CG1111A Engineering Principles and Practice I

The A-maze-ing Race Project 2023

10 October 2023

Introduction

Welcome to the grand project of CG1111A: The A-maze-ing Race!

In this race, your mBot needs to find its way through a maze in the shortest time. Similar to its namesake TV program, your mBot will be facing a number of challenges at intermediate waypoints while attempting to complete the race. To successfully meet all the requirements, you need to have a good grasp of many of the principles you have learnt in CG1111A and apply them into good practice!

Key Project Requirements

1. The mBot must not bump into any wall. Your mBot shall accomplish this with the help of one ultrasonic sensor on one side, and one infrared (IR) proximity sensor on the other side (no restriction on which of these two sensors to place on the left or right). You need to come up with your own algorithms to meet this requirement. Note that there will be penalty points for bumping into walls (including wires brushing against the wall), even if your mBot doesn't get stuck.

2. When your mBot is not making a turn, it must travel as straight as possible. (It must not drive in a zig-zag manner like a car driven by a drunken driver.)

3. All turns in the maze are dictated by "waypoint challenges". Your mBot must not make any automatic turn without decoding a waypoint challenge.

4. When making a turn, your mBot must not over- or under-manoeuvre too much.

5. At each waypoint challenge, there will be a black strip on the maze floor. Your mBot needs to detect the black strip, stop, solve the waypoint challenge directly underneath it, and act according to the turn instruction decoded from the waypoint challenge.

6. Waypoint Challenge: Colour-sensing

   You need to build your own colour-sensing circuit on a mini-breadboard, and place it underneath your mBot. At each waypoint challenge grid, besides the black strip, there will be a colour paper directly underneath your mBot. (Please refer to the video "Black Strip and Colour Paper's Positions.mp4" in Canvas for an illustration of how the black strip and the colour paper will be placed.) Depending on the colour of the paper, your mBot needs to execute one of the following five types of turns:

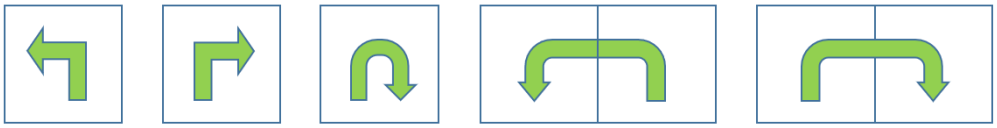


Figure 1: The five types of turns that your mBot needs to execute.

Note:
For the "two successive left-turns in two grids" and the "two successive right-turns in two grids", there will not be any black strip in the second grid to guide the mBot to execute the second turn. Your mBot needs to be hard coded to make these successive turns.

Table I summarizes how the colours are to be interpreted. Colour paper samples will be provided.

Table I: Colour Interpretation for the Colour-sensing Challenge

Colour	Interpretation
Red	Left-turn
Green	Right turn
Orange	180° turn within the same grid
Purple	Two successive left-turns in two grids
Light Blue	Two successive right-turns in two grids

7. End of Maze:

   At the end of the maze, there will also be a black strip. The colour of the paper underneath the mBot at this grid will be **white**. Upon decoding that it has reached the end of the conquest, the mBot must **stop moving**, and **play a celebratory tune** of your choice (Yay!).

Key Knowledge and Skills Needed

- DC Circuit Principles
- Arduino/mBot programming (self-learn)
- Analog-to-digital conversion
- How to read/interpret datasheets
- Circuit building skills
- Principles of IR proximity sensor
- Principles of colour sensor
- Principles of ultrasonic range sensor
- Hardware wiring and debugging skills
- Equipment usage (multimeter, etc.)

Key Components

1) mBot Robotic Platform

Figure 2: mBot robotic platform.

This is the robotic platform that we will be using for the project. The hardware comes with two key sensors, namely, a line sensor, and an ultrasonic sensor. You will be **building additional sensors** and integrating them with the mBot to meet the project specifications.

You can learn a lot about how the mBot works by going through its default factory firmware. For instance, it contains code that lets the mBot follow a black line once fully assembled. You will notice that the code looks very similar to Arduino program codes. This is because the mCore (i.e., mBot's brain) uses the ATmega328 microcontroller, which is also used by the Arduino Uno. You will be using the Arduino IDE to program the mBot as well.

Since the mBot can already perform the line-following trick right out of the box, it will be trivial if our project maze also has a line for it to follow! Rather, your mBot will be relying on two side proximity sensors to allow it to align itself within the maze. One of these will be the ultrasonic sensor. Instead of mounting it at the front, you will mount it either on **mBot's left or right** (your choice). The other side's proximity sensor will be your self-built infrared (IR) proximity sensor which you have already learnt in our previous studio.

2) Makeblock Ultrasonic Sensor

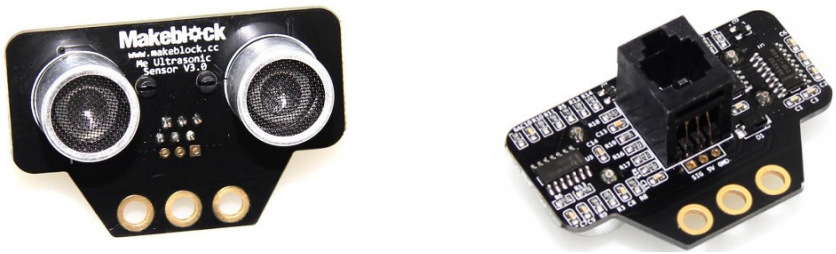


Figure 3: Makeblock ultrasonic sensor.

This ultrasonic sensor works in a similar manner as the one you have used in our previous studio. Although the ultrasonic sensor is a very accurate range measurement device, you need to take note

of its operating range, which is specified as **3 cm** to **400 cm**. The minimum range of 3 cm implies that you may not be able to detect an obstacle if it is less than 3 cm away. Hence, you should mount your ultrasonic sensor **inward by a few cm**, so that the sensor is still at least 3 cm from the wall even when your mBot touches the wall.

