mBot Project Report

Studio Group: 4B

Team Number: 4b-2

Team members:

Siew Yang Zhi

Sridharan Arvind Srinivasan

Stefan Choo Bin Hao

Wu Luoyu

1 Overall algorithm of *mBot*

The overall algorithm of *mBot* that allows it to overcome the challenges while completing the maze race is illustrated in Fig. 1. The *mBot* is set to move forward until a black strip is encountered. To keep the *mBot* moving in a straight line, the algorithm constantly checks if it is too close to one side of the walls and adjusts its movement to the other side. When a black strip is encountered, the *mBot* will stop to detect the colour above. If it detects red, green or yellow, it will make a left turn, right turn or 180° within the same square respectively. If it detects purple (or light blue), it will make a left-turn (or right-turn) in the first grid and move forward until it is 10 cm away from the wall. It then makes the second left-turn (or right-turn) in the second grid. After finishing these actions, the *mBot* will continue moving forward until it detects another black strip. If it detects black, it will stop and play a celebratory tone, indicating the end of the maze.

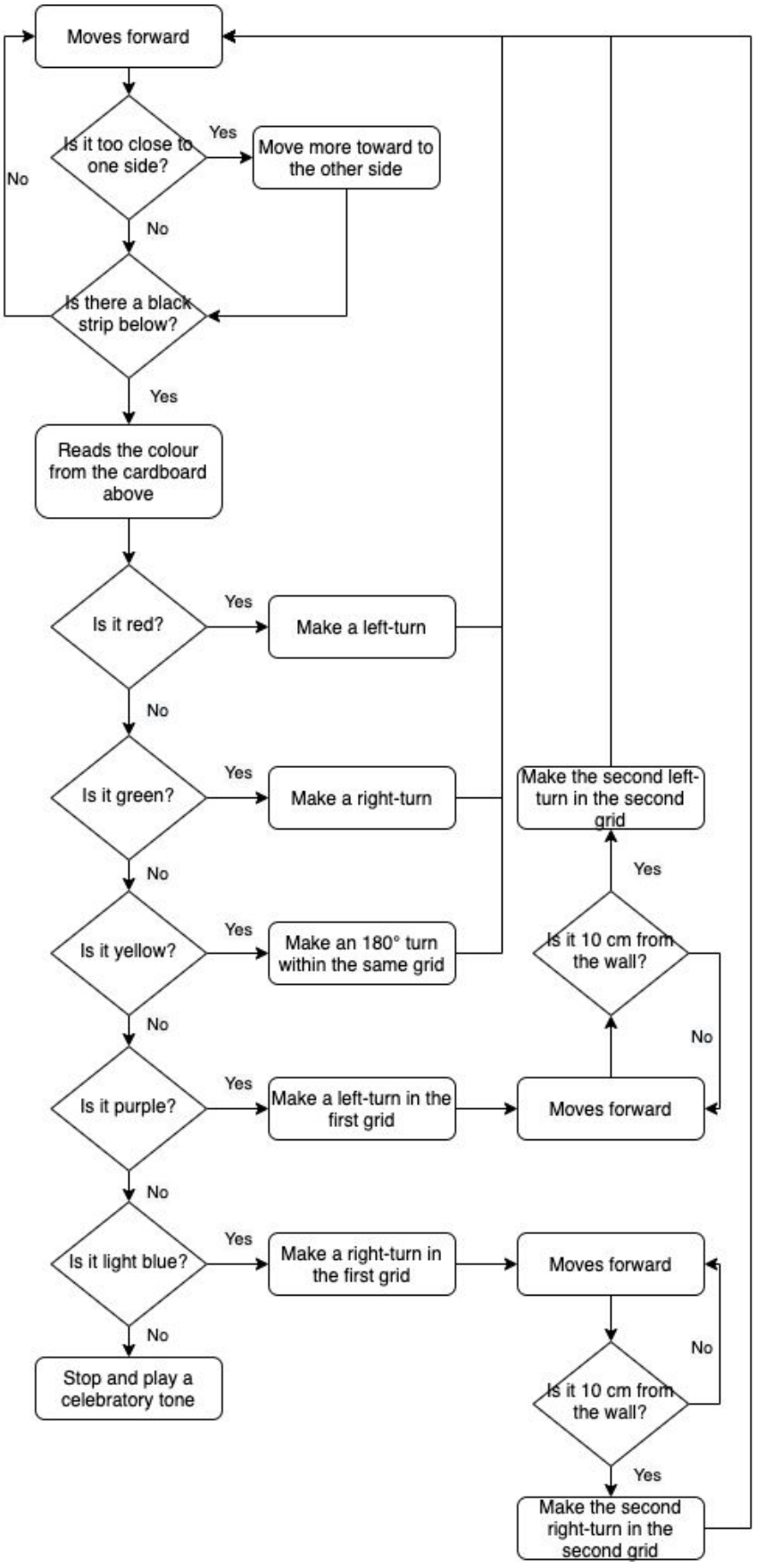

Fig. 1. Overall algorithm of *mBot*

2 Implementation of subsystems in *mBot*

This section describes the various subsystems in *mBot* and how they enable the *mBot* to complete the maze successfully. These subsystems include the motors, the colour-sensor, the Infrared (IR) sensors, ultrasonic sensor and the line follower sensor.

2.1 The motors

The motors are used to move the *mBot*. To make the *mBot* move in a particular direction, there needs to be an imbalance between the speeds of the motors i.e. to move to the left, the left motor should be moving slower than the right (Used in course correction). To make the *mBot* turn on the spot, one motor moves backwards while the other moves forward, allowing it to rotate along the axis between the two motors. This prevents it from moving forward while turning and potentially bumping into any walls. These principles are codified in the piece of code named *movement.h*.

