot can move forward exactly 1 grid after turning the appropriate direction. moveForward() is then the default forward movement function called when there are no waypoints, and is constantly called with a time delay of 100ms with no waypoint detected. This function will call getDist() which takes into account the distance from the walls using the IR sensors. The function will automatically readjust according to the return value of getDist(). The Ultrasonic sensor is also used in both moveForward() and forwardGrid() to prevent the mBot from crashing into the wall in case it gets too close. Whenever a waypoint is detected, the function stopMove() is also used to stop the mBot.

IR Sensor Circuit Diagram (Two sides share the same circuit diagram):

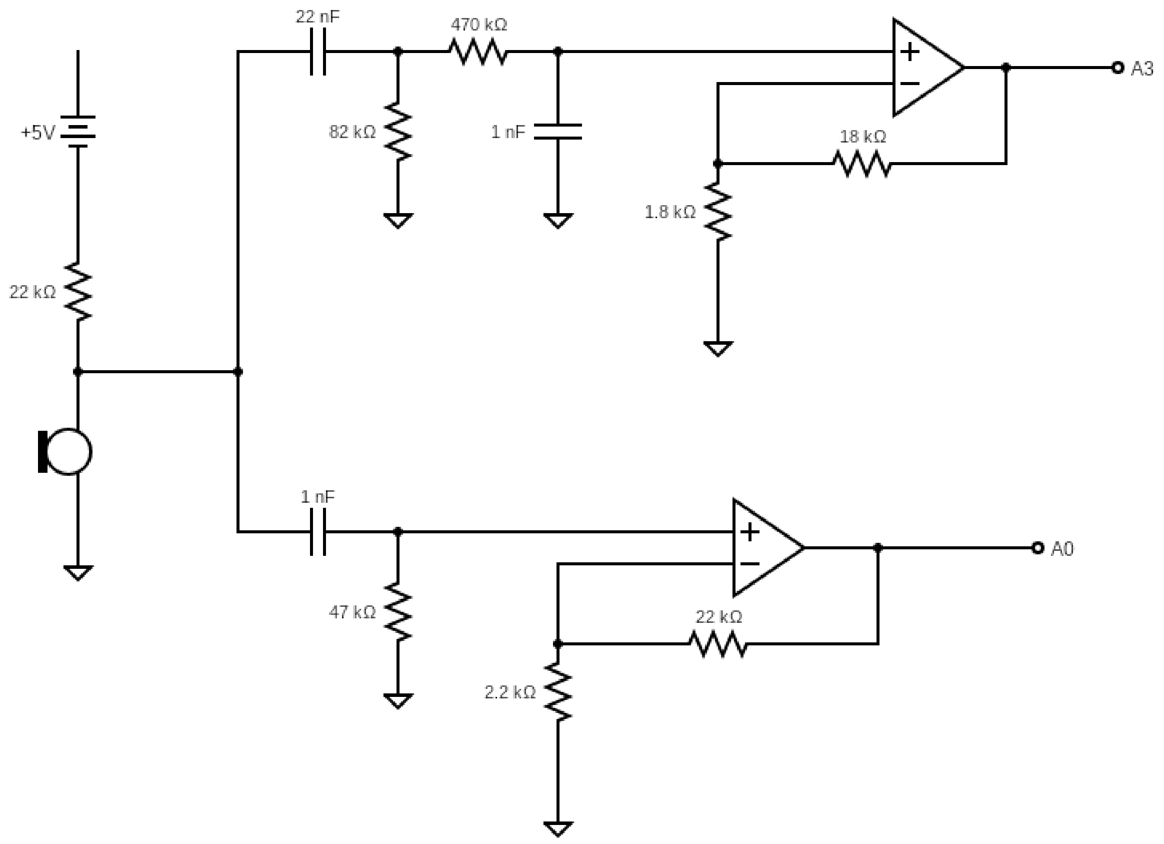


Audio Processing

For the detection of sound in the sound-based challenge, we used a microphone in conjunction with both a band-pass and high pass filter, to only detect sounds of frequencies 100Hz-300Hz and above 3000Hz, respectively. If low frequency (100-300Hz) audio is detected, the bot will perform a left turn, and if high frequency (>3000Hz) audio is detected, then the bot will perform a right turn. The functions used are getSound(), which will return the type of audio detected, and soundWaypoint(), which will execute the instruction based on the value from getSound(). The maximum time spent on detecting sound amounts to at least 5 seconds to ensure accuracy, before deciding that the end of maze is detected.

Final theoretical cutoff frequencies:
Band Pass Filter: 88 - 339HZ
High Pass Filter: 3386HZ

Circuit diagram:



Colour Sensing

To determine the correct colour for the waypoint challenges, we used the LDR built into the mCore, together with the MeRGBLed object to shine the RGB lights. A separate file, colourcal.ino, was used to calibrate specific values for all the various colours used in the maze. We first calibrated black and white LDR values using setBalance(), and by using white and black objects to obtain baseline RGB values, scaled from 0-255. The code then uses a function, getColourValues(), to retrieve the RGB values of the object being detected. These values were all kept as arrays and stored in the main file to complete the waypoint challenges.