The Makeblock ultrasonic sensor is to be connected to your mCore using an RJ25 cable. As it only requires digital pins, you should connect it to **either port 1 or port 2**, so as to free up mCore's analog pins (A0 – A3) that are contained within ports 3 and 4. The figure below shows the pin mappings within each of the four RJ25 ports:

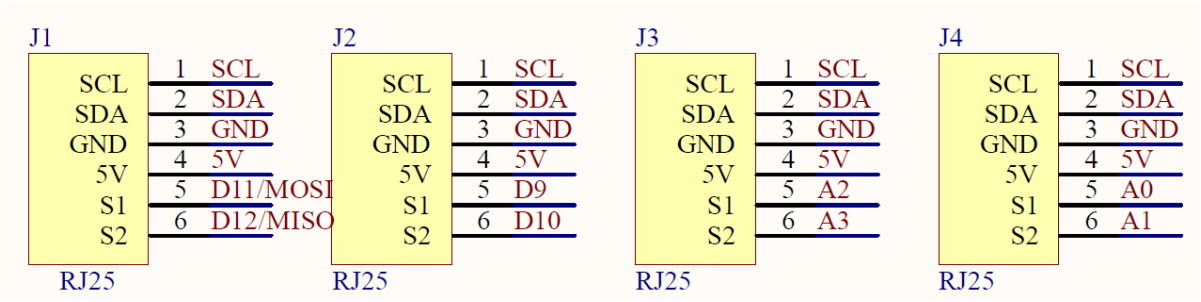


Figure 4: Pin mappings within each RJ25 port.

From the sample code given on Page 8 of our "mBot introduction.pdf" file, notice that you can specify a timeout for the ranging operation. It is recommended that you **choose this timeout value wisely** (making informed decision considering speed of sound, maximum range of side wall, etc., **like what an engineer would have done**) so that your mBot does not waste time waiting in vain for reflected pulses beyond what would be possible in your maze setting; otherwise, your mBot will become **less responsive** as it cannot perform any other decision making while it is waiting for the timeout to lapse.

3) Makeblock Line Sensor

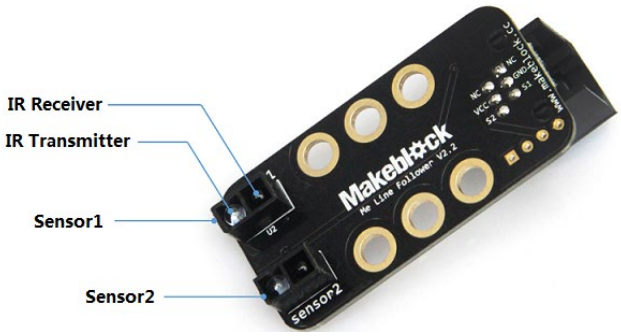


Figure 5: Makeblock line sensor.

This line sensor uses infrared (IR) rays to detect whether there is any black line. It is to be mounted underneath the mBot in front of its mini caster wheel. There are two pairs of IR transmitter+receiver (Sensor 1 & Sensor 2) on this line sensor. Page 10 of our "mBot introduction.pdf" file provides an example of how to read the results of Sensors 1 & 2. Notice that each of these two sensors only returns a Boolean result, either "IN" the black line (i.e., TRUE), or "OUT" of the black line (i.e., FALSE), unlike the IR proximity sensor from our studio that can be used to estimate an obstacle's distance.

How does it work? Most black objects readily absorb IR; hence when the sensor is above a black line, no IR is reflected, and the sensor returns an "IN"; otherwise, it returns an "OUT".

The Makeblock line sensor also uses the RJ25 cable. Just like the ultrasonic sensor, it only requires digital pins, hence you should also connect it to **either port 1 or port 2**, so as to free up mCore's analog pins (A0 – A3) that are contained within ports 3 and 4.

Notes:

i) The top side of this line sensor has two blue LEDs that conveniently show you the sensing results. (If the blue LEDs are not lit when the line sensor is on a white surface, your line senor may be faulty; consult the professor or the TA.)
ii) Strangely as it is, some black paper does reflect IR pretty well. Be reassured that we won't use such black paper during the project evaluation. :-)

4) RJ25 Adapter

Previously, we have seen in Figure 4 that each RJ25 port on the mCore contains 6 pins, namely, SCL, SDA, GND, 5V, S1, and S2. SCL and SDA are two lines used for communications using the I2C Communication Protocol, which you can safely ignore in CG1111A. GND and 5V are the ground and 5V lines used for supplying power to peripheral devices connected to this port. Finally, S1 and S2 are two "signal" wires that are connected to two input/output (I/O) pins on the mCore, very much like the I/O pins on your Arduino Uno. The exact I/O pin numbers depend on which of the four RJ25 ports you are using. For example, if you are using port 4, S1 is connected to analog pin A0, while S2 is connected to analog pin A1 (see Figure 4). Now, how do you tap these pins when you want to interface your custom-built sensors to the mBot? The solution is to use the RJ25 adapter shown in Figure 6.