2.2 The colour sensor

2.2.1 Colour sensing process

As illustrated in Fig. 2, to sense a colour, the RGB LED first flashes a red light before reading in the voltage value at the LDR after a delay of 50 milliseconds. The delay of 50 milliseconds is to ensure that the RGB LED has enough time to change its color to ensure that the LDR is reading the correct color. The *mBot* would read in the voltage readings from the LDR in intervals of 5 milliseconds for 5 times and take the average voltage across these 5 readings.

The LDR's resistance value would change based on the amount of light being reflected by the color paper as a result, the voltage read by the *mBot* would also change. As the amount of light reflected by each color is different, each color would output a unique voltage which can be then used to determine the colors. The *mBot* will then repeat the above process with the LED emitting a green light followed by a blue light.

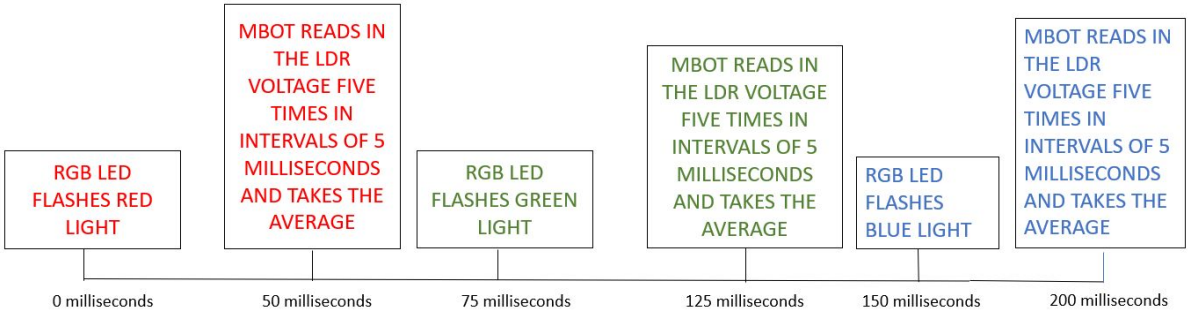

Fig 2. Color sensing process

After taking the RGB values of the colour paper it is sensing, the colour sensor then converts them to HSV (hue, saturation, value), which is another way of representing colours. The colour sensor outputs the Hue values which are used to determine what color is on the paper above it. This is possible because each color has a unique Hue value.

The *mBot* also checks the V component of HSV which stands for value. This is because the V component determines the brightness of the color, thus if the V components value is zero, the color detected must be black.

2.2.2 The working principle of the color sensor

The color sensor works using the principle that all colors are made up of varying amounts of the three primary colors - red , green and blue. Different colored objects absorb different amounts of each of the three colours of light, resulting in the intensity of the reflected light for each of the three colours to vary. The LDR's - Light Dependent Resistor - resistance will vary based on the reflected light intensity it perceives, for each of the three colours.