In the main file, getColour() was used to detect the colour of the board above the challenge by retrieving the RGB values. The obtained colour values are then compared to established colour values we determined earlier. If the values are close enough, the program will decide the detected values correspond to that particular colour. The algorithm uses a least square distance of the RGB values. Based on which colour is determined, and using colourWaypoint(), the bot will then perform the movement corresponding to the challenge requirement.

Waypoint/Endpoint Detection

For waypoint detection, we used built-in MeLineFollower object located beneath the mBot to determine if there was a black strip on the floor, using the checkWaypoint() function. When a black strip is detected, the mBot will stop immediately and proceed to check for the colour/sound challenges, then execute the appropriate action. If neither challenge is detected, the bot will then decide that it has reached the finish line and play a celebratory tune.

Celebratory Music

The celebratory tune is played when the mBot determines that it has reached the finish line, after it runs through the code for the color and sound challenge. The code utilizes MeBuzzer, and uses the tone() function to execute the melody sheet stored in the arrays “melody” and “noteDurations” when the celebratory_music() function is called. It then uses the noTone() function to stop the buzzer after the celebratory tune has finished playing. The melody sheet for the celebratory tune was referenced online from an open-source project by Dipto Pratyaksa[1].

[1] http://www.linuxcircle.com/2013/03/31/playing-mario-bros-tune-with-arduino-and-piezo-buzzer/

Calibration

Movement

The amount of turning time required for the bot to turn ninety degrees to the left or right, or to move forward one grid was determined through experimentation. Since changing the motor speed would change the time values, the time values were then made to be inversely proportional to the motor speed.

Colour Sensor

As mentioned above, the colour sensor was first calibrated with a white and black object in normal lighting conditions using colourcal.ino, and used the obtained RGB values to determine the lower and upper limits for the colour values. We then measured the values for the colours that would be used in the maze, such as red and green, and stored them in arrays. In the actual challenge, the values detected by the mBot would then be compared with the previously determined values to figure out which colour it is, and thus allow the bot to carry out the correct action.

The formula used for scaling RGB Values is as follows:

$$Current\ RGB\ Value = \frac{Current\ LDR\ Value - Black\ LDR\ Value}{White\ LDR\ Value - Black\ LDR\ Value}$$

Work Division

The initial building of the mbot was done by Zihao and the initial movement of the mBot was handled by Wira. The general movement of the mBot which required the readjustments between the walls was then handled by Walter, who also coded the IR sensor and the ultrasonic sensor. Wira was responsible for coding the color challenge, and was assisted by Brendan and Shuyi during the testing phase. Together, they made sure the mBot was able to correctly detect the colors during the color challenge and make the correct actions afterward. The building of the sound filter circuit and side IR sensors circuit were done by Zihao, who also wrote the code for the sound challenge and the pseudocode for the IR sensor. The celebratory tune played after the mBot reaches the finish line was coded by Shuyi. Wira and Walter were also responsible for merging all the codes together, and making sure the final code was free of bugs. Throughout the project, everyone was involved in brainstorming for ideas to improve the mBot and it’s capabilities.

Difficulties Faced

During the project, our group faced a few problems and have made efforts to overcome them. These problems mainly surfaced when we started working on the color and sound challenges.

While working on the sound challenge, our group spent a lot of time on the circuitry due to the insensitive microphone and possibly damaged components. Zihao realized that some of the pins on the breadboard we worked on might have been faulty, and exchanged for a new breadboard to build the circuit on. Halfway through the project, we also encountered the issue that the filter worked but the operational amplifier did not work, which might be caused by the incorrect gain value used for the op-amp in the bandpass and high pass filters. Due to the insensitive microphone, we initially decided to use a very high gain, but this resulted in the circuit amplifying unwanted ambient sounds during the sound challenge testing. The high gain also resulted in the signal reaching saturation point after amplification, causing the sine wave signal to become a square wave signal. To solve this problem, our group did many trial and error tests to determine the correct values of resistors used for the operational amplifier circuit, so that we can achieve the right amount of gain.

Nearer to the actual maze, we then discovered that the pin connection from the mic to the breadboard was also faulty, causing unreliable analogRead values coming from the mic. There was then a need to adjust the position of pin on the breadboard as it was determined that the issue was with the pinhole of the breadboard.

Meanwhile, we also encountered problems when working on the color challenge. While primary colors such as red and green were easily detected by the light sensor on the mBot, it had a hard time differentiating between non-primary colors such as purple, blue and yellow. To combat this issue, our group came up with the idea of using a tube made using a piece of black paper to surround the light sensor and RGB lights. This resulted in higher accuracy during the color challenge, as the light sensor is less likely to take in light from the environment. We also made changes to the code, so that the mBot will be able to differentiate between the different colors more reliably. This included using the least square distance method between obtained values and reference values. This method replaced the original method of using threshold ranges for each colour and was found to be much more reliable. Our group then tested the mBot extensively to ensure the outcome is consistent for the color challenge.

In the end however, while our mBot had decoded every waypoint in the final maze perfectly, we had forgotten to account for the slower motor, due to calibrating its turns when the battery was at a higher level than during the actual test, and ended up getting stuck once due to the pole.

- END -