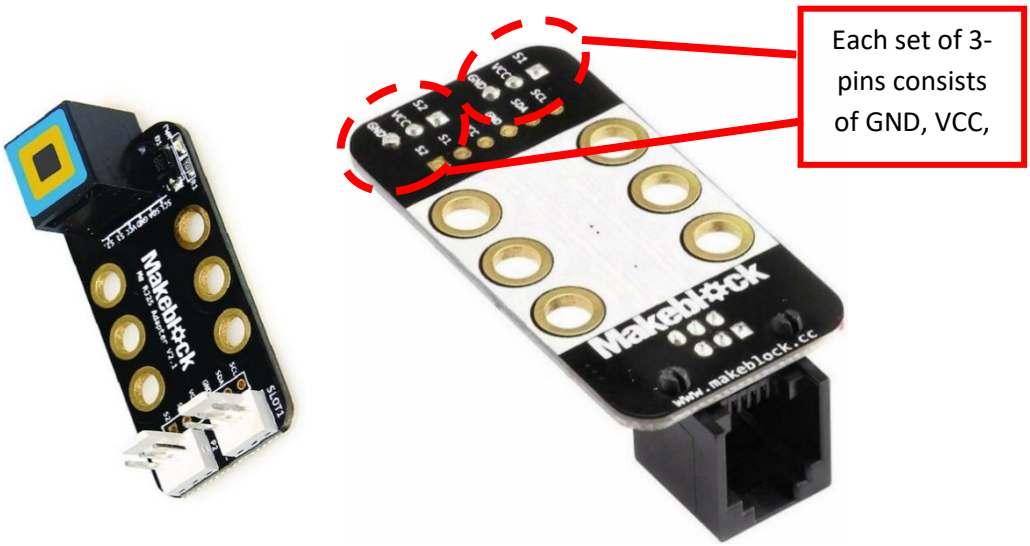
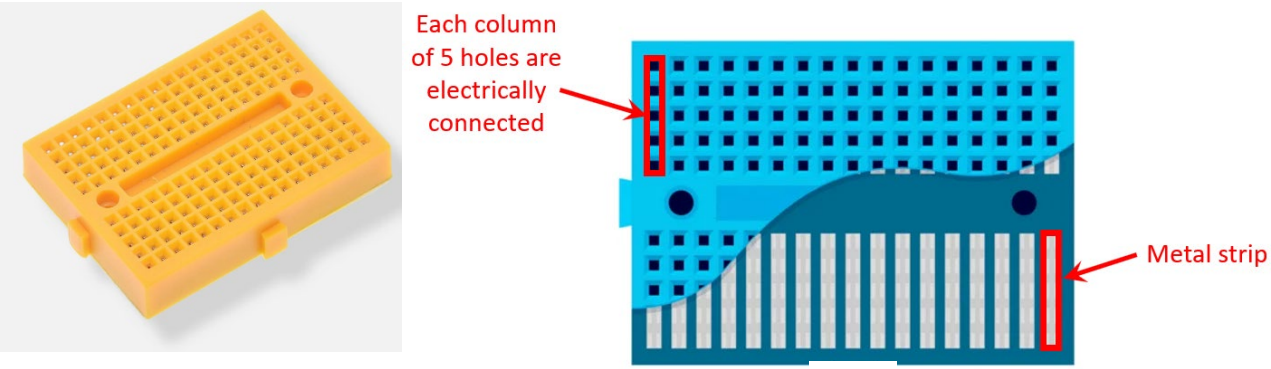


Figure 6: RJ25 adapter.

The RJ-25 adapter breaks out the wires into two sets of 3-pins, consisting of **GND**, **VCC** (~ 5V), and a signal pin (either **S1** or **S2**). It is very convenient for interfacing with your own custom-built circuits.

5) 170-wire Mini-breadboard



(a) (b)

Figure 7: (a) 170-wire mini-breadboard, (b) Anatomy of the mini-breadboard.

Instead of using the regular full-size breadboard on your mBot which doesn't look cool, you are given several of these 170-wire mini-breadboards. As can be seen in Figure 7(a), there is only a single gap in the middle. On each side of the gap, there are 17 columns of 5 holes, where each column of 5 holes are electrically connected (just like your regular full-size breadboard's column). Figure 7(b) shows the anatomy of the mini-breadboard, illustrating its internal electrical connectivity.

When constructing your circuits on the mini-breadboard, you must ensure that the components **fit snugly** into the holes so that they don't come off easily when your mBot bumps into anything. You can use the **single-core wires** (those without header pins) in the lab, and **trim them** to the required lengths using the wire cutters in the lab.

You can use the double-sided tape underneath the mini-breadboard to stick it to your mBot. However, as it gets damaged when you try to remove it (quite messy because all the metal strips come off), you may want to stick it only after you have finalized where you want to place it on the mBot.

6) HD74LS139P 2-to-4 Decoder IC Chip

We have seen that the ultrasonic sensor and the line sensor already take up two ports on your mCore. The remaining two ports are only left with a total of four signal pins. As your colour sensor requires four signal pins (three pins for turning on/off the individual Red, Green, and Blue LEDs, and one pin for sensing the LDR circuit's voltage), while your IR proximity sensor requires two signal pins (one pin for turning on/off IR emitter, and one pin for sensing the IR detector's voltage), you realize that there are not enough pins to accomplish the mission. This is where the 2-to-4 decoder IC chip comes to the rescue.

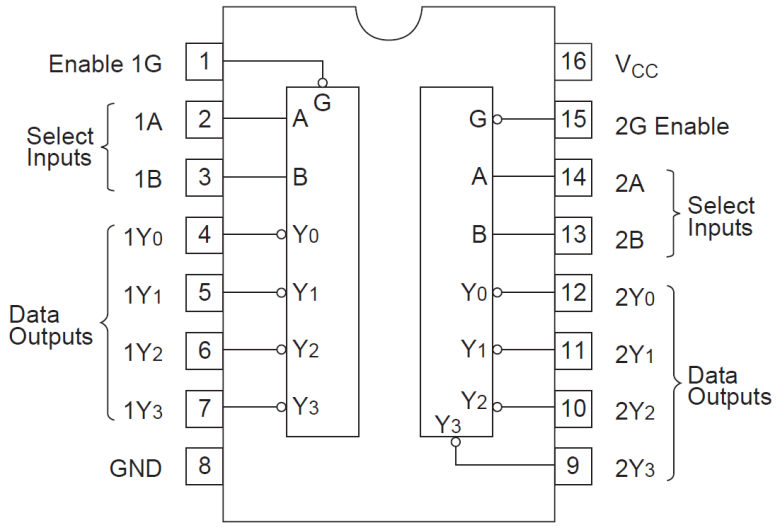


Figure 8: Pin layout of HD74LS139P 2-to-4 decoder IC chip.