The *mBot* would then proceed to read in the various voltages for each of the three colours and use these three voltages to get the HSV (hue, saturation, value) of the color that it is attempting to identify. Each color has a unique HSV value and thus with the HSV value, we are able to identify what is the color the *mBot* is detecting.

We would have to calibrate the *mBot* with a black colored paper to take into account the effects the surrounding light has on the *mBot* when it is detecting the color.

2.3 The infrared (IR) sensors

The two IR sensors at the sides are used to make the *mBot* remain centered throughout the course. They are wired following the circuit as shown in Fig. 3 and attached on the sides of the *mBot* as shown in Fig. 4 and Fig. 5. The outputs (distances) from the two sensors are taken and the ratio between the two is calculated. Based on this ratio (left distance/right distance), the speeds of the wheels are changed to keep the *mBot* in the middle, i.e if the ratio is less than 1, implies the *mBot* is closer to the left than the right and so the right motor is slowed down so that the *mBot* moves back to the center. Similarly, if the ratio is greater than one, implies the *mBot* is closer to the right than the left and so the left motor is slowed down so that the *mBot* moves back to the center

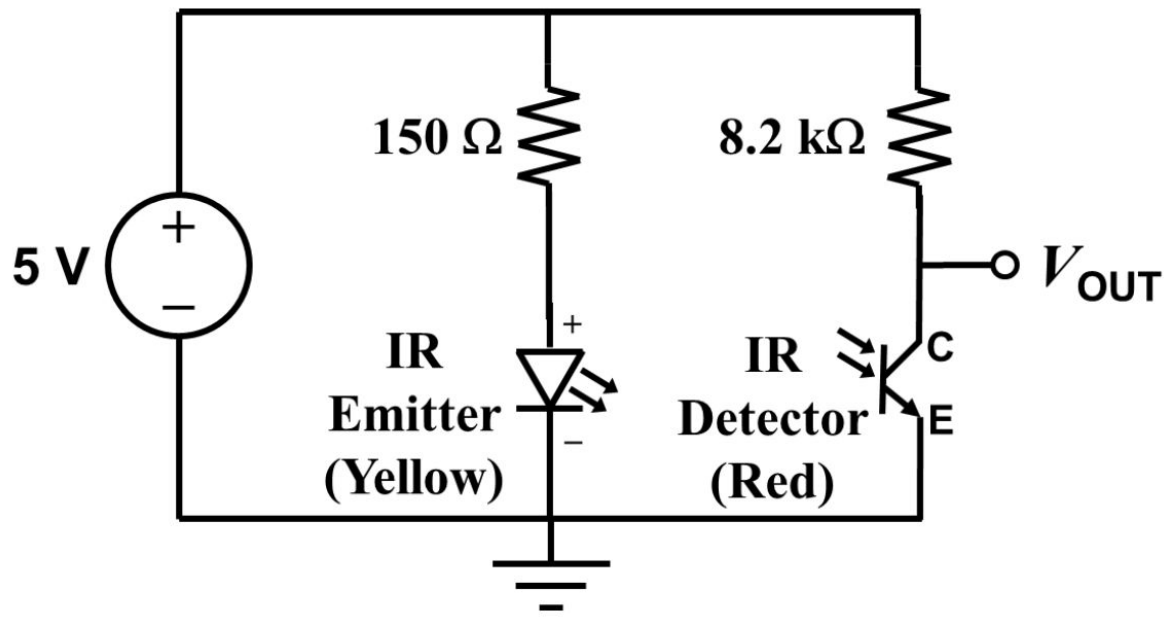

Fig. 3: IR sensor circuit

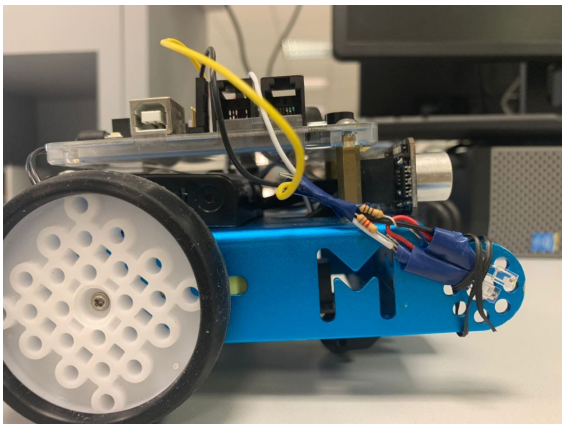

Fig. 4: Photo of the right IR sensor on *mBot*

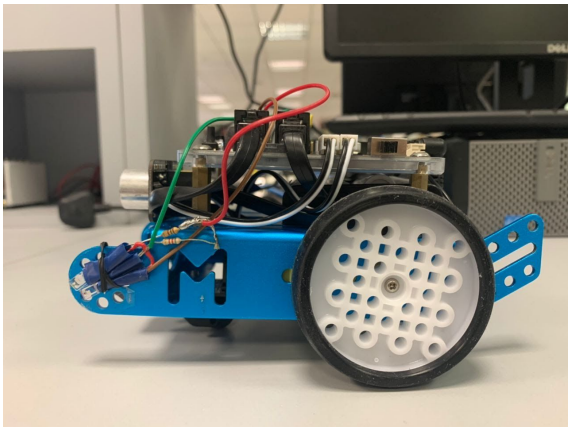

Fig. 5: Photo of the left IR sensor on *mBot*

2.4 The line sensor