The HD74LS139P 2-to-4 decoder IC chip consists of two sets of 2-to-4 decoders. As seen in Figure 8, the first decoder comprises pins 2-7, while the second decoder comprises pins 9-15. Pin 8 is GND, and Pin 16 is Vcc (to be connected to 5 V). You only need to use one of the two sets of decoders within the IC chip for this project. Each decoder consists of one "Enable" pin "G", two "Select Inputs" pins "A" and "B", and four "Data Outputs" pins Y_0, Y_1, Y_2, and Y_3. Their purposes are described below:

- **"Enable" pin "G":** for activating/deactivating each decoder.
- **"Select Inputs" pins "A" and "B":** for selecting which *one* of the four "Data Outputs" pins, "Y_0" to "Y_3", is to be activated.
- **"Data Outputs" pins "Y_0" to "Y_3":** if pin "G" is enabled, *one* of these four pins will be *active*.

Notice in Figure 8 that there are circles ("o") at pin "G", as well as pins "Y_0" to "Y_3", implying that these are "Active LOW" pins. The implication is that a logic LOW is regarded as "active". Hence, to enable a decoder, its "Enable" pin "G" needs to be connected to logic LOW (**you cannot leave it "floating"**). Similarly, when one of the four "Data Outputs" pins, "Y_0" to "Y_3", is activated, it will output a logic LOW (close to 0 V, but not 0 as there is some overhead voltage); otherwise, it will output a logic HIGH (lower than 5 V, as there is some overhead voltage). Table II summarizes the behavior of the decoder.

Table II: Function Table Summarizing the Behavior of the HD74LS139P 2-to-4 decoder

Inputs			Outputs			
Enable	Select					
G	B	A	Y_0	Y_1	Y_2	Y_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H ; high level, L ; low level, X ; irrelevant

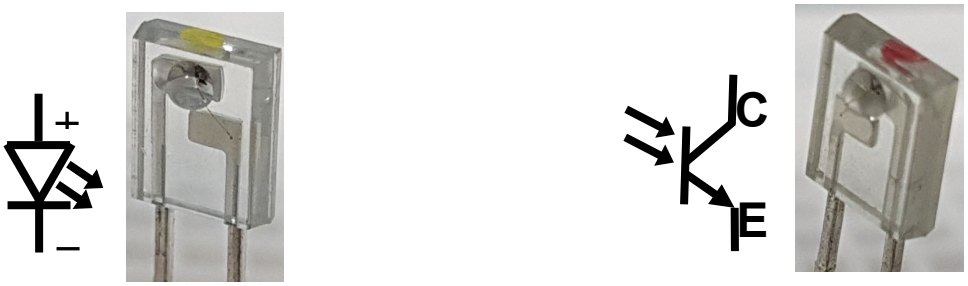
With the 2-to-4 decoder, you can now use just two signal pins from your mCore to control the turning on/off of the three colour LEDs (i.e., red, green, blue) as well as one IR emitter. This leaves your mCore with two remaining signal pins for reading the LDR circuit's voltage and the IR detector's voltage. Perfect!

From the HD74LS139P 2-to-4 decoder IC chip's datasheet, we also note that its **current limit** for each of the data output pins "Y_0" to "Y_3" is only **8 mA** (see Table III below). Hence, when **choosing the resistors** for your colour LEDs, one of the constraints is to ensure that this 8 mA current limit is never exceeded (use KVL, Ohm's Law, I-V characteristics of LEDs and IR emitter).

Table III: Recommended Operating Conditions of the HD74LS139P 2-to-4 decoder IC Chip

Item	Symbol	Min	Typ	Max	Unit
Supply voltage	V_{CC}	4.75	5.00	5.25	V
Output current	I_{OH}	—	—	–400	µA
	I_{OL}	—	—	8	mA
Operating temperature	Topr	–20	25	75	°C

7) IR Emitter and Detector



IR Emitter (Yellow Dot) IR Detector (Red Dot)

Figure 9: IR emitter and detector.

The IR emitter (LTE-302) and IR detector (LTR-301) that you are using for the project are the same as the ones you have used in our previous studio on photoelectric sensors. However, as you need to be able to **turn on and off the IR emitter at regular intervals to adapt to the amount of ambient IR** (more about this later in the section "Some Helpful Tips"), you will be using the HD74LS139P 2-to-4 decoder to control its turning on and off. Unlike the colour LEDs which work pretty well with currents below 8 mA, the IR proximity sensor will have very short sensitivity range if your IR emitter uses such a low current. Hence, we want to increase its current. Table IV below shows the IR emitter's specifications from its datasheet. From the table, we noted that its maximum rated continuous forward current is 50 mA. To operate at this current, we cannot drive the IR emitter using the HD74LS139P 2-to-4 decoder directly. Instead, we will use an additional IC chip to help us – the L293D motor driver chip!

Note: The IR emitter can withstand higher current pulses beyond 50 mA so long as the average dissipated power for the chosen duty cycle (like PWM) is < 75 mW. However, to make things simple (and also to avoid accidentally damaging your IR emitter), you can stick to a maximum current of 50 mA by choosing an appropriate current-limiting resistor. Note that the voltage across the IR emitter would be around 1.5 V when the current flowing through it is 50 mA. Use circuit laws to calculate the current-limiting resistance value you need.

Table IV: IR emitter's specifications from Datasheet

PARAMETER	MAXIMUM RATING	UNIT
Power Dissipation	75	mW
Peak Forward Current (300pps, 10 μ s pulse)	1	A
Continuous Forward Current	50	mA
Reverse Voltage	5	V
Operating Temperature Range	-40℃ to + 85℃	
Storage Temperature Range	-55℃ to + 100℃	
Lead Soldering Temperature [1.6mm(.063") From Body]	260℃ for 5 Seconds	

Forward Voltage	V_F		1.2	1.6	V	I_F = 20mA	

Figure 10 shows an example of how you can use an output pin (e.g., Y_0) from the 2-to-4 decoder to drive a particular output pin of the L293D chip to either logic HIGH or LOW. In this example, when pin 7 is LOW, pin 6 will also be LOW, and the IR emitter will turn on. Conversely, when pin 7 is HIGH, the IR emitter will turn off.

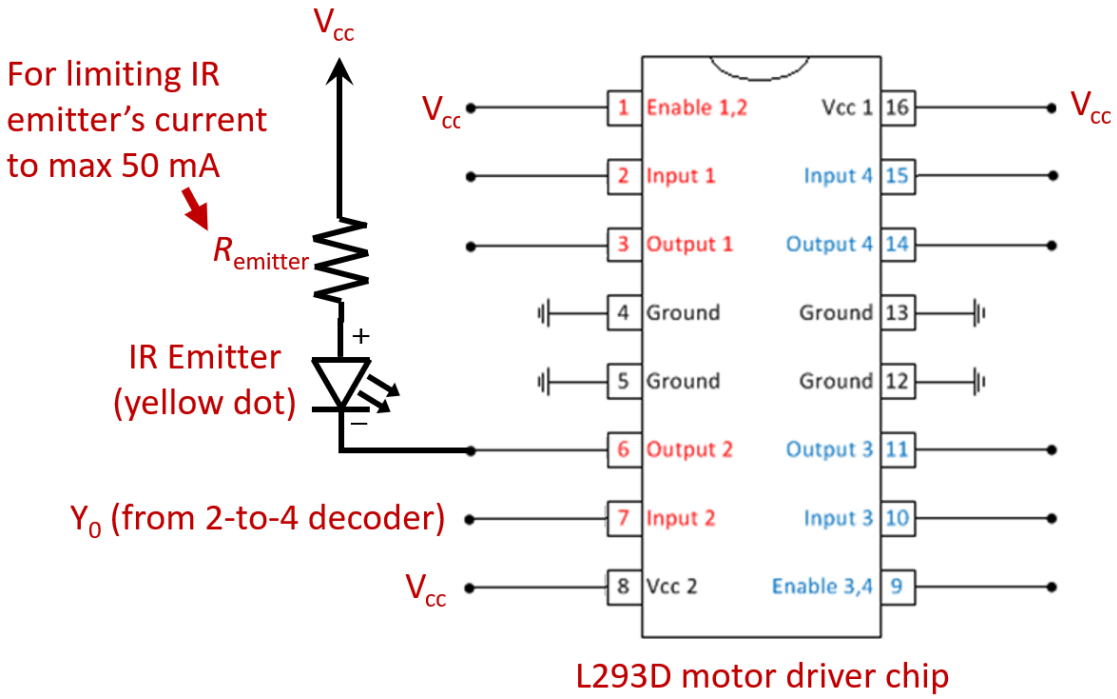


Figure 10: Using L293 motor driver chip to bypass 2-to-4 decoder's 8 mA current limitation.

For the IR detector circuit, you could try to experiment what resistance value to use for $R_{receiver}$ to make it more sensitive for your setting. The 8.2 kΩ used in the studio may not be the best value for your setting. A larger $R_{receiver}$ makes the receiver voltage more responsive. However, it shouldn't be too large or it may result in saturation easily in the presence of even a little IR.

Figure 11 shows an example of how to connect your IR detector circuit. As your mCore needs to measure the voltage of the IR detector, you need to use one of its analog sensing pins contained in either port 3 or 4. Use the RJ25 adapter to access this signal pin.

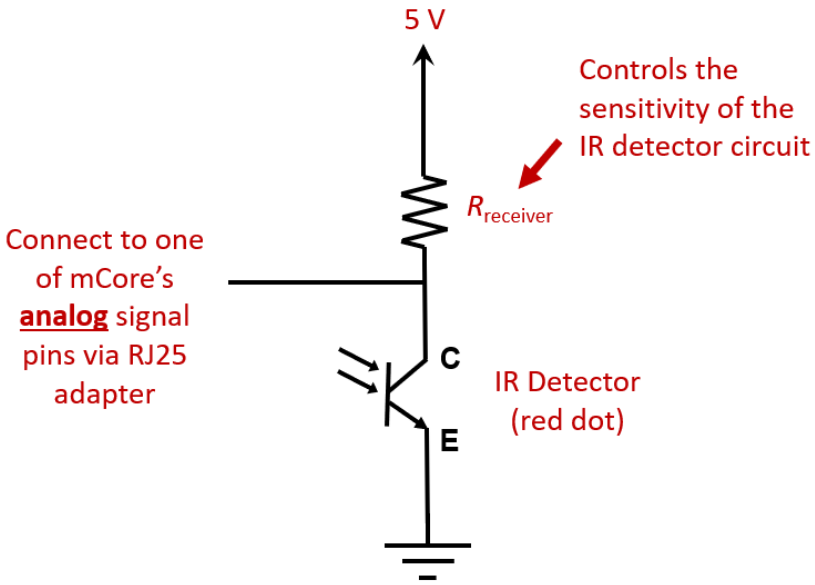


Figure 11: IR detector circuit.

8) Light Dependent Resistor (LDR)

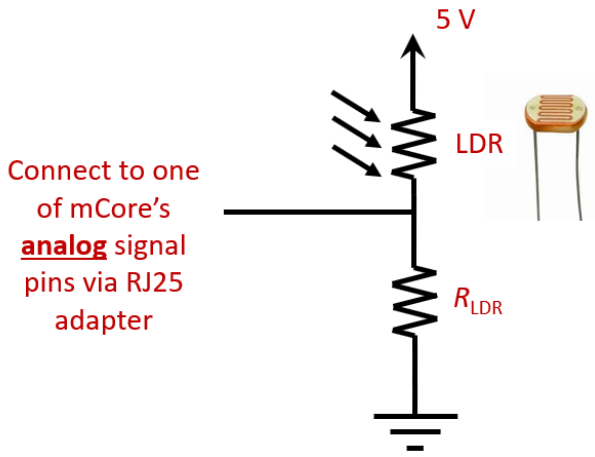


Figure 12: LDR circuit for detecting light intensity.