The line sensor consists of two IR LEDs and functions by measuring the amount of light reflected by said LEDs. This allows it to differentiate between black/darker colors (less reflected light) and white/lighter colors (more reflected light). Using this we are able to ascertain where the black strip is. Since the maze floor is white, the only place where the line sensor would receive an output of black is the black strip, making it easy to know if the *mBot* is on the black strip.

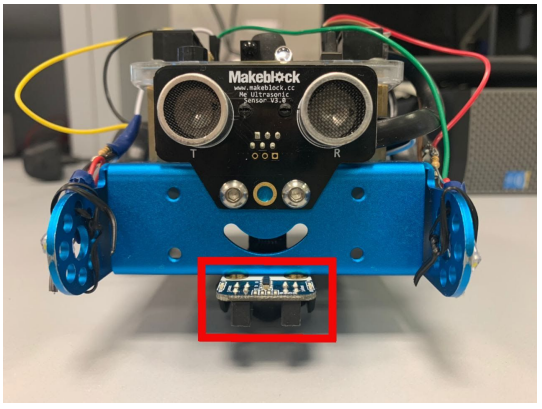

Fig. 6: Photo of the line sensor on *mBot*

2.5 The ultrasonic sensor

The ultrasonic sensor is placed in the front of the *mBot* as shown in Fig. 7 and used to prevent it from hitting the wall in front during two successive left-turns or two successive right-turns in two grids. Ideally, the ultrasonic sensor is coded such that after the first left-turn or right-turn, the *mBot* will make the second turn when it is 10 cm away from the wall in front. For example, in Fig. 8, the *mBot* is required to make two consecutive right-turns for this particular challenge. After making the first right-turn, it will make the second right-turn when it reaches the red line which is 10 cm away from the wall in front.

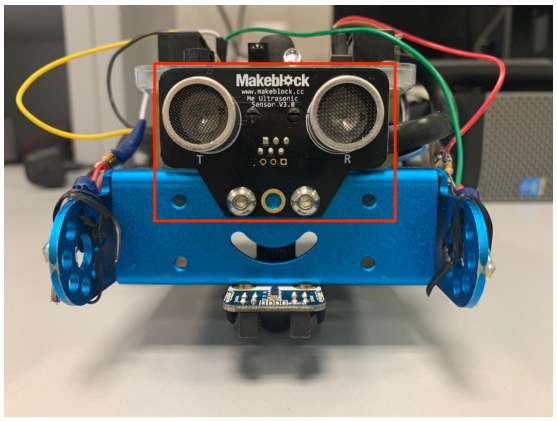

Fig 7: Photo of ultrasonic sensor on *mBot*

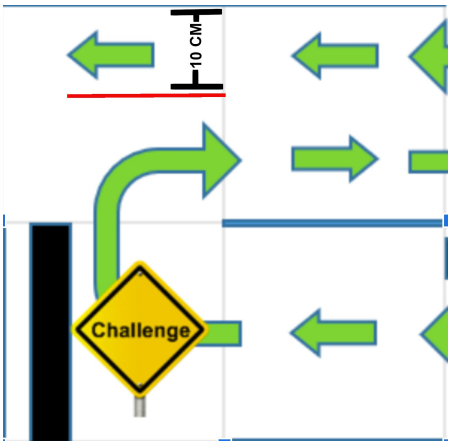

Fig 8: The way-point challenge for two consecutive right-turns

3 Steps taken for calibration

The colour sensor is calibrated by taking a reading of the colour black to take into account the different intensities of light reflected by the separate colours as the LEDs are not uniformly bright.