The light dependent resistor (LDR) issued for the project is the same as the one used in your previous studio on photoelectric sensors. You will use this LDR together with red, green, and blue LEDs to build a colour sensor, to be placed underneath the mBot. The resistance value of the biasing resistor R_{LDR} used in the studio was 100 kΩ. You can try experimenting with other values if you like. As your mCore also needs to measure the voltage from the LDR circuit, you need to use one of its analog sensing pins contained in either port 3 or 4, via the RJ25 adapter.

9) Red, Green, and Blue LEDs

For this project, you will be building your colour sensor using individual red, green, and blue LEDs, instead of the RGB LED Lamp that you have used in our previous colour sensing studio activity. The reason is because the HD74LS139P 2-to-4 decoder's data output pins "Y_0" to "Y_3" are active LOW (i.e., it gives a logic LOW when the output pin is "active"), meaning that we cannot use a common-cathode RGB LED Lamp if we want to only turn on one colour LED at a time.

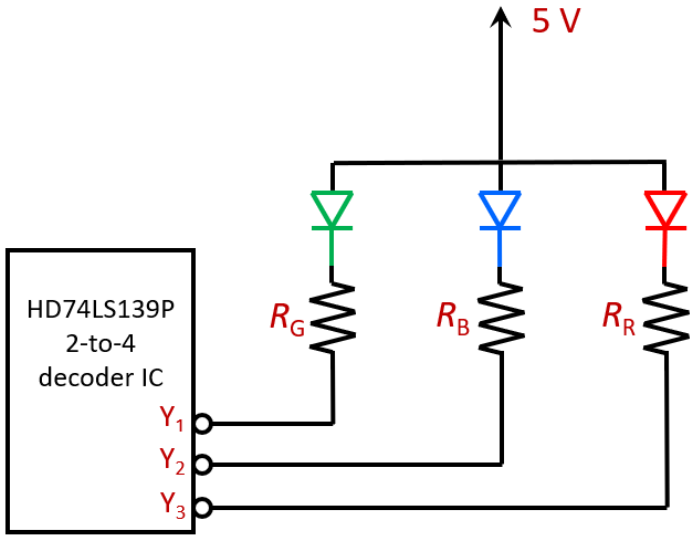


Figure 13: Controlling the red, green, and blue LEDs.

Figure 13 shows how you can connect the three colour LEDs to the HD74LS139P 2-to-4 decoder. When choosing the biasing resistance values for R_G, R_B, and R_R, do not pick values that would cause the current to exceed the 8 mA maximum limit that the decoder output pins can take. Use KVL, Ohm's Law, I-V characteristics of each individual colour LED (different for each colour LED), and the active LOW voltage of the decoder (not 0 V) in your calculations. Make informed calculations like an engineer. If you choose a resistance value that is too small, the output pin's logic LOW voltage will rise to keep the current under 8 mA due to the IC chips's self-protection mechanism, and your circuit's behavior will become **unpredictable**. Besides calculating the minimum allowable resistance values for R_G, R_B, and R_R, you may also want to **further reduce the brightness** of any particular colour LED if it is too bright (i.e., by choosing a resistance value that is larger than the minimum allowable value).

Note: Whether an LED is considered too "bright" is not necessarily the brightness that you perceive using your eyes. Rather, the "brightness" should be from the **perspective of the LDR**. For example, if the LDR always detects a lot of reflected light for a particular LED colour regardless of what colour paper you are trying to sense, that LED is probably too bright and you need to reduce its current; otherwise your colour sensor will not be very sensitive. From experience, red LED tends to have this issue, and needs a much lower current than 8 mA.

Some Helpful Tips

1) Tackling Ambient IR Variations

As you would have noticed during your studio on IR proximity sensors, the voltage of the IR receiver in your setup depends not just on the amount of reflected IR rays from the obstacle, but also on how much IR is present in the surroundings (e.g., from sunlight entering the windows, lightings in the lab, warm objects nearby, etc.). Figure 14 below shows an example of how the IR detector's voltage can be affected by the amount of ambient IR. Hence, it is of utmost importance that you develop a robust IR proximity sensor subsystem that can adapt to the ambient IR!

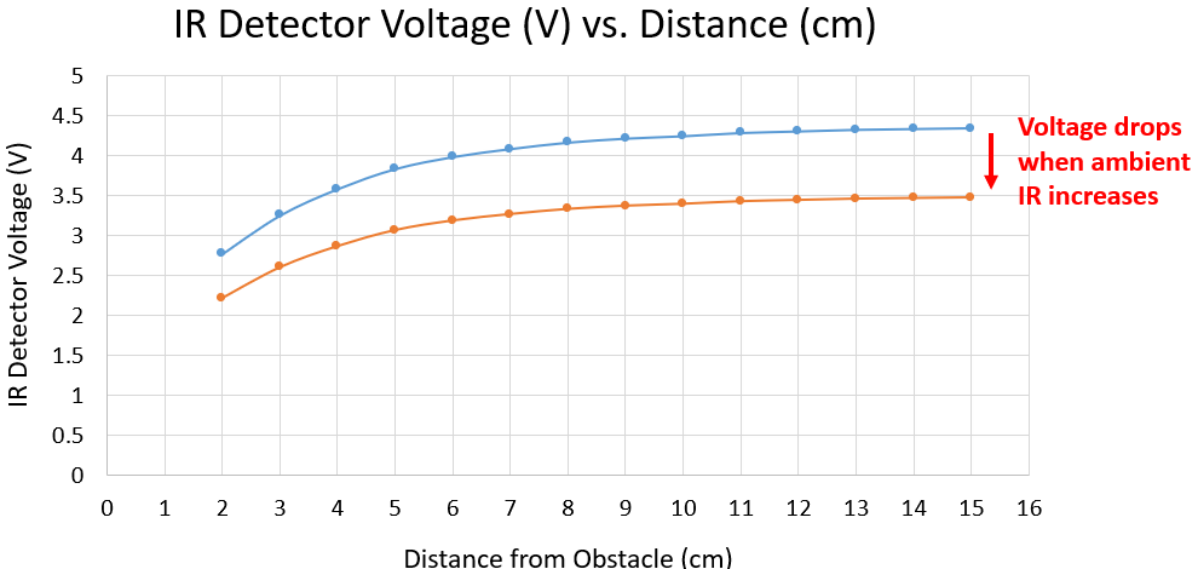


Figure 14: IR detector's voltages become lower when ambient IR increases.