4 Work division

Motor: Arvind

Colour sensor: Yang Zhi, Stefan

IR sensor: Stefan, Luoyu

Ultrasonic sensor: Arvind

Written report and documentation: Luoyu, Yang Zhi, Arvind, Stefan

5 Challenges and actions taken

In this section, the three main challenges we encountered during the prototyping stage and the actions taken to address them will be discussed.

5.1 Slow detection of ultrasonic sensor

Initially we relied on the ultrasonic sensor to prevent the *mBot* from bumping into the wall in front throughout the entire course. However, since we set the *mBot* at its maximum speed, the ultrasonic sensor was unable to stop the *mBot* in time before the black strip. This is probably due to how the ultrasonic sensor works as it takes time (10ms) to generate the input pulse itself. This makes one cycle of distance sensing itself take quite a while which reduces the responsiveness of the sensor, leading it to overshoot the thresholds set in some cases. As the *mBot* ended up stopping too close to the wall due to this overshooting, it often failed to turn smoothly without sliding against the wall or turn successfully at all.

To rectify this, we used the line follower sensors instead as they were faster to respond compared to the ultrasonic sensor. In the MeCore Library, the function readSensors is coded such that it returns 0 when the sensors are atop a black line. By making use of this function, the *mBot* stopped immediately whenever a black strip was detected.

The ultrasonic sensor was nevertheless needed for making two successive turns within two grids. This was because before the second turn needed to be taken, there was no indication for where the *mBot* was to stop and make the turn as there was no black strip. Hence, using the ultrasonic sensor was the only means to ensure that the *mBot* turned again in the second grid. The details of the ultrasonic sensor are covered in Section 2.5.

5.2 Inaccurate colour sensing using RGB values

In the beginning, we adopted the method provided in Week 10 Studio 1 which detected the amount of red, green and blue light reflected from the object shone on by the LED on the *mBot*. As we understood the inconsistency of the absolute values of the three primary colours due to changing ambience, we decided to make use of the relative values of the three RGB colours. For example, if there is more green than red and the difference between green and blue values is less than 10, we determined it to be light blue. If there is more green than red and the difference between red and blue is less than 5, the colour would be green. Even though this method was more promising, it still relied on the absolute values to a small extent (e.g. taking the absolute difference between two colours) and thus gave inaccurate detection from time to time.

To improve on this, the LDR (Fig.9) was covered with foam to ensure that the light it can receive is only from the coloured paper above. This is to prevent ambient light as well as light from the RGB LED to enter it. We also used an alternative representation of the RGB color model, which is HSV. In the RGB colour model as shown in Fig. 10, it uses an additive method where the more of each colour of the three primary colours is added, the brighter it becomes. The HSV colour model as illustrated in Fig. 11, on the other hand, uses three parameters including hue, saturation and value to describe colours. The advantage of using the HSV model is that, theoretically only the Hue value is needed to capture a general colour (e.g. red, blue). This is because the Hue component describes the colour itself in the form of an angle between [0,360] degrees. For example, 0 degree means red; 120 means green; 240 means blue; 60 degrees is yellow. In the improved version of the codes, RGB values were thus converted to HSV values for determination of colour, which was empirically proven to be almost always accurate.

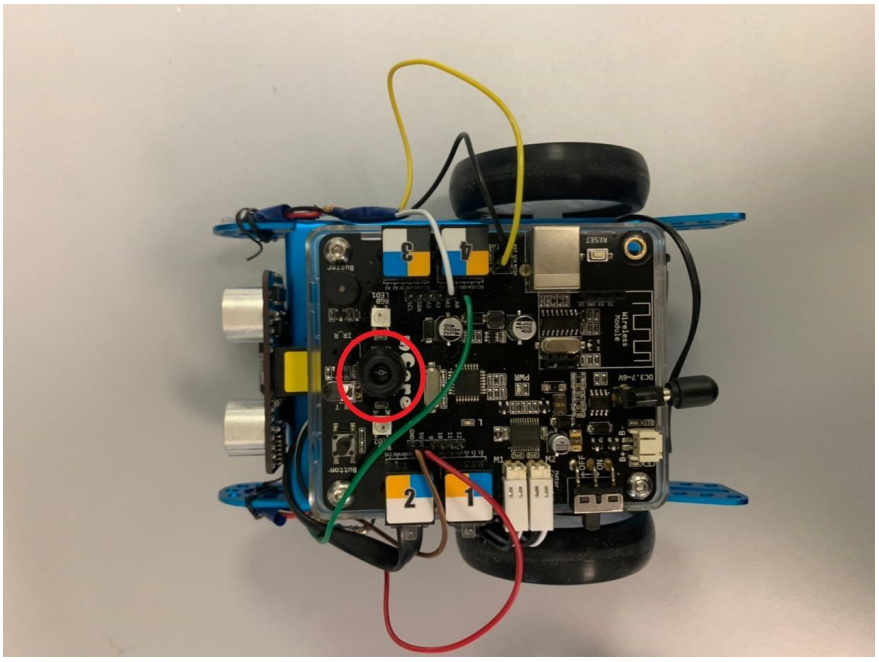

Fig 9. Foam tip to reduce the RGB LEDs and ambient light's effect on the readings

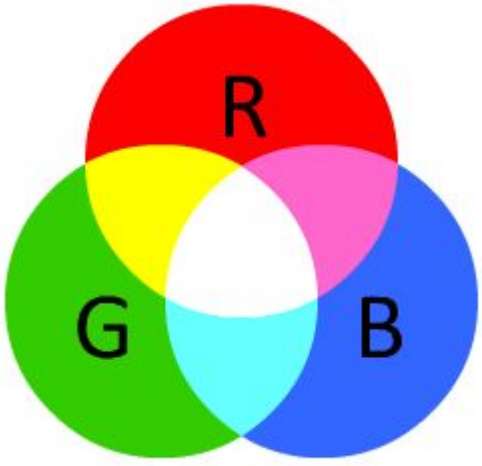

Fig 10. RGB colour model

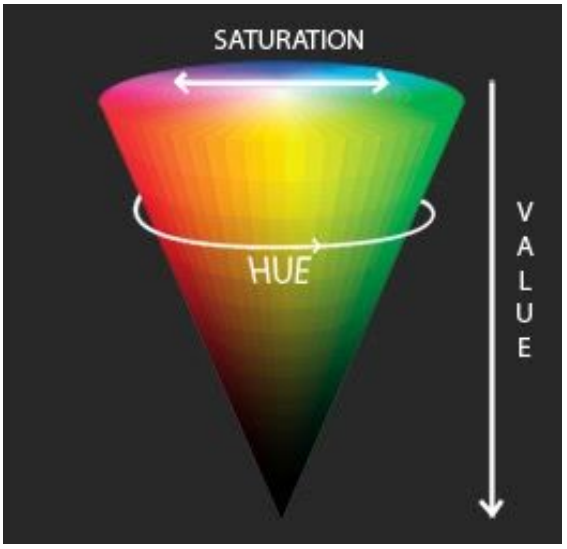

Fig 11. HSV colour model

5.3 Ineffective IR sensor using absolute values

To ensure that the *mBot* moves in a straight line, we assembled and attached an IR sensor and emitter on each side of the *mBot* (as shown in Fig. xx). When the *mBot* was too close to one side of the wall, it needed to turn slightly to the other side. Initially, the code was written in a

way such that the *mBot* would turn at a fixed angle when it reached a certain absolute distance to either side of the wall. However, the course correction using this method resulted in the *mBot* moving in a zig-zag manner.

To rectify this, we recognised the fact that the extent of the change of direction when the *mBot* is too closed to one side of the wall depended on the relative distances of the *mBot* to the left wall and right wall. The change in speed of the two motors thus depended on the relative distances as well. Hence, to represent this relation, we calculated the ratio of the two distances which was then exponentiated to be more pronounced. The resulting factor was then incorporated to the two motors' speeds to make the wheels rotate faster or slower accordingly.

In addition, to avoid unnecessary turning into the space where a wall is missing on one side of the maze, the distance used in calculations was capped at 7 cm.