As the ambient IR can change dynamically within different parts of the maze (can even depend on your mBot's direction), we need to adapt our range estimation algorithm to the changing ambient IR. **As an engineer, how would you solve this issue?** The trick is that we can turn off the IR emitter intermittently, during which our IR detector's voltage will be **solely due to the ambient IR**. We shall refer to this voltage as the "baseline" voltage (which is representative of the amount of ambient IR at that moment). Then, when we turn on our IR emitter, the **amount of dipping** in the IR detector voltage can be used to estimate the distance away from the maze wall. Note that for the same maze wall distance, the amount of dipping in terms of absolute voltage may be slightly dependent on the baseline voltage. We will leave the solution to aspiring engineers like you.

Figure 15 illustrates how the IR detector voltage may vary when the IR emitter turns on and off. Note that the IR detector voltage takes some time (say, 1 ms) to stabilize after the IR emitter is turned on. **Do not read the IR detector voltage immediately after turning on the IR emitter**!

Note: When the IR emitter and IR detector are placed immediately side by side, there is actually some IR leaking from the side of the emitter directly to the detector. This makes the detector less sensitive to the IR reflected from the obstacle. You may want to leave a **gap** equivalent to one breadboard column (about 2 mm) between the IR emitter and IR detector.

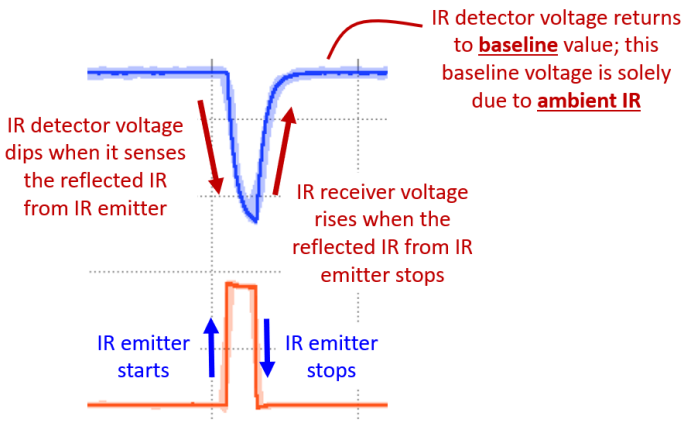


Figure 15: Illustration of how the IR detector voltage may vary when the IR emitter turns on and off.

2) Tackling Ambient Lighting Variations

Even though the colour sensor is somewhat shielded by mBot's shadow, the LDR voltage readings corresponding to red, green, and blue LED lights still vary for the same piece of colour paper when the ambient lighting varies. You should build a chimney using black paper and shield your entire colour sensing circuit from the ambient lighting. In addition, it is **strongly recommended** that you build an additional layer of skirting around your mBot using black paper to further shield off the effect of ambient lighting from your colour sensor. Also block off any holes on the mBot chassis (e.g., top, side, rear, etc.) where stray light can enter the colour sensor's region. In other words, it works best when it is totally pitch-dark underneath the mBot. This **saves you a lot of time** in your colour sensor's calibration later on.

You should start calibrating your colour sensor only when it has been mounted underneath your robot, and the black paper chimney/skirting have been completed. In other words, **only calibrate it in the exact same condition as how the robot will be operated**. Any premature calibration could be a waste of effort as it does not account for the actual operating conditions. If the black paper chimney/skirting is done well, you can even perform the calibration at home using the colour paper samples provided.

To ensure that you have built a robust colour sensor, you must test your mBot in not just one maze in the lab. Instead, you should try it in **all the mazes** in the lab. Your mBot is only regarded as robust if the colour sensor works perfectly in all mazes. If you tune your colour sensor to work in just one particular maze, it demonstrates **poor engineering skills**.

3) Building Neat and Tight Breadboard Circuits

You should take pride in the breadboard circuits you build. There should be minimal amount of wires protruding. This can be accomplished by using the **trimmable single-core wires** in the lab. Feel free to use these trimmable wires, wire-cutters, and resistors in the DSA Lab for building your circuits.

Bear in mind that your circuits also need to be rugged so that the components do not fall off easily when you knock the mBot accidentally during transportation, or when the mBot knocks onto a maze wall. The result will be disastrous if a component falls off or gets shifted during the final project demo!

Do not use unnecessarily long wires, and **do not abuse the wires**. The colour ribbon wires given to you contains tiny strands of copper wires inside. **If you twist the wires too much, they will break inside** – this kind of problem is **very hard to debug**, and you certainly do not want to frustrate yourself spending time over this kind of bug.

Project Evaluation

1) Mock Evaluation:

   All groups except B02: **Week 12 Studio 2 timeslot**

   Group B02: **Week 13 Studio 1 timeslot**

   This will be a good gauge of your mBot's readiness. The evaluation procedure will be the same as the final evaluation. Hence, you should get your mBot ready as if it is the final evaluation, so that you know what are your mBot's shortfalls and improve upon them.

2) Final Evaluation: **Week 13 Studio 2** timeslot

Rules

1. At each challenge, if your mBot turns in the wrong direction, it will be teleported to one grid before the challenge to make a second attempt, and if necessary, a third attempt, while the clock continues to run. If it fails at the third attempt, it will be manually turned to the correct direction.

2. If your mBot gets stuck to a wall, you can move it to the correct position within the same grid.

3. You are not allowed to add any commercial-off-the-shelf sensors that are not issued by us.

4. The actual maze layout for the A-maze-ing race **will not** be revealed beforehand. The figure below shows a **sample** maze layout.

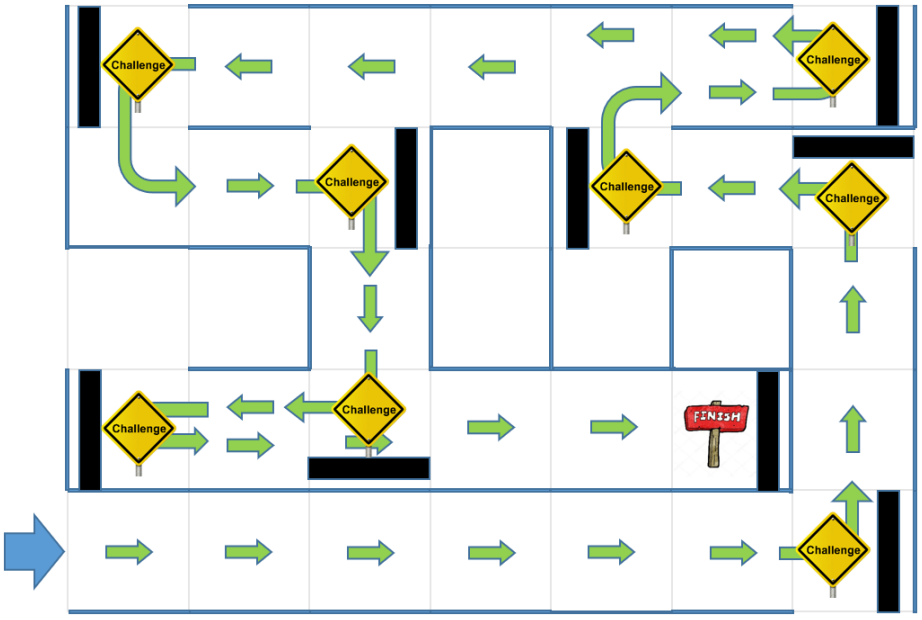


Figure 16: A sample maze layout.

5. As can be seen in the sample maze layout, **some of the walls may be missing**. Your mBot must be able to handle such missing walls (e.g., continue going straight even if one or two side walls is/are missing).

   Note:
   - There will always be a maze wall in front of the mBot within any waypoint challenge grid.

6. Your mBot will be tested in **two maze tables** (one specified by us, and one of your own choice) to ensure that you did not overly tune your mBot to work in just one maze. Your project demo marks will be the average of your mBot's performance in these two mazes.

7. You are **not allowed to perform any calibration** at the start of your project demo. Hence, **all calibrations and software upload** must have been **completed** before surrendering the mBots to the instructors.

8. You are allowed to fine-tune your mBot's hardware and software and go for a re-run of the demo if the following conditions occur, subject to a **20% penalty** of the total demo marks:
   a. If your mBot fails to decode more than half of the colour challenges.
   b. If your mBot keeps bumping into walls or getting stuck in the maze.

Grading Criteria

Criterion	Marks
Project Demo	25
Neatness of wirings and robot	5
Algorithms and coding (e.g., elegance of algorithms, well commented codes, etc.)	10
Short team report	10
Total	**50**

The following are the key grading criteria during the project demo:

- Number of bumps into the maze walls (regardless of whether it gets stuck); note that wires brushing against the wall are also regarded as bumps.
- Successfully decoding the challenges: for each challenge, your mBot has up to 3 tries. You get full marks for a challenge if you succeed in the 1st attempt; some penalty marks will be incurred if you succeed in the 2nd or the 3rd attempt.
- Whether your mBot can travel in a near-straight line when not turning.
- Whether your mBot can execute a turn accurately without over- or under-manoeuvring too much.
- Whether the mBot plays a celebratory tone upon detecting the end of the maze, and stops moving.
- Total time taken to complete the maze.

Deliverables (Short Team Report and Source Codes)

Deadline: **20 Nov 2023 (Monday), 2359 hrs**

(10% will be deducted for every day it is late)

Zip all files (source code + PDF report) into a single .zip file (one submission per team), name it according to your StudioGroup-SectionNo-TeamNo, e.g., **B01-S1-T2.zip** (for Studio Group B01, Section 1, Team 2), and upload it into the "Project Submission" folder in Canvas. You must include the following:

1. Program source codes. The codes must be well documented by providing appropriate comments.

2. A concise written report, in **PDF format**. You may use your own discretion for the number of pages. Your report must describe your design in detail, especially about how the required features are met. Please include:

   - A cover page with your Studio Group Number, Section Number, and Team Number, along with the **team members' names**.
   - **Pictures** of your **mBot** and **sensor breadboard circuits**. These will be used for grading the neatness of the wirings and the robot.
   - Description of the overall algorithm your mBot uses to solve the maze. You may include pseudocode/flowcharts or other pictorial aids to assist in your explanation.
   - Implementation details of the various subsystems – algorithms for keeping mBot straight, colour sensing, IR proximity sensing, etc.
   - Steps taken for calibrating/improving the robustness of your custom-built sensors.
   - Details about work division within your team – each member's role in the project.
   - Any significant difficulties and the steps taken to overcome them.

Peer Evaluation

You will submit a confidential peer evaluation about your teammates' contribution in the project. This will be used to moderate the marks among the team members. A non-contributing team member can get several grades lower than the rest of the team. If there is any teammate who is not contributing to the project despite nudging, **please inform us as early as possible**.

Academic Integrity

Plagiarism will not be tolerated. You are **not** supposed to share any code with other teams. You **may** discuss the project requirements or your solution strategies at a high-level, without sharing details at the code level. We do **not** distinguish between those who copy others' work, and those who allow their work to be copied. If you are involved in plagiarism, you will be given 0 mark for the project, and referred to the University for disciplinary